Restricted Power - Computational Complexity Results for Strategic Defense Games

de Haan, R.; Wolf, P.

DOI
10.4230/LIPIcs.FUN.2018.17

Publication date
2018

Document Version
Final published version

Published in
9th International Conference on Fun with Algorithms

License
CC BY

Link to publication

Citation for published version (APA):
de Haan, R., & Wolf, P. (2018). Restricted Power - Computational Complexity Results for Strategic Defense Games. In H. Ito, S. Leonardi, L. Pagli, & G. Prencipe (Eds.), *9th International Conference on Fun with Algorithms: FUN 2018, June 13-15, 2016, La Maddalena Island, Italy* [17] (Leibniz International Proceedings in Informatics; Vol. 100). Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing. https://doi.org/10.4230/LIPIcs.FUN.2018.17

Restricted Power – Computational Complexity Results for Strategic Defense Games

Ronald de Haan
Institute for Logic, Language and Computation, University of Amsterdam, the Netherlands
me@ronalddehaan.eu
https://orcid.org/0000-0003-2023-0586

Petra Wolf
Wilhelm-Schickard-Institut, University of Tübingen, Germany
wolfp@informatik.uni-tuebingen.de

Abstract

We study the game *Greedy Spiders*, a two-player strategic defense game, on planar graphs and show PSPACE-completeness for the problem of deciding whether one player has a winning strategy for a given instance of the game. We also generalize our results in metatheorems, which consider a large set of strategic defense games. We achieve more detailed complexity results by restricting the possible strategies of one of the players, which leads us to Σ_2^p- and Π_2^p-hardness results.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Computational complexity, generalized games, metatheorems

Digital Object Identifier 10.4230/LIPIcs.FUN.2018.17

Funding Ronald de Haan is supported by the Austrian Science Fund (FWF), project J4047.

Acknowledgements We thank Janosch Döcker and Britta Dorn for proof-reading.

1 Introduction

With computational devices in nearly everyone's pockets nowadays, the opportunities to play puzzle games on these devices are plentiful. What makes such games so addictive that they are played every day by millions of people? One possible answer to the suggested question is that (generalized variants of) these games are computationally intractable [9, 13], which could explain why it can be so challenging to find a solution or to get a good score.

In this paper, we analyze the two-player strategic defense game *Greedy Spiders* [3] from a computational complexity perspective. In the game Player 1 must prevent Player 2 from reaching designated positions. In particular, we show that the problem of deciding whether Player 1 or 2 has a winning strategy is PSPACE-complete. We also generalize this result to state two metatheorems, which can be applied to a larger set of strategic defense games. These metatheorems additionally claim that the problem becomes Σ_2^p-hard if we restrict the possible strategies of Player 1 to those that can be specified by a polynomial-time computable algorithm that is to be submitted at the beginning of the game – the problem is Π_2^p-hard if we restrict Player 2 in a similar way. In both cases, the question is whether Player 1 has a winning strategy. We get hardness results for the complementary classes, if we ask whether Player 2 has a winning strategy.

2 Related Work

While two-player board games have been studied well from a computational complexity point of view in the 80's [15, 16], two-player computer games are rarely examined till now. Despite the fact that the first gaming console was released in 1983 [4], it took until 2000 until the first computer games where studied in terms of their computational complexity. A good survey is given by Demaine et al. [9] and Kendall et al. [13]. While the first results where obtained by examining concrete games, Demaine et al. [10] made a first approach to find more general structures in games by developing a directed graph based framework for which they showed several hardness results for different versions. The framework was intended to be "a natural problem to reduce from." That approach instantly led to complexity results for games like *Sokoban*, *Rush Hour*, *Pushing blocks*, and many more. This proposal was taken up by Forišek [12], who coined the term "metatheorem" to describe complexity results for abstracted games consisting of a combination of game elements, which are often implemented in real computer games. The reduction from a metatheorem to a concrete computer game is obtained by proving that all the elements of one metatheorem can be implemented with the mechanics provided by the game. Note that this is often much easier than finding an individual reduction from a computationally hard formal problem to a certain computer game. Viglietta [19] further developed this approach with several metatheorems that are particularly useful for platform- and puzzle-games. His metatheorems have been used to study the complexity of the best-known Nintendo games [6]. Demaine, Lockhart and Lynche also continued the study of metatheorems for platform-games [11]. To our knowledge, very few of the currently known metatheorems are suitable to describe two-player games (the only one known to us are given by Demaine and Hearn [10]) and most of the known metatheorems are applicable to single-player platform- and puzzle-games only.

3 Greedy Spiders

We describe the game *Greedy Spiders* [3] as a two-player game – in the version of the game for iOS and Android devices, the user plays as Player 1, and the moves of Player 2 are determined by the application. In this paper, we consider the most basic variant of the game. We describe the game in intuitive terms, before we give a fully detailed formal description of the game.

Informal Description of the Game

Greedy Spiders is a turn-based strategic defense two-player game played on planar graphs (that represent spider webs). Initially, some nodes of the graph are occupied by spiders, and some nodes of the graph are occupied by flies. The players alternate turns, and Player 1 plays first. In each turn of Player 1, she removes an edge from the graph, and Player 2 in each of her turns moves a subset of the spiders (possibly all) along a remaining edge of the graph to an adjacent node. The flies cannot move. Player 2 wins whenever some spider occupies the same node as some fly, and Player 1 wins whenever there is no path anymore from any of the spiders to any of the flies.

Formal Description of the Game

A *game situation* (for *Greedy Spiders*) is represented by a triple $C = (G, S, F)$, where $G = (V, E)$ is an undirected planar graph, $S \subseteq V$ is the set of nodes that are occupied by spiders, and $F \subseteq V$ is the set of nodes that are occupied by flies.

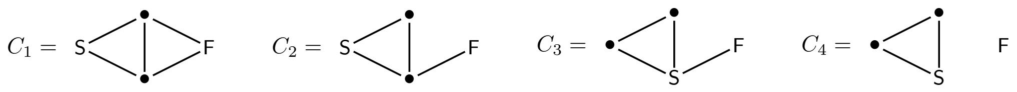

Figure 1 Example of a run $\sigma = (C_1, \dots, C_4)$ that is winning for Player 1.

A *valid move for Player 1* consists of a tuple (C_1, C_2), where $C_1 = (G_1, S_1, F_1)$ and $C_2 = (G_2, S_2, F_2)$ are game situations, such that $S_1 = S_2$, $F_1 = F_2$, and G_2 is obtained from G_1 by removing one edge, that is, $G_1 = (V, E)$ and $G_2 = (V, E \setminus \{e\})$ for some $e \in E$. A *valid move for Player 2* consists of a tuple (C_1, C_2), where $C_1 = (G_1, S_1, F_1)$ and $C_2 = (G_2, S_2, F_2)$ are game situations, for which holds that $G_1 = G_2 = (V, E)$; that $S_2 = \{ f(s) \mid s \in S_1 \}$, where $f \colon S_1 \to V$ is an injective function such that for all $s \in S_1$ it holds that $\{s, f(s)\} \in E$ or $f(s) = s$; and that $F_1 = F_2$.

A game situation (G, S, F) is *winning for Player 1* if for each $s \in S$ and each $f \in F$, there is no path in G from s to f. A game situation (G, S, F) is *winning for Player 2* if $S \cap F \neq \emptyset$. A game situation is *terminal* if it is winning for either of the players, and it is *non-terminal* otherwise.

A *run* σ of the game is a finite sequence $(C_1, \dots, C_n)$ of game situations where (1) for each odd $i \in [n-1]$ it holds that (C_i, C_{i+1}) is a valid move for Player 1, (2) for each even $i \in [n-1]$ it holds that (C_i, C_{i+1}) is a valid move for Player 2, (3) for each $i \in [n-1]$ it holds that C_i is non-terminal, and (4) C_n is terminal. (For each $u, v \in \mathbb{N}$, we use $[u]$ to denote the set $\{1, \dots, u\}$ and $[u, v]$ to denote the set $\{u, \dots, v\}$.) The run σ is winning for either of the players if and only if C_n is.

▶ **Example 1.** See Figure 1 for an example of a run σ of the game *Greedy Spiders* that is winning for Player 1. In this figure (as in all figures in this paper), nodes that are occupied by a spider are marked with S and nodes that are occupied by a fly are marked with F.

We invite the reader to play the game and to verify that there is in fact a winning strategy for Player 1 for the initial game situation C_1 depicted in Figure 1.

A *strategy for Player 1* for an initial game situation C_1 is a finite tree T where each node is labeled with a pair (C, j), where C is a game situation and $j \in [2]$, that satisfies the following conditions:

(a) the root of T is labeled with $(C_1, 1)$;
(b) whenever a node is labeled with $(C, 1)$, for some non-terminal game situation C, it has one single child that is labeled with $(C', 2)$ such that (C, C') is a valid move for Player 1;
(c) whenever a node is labeled with $(C, 2)$, for some non-terminal game situation C, it has m children nodes that are labeled with $(C_1, 1), \dots, (C_m, 1)$, respectively, where $\{(C, C_1), \dots, (C, C_m)\}$ is the set of all valid moves for Player 2 that have C as first component; and
(d) whenever a node is labeled with (C, j), for some terminal game situation C and some $j \in [2]$, it has no children (i.e., it is a leaf).

In the remainder of this paper, we will often slightly abuse notation by identifying a node of a strategy T with the pair (C, j) with which it is labeled. A strategy T for Player 1 is *winning* if all its leaves are labeled with pairs (C, j) where C is winning for Player 1. (In fact, it can easily be verified that this can only be the case if each leaf is labeled with a pair $(C, 2)$ for some game situation C that is winning for Player 1.) Note that any root-to-leaf path in the strategy T corresponds to a run of the game.

Intuitively, a strategy for Player 1 specifies a sequence of valid moves for Player 1 for each possible combination of valid moves that Player 2 makes. A winning strategy for Player 1 specifies what moves Player 1 can make to ensure that she wins the game. (Winning) strategies for Player 2 are defined analogously. Since *Greedy Spiders* is a zero-sum game, there is a winning strategy for Player 1 if and only if there is no winning strategy for Player 2.

Decision Problem

We consider the following decision problems in this paper.

WINNER DETERMINATION FOR PLAYER 1 *Input:* An initial game situation C_1.
Question: Is there a winning strategy for Player 1 for the game situation C_1?

WINNER DETERMINATION FOR PLAYER 2 *Input:* An initial game situation C_1.
Question: Is there a winning strategy for Player 2 for the game situation C_1?

Because the game *Greedy Spiders* never ends in a tie (either Player 1 or Player 2 wins), these problems are complementary. That is, Player 1 has a winning strategy if and only if Player 2 does not have a winning strategy.

4 Preliminaries

We assume the reader to be familiar with basic notions from the theory of computational complexity, such as the complexity classes P and NP, and polynomial-time (many-to-one) reductions. For more details, we refer to textbooks on the topic (e.g., see [7]).

The class PSPACE consists of all decision problems that can be solved by an algorithm that uses a polynomial amount of space (memory). Alternatively, one can characterize the class PSPACE as all decision problems for which there exists a polynomial-time reduction to the problem TQBF, that is defined using quantified Boolean formulas as follows. A quantified Boolean formula (in prenex form) is a formula of the form $Q_1x_1Q_2x_2\dots Q_nx_n.\psi$, where all x_i are propositional variables, each Q_i is either an existential or a universal quantifier, and ψ is a (quantifier-free) propositional formula over the variables $x_1,\dots,x_n$ (called the *matrix*). Truth for such formulas is defined in the usual way. The problem TQBF consists of deciding whether a given quantified Boolean formula is true. It is well-known that the problem TQBF is PSPACE-complete, and that it remains PSPACE-hard even when restricted to quantified Boolean formulas whose matrix is in 3CNF.

The class PSPACE can also be characterized using alternating Turing machines (ATMs). A problem is in PSPACE if and only if it can be solved in polynomial time by an alternating Turing machine [8]. We refer to textbooks on complexity theory for more details (e.g., see [7]).

One can also restrict the number of quantifier alternations occurring in quantified Boolean formulas, i.e., the number of times where $Q_i \neq Q_{i+1}$. For each constant $k \geq 1$ number of alternations, this leads to a different complexity class. These classes together constitute the Polynomial Hierarchy. We consider the complexity classes Σ^{p}_k, for each $k \geq 1$. The complexity class Σ^{p}_k consists of all decision problems for which there exists a polynomial-time reduction to the problem $\mathrm{TQBF}_{\exists,k}$, that is defined as follows. Instances of the problem are quantified Boolean formulas of the form $\exists x_1 \dots \exists x_{\ell_1} \forall x_{\ell_1+1} \dots \forall x_{\ell_2} \dots Q_k x_{\ell_{k-1}+1} \dots Q_k x_{\ell_k}.\psi$, where $Q_k = \exists$ if k is odd and $Q_k = \forall$ if k is even, where $1 \leq \ell_1 \leq \dots \leq \ell_k$, and where ψ is quantifier-free. The problem is to decide if the quantified Boolean formula is true. For each $k \geq 1$, the dual problem $\mathrm{TQBF}_{\forall,k}$ is defined analogously, where the first quantifier of the formula is universal rather than existential. The complexity class Π^{p}_k consists of all decision problems for which there exists a polynomial-time reduction to the problem $\mathrm{TQBF}_{\forall,k}$. The class NP coincides with Σ^{p}_1, and the class co-NP coincides with Π^{p}_1.

5 Complexity Results for Greedy Spiders

In this section, we show that the problems Winner Determination for Player 1 and Winner Determination for Player 2 for *Greedy Spiders* are PSPACE-complete. Since these problems are complementary, we focus on Winner Determination for Player 1. The result for Winner Determination for Player 2 will then follow immediately, because PSPACE is closed under complement. We begin with showing membership in PSPACE.

▶ **Lemma 2.** Winner Determination for Player 1 *for* Greedy Spiders *is in* PSPACE.

Proof. Let $C_1 = (G, S, F)$ be an initial game situation, where $G = (V, E)$. Since each valid move for Player 1 removes an edge from G, we know that every possible run σ that starts with C_1 is of length at most $2|E|-1$. Therefore, the problem can be solved in polynomial time by an alternating Turing machine. We describe the algorithm that is implemented by such an alternating Turing machine. The algorithm starts with a partial run $\sigma = (C_1)$ that is extended to a complete run. Then, whenever the partial run $\sigma = (C_1, \dots, C_\ell)$ ends with a non-terminal game situation C_ℓ and is of odd length, the algorithm uses existential nondeterminism to guess a game situation $C_{\ell+1}$ such that $(C_\ell, C_{\ell+1})$ is a valid move for Player 1, resulting in the partial run $(C_1, \dots, C_{\ell+1})$. Whenever the partial run $\sigma = (C_1, \dots, C_\ell)$ ends with a non-terminal game situation C_ℓ and is of even length, the algorithm uses universal nondeterminism to guess a game situation $C_{\ell+1}$ such that $(C_\ell, C_{\ell+1})$ is a valid move for Player 2, resulting in the partial run $(C_1, \dots, C_{\ell+1})$. Whenever the partial run $\sigma = (C_1, \dots, C_\ell)$ ends with a terminal game situation C_ℓ, the algorithm accepts if and only if C_ℓ is winning for Player 1. ◀

Next, to show PSPACE-hardness of Winner Determination for Player 1 for *Greedy Spiders*, we will need a technical lemma that states that TQBF is PSPACE-hard even when restricted to instances with a matrix in 3DNF whose incidence graph is planar.

Let $\varphi = Q_1 x_1 \dots Q_n x_n.\psi$ be a quantified Boolean formula, where ψ is a quantifier-free DNF formula. Suppose that $\psi = d_1 \vee \dots \vee d_m$. The *incidence graph* G_φ of φ is a bipartite graph that is defined as follows. The nodes V_φ of G_φ are the literals and the terms of ψ, i.e., $V_\varphi = \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\} \cup \{d_1, \dots, d_m\}$. A node corresponding to a literal l is connected by an edge to a node corresponding to a term d_j if and only if l occurs in the term d_j. The incidence graph of a formula with a matrix in CNF is defined analogously. (Often a variant of incidence graphs with vertices only for variables, not literals, is used.)

▶ **Lemma 3.** TQBF *is* PSPACE*-hard even when restricted to quantified Boolean formulas (in prenex form) whose incidence graph is planar and whose matrix is a 3DNF formula.*

Proof. It has been shown that TQBF remains PSPACE-hard when restricted to quantified Boolean formulas (in prenex form) whose matrix is a 3CNF formula and whose incidence graph is planar [14, Theorem 1]. This result can easily be adapted to work also for incidence graphs with vertices for literals (by introducing existentially quantified copies of variables and adding clauses to ensure that copies are assigned the same truth value). Then, since PSPACE is closed under complement, and the negation of a quantified Boolean formula whose matrix is in 3CNF is equivalent to a formula whose matrix is in 3DNF, the result follows. ◀

▶ **Theorem 4.** Winner Determination for Player 1 *for* Greedy Spiders *is* PSPACE*-complete.*

Proof. Membership in PSPACE is shown in Lemma 2. We show PSPACE-hardness by means of a polynomial-time reduction from TQBF. Take an arbitrary instance $\varphi = \exists x_1.\forall x_2 \dots \exists x_{n-1}.\forall x_n.\psi$, where $\psi = d_1 \vee \dots \vee d_m$ is a quantifier-free 3DNF formula with n

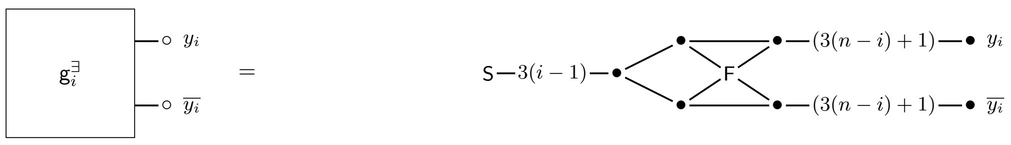

Figure 2 Gadget $\mathsf{g}_i^{\exists}$ for variable x_i, for odd i.

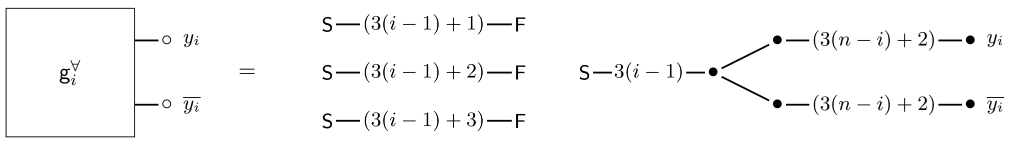

Figure 3 Gadget $\mathsf{g}_i^{\forall}$ for variable x_i, for even i.

variables and m terms – without loss of generality we may assume that the odd-numbered variables x_i are existentially quantified, and that the even-numbered variables x_i are universally quantified. Moreover, by Lemma 3, we may assume that the incidence graph of φ is planar. Also, without loss of generality, we may assume that n is even and that $m \geq 2$.

We construct a game situation $C_1 = (G, S, F)$ as follows. We construct the planar graph $G = (V, E)$, together with the sets $S \subseteq V$ and $F \subseteq V$ by connecting various gadgets for the variables and terms of φ.

The idea of the reduction is as follows. We introduce gadgets $\mathsf{g}_i^{\exists}$ that allow Player 1 to choose a truth assignment for variable x_i, for odd i. Similarly, for even i, we have gadgets $\mathsf{g}_i^{\forall}$ that allow Player 2 to choose a truth assignment for variable x_i. These choices are made one after the other, so that they can depend on the truth assignment of preceding variables. The choices in these first gadgets consist of sending a spider on one of two paths. Then, we have gadgets k_i and k'_i, that serve to let the spiders from gadgets $\mathsf{g}_i^{\exists}$ and $\mathsf{g}_i^{\forall}$ pass onwards, while giving Player 1 time to cut free flies in all but one of the gadgets h_j representing the terms of ψ. If the chosen truth assignment satisfies a term d_j, Player 1 can safely leave the fly in gadget h_j unprotected (and cut free the flies in all other gadgets $\mathsf{h}_{j'}$). In order to make this function properly, we additionally have gadgets f_ℓ, forcing Player 1 to cut free a fly in this gadget in one of her first ℓ turns. Figure 7 illustrates this for an example.

For each existentially quantified variable x_i – that is, for every odd $i \in [n]$ – we add the gadget $\mathsf{g}_i^{\exists}$ as depicted in Figure 2. For each universally quantified variable x_i – that is, for every even $i \in [n]$ – we add the gadget $\mathsf{g}_i^{\forall}$ as depicted in Figure 3. In these figures, nodes in S are marked with S and nodes in F are marked with F. Also, each edge that is marked with a number ℓ represents a path containing ℓ edges (where each of the non-depicted nodes are neither in S nor in F). In particular, if $\ell = 0$, the two nodes adjacent to this edge coincide.

Intuitively, the gadgets $\mathsf{g}_i^{\exists}$ and $\mathsf{g}_i^{\forall}$ simulate the quantification over the truth assignments to the variables $x_1, \dots, x_n$. For each $i \in [n]$, in Player 1's $(3(i-1)+1)$-th, $(3(i-1)+2)$-th and $(3(i-1)+3)$-th turn, she is forced to make a move in gadget $\mathsf{g}_i^{\exists}$ or $\mathsf{g}_i^{\forall}$ (depending on the parity of i), in order to prevent the spider in this gadget from capturing a fly in this gadget. Moreover, in gadgets $\mathsf{g}_i^{\exists}$, her choices for these moves determine which of the two paths leading to the nodes labeled y_i and $\overline{y_i}$, respectively, are still available to the spider in this gadget. In the gadgets $\mathsf{g}_i^{\forall}$, Player 2 is free to choose on which of the two paths, leading to the nodes labeled y_i and $\overline{y_i}$, respectively, the spider in this gadget moves. Moving a spider on the path towards y_i corresponds to setting variable x_i to true, and moving a spider on the path towards $\overline{y_i}$ corresponds to setting variable x_i to false. Thus, in this way, Player 1 can choose the truth values for the odd-numbered variables x_i and Player 2 can choose the truth values for the even-numbered variables x_i.

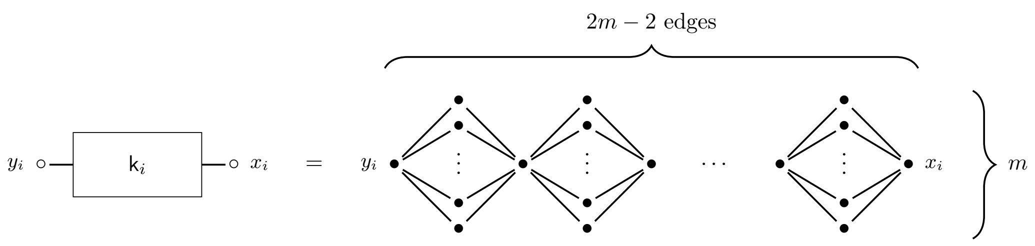

Figure 4 Secondary gadget k_i for literal x_i. The secondary gadget $\mathsf{k}_i^{\neg}$ for literal $\overline{x_i}$ is entirely similar, replacing y_i by $\overline{y_i}$ and x_i by $\overline{x_i}$.

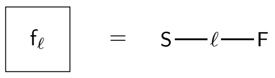

Figure 5 Gadget f_ℓ in which Player 1 is forced to remove an edge in her ℓ-th turn (at the latest).

Then, for each $i \in [n]$, we identify the node labeled with y_i in the gadget $\mathsf{g}_i^{\exists}$ or $\mathsf{g}_i^{\forall}$ with the node labeled with y_i in the gadget k_i that is depicted in Figure 4. We similarly identify the node labeled with $\overline{y_i}$ in the gadget $\mathsf{g}_i^{\exists}$ or $\mathsf{g}_i^{\forall}$ with the node labeled with $\overline{y_i}$ in the gadget $\mathsf{k}_i^{\neg}$, which is entirely similar to the gadget depicted in Figure 4 – the only difference is that the node label y_i is replaced by $\overline{y_i}$ and the node label x_i is replaced by $\overline{x_i}$. These gadgets consist of $m - 1$ successive pieces, each consisting of m parallel paths of length 2 – here m is the number of terms occurring in the matrix ψ of the quantified Boolean formula φ. Intuitively, the purpose of these gadgets k_i and $\mathsf{k}_i^{\neg}$ is to ensure that there remains a path of length $2m-2$ from the node labeled with y_i to the node labeled with x_i, even after the next $2m - 2$ moves (and similarly for the nodes labeled with $\overline{y_i}$ and $\overline{x_i}$).

For each even $\ell \in [3n + 1, 3n + 2m - 2]$ (so not the odd values), we add the gadget f_ℓ, as depicted in Figure 5. These gadgets force Player 1 to make a move in gadget f_ℓ in her ℓ-th turn (at the latest). As a result, Player 1 has no way of preventing any spider to move from a node labeled with y_i to a node labeled with x_i in her $(3n+1)$-th until her $(3n+2m-2)$-th turn (while also preventing the flies in the gadgets f_ℓ from getting captured by a spider). However, notably, for each odd $\ell \in [3n + 1, 3n + 2m - 2]$, Player 1 is not forced to delete any particular edge in the graph in her ℓ-th turn (in order to avoid losing directly after that turn). This free choice for Player 1 will play a role in the next type of gadget that we will add.

For each term d_j of ψ, we add the gadget h_j, as depicted in Figure 6. The leftmost nodes in this gadget are labeled with x_i or $\overline{x_i}$. We identify these leftmost nodes with the nodes in gadgets k_i and $\mathsf{k}_i^{\neg}$ that have identical labels. Suppose that $d_j = (l_{j,1} \wedge l_{j,2} \wedge l_{j,3})$, where each $l_{j,u}$, for $u \in [3]$, is either x_i or $\overline{x_i}$ for some $i \in [n]$. Then the leftmost nodes in the gadget h_j coincide with the nodes in gadgets k_i and $\mathsf{k}_i^{\neg}$ that are labeled with $\overline{l_{j,u}}$, denoting the complementary literal of $l_{j,u}$. For example, if $d_j = (x_1 \wedge \overline{x_2} \wedge x_3)$, then the leftmost nodes in the gadget h_j are identified with the nodes labeled with $\overline{x_1}$, x_2 and $\overline{x_3}$ in gadgets $\mathsf{k}_1^{\neg}$, k_2 and $\mathsf{k}_3^{\neg}$.

Intuitively, the gadgets h_j all contain a fly that needs to be protected from the incoming spiders on the paths from x_i and $\overline{x_i}$. Player 1 has time to remove the edges adjacent to the flies in exactly $m - 1$ of these gadgets h_j – she has time to do this in her ℓ-th turns, for odd values of $\ell \in [3n + 1, 3n + 2m - 2]$. In other words, Player 1 needs to choose exactly one $j \in [m]$ such that the fly in gadget h_j is out of reach of the spiders, for her next two turns.

Finally, we add the gadgets f_ℓ, as depicted in Figure 5, for both $\ell \in [3n+2m-1, 3n+2m]$ to ensure that after rescuing the flies in all but one of the gadgets h_j, Player 1 has to make a

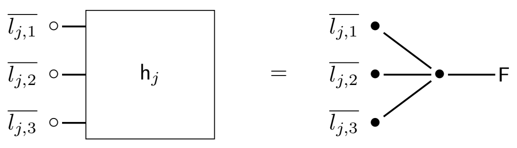

Figure 6 Gadget h_j for the term $d_j = (l_{j,1} \wedge l_{j,2} \wedge l_{j,3})$.

move in these gadgets in her next two turns. In other words, if the fly in the unique gadget h_j whose safety she did not ensure by deleting its adjacent edge is being approached by some spider within distance 2, this spider will then be able to capture the fly. If this is not the case, Player 1 can ensure the safety of this final fly in her $(3n + 2m + 1)$-th turn.

Clearly, this reduction runs in polynomial time. Moreover, since the incidence graph of φ is planar, the graph G that we constructed is also planar.

Verifying the correctness of this reduction is straightforward using the intuitions behind and explanations of the workings of the gadgets $\mathsf{g}_i^{\exists}$, $\mathsf{g}_i^{\forall}$, k_i, $\mathsf{k}_i^{\neg}$, f_ℓ, and h_j – that we gave above – together with the following observations.

The first observation is that for each odd $i \in [n]$, Player 1 can decide which of the two paths, towards the nodes labeled with y_i or $\overline{y_i}$, are left open for the spider in gadget $\mathsf{g}_i^{\exists}$, and she can base this choice on her choices in the gadgets $\mathsf{g}_{i'}^{\exists}$, for odd $i' \in [i]$ and Player 2's choices in the gadgets $\mathsf{g}_{i'}^{\forall}$, for even $i' \in [i]$. Similarly, for each even $i \in [n]$, Player 2 can decide which of the two paths, towards the nodes labeled with y_i or $\overline{y_i}$, are taken by the spider in gadget $\mathsf{g}_i^{\forall}$, and she can base this choice on her choices in gadgets $\mathsf{g}_{i'}^{\forall}$, for even $i' \in [i]$ and Player 1's choices in the gadgets $\mathsf{g}_{i'}^{\exists}$, for odd $i' \in [i]$.

The second observation is that whenever a truth assignment satisfies ψ, it must satisfy some term d_j of ψ. This means that it must satisfy all literals in d_j, and thus must make all their complements false. Therefore, if (and only if) the spiders are on their way towards the nodes labeled with x_i and $\overline{x_i}$ in such a way that the corresponding truth assignment satisfies ψ (and thus satisfies d_j for some $j \in [m]$), Player 1 can safely leave the fly in gadget h_j unprotected during her $(3n + 1)$-th until $(3n + 2m)$-th turn.

This concludes our proof of PSPACE-hardness. ◀

▶ **Example 5.** Consider the quantified Boolean formula $\varphi = \exists x_1.\forall x_2.\exists x_3.\forall x_4.[d_1 \vee d_2]$, where $d_1 = (x_1 \wedge x_2 \wedge x_3)$ and $d_2 = (x_1 \wedge \overline{x_2} \wedge x_3)$. The game situation $C_1 = (G, S, F)$ as constructed in the proof of Theorem 4 is depicted (schematically) in Figure 7. (Note that the last universally quantified variable (x_4) does not occur in the terms d_1 and d_2 – its presence makes n even.)

▶ **Corollary 6.** *Winner Determination for Player 2 for* Greedy Spiders *is* PSPACE-*complete.*

Proof. This follows directly from Theorem 4, since PSPACE is closed under complement and the problems Winner Determination for Player 1 and Winner Determination for Player 2 are complementary. ◀

6 Metatheorems

For our metatheorems, we consider games that are turn-based two-player games modeled on graphs. In the unrestricted version, the players alternate turns and every player has unlimited resources in every turn to calculate her next move. A player is called *strategically*

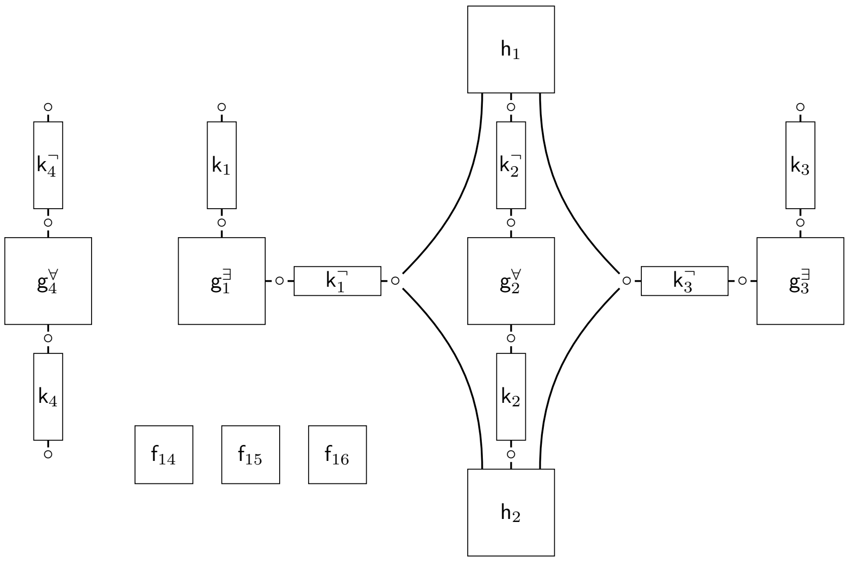

Figure 7 The game situation $C_1 = (G, S, F)$ that is constructed from the quantified Boolean formula $\varphi = \exists x_1.\forall x_2.\exists x_3.\forall x_4.[(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \overline{x_2} \wedge x_3)]$ in the proof of Theorem 4 – see Example 5. (The nodes that are drawn in this picture between two gadgets are the nodes that belong to both gadgets, and are identified – e.g., the node drawn between the gadgets $g_1^{\exists}$ and $k_1^{\neg}$ is the node labeled with $\overline{y_1}$ that appears in both gadgets.)

restricted if she chooses in her first move a deterministic polynomial-time algorithm with a polynomial-size description, that determines all of her moves on that game instance. The algorithm is then disclosed to the opposing player. This means that the other player can then calculate the reaction of her opponent for any possible situation in polynomial time.

We consider the following game mechanics. A game is said to implement *defense positions*, if there exist positions which must not be reached by attackers of Player 2. Paths that can be eliminated or permanently blocked by Player 1 are called *destroyable paths.* Player 1 has the ability to destroy one destroyable path in each of her turns, while Player 2, in each of her turns, moves all of her attackers (towards the defense positions of Player 1) over one edge each. Player 1 wins the game if there is no path left from any attacker of Player 2 to any defense position of Player 1. Conversely, Player 2 wins if at least one of her attackers has reached a defense position of Player 1.

The decision problems WINNER DETERMINATION FOR PLAYER 1 and WINNER DETERMINATION FOR PLAYER 2 for games that implement defense positions and destroyable paths are defined analogously as for the game of *Greedy Spiders* (see Section 3).

▶ **Metatheorem 7.** *For a round-based two-player game implementing defense positions and destroyable paths, the problem* WINNER DETERMINATION FOR PLAYER 1 *is:*
(1) PSPACE*-hard if neither of the players is strategically restricted;*
(2) Σ_2^{p}*-hard, if Player 1 is strategically restricted; and*
(3) Π_2^{p}*-hard, if Player 2 is strategically restricted.*
These hardness results hold even when the game is restricted to planar graphs.

Proof (idea). Statement (1) follows as a corollary from the proof of our hardness result for Theorem 4. The reduction used in this proof is entirely based on the game mechanics of defense positions and destroyable paths. We will prove Statement (2) by modifying the hardness reduction from the proof of Theorem 4 to a reduction from the Σ_2^{p}-complete problem

$\mathrm{TQBF}_{\exists,2}$ – we will explain this in more detail below. Similarly, we will prove Statement (3) by modifying the same reduction to a reduction from an appropriate Π_2^{p}-complete variant of the problem $\mathrm{TQBF}_{\forall,3}$ – we also work this out in more detail below. All these reductions also work when restricted to planar graphs. ◀

Asking the converse question (i.e., whether Player 2 can win) leads to the following metatheorem.

▶ **Metatheorem 8.** *For a round-based two-player game implementing defense positions and destroyable paths, the problem* Winner Determination for Player 2 *is:*
(1) PSPACE*-hard if neither of the players is strategically restricted;*
(2) Π_2^{p}*-hard, if Player 1 is strategically restricted; and*
(3) Σ_2^{p}*-hard, if Player 2 is strategically restricted.*
These hardness results hold even when the game is restricted to planar graphs.

Proof. Because the problems Winner Determination for Player 1 and Winner Determination for Player 2 are complementary, these statements follow directly from Metatheorem 7. ◀

We now turn to proving Metatheorem 7(2–3).

Proof of Metatheorem 7(2). We describe how the hardness reduction from the proof of Theorem 4 can be used to form a reduction from $\mathrm{TQBF}_{\exists,2}$ to Winner Determination for Player 1 where Player 1 is strategically restricted. Let $\varphi = \exists x_1 \dots \exists x_{n_1} \forall x_{n_1+1} \dots \forall x_n . \psi$ be an instance of $\mathrm{TQBF}_{\exists,2}$. Without loss of generality, we may assume that ψ is in 3DNF and has a planar incidence graph.

We consider the formula $\varphi' = \exists x_1 \forall y_1 \dots \exists x_{n_1-1} \forall y_{n_1-1} \exists x_{n_1} \forall x_{n_1+1} \exists y_{n_1+1} \dots \forall x_{n-1} \exists y_{n-1}$ $\forall x_n . \psi$, where the variables in $Y = \{y_1, \dots, y_{n_1-1}, y_{n_1+1}, \dots, y_{n-1}\}$ are fresh variables that do not occur in ψ. That is, φ' differs from φ only in that variables from Y are added to the quantifier prefix to ensure that existential and universal quantifiers alternate. We know that φ is true if and only if φ' is true. Then, because the quantifiers in φ' alternate between existential and universal quantifiers, we can employ the reduction from the proof of Theorem 4 to construct a game situation C_1 where Player 1 has a winning strategy if and only if φ' is true (which is the case if and only if φ is true).

All that remains to show that whenever Player 1 has a winning strategy for C_1, she can – in her first turn – submit an algorithm (whose description is of polynomial size) that computes the moves of her winning strategy in polynomial time. By construction of the game instance C_1, and because the variables $y_1, \dots, y_{n_1-1}$ do not occur in ψ, we know that any winning strategy for Player 1 does not depend on Player 2's moves in the gadgets $\mathsf{g}_i^{\forall}$ corresponding to the variables $y_1, \dots, y_{n_1-1}$. Moreover, since the variables $y_{n_1+1}, \dots, y_{n-1}$ do not occur in ψ, Player 1's optimal strategy in the gadgets $\mathsf{g}_i^{\exists}$ corresponding to the variables $y_{n_1+1}, \dots, y_{n-1}$ is easy to determine. Player 1's only moves that depend on the choice of Player 2 in the gadgets $\mathsf{g}_i^{\forall}$ are Player 1's moves in the gadgets h_j, and Player 1's optimal moves in these latter gadgets are easy to determine – these moves correspond to evaluating ψ once the truth value of each variable is set. Therefore, the optimal moves for carrying out her winning strategy can be generated by a polynomial-time algorithm that she can submit at the beginning of the game. Thus, this reduction works for the case where Player 1 is strategically restricted. ◀

In order to prove Metatheorem 7(3), we consider a Σ_2^{p}-complete variant of $\mathrm{TQBF}_{\exists,3}$.

▶ **Lemma 9.** *There is a class of quantified Boolean formulas of the form* $\varphi = \exists x_1 \dots \exists x_{\ell_1} \forall y_1 \dots \forall y_{\ell_2} \exists z_1 \dots \exists z_{\ell_3}.\psi$ *with the following properties:*

(1) $\mathrm{TQBF}_{\exists,3}$ *restricted to this class of quantified Boolean formulas is* Σ^{p}_2*-complete;*

(2) *each quantified Boolean formula* φ *in this class has a matrix in 3CNF and has a planar incidence graph; and*

(3) *for each quantified Boolean formula* $\varphi = \exists x_1 \dots \exists x_{\ell_1} \forall y_1 \dots \forall y_{\ell_2} \exists z_1 \dots \exists z_{\ell_3}.\psi$ *in this class, and for each truth assignment* $\alpha : \{x_1, \dots, x_{\ell_1}, y_1, \dots, y_{\ell_2}\} \to \{0, 1\}$*, it can be decided in polynomial time (given* φ *and* α*) if there exists a truth assignment* $\beta : \{z_1, \dots, z_{\ell_3}\} \to \{0, 1\}$ *such that* $\psi[\alpha \cup \beta]$ *evaluates to true, and such a truth assignment* β *can be computed in polynomial time, if it exists.*

Proof. We provide a reduction from $\mathrm{TQBF}_{\exists,2}$ to $\mathrm{TQBF}_{\exists,3}$ and show that the class of quantified Boolean formulas that are produced by this reduction has Properties (1)–(3). Hardness for Σ^{p}_2 for the problem $\mathrm{TQBF}_{\exists,3}$ restricted to this class of quantified Boolean formulas follows immediately from this reduction.

Let φ be an instance of $\mathrm{TQBF}_{\exists,2}$. Without loss of generality, we may assume that φ has a matrix ψ in 3DNF. We then transform the matrix ψ to 3CNF using the standard Tseitin transformation [18], by adding additional existentially quantified variables at the end of the quantifier prefix – this will result in an equivalent quantified Boolean formula φ' with an "$\exists\forall\exists$" quantifier prefix. We then transform φ' into an equivalent quantified Boolean formula φ'' with a matrix in 3CNF and a planar incidence graph using the gadgets used in the proof that 3SAT restricted to planar formulas is NP-hard [14, Theorem 1] – this will add additional existentially quantified variables at the end of the quantifier prefix.

The reduction clearly results in quantified Boolean formulas that satisfy Property (2). The resulting formulas also satisfy Property (3). Once the variables from the original quantified Boolean formula φ have been instantiated, only clauses corresponding to the introduced gadgets in the two-step reduction described above (containing only existentially quantified variables) remain – finding satisfying truth assignments for these remaining clauses can be done in polynomial time. This is because both steps in the reduction have the property that given any satisfying truth assignment α for the matrix of the original formula, one can compute in polynomial time a truth assignment β such that $\alpha \cup \beta$ satisfies the matrix of the constructed formula – and that both steps of the reduction are reversible in polynomial time. For the first step of the reduction (where the matrix ψ is transformed to 3CNF) this is the case because the introduced clauses form a renamable Horn formula – thus after instantiating the formula with α, a renamable Horn formula remains, and a satisfying truth assignment for renamable Horn formulas can be found in polynomial time. For the second step of the reduction (where the formula is transformed to an equivalent formula that has a planar incidence graph) this property follows directly from the shape of the gadgets used in the reduction [14, Theorem 1].

As a result of Property (3), we get membership in Σ^{p}_2 for the problem $\mathrm{TQBF}_{\exists,3}$ restricted to quantified Boolean formulas produced by the reduction above. Together with Σ^{p}_2-hardness, this gives us Property (1). ◀

The main idea behind the proof of Σ^{p}_2-hardness is to apply Tseitin transformations [18] to inputs of the problem $\mathrm{TQBF}_{\exists,2}$. We denote the problem $\mathrm{TQBF}_{\exists,3}$ restricted to the class of quantified Boolean formulas identified in Lemma 9 by $\mathrm{TQBF}^{\star}_{\exists,3}$. Similarly, we consider the Π^{p}_2-complete dual problem $\mathrm{TQBF}^{\star}_{\forall,3}$, that concerns formulas that are equivalent to the negation of instances of $\mathrm{TQBF}^{\star}_{\exists,3}$.

Proof (sketch) of Metatheorem 7(3). We modify the proof of Theorem 4 to a reduction from the problem $\text{TQBF}^{\star}_{\forall,3}$. These modifications are entirely analogous to the modifications in the proof of Metatheorem 7(2). That is, we introduce new variables (not occurring in the matrix of the quantified Boolean formula) to ensure that existential and universal quantifiers alternate strictly.

In the resulting game, whenever Player 2 has a winning strategy that corresponds to a way of assigning the universally quantified variables that makes the remaining formula false (for any assignment to the existentially quantified variables), the optimal moves for carrying out this strategy can be generated by a polynomial-time algorithm that she can submit at the beginning of the game. This is because her only moves that (non-trivially) depend on the choice of Player 1 in the gadgets $\mathsf{g}_i^{\exists}$ are her moves in the gadgets $\mathsf{g}_i^{\forall}$ corresponding to the variables in the third quantified block and her moves in the gadgets h_j, and Player 2's optimal moves in these latter gadgets are easy to determine – this is due to Lemma 9(3). Thus, this reduction works for the case where Player 2 is strategically restricted. ◀

7 Application of Metatheorems

In this section we describe how to apply our metatheorems to tower-defense games. Games of this genre can be described as two-player games where the defending Player 1 must prevent the attackers of Player 2 from reaching designated locations on the playing field. For this purpose Player 1 can place towers on the field which damage every attacker in their reach. To place the towers, Player 1 usually has to pay some amount of a currency which is steadily credited to Player 1 over time. In most tower-defense games Player 1 is played by the user, while Player 2 is played by the computer. The strategy of Player 2 is fixed per instance, but differs from instance to instance, so we will apply Metatheorem 7(3).

To apply Metatheorem 7(3), we have to show that all elements of the metatheorem can be modeled within the game. Defense positions are naturally a part of tower-defense games, since they all include positions which have to be protected from the attacking enemies. Destroyable paths are implemented in the following way. A path is said to be destroyed if no attacker can cross it (and survive). Therefore we can destroy a path by placing a strong enough tower somewhere on the path to kill every attacker in its reach. The accessible environment of this tower is regarded as the destroyed path. Every spot on the map where a tower can be placed represents therefore a destroyable path. While most tower defense-games are not round-based in a strong sense, we can still model them as round-based. To implement the game elements, we only have to consider one type of attackers and one type of towers. Since Player 1 earns coins of a currency every fixed amount of time we can graduate the time in steps which are as long as it takes Player 1 to earn enough coins to buy one tower instance. The step range of the attackers of Player 2 is therefore as long as the distance they can walk in one time step. Thus we can assume the game to be round-based. Since all criteria of Metatheorem 7(3) can be implemented, this shows that tower-defense games in general are Π_2^{p}-hard.

In concrete terms, the above described implementation works among others for games like *Bloons Tower Defense 5* [2], *Warcraft 3* [1], and *Starcraft* [5].

8 Conclusion

We showed PSPACE-completeness for the problem of deciding whether Player 1 has a winning strategy for the game *Greedy Spiders*, as well as for the problem of deciding whether

Player 2 has a winning strategy. Afterwards we generalized the idea of our proof to give two metatheorems referring to [11, 12, 19], which granulate the computational complexity of the core element of the game by restricting the computational power of the players. In particular, we showed that WINNER DETERMINATION FOR PLAYER 1 in a turn-based two-player game containing defense positions and destroyable paths is in general PSPACE-hard, becomes Σ_2^p-hard if Player 1 is strategically restricted, and Π_2^p-hard if Player 2 is strategically restricted. The reverse question of WINNER DETERMINATION FOR PLAYER 2 is in general PSPACE-hard, becomes Π_2^p-hard if Player 1 is strategically restricted, and Σ_2^p-hard if Player 2 is strategically restricted. Finally, we discussed the applicability of our metatheorems on tower-defense games and mentioned some specific games to which our metatheorems can be applied.

Finding metatheorems for the computational complexity of computer games has recently become more and more of a focus. With tower-defense games, we grazed with our metatheorems a previously untouched game genre in terms of computational complexity and provided new tools to investigate them. As most metatheorems are discovered in the area of platform- and puzzle-games, they can only be applied to single-player games. Therefore with our metatheorems, we give new impulses in looking for metatheorems, which describe multiplayer (specifically two-player) games. To our knowledge, our results are the first hardness results for the complexity classes Σ_2^p and Π_2^p in the field of computational complexity of computer games.

A possibility for further research in this field is to look at two-player games and restrict the computational power of one of the players. This approach could also be applied to well studied board games like *Chess*, *Checkers*, or *Mill*. In general the field of multiplayer strategy games seems to afford more yet undiscovered metatheorems and should be investigated in the future. Beside tower-defense games, our metatheorem should also be applicable to other strategic games, such as war simulations or any game in which one player has the role of a defender who has to prevent the other player (with the role of an attacker) from reaching certain locations in the game. Over the last few years, more and more complex and modern games have been explored, resulting in metatheorems which are applicable to state of the art games. Since many modern computer games provide scripting languages with whom the players can modify the game, the games themselves are instantly Turing-complete. We think that examining restricted versions of these games is still worth a try and can lead to metatheorems for the essential elements of the games, taking off the focus from the scripting languages.

References

1 Blizzard Entertainment: Warcraft III. `http://eu.blizzard.com/en-gb/games/war3/`. Accessed: 2018-02-17.

2 Bloons Tower Defense 5. `http://bloons.wikia.com/wiki/Bloons_Tower_Defense_5`. Accessed: 2018-02-17.

3 Greedy Spiders. `http://greedyspiders.com/`. Accessed: 2018-02-17.

4 Nintendo Entertainment System (NES). `http://www.pcgames.de/Nintendo-Entertainment-System-NES-Konsolen-255246/`. Accessed: 2018-02-01.

5 StarCraft: Remastered. `https://starcraft.com/en-us/`. Accessed: 2018-02-17.

6 Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo Games are (Computationally) Hard. *Theoretical Computer Science*, 586:135–160, 2015.

7 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.

8 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. of the ACM*, 28(1):114–133, 1981.

9 Erik D. Demaine. Playing Games with Algorithms: Algorithmic Combinatorial Game Theory. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 18–33. Springer, 2001.

10 Erik D. Demaine and Robert A. Hearn. Constraint Logic: A Uniform Framework for Modeling Computation as Games. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, 2008 (CCC 2008)*, pages 149–162. IEEE, 2008.

11 Erik D. Demaine, Joshua Lockhart, and Jayson Lynch. The Computational Complexity of Portal and Other 3D Video Games. *arXiv preprint 1611.10319*, 2016.

12 Michal Forišek. Computational Complexity of Two-Dimensional Platform Games. In *Proceedings of the 5th International Conference on Fun with Algorithms (FUN 2010)*, pages 214–227. Springer, 2010.

13 Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A Survey of NP-complete Puzzles. *ICGA Journal*, 31(1):13–34, 2008.

14 David Lichtenstein. Planar Formulae and Their Uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

15 John Michael Robson. The Complexity of Go. In *IFIP Congress*, pages 413–417, 1983.

16 John Michael Robson. N by N Checkers is Exptime complete. *SIAM J. Comput.*, 13(2):252–267, 1984.

17 Jörg Siekmann and Graham Wrightson, editors. *Automation of reasoning. Classical Papers on Computer Science 1967–1970*, volume 2. 1983.

18 G. S. Tseitin. Complexity of a Derivation in the Propositional Calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, 8:23–41, 1968. English transl. repr. in [17].

19 Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory Comput. Syst.*, 54(4):595–621, 2014. `doi:10.1007/s00224-013-9497-5`.