THE COMPUTATIONAL COMPLEXITY COLUMN

BY

VIKRAMAN ARVIND

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

Combinatorial games are a fascinating topic, as both recreational and serious mathematics. One aspect of the mathematical theory deals with assigning values that quantify positions in games. This has lead to deep connections between numbers and games and a rich algebraic theory as discovered by Conway.

A natural algorithmic question that arises is the following: given a position in a combinatorial game as input, determine which player has a winning strategy. In their article, Steve Fenner and John Rogers give an excellent self-contained introduction to the computational complexity of combinatorial games. They first explain the basic theory of combinatorial games, with focus on poset games. With this background, they survey the complexity-theoretic results in this field and discuss several open questions.

Combinatorial Game Complexity: An Introduction with Poset Games

Stephen A. Fenner
University of South Carolina,
Computer Science and Engineering Department

John Rogers
DePaul University,
School of Computing

Abstract

Poset games have been the object of mathematical study for over a century but little has been written on the computational complexity of determining important properties of these games. In this introduction we define combinatorial games and focus for the most part on impartial poset games, of which Nim is perhaps the best-known example. We present the complexity results known to date, some discovered very recently.

An extended version of this paper, with detailed proofs, is available online at `http://www.cse.sc.edu/~fenner/papers/games.html` and on `arXiv.org`.

1 Introduction

Combinatorial games have long been studied (see [5, 1], for example) but the record of results on the complexity of questions arising from these games is rather spotty. Our goal in this introduction is to present several results—some old, some new—addressing the complexity of the fundamental problem given an instance of a combinatorial game:

> Determine which player has a winning strategy.

A secondary, related problem is

> Find a winning strategy for one or the other player, or just find a winning first move, if there is one.

The former is a decision problem and the latter a search problem. In some cases, the search problem clearly reduces to the decision problem, i.e., having a solution for the decision problem provides a solution to the search problem. In other cases this is not at all clear, and it may depend on the class of games you are allowed to query.

We give formal definitions below, but to give an idea of the subject matter, we will discuss here the large class of games known as the *poset games*. One of the best known of these is Nim, an ancient game, but given its name by Charles Bouton in 1901 [2]. There are many others, among them, Hackendot, Divisors, and Chomp [5]. Poset games not only provide good examples to illustrate general combinatorial game concepts, but they also are the subject of a flurry of recent results in game complexity, which is the primary focus of this article.

The rest of this section gives some basic techniques for analyzing poset games. Section 2 lays out the foundations of the general theory of combinatorial games, including numeric and impartial games. The rest of the paper is devoted to computational complexity. Section 3 gives an upper bound on the complexity of so-called "N-free" games, showing that they are solvable in polynomial time. Section 4 gives lower bounds on the complexity of some games, showing they are hard for various complexity classes. The section culminates in two recent **PSPACE**-completeness results—one for impartial poset games, and the other for "black-white" poset games. Section 5 discusses some open problems.

An extended version of this paper, with detailed proofs, is available online at `http://www.cse.sc.edu/~fenner/papers/games.html` and on `arXiv.org`.

1.1 Poset games

Definition 1.1. A *partial order* on a set P (hereafter called a *poset*) is a binary relation $\leq$ on P that is reflexive, transitive, and antisymmetric (i.e., $x \leq y$ and $y \leq x$ imply $x = y$). For any $x \in P$, define $P_x := \{y \in P \mid x \nleq y\}$.

We identify a finite poset P with the corresponding *poset game*: Starting with P, two players (Alice and Bob, say) alternate moves, Alice moving first, where a move consists of choosing any point x in the remaining poset and removing all y such that $x \leq y$, leaving P_x remaining. Such a move we call *playing x*. The first player unable to move (because the poset is empty) loses.[1]

Poset games are *impartial*, which means that, at any point in the play, the set of legal moves is the same for either player. There is a rich theory of impartial games, and we cover it in Section 2.5.

[1]Games can be played on some infinite posets as well, provided every possible sequence of moves is finite. This is true if and only if the poset is a well-quasi-order (see, e.g., Kruskal [18]).

In an impartial game, the only meaningful distinction between players is who plays first (and we have named her Alice). Since every play of a poset game has only finitely many moves, one of the two players (but clearly not both!) must have a winning strategy. We say that a poset P is an $\exists$-*game* (or *winning position*) if the first player has a winning strategy, and P is a $\forall$-*game* (or *losing position*) if the second player has a winning strategy. In the combinatorial game theory literature, these are often called $\mathcal{N}$-games ("Next player win") and $\mathcal{P}$-games ("Previous player win"), respectively. We get the following concise inductive definition for any poset P:

P is an $\exists$-game iff there exists $x \in P$ such that P_x is a $\forall$-game.
P is a $\forall$-game iff P is not an $\exists$-game (iff, for all $x \in P$, P_x is an $\exists$-game).

We call the distinction of a game being a $\forall$-game versus an $\exists$-game the *outcome* of the game.

There are at least two natural ways of combining two posets to produce a third.

Definition 1.2. For posets $P = \langle P, \leq_P \rangle$ and $Q = \langle Q, \leq_Q \rangle$,

- define $P + Q$ (the *parallel union of P and Q*) to be the disjoint union of P and Q, where all points in P are incomparable with all points in Q:

$$P + Q := \langle P \mathbin{\dot\cup} Q, \leq \rangle ,$$

where $\leq := \leq_P \mathbin{\dot\cup} \leq_Q$.

- Define P/Q (or $\frac{P}{Q}$—the *series union of P over Q*) to be the disjoint union of P and Q where all points in P lie above (i.e., are $\geq$ to) all points in Q:

$$\frac{P}{Q} := \langle P \mathbin{\dot\cup} Q, \leq \rangle ,$$

where $\leq \; := \; \leq_P \mathbin{\dot\cup} \leq_Q \mathbin{\dot\cup} (Q \times P)$.

Note that $+$ is commutative and associative, and that $/$ is associative but not commutative. Using these two operations, let's build some simple posets. Let C_1 be the one-element poset. For any $n \in \mathbb{N}$, let

1. $C_n := \underbrace{C_1/C_1/\ldots/C_1}_{n}$ is the chain of n points (totally ordered). This is also called a *NIM stack*.

2. $A_n := \underbrace{C_1 + C_1 + \cdots + C_1}_{n}$ is the antichain of n pairwise incomparable points.

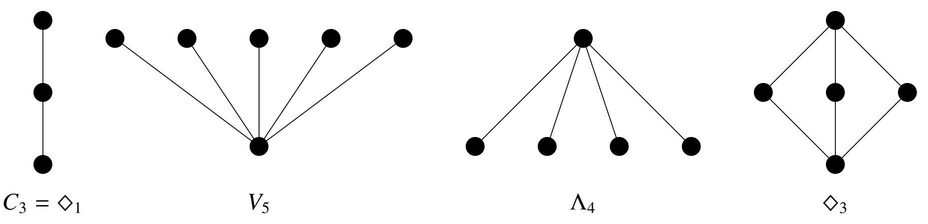


Figure 1: Some simple posets constructed from individual points via parallel and series union.

3. $V_n := A_n/C_1$ is the n-antichain with a common lower bound.

4. $\Lambda_n := C_1/A_n$ is the n-antichain with a common upper bound.

5. $\diamond_n := C_1/A_n/C_1$ is the n-antichain with common upper and lower bounds.

Some examples are shown in Figure 1.

Exercise 1.3. Find a simple way, given m and n, to determine whether A_m/A_n is an $\exists$-game or a $\forall$-game.

Exercise 1.4. Show that P/Q is an $\exists$-game if and only if either P is an $\exists$-game or Q is an $\exists$-game.

1.1.1 More examples

The best-known poset game is NIM, an ancient game first formally described and "solved" by C. L. Bouton in 1902 [2]. Here, the poset is a union of disjoint chains, i.e., of the form $C_{n_1} + C_{n_2} + \cdots + C_{n_k}$ for some positive integers $n_1, \ldots, n_k$. A move then consists of choosing a point in one of the chains and remove that point and everything above it.

Other families of poset games include

CHOMP, introduced in 1974 by D. Gale [11], which, in its finite form, is represented by a rectangular arrangement of squares with the leftmost square in the bottom row removed. This is a poset with two minimal elements (first square on the second row, second square on bottom row). Every element in a row is greater than all of the elements to the left and below so playing an element removes it and all elements to the right and above.

HACKENDOT, attributed to von Newmann, where the poset is a forest of upside-down trees (roots at the top). HACKENDOT was solved in 1980 by Úlehla [29].

DIVISORS, introduced by F. Schuh [22], the poset is the set of all positive divisors (except 1) of a fixed integer n, partially ordered by divisibility. DIVISORS is a multidimensional generalization of CHOMP. CHOMP occurs as the special case where $n = p^m q^n$ for distinct primes p, q.

1.2 Dual symmetry

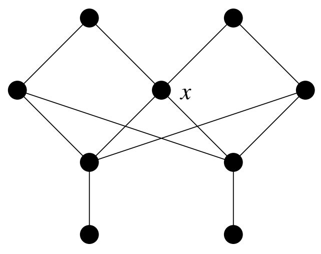
Some poset games can be determined (as $\exists$-games or $\forall$-games just by inspection). For example, suppose a poset P has some kind of dual symmetry, that is, there is an order-preserving map $\varphi : P \to P$ such that $\varphi \circ \varphi = \mathrm{id}$.

Fact 1.5. *Let P be a poset and let $\varphi : P \to P$ be such that $\varphi \circ \varphi = \mathrm{id}_P$ and $x \leq y \iff \varphi(x) \leq \varphi(y)$ for all $x, y \in P$.*

- *If φ has no fixed points, then P is a $\forall$-game.*
- *If φ has a minimum fixed point (minimum among the set of fixed points), then P is an $\exists$-game.*

Proof. If φ has no fixed points, then Bob can answer any x played by Alice by playing $\varphi(x)$. If φ has a least fixed point z, then Alice plays z on her first move, leaving P_z, which is symmetric with no fixed points, and thus a $\forall$-game. □

For example, the poset below is symmetric with a unique fixed point x, which Alice can win by playing on her first move:



1.3 Strategy stealing

Another class of posets that are easy to determine by inspection are those with an *articulation point*, i.e., a point that is comparable with every other point in the poset. For example, minimum and maximum points of P are articulation points.

Fact 1.6. *If a poset P contains an articulation point, then P is an $\exists$-game.*

Proof. Let x be some articulation point of P. If x is a winning first move for Alice, then we are done. If x is a losing first move for Alice, then there must be some winning response y for Bob if Alice first plays x. But if Alice plays x, then all points $\geq x$ are now gone, and so we have $y < x$. This means that the game after Bob moves is P_y, which is a $\forall$-game by assumption. But then, Alice could have played y instead on her first move, leaving the $\forall$-game P_y for Bob, and thus winning. $\square$

We call this "strategy stealing" because Alice steals Bob's winning strategy. The interesting thing about this proof is how nonconstructive it is. It shows that Alice has a winning first move, but gives virtually no information about what that first move could be. All we know is that the winning first play must be $\leq x$. If x is a maximum point of P, then the proof gives no information whatsoever about Alice's winning first move. Several poset games, including Chomp, have initial posets with maximum points, so we know that they are $\exists$-games. But determining a winning first move for Alice in Chomp appears quite difficult, and no fast algorithm is known. This suggests that, in the case of Chomp at least, the search problem (finding a winning first move) is apparently difficult, whereas the decision problem ($\exists$-game or $\forall$-game?) is trivial. The search versus decision issue is discussed further in Section 4.1, below.

Exercise 1.7. Show that the winning first moves in any poset form an antichain.

1.4 Black-white poset games

Many interesting games are not impartial because the legal moves differ for the players. In chess, for example, one player can only move white pieces and the other only black pieces. We will informally call a game "black-white" when each player is assigned a color (black or white) and can only make moves corresponding to their color.[2] Many impartial games have natural black-white versions. Here, then, is a black-white version of a poset game:

Definition 1.8. A *black-white poset game* consists of finite poset P, each of whose points are colored either black or white. The same rules apply to black-white poset games as to (impartial) poset games, except that one player (Black) can only play black points and the other player (White) can only play white points. (All points above a played point are still removed, regardless of color.)

One could generalize this definition by allowing a third color, grey, say, where grey points can be played by either player. We will not pursue this idea further.

[2]A different, popular color combination is red-blue. We use black-white so that illustrations are faithfully rendered on a black-and-white printer.

Other "colored" games include red-blue Hackenbush and red-green-blue Hackenbush [1].

Combinatorial games that are not impartial are known as *partisan*. In partisan games, we must make a distinction between the two players beyond who moves first. Generically, these players are called Left and Right. There is a surprisingly robust general theory of combinatorial games, both impartial and partisan, developed in [1, 5], and we give the basics of this theory in the next section.

2 Combinatorial game theory basics

In this section we give some relevant definitions and a few facts from the general theory of combinatorial games. We give enough of the theory to understand later results. Thorough treatments of this material, with lots of examples, can be found in [1, 5] as well as other sources, e.g., the recent book by Siegel [23]. Our terminology and notation vary a little bit from [1, 5], but the concepts are the same. When we say, "game," we always mean what is commonly referred to as a *combinatorial game*, i.e., a game between two players, say, Left and Right, alternating moves with perfect information, where the first player unable to move loses (and the other wins). In their fullest generality, these games can be defined abstractly by what options each player has to move, given any position in the game.

To save space, we will omit proofs of the results of this section, leaving them as exercises to the reader. These proofs are usually straightforward applications of previous results in this section, or induction, or both. The extended paper contains the full proofs.

2.1 Notation

We let $\mathbb{N}$ denote the set $\{0, 1, 2, \ldots, \}$ of natural numbers. We let $|X|$ denote the cardinality of a finite set X. We use the relation ":=" to mean "equals by definition." We extend the definition of an operator on games to an operator on sets of games in the customary way; for example, if $*$ is a binary operation on games, and G and H are sets of games, then $G * H := \{g * h \mid g \in G \wedge h \in H\}$, and if g is a game, then $g * H := \{g\} * H$, and so on.

2.2 Basic definitions

Definition 2.1. A *game* is an ordered pair $G = (G^L, G^R)$, where G^L and G^R are sets of games. The elements of G^L (respectively, G^R) are the *left options* (respectively, *right options*) of G. An *option* of G is either a left option or a right option of G.

It is customary to write $\{G^L|G^R\}$ or $\{\ell_1, \ell_2, \ldots | r_1, r_2, \ldots\}$ rather than (G^L, G^R), where $G^L = \{\ell_1, \ell_2, \ldots\}$ and $G^R = \{r_1, r_2, \ldots\}$. We will do the same.

For this and the following inductive definitions to make sense, we tacitly assume that the "option of" relation is well-founded, i.e., there is no infinite sequence of games $g_1, g_2, \ldots$ where g_{i+1} is an option of g_i for all i.[3] A *position* of a game G is any game reachable by making a finite series of moves starting with G (the moves need not alternate left-right). Formally,

Definition 2.2. A *position* of a game G is either G itself or a position of some option of G. We say that G is *finite* iff G has a finite number of positions.[4]

Starting with a game G, we imagine two players, Left and Right, alternating moves as follows: the initial position is G; given the current position P of G (also a game), the player whose turn it is chooses one of her or his options of P (left options for Left; right options for Right), and this option becomes the new game position. The first player faced with an empty set of options loses. The sequence of positions obtained this way is a *play* of the game G. Our well-foundedness assumption implies that every play is finite, and so there must be a winning strategy for one or the other player. We classify games by who wins (which may depend on who moves first) when the players play optimally. This is our broadest and most basic classification. Before giving it, we first introduce the "mirror image" of a game G: define $-G$ to be the game where all left options and right options are swapped at every position, as if the players switched places. Formally,

Definition 2.3. For any game G, define $-G := \{-G^R | -G^L\}$.

It is a good warm-up exercise to prove—inductively, of course—that $-(-G) = G$ for every game G. For impartial games, e.g., poset games, the "$-$" operator has no effect; for black-white poset games, this is tantamount to swapping the color of each point in the poset.

We can consider the following definition to be the most fundamental property of a game:

Definition 2.4. Let G be a game. We say that $G \geq 0$ (or $0 \leq G$) iff there is no right option g^R of G such that $-g^R \geq 0$. We will say $G \leq 0$ to mean that $-G \geq 0$.

So $G \geq 0$ if and only if no right option g^R of G satisfies $g^R \leq 0$. Symmetrically, $G \leq 0$ if and only if no left option g^L of G satisfies $g^L \geq 0$. In terms of strategies, $G \geq 0$ means that G is a *first-move loss for Right* or a *second-move win for Left.*

[3]This follows from the Foundation Axiom of set theory, provided ordered pairs are implemented in some standard way, e.g., $(x, y) := \{\{x\}, \{x, y\}\}$ for all sets x and y.

[4]Finite games are sometimes called *short games*; see [23].

If Right has to move first in G, then Left can win. Symmetrically, $G \leq 0$ means that G is a *first-move loss for Left* or a *second-move win for Right*.

The $\leq$ notation suggests that a partial order (or at least, a preorder) on games is lurking somewhere. This is true, and we develop it below.

Definition 2.4 allows us to partition all games into four broad categories.

Definition 2.5. Let G be a game.

- G is a *zero game* (or a *first-move loss*, or $\mathcal{P}$-game) iff $G \leq 0$ and $G \geq 0$.
- G is *positive* (or a *win for Left*, or $\mathcal{L}$-game) iff $G \geq 0$ and $G \not\leq 0$.
- G is *negative* (or a *win for Right*, or $\mathcal{R}$-game) iff $G \leq 0$ and $G \not\geq 0$.
- G is *fuzzy* (or a *first-move win*, or $\mathcal{N}$-game) iff $G \not\leq 0$ and $G \not\geq 0$.

These four categories, $\mathcal{P}$ (for previous player win), $\mathcal{L}$ (for Left win), $\mathcal{R}$ (for Right win), and $\mathcal{N}$ (for next player win), partition the class of all games. The unique category to which G belongs is called the *outcome* of G, written $o(G)$.

For example, the simplest game is the *endgame* $0 := \{|\}$ with no options, which is a zero game ($o(0) = \mathcal{P}$). The game $1 := \{0|\}$ is positive ($o(1) = \mathcal{L}$), and the game $-1 := \{|0\}$ is negative $o(-1) = \mathcal{R}$, while the game $* := \{0|0\}$ is fuzzy ($o(*) = \mathcal{N}$).

2.3 Game arithmetic, equivalence, and ordering

Games can be added, and this is a fundamental construction on games. The sum $G + H$ of two games G and H is the game where, on each move, a player may decide in which of the two games to play. Formally:

Definition 2.6. Let G and H be games. We define

$$G + H := \{(G^L + H) \cup (G + H^L) \mid (G^R + H) \cup (G + H^R)\} .$$

In Section 1 we used the + operator for the parallel union of posets. Observe that this corresponds exactly to the + operator on the corresponding games, i.e., the game corresponding to the parallel union of posets P and Q is the game-theoretic + applied to the corresponding poset *games* P and Q.

We write $G - H$ as shorthand for $G + (-H)$. One can easily show by induction that + is commutative and associative when applied to games, and the endgame 0 is the identity under +. This makes the class of all games into a commutative monoid (albeit a proper class). One can also show for all games G and H that $-(G + H) = -G - H$. Furthermore, if $G \geq 0$ and $H \geq 0$, then $G + H \geq 0$. It is *not* the case, however, that $G - G = 0$ for all G, although $G - G$ is always a zero game. These easy results are important enough that we state and prove them formally.

Lemma 2.7. *For any games G and H,*

1. *$G - G$ is a zero game.*
2. *Suppose $G \geq 0$. Then $H \geq 0$ implies $G + H \geq 0$, and $H \not\leq 0$ implies $G + H \not\leq 0$.*
3. *Suppose $G \leq 0$. Then $H \leq 0$ implies $G + H \leq 0$, and $H \not\geq 0$ implies $G + H \not\geq 0$.*
4. *$-(G + H) = -G - H$.*

The outcome $o(G)$ of a game G is certainly the first question to be asked about G, but it leaves out a lot of other important information about G. It does not determine, for example, the outcome when G is added to a fixed game X. That is, it may be that two games G and H have the same outcome, but $o(G+X) \neq o(H+X)$ for some game X. Indeed, defining $2 := \{1|\}$, one can check that $o(1) = o(2) = \mathcal{L}$, but we have $o(2-1) = \mathcal{L}$ (left wins by choosing $1 \in 2^L$ when she gets the chance), whereas we know already from Lemma 2.7 that $o(1 - 1) = \mathcal{P}$.

Behavior under addition leads us to a finer classification of games.

Definition 2.8. Let G and H be games. We say that G and H are *equivalent*, written $G \approx H$, iff $o(G + X) = o(H + X)$ for all games X.[5]

It follows immediately from the definition that $\approx$ is an equivalence relation on games, and we call the equivalence classes *game values*. We let **PG** denote the Class[6] of all game values.[7] Letting X be the endgame 0 in the definition shows that equivalent games have the same outcome. Using the associativity of +, we also get that $G \approx H$ implies $G + X \approx H + X$ for any game X. Thus + respects equivalence and naturally lifts to a commutative and associative Operation (also denoted +) on **PG**.

The remaining goal of this subsection is finish showing that $\langle \mathbf{PG}, +, \leq \rangle$ is a partially ordered abelian Group. We have built up enough basic machinery that we can accomplish our goal in a direct, arithmetic way, without referring to players' strategies.

[5]In much of the literature, the overloaded equality symbol = is used for game equivalence. We avoid that practice here, preferring to reserve = for set theoretic equality. There are some important game properties that are not $\approx$-invariant.

[6]We will start to capitalize words that describe proper classes.

[7]Since each game value itself is a proper Class, we really cannot consider it as a member of anything. A standard fix for this in set theory is to represent each game value v by the *set* of elements of v with minimum rank, so **PG** becomes the Class of all such sets.

Lemma 2.9. *A game G is a zero game if and only if $G + H \approx H$ for all games H.*

Corollary 2.10. *A game G is a zero game if and only if $G \approx 0$ (where 0 is the endgame).*

Here is our promised Preorder on games.

Definition 2.11. Let G and H be games. We write $G \leq H$ (or $H \geq G$) to mean $H - G \geq 0$ (equivalently, $G - H \leq 0$). As usual, we write $G < H$ to mean $G \leq H$ and $H \nleq G$.[8]

You can interpret $G < H$ informally as meaning that H is more preferable a position for Left than G, or that G is more preferable for Right than H. For example, if Left is ever faced with moving in position G, and (let us pretend) she had the option of replacing G with H beforehand, she always wants to do so.

Proposition 2.12. *The $\leq$ Relation on games is reflexive and transitive.*

Proposition 2.13. *For any two games G and H, $G \approx H$ if and only if $G - H$ is a zero game, if and only if $G \leq H$ and $G \geq H$.*

The last two propositions show that the binary Relation $\leq$ on games is a Preorder that induces a partial Order on **PG**. Proposition 2.13 also gives a good working criterion for proving or disproving game equivalence—just check whether $G - H$ is a second player win—without having to quantify over all games.

Proposition 2.14. *$\langle$**PG**$, +\rangle$ is an abelian Group, where the identity element is the $\approx$-equivalence class of zero games, and inverses are obtained by the negation Operator on games.*

Finally, $\leq$ is translation-invariant on **PG**, making it a partially ordered abelian Group:

Corollary 2.15. *For any games G, H, and X, if $G \leq H$ then $G + X \leq H + X$.*

We next look at two important subclasses of games—the numeric games and the impartial games.

[8] We now have two ways of interpreting the expression "$G \geq 0$": one using Definition 2.4 directly and the other using Definition 2.11 with 0 being the endgame. One readily checks that the two interpretations coincide.

2.4 Numeric games

A numeric game is one where at each position all the left options are $<$ all the right options. Formally,

Definition 2.16. A game G is *numeric* iff $\ell < r$ for every $\ell \in G^L$ and $r \in G^R$, and further, every option of G is numeric.

One can show that G is numeric if and only if $\ell < G$ for every $\ell \in G^L$ and $G < r$ for every $r \in G^R$. If H is also numeric, then either $G \leq H$ or $H \leq G$. The $+$ and $-$ operations also yield numeric games when applied to numeric games.[9] Numeric games have a peculiar property: making a move only worsens your position (for Left this means having to choose a smaller game; for Right, having to choose a larger game). Thus neither player wants to make a move—if they were given the option to skip a turn, they would always take it. For these games, an optimal play is easy to describe: Left always chooses a maximum left option (i.e., one that does the least damage), and Right always chooses a minimum right option, assuming these options exist.[10] This intuitive idea is formalized in the following theorem, which is referred to in the literature as the "dominating rule." It applies to all games, not just numeric games.

Theorem 2.17. *Let G be a game. If $y \leq \ell$ for some $\ell \in G^L$, then $G \approx \{y, G^L | G^R\}$. Similarly, if $y \geq r$ for some $r \in G^R$, then $G \approx \{G^L | G^R, y\}$.*

If $y \leq \ell \in G^R$, then we say that y is *dominated* by ℓ in G. Similarly, if $y \geq r \in G^R$, then y is *dominated* by r in G. We obtain equivalent games by removing dominated options. A player never needs to play a dominated option; it is just as well (or better) to choose an option that dominates it.

Numeric games are called such because their values act like real numbers; for one thing, their values are totally ordered by $\leq$. These games are constructed in a way somewhat akin to how the real numbers are constructed from the rationals via Dedekind cuts. The left options of a game form the left cut, the right options the right cut, and the game itself represents a number strictly between the two. The differences are that the two cuts might be bounded away from each other (one or the other may even be empty), and the left cut might contain a maximum element.

2.4.1 Finite numeric games

The values of *finite* numeric games form a subgroup of **PG** naturally isomorphic (in an order-preserving way) to the dyadic rational numbers under addition, ac-

[9] The property of being numeric is *not* invariant under $\approx$. One can easily concoct two equivalent games, one of which is numeric and the other not.

[10] In general, Left can win by choosing any option $\ell \geq 0$, and Right can win by choosing any option $r \leq 0$.

cording to the following "simplicity rule":

Definition 2.18. Let G be a finite numeric game. The *(numerical) value* of G, denoted $v(G)$, is the unique rational number $a/2^k$ such that

1. k is the least nonnegative integer such that there exists an integer a such that $v(\ell) < a/2^k$ for all $\ell \in G^L$ and $a/2^k < v(r)$ for all $r \in G^R$, and

2. a is the integer with the least absolute value satisfying (1.) above.

So for example, the endgame 0 has value $v(0) = 0$, the game 1 has value $v(1) = 1$, and the game -1 has value $v(-1) = -1$, as the notation suggests. Intuitively, $|v(G)|$ indicates the number of "free moves" one of the players has before losing (Left if $v(G) > 0$, and Right if $v(G) < 0$). In fact, for any two finite numeric games P and Q, one can show that $v(P+Q) = v(P)+v(Q)$ and that $v(-P) = -v(P)$. Also, $P \leq Q$ if and only if $v(P) \leq v(Q)$.[11] The valuation map v is not one-to-one on games, but induces a one-to-one map on *values* of numeric games.

To illustrate the simplicity rule, consider the game $h := \{0|1\}$. The rule says that $v(h)$ is the simplest dyadic rational number strictly between 0 and 1, namely, $1/2$. First note that Left can always win h whether or not she plays first, so $h > 0$. If v respects $+$, then we should also have $h + h \approx 1$. Let us check this. First consider $1 - h$:

$$\begin{aligned}1 - h &= 1 + (-h) = \{0|\} + \{-1|0\} = \{0 - h, 1 - 1|1 + 0\} \\ &= \{-h, 0|1\} \approx \{0|1\} = h\end{aligned}$$

(the equivalence is by the dominating rule and $-h < 0$). Thus

$$h + h \approx h + (1 - h) \approx 1 .$$

Black-white poset games are numeric [10]. Here we identify Black with Left and White with Right. So for example, an antichain of k black points has numeric value k, and an antichain of k white nodes has numeric value $-k$. Figure 2 shows the numeric value of two simple, two-level black-white poset games.

Exercise 2.19. Use the simplicity rule to prove the values in Figure 2.

The numerical values of arbitrary numeric games (not necessarily finite) form an ordered, real-closed field **No** into which the real numbers embed, but which also contains all the ordinals as well as infinitesimals [5]. Donald Knuth dubbed **No** the *surreal numbers* [17], and they are formed via a transfinite construction. The dyadic rationals are those constructed at finite stages, but numbers constructed through stage ω already form a proper superset of $\mathbb{R}$.

[11]One can define a purely game-theoretic multiplication operation on numeric games in such a way that $v(PQ) = v(P)v(Q)$ for all P and Q. See [5] for details.

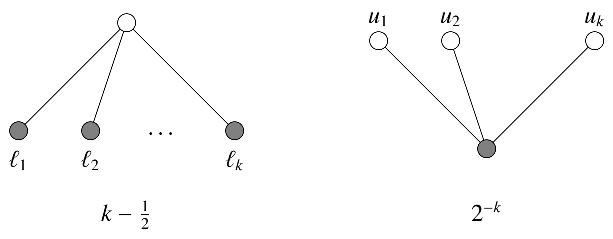


Figure 2: The numerical values of two simple black-white poset games. The left has value $k - \frac{1}{2}$ and the right has value 2^{-k}, for $k \geq 1$.

2.5 Impartial games and Sprague-Grundy theory

A game is *impartial* if at every position, the two players have the same options. Formally,

Definition 2.20. A game G is *impartial* iff $G^L = G^R$ and every $g \in G^L$ is impartial.

Equivalently, G is impartial if and only if $G = -G$. This means that values of impartial games are those that have order two in the group $\langle \mathbf{PG}, + \rangle$.

Examples of impartial games include 0 and $*$. Families of impartial games include Nim, Geography, Node Kayles, and poset games.[12] There is a beautiful theory of impartial games, developed by R. P. Sprague and P. M. Grundy [25, 14] that predates the more general theory of combinatorial games described in [1, 5]. We develop the basics of this older theory here. First note that, since there are no Left/Right biases, all impartial games are either zero ($\mathcal{P}$) or fuzzy ($\mathcal{N}$), and we can assume that Left always moves first. We will call impartial zero games $\forall$*-games* ("for all first moves ...") and impartial fuzzy games $\exists$*-games* ("there exists a first move such that ..."). In this section only, we restrict our attention to impartial games, so when we say "game," we mean impartial game.

Two (impartial) games G and H are equivalent ($G \approx H$) if and only if $G + H$ is a $\forall$-game, because $H = -H$ (Sprague and Grundy defined this notion for impartial games). One can associate an ordinal number with each game, which we call the *g-number*[13] of the game, such that two games are equivalent if and only if they have the same g-number. The g-number of a finite game is a natural number. We will restrict ourselves to finite games.

Definition 2.21. Let A be any coinfinite subset of $\mathbb{N}$. Define mex A (the *$\underline{m}$inimum*

[12]Impartiality is not $\approx$-invariant.

[13]also called the *Grundy number* or the *NIM number*—not to be confused with the value of a numerical game

excluded element from A) to be the least natural number not in A, i.e.,

$$\operatorname{mex} A := \min(\mathbb{N} - A) \ .$$

More generally, for $i = 0, 1, 2, \ldots$, inductively define

$$\operatorname{mex}_i A := \min\left(\mathbb{N} - (A \cup \{\operatorname{mex}_0(A), \ldots, \operatorname{mex}_{i-1} A\})\right) \ ,$$

the i'th least natural number not in A. (So in particular, $\operatorname{mex}_0 A = \operatorname{mex} A$.)

Definition 2.22. Let G be any (finite) game. Define the *g-number* of G as

$$g(G) := \operatorname{mex} \textit{g-set}(G) \ ,$$

where $\textit{g-set}(G) := \{g(x) \mid x \in G^L\}$ is called the *g-set* of G.

That is, $g(G)$ is the least natural number that is not the g-number of any option of G, and the set of g-numbers of options of G is $\textit{g-set}(G)$. For example, $\textit{g-set}(0) = \emptyset$, and so $g(0) = 0$. Also, $\textit{g-set}(*) = \{g(0)\} = \{0\}$, and so $g(*) = 1$.

Exercise 2.23. Prove the following for any finite poset P and any $n \in \mathbb{N}$.

1. $g(P) \leq |P|$. (Generally, $g(G) \leq \left|G^L\right|$ for any impartial G.)
2. $g(C_n) = n$ for all $n \in \mathbb{N}$.
3. $g(A_n) = n \bmod 2$.
4. $g(V_n) = (n \bmod 2) + 1$.

What is $g(\Lambda_n)$? What is $g(\Diamond_n)$?

Exercise 2.24. Describe $g(A_m/A_n)$ simply in terms of m and n.

Here is the connection between the g-number and the outcome of a game.

Proposition 2.25. *A game G is a $\forall$-game if and only if $g(G) = 0$.*

Proof idea. If $g(G) \neq 0$, then there is some option x of G that Left can play such that $g(x) = 0$, but if $g(G) = 0$, then no move Left makes can keep the g-number at 0. □

The central theorem of Sprague-Grundy theory—an amazing theorem with a completely nonintuitive proof—concerns the g-number of the sum of two games.

Definition 2.26. For any $m, n \in \mathbb{N}$, define $m \oplus n$ to be the natural number k whose binary representation is the bitwise exclusive OR of the binary representations of m and n. We may also call k the *bitwise XOR* of m and n.

For example, $23 \oplus 13 = 10111 \oplus 01101 = 11010 = 26$.

Theorem 2.27 (Sprague, Grundy [25, 14])**.** *For any finite games G and H,*

$$g(G + H) = g(G) \oplus g(H) .$$

Corollary 2.28. *Two impartial games G and H are equivalent if and only if $g(G) = g(H)$.*

Proof. G and H are equivalent iff $G + H$ is a $\forall$-game, iff $g(G + H) = 0$ (Proposition 2.25), iff $g(G) \oplus g(H) = 0$ (Theorem 2.27), iff $g(G) = g(H)$. □

Since every natural number n is the g-number of the poset game C_n, this means that every game is equivalent to a single NIM stack.

We can use Theorem 2.27 to solve Nim. Given a Nim game $P = C_{n_1} + \cdots + C_{n_k}$, we get $g(P) = n_1 \oplus \cdots \oplus n_k$. If this number is nonzero, then let i be largest such that $(g(P))_i = 1$. Alice can win by choosing a j such that $(n_j)_i = 1$ and playing in C_{n_j} to reduce its length (and hence its g-number) from n_j to $n_j \oplus (g(P))_i$. This makes the g-number of the whole Nim game zero.

Theorem 2.27 shows how the g-number behaves under parallel unions of posets (Definition 1.2). How does the g-number behave under series unions? Unfortunately, $g(P/Q)$ might not depend solely on $g(P)$ and $g(Q)$. For example, $g(V_2) = g(C_1) = 1$, but $g(C_1/V_2) = g(\Diamond_2) = 3$ whereas $g(C_1/C_1) = g(C_2) = 2$. However, *g-set*(P/Q) *does* depend solely on *g-set*(P) and *g-set*(Q) for any posets P and Q, and this fact forms the basis of the Deuber & Thomassé algorithm of the next section.

There is one important case where $g(P/Q)$ does only depend on $g(P)$ and $g(Q)$:

Fact 2.29. *For any finite poset P and any $k \geq 0$,*

$$g\left(\frac{P}{C_k}\right) = g(P) + k .$$

This can shown by first showing that $g(P/C_1) = g(P) + 1$, then using induction on k. By Fact 2.29, we get that $g(\Diamond_n) = 1 + g(\Lambda_n)$ for example.

3 Upper bounds

When asking about the computational difficulty of determining the outcome of a game, we really mean a family of similar games, represented in some way as finite inputs. In discussing game complexity, we will abuse terminology and refer to a family of games simply as a game. (The same abuse occurs in other areas of complexity, notably circuit complexity.) We will also use the same small-caps notation to refer both to a family of games and to the corresponding decision problem about the outcomes.

Perhaps the most common upper bound in the literature on the complexity of a game is membership in **PSPACE**. Without pursuing it further, we will just mention that, if a game G of size n satisfies: (i) every position of G has size polynomial in n; (ii) the length of any play of G is polynomial in n; and (iii) there are polynomial-time (or even just polynomial-space) algorithms computing the "left option of" and "right option of" relations on the positions of G, then $o(G)$ can be computed in polynomial space. These properties are shared by many, many games.

In this section we will give some better upper bounds on some classes of finite poset games, the best one being that N-free poset games are in **P** [6]. We will assume that a poset is represented by its Hasse diagram, a directed acyclic graph (DAG) in which each element is represented as a node and an arc is placed from a node for element x to the node for y when $x < y$ and there is no element z such that $x < z < y$. The poset is the reflexive, transitive closure of the edge relation of the DAG.

3.1 N-free games

With the Hasse diagram representation, we can apply results from graph theory to devise efficient ways to calculate g-numbers for certain classes of games. A good example is the class of N-free poset games. An "N" in a poset is a set of four elements $\{a, b, c, d\}$ such that $a < b$, $c < d$, $c < b$, and the three other pairs are incomparable. When drawn as a Hasse diagram the arcs indicating comparability form the letter "N". A poset is *N-free* if it contains no N as an induced subposet. We let N-Free denote the class of N-free poset games.

Valdes, Tarjan, and Lawler [30] show that an N-free DAG can be constructed in linear time from a set of single nodes. New components are created either by applying parallel union ($G+H$) or by applying series union (G/H). As with posets, the parallel union is the disjoint union of G and H. The series union is a single DAG formed by giving to every element in H with out-degree 0 (the sinks in H) an arc to every element in G with in-degree 0 (the sources in G). This gives the Hasse diagram of the series union of the corresponding posets. Their algorithm provides a sequence of + and / operations that will construct a given N-free DAG from single points.

Deuber & Thomassé [6] show that N-Free $\in$ **P** by applying this construction to demonstrate how to calculate the g-number of an N-free poset game based on the sequence of construction steps obtained by the VTL algorithm above. Their algorithm, which we now describe, works by keeping track of the g-sets of the posets obtained in the intermediate steps of the construction, rather than the g-numbers. There is no need to store the g-numbers, because the g-number of any poset can always be easily computed from its g-set by taking the mex.

The g-number of a single node is 1. This is the base case.

Fact 3.1. *Given posets P and Q, the g-set of the parallel union $P + Q$ is*

$$\begin{aligned} \textit{g-set}(P+Q) &= \{g(P+Q_q) : q \in Q\} \cup \{g(P_p + Q) : p \in P\} \\ &= \{g(P) \oplus g(Q_q) : q \in Q\} \cup \{g(P_p) \oplus g(Q) : p \in P\} \,. \end{aligned}$$

The second equality follows from the Sprague-Grundy theorem. This is easy to see if you consider the root of the game tree for $P + Q$. Each of its children results from playing either an element in P or one in Q. The left-hand set in the union contains the g-numbers of the games resulting from playing an element in Q; the right-hand set from playing an element in P. Their union is the g-set of $P + Q$, so its g-number is the mex of that set.

To calculate the g-set of a series union, we will need the definition of the *Grundy product* of two finite sets of natural numbers:

$$A \odot B := B \cup \{\operatorname{mex}_a B \mid a \in A\} \,.$$

$A \odot B$ is again a finite set of natural numbers that is easy to compute given A and B. Basically, $A \odot B$ unions B with the version of A we get after re-indexing the natural numbers to go "around" B. Notice that $\operatorname{mex}(A \odot B) = \operatorname{mex}_{\operatorname{mex} A} B$. We will use this fact below.

Lemma 3.2 (Deuber & Thomassé [6])**.** *For any finite posets P and Q, $\textit{g-set}(P/Q) = \textit{g-set}(P) \odot \textit{g-set}(Q) = \textit{g-set}(Q) \cup \{\operatorname{mex}_i(\textit{g-set}(Q)) : i \in \textit{g-set}(P)\}$.*

The left-hand set of the union results from playing an element in Q, which removes all of the elements in P. Using induction, we can see what happens when an element in P is played.

Proof of Lemma 3.2. The fourth equality uses the inductive hypothesis.

$$\begin{aligned} \textit{g-set}(P/Q) &= \{g((P/Q)_r) : r \in P/Q\} \\ &= \{g((P/Q)_p) : p \in P\} \cup \{g((P/Q)_q) : q \in Q\} \\ &= \{g((P_p/Q)) : p \in P\} \cup \{g(Q_q) : q \in Q\} \\ &= \{\operatorname{mex}(\textit{g-set}(P_p) \odot \textit{g-set}(Q)) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_{\operatorname{mex} \textit{g-set}(P_p)} \textit{g-set}(Q) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_{g(P_p)}(\textit{g-set}(Q)) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_i(\textit{g-set}(Q)) : i \in \textit{g-set}(P)\} \cup \textit{g-set}(Q) \\ &= \textit{g-set}(P) \odot \textit{g-set}(Q) \end{aligned}$$

□

In particular, the g-number of P/Q is greater than or equal to the sum of the g-numbers of P and Q. Notably, it's an equality if Q is C_n for some n (Fact 2.29) and the reason is that the g-set of C_n has no gaps, that is, it contains all of the values from 0 to $n - 1$. It's easy to see that it's true when P and Q are both singletons. Their g-numbers are both 1 and forming their series-union creates a NIM stack of size 2 and that has g-number 2.

Another way to understand Lemma 3.2 is to consider the game tree of P/Q, and we'll look at the simple case where P is an arbitrary game with g-number k and Q is a singleton. Consider the root node r of the game tree of P/Q. One of its children represents playing the single element in Q and that child has g-number 0. The rest of r's children represent game configurations reached by playing an element in P. By the induction hypothesis the g-number of each of these nodes will be one more than in P's game tree where they had g-numbers 0 to $k - 1$, and perhaps g-numbers $k + 1$ and larger. So in P/Q's tree they have g-numbers 1 to k, with perhaps g-numbers $k + 2$ or larger. Because the child reached by playing Q's single element has g-number 0, the first missing value in the g-set formed from these g-numbers is $k + 1$.

Now using Fact 3.1 and Lemma 3.2, the decomposition described in [30] can generate a binary tree where each internal node is labeled with a poset P and an operation (parallel union or series union), and its children are the two posets combined to form P. Starting with each leaf, where the poset is a singleton and the g-set is $\{0\}$, and moving up the tree, one can apply Fact 3.1 and Lemma 3.2 to compute the g-set of the root (and none of the g-numbers involved exceed the size of the final poset). This can all be done in time $O(n^4)$.

3.2 Results on some classes of games with N's

General results for classes of games containing an "N" have been few. In 2003, Steven Byrnes [3] proved a poset game periodicity theorem, which applies to, among others, Chomp-like games, which contain many "N"-configurations.

Here's the theorem, essentially as stated in the paper:

Theorem 3.3. *In an infinite poset game X, suppose we have two infinite chains C ($c_1 < c_2 < \cdots$) and D ($d_1 < d_2 < \cdots$), and a finite subset A, all pairwise disjoint, and assume that no element of C is less than an element of D. Let $A_{m,n} = A \cup C \cup D - \{x \in X | x \geq c_{m+1}\} - \{x \in X | x \geq d_{n+1}\}$ (that is, $A_{m,n}$ is the position that results from starting with the poset $A \cup C \cup D$, then making the two moves c_{m+1} and d_{n+1}). Let k be a nonnegative integer. Then either:*

1. *there are only finitely many different $A_{m,n}$ with g-number k; or*

2. *we can find a positive integer p such that, for large enough n,* $g(A_{m,n}) = k$ *if and only if* $g(A_{m+p,n+p}) = k$.

Thus, as the poset A expands along the chains C and D, positions with any fixed g-number have a regular structure.

A simple example of a class of games covered by the theorem is the family of two-stack Nim games, where A is empty and $A_{m,n}$ consists of an m-chain and an n-chain. The g-number 0 occurs for every $A_{n,n}$ so the periodicity is 1. The g-number 1 occurs for every $A_{2n,2n+1}$ and so has periodicity 2. In fact, one can find a periodic repetition for every g-number. The surprising thing is that this is still true when you allow elements in one chain to be less than elements in the other.

Another family contains Chomp, described in Section 1.1.1. We can generalize Chomp to games where the rows do not have to contain the same number of elements. Byrnes showed that for such games there is a periodicity in the g-numbers when we fix the size of all but the top two rows.

As Byrnes claims, this yields a polynomial-time decision algorithm for each family generated from a fixed A but not a uniformly polynomial-time algorithm across the families, as the time is parameterized by A.

3.2.1 Bounded-width poset games

If a poset P has width k, that is, if k is the maximum size of any antichain in P, then there are only $|P|^k$ many positions at most in the game: if $x_0, x_1, \ldots, x_{n-1} \in P$ are the elements chosen by the players in the first n moves of the game, then the resulting position is completely determined by the minimal elements of the set $\{x_0, \ldots, x_{n-1}\}$, i.e., an antichain of size $\leq k$.

This means that, for constant k, one can compute the g-number of P in polynomial time using dynamic programming. The exponent on the running time depends on k, however. For certain families of bounded-width posets, one can beat the time of the dynamic programming algorithm; for example, one can compute the g-number of width-2 games in linear time.

4 Lower bounds

In this section we give some lower bounds on game complexity. There is a vast literature on combinatorial game complexity, and we make no attempt to be thorough, but rather concentrate on poset game complexity.

4.1 A note about representations of games

The complexity of a game depends quite a bit on its representation. The choice of representation is usually straightforward, but not always. For example, how should we represent an N-free poset? Just by its Hasse diagram, or by an expression for the poset in terms of single points and parallel union and series union operators? The results of Valdes, et al. [30] show that one representation can be converted into the other in polynomial time, so the choice of representation is not an issue unless we want to consider complexity classes within **P** or more succinct representations of posets, as we will do below. There, fortunately, our hardness results apply to either representation.

Even if the representation of a game is clear, the results may be counterintuitive. For example, how should we represent members of the class of *all* finite games? In Section 2, we defined a game as an ordered pair of its left and right options. We must then represent the options, and the options of options, and so on. In effect, to represent an arbitrary finite game explicitly, we must give its entire game tree (actually, game DAG, since different sequences of moves may end up in the same position). Under this representation, there is a straightforward algorithm to compute the outcome of any game: use dynamic programming to find the outcome of every position in the game. Since every position is encoded in the string representing the game, this algorithm runs in polynomial time.

What makes a game hard, then, is that we have a succinct representation for it that does not apply to all games. For example, the obvious representation of a poset game is the poset itself, and the number of positions is typically exponential in the size of the poset. Subfamilies of poset games may have even more succinct representations. For example, a Nim game can be represented as a finite list of natural numbers in binary, giving the sizes of the stacks, and a game of Chomp can be represented with just two natural numbers m and n in binary, giving the dimensions of the grid. Notice that this Chomp representation is significantly shorter than what is needed to represent an arbitrary *position* in a Chomp game; the latter is polynomial in $m + n$.

In what sense does finding a winning strategy in Chomp reduce to determining the outcome of Chomp games? We already know that every Chomp game is an $\exists$-game because it has a maximal point. We could find a winning strategy if we were able to determine the outcome of every Chomp position, but even writing down a query to an "outcome oracle" takes time linear in $m + n$, which is exponential in the input size. The more modest goal of finding a winning first move may be more feasible, because the position after one move is simple enough to describe by a polynomial-length query string. To our knowledge, no efficient algorithm is known to determine the outcome of an arbitrary Chomp position after a single move, even allowing time $(m + n)^{O(1)}$.

We will more to say about representations below when we discuss lower bounds for poset games within the complexity class **P**.

4.2 Some PSPACE-hard games

Many games have been shown **PSPACE**-hard over the years. Early on, Even and Tarjan showed that Hex generalized to arbitrary graphs is **PSPACE**-complete [7]. A typical proof of **PSPACE**-hardness reduces the **PSPACE**-complete True Quantified Boolean Formulas (TQBF [26]) problem to the outcome of a game. We can consider a quantified Boolean formula $\varphi = (\exists x_1)(\forall x_2)\cdots\psi$ (where ψ is a Boolean formula in conjunctive normal form (cnf)) itself as a game, where players alternate choosing truth values for $x_1, x_2, \ldots$, the first player (Right, say) winning if the resulting instantiation of ψ is true, and Left winning otherwise.[14]

TQBF seems ideal for encoding into other games. Thomas Schaefer showed a number of interesting games to be **PSPACE**-hard this way [21]. One interesting variant of TQBF that Schaefer proved **PSPACE**-complete is the game where a positive Boolean formula ψ is in cnf with no negations, and players alternate choosing truth values for the Boolean variables. Schaefer called this game G_{pos}(POS CNF). Unlike TQBF, however, the variables need not be chosen in order; players may choose to assign a truth value to any unassigned variable on any move. Left (who moves first) wins if ψ is true after all variables have been chosen, and Right wins otherwise. Since ψ is positive, Left always wants to set variables to 1 and Right to 0.

As another example, consider Geography. The input is a directed graph G and a designated vertex s of G on which a token initially rests. The two players alternate moving the token on G from one node to a neighboring node, trying to force the opponent to move to a node that has already been visited. Geography is a well-known **PSPACE**-complete game [21, 24]. In [19], Lichtenstein & Sipser show that Geography is **PSPACE**-complete even for bipartite graphs.

An obvious way to turn Geography into a black-white game is to color the nodes of graph G black and white. Each player is then only allowed to move the token to a node of their own color. Since moves are allowed only to neighboring nodes, the black-white version is equivalent to the uncolored version on bipartite graphs. The standard method of showing that Geography is **PSPACE**-complete is via a reduction from True Quantified Boolean Formulas (TQBF) to Geography (see for example [24]). Observe that the graph constructed in this reduction is not bipartite. That is, there are nodes that potentially may be played by both players. Hence, we cannot directly conclude that the black-white version is **PSPACE**-

[14]This is technically not a combinatorial game by our definition, because the end condition is different. One can modify the game slightly to make it fit our definition, however.

complete. However, in [19] Lichtenstein & Sipser show that GEOGRAPHY is indeed **PSPACE**-complete for bipartite graphs.

We now consider the game NODE KAYLES. This game is defined on an undirected graph G. The players alternately play an arbitrary node from G. In one move, playing node v removes v and all the direct neighbors of v from G. In the black-white version of the game, we color the nodes black and white. Schaefer [21] showed that determining the winner of an arbitrary NODE KAYLES instance is **PSPACE**-complete. He also extended the reduction to bipartite graphs, which automatically yields a reduction to the black-white version of the game (see [12]). Therefore, black-white NODE KAYLES is also **PSPACE**-complete.

The game of COL [1] is a two-player combinatorial strategy game played on a simple planar graph, some of whose vertices may be colored black or white. During the game, the players alternate coloring the uncolored vertices of the graph. One player colors vertices white and the other player colors vertices black. A player is not allowed to color a vertex neighboring another vertex of the same color. The first player unable to color a vertex loses. A well-known theorem about COL is that the value of any game is either x or $x + *$ where x is a number. Removing the restriction that COL games be played on planar graphs and considering only those games in which no vertex is already colored, we get a new game, GENCOL (generalized COL). It is shown in [10] that GENCOL is **PSPACE**-complete; furthermore, GENCOL games only assume the two very simply game values 0 and $*$.

In [20], Stockmeyer & Chandra give examples of games that are complete for exponential time and thus provably infeasible.

4.3 Lower bounds for poset games

Until recently, virtually no hardness results were known relating to poset games, and the question of the complexity of determining the outcome of a game was wide open, save the easy observation that it is in **PSPACE**.

For the moment, let PG informally denote the decision problem of determining the outcome of a arbitrary given (impartial) poset game, that is, whether or not the first player (Alice) can win the game with perfect play. The first lower bound on the complexity of PG we are aware of, and it is a modest one, was proved by Fabian Wagner [31] in 2009. He showed that PG is **L**-hard[15] under **FO**-reductions (First-Order reductions). This is enough to show, for example, that PG $\notin$ $\mathbf{AC}^0$. Soon after, Thomas Thierauf [28] showed that PG is hard for **NL** under $\mathbf{AC}^0$ reductions.[16] A breakthrough came in 2010, when Adam Kalinich, then a high

[15]**L** is short for LOGSPACE.

[16]**NL** is nondeterministic LOGSPACE.

school student near Chicago, Illinois, showed that PG is hard for $\mathbf{NC}^1$ under $\mathbf{AC}^0$ reductions [16]. For the proof, he invents a clever way to obliviously "flip" the outcome of a game, i.e., to toggle the outcome between $\exists$ and $\forall$. This allows for the simulation of a NOT-gate in an $\mathbf{NC}^1$ circuit. (An OR-gate can be simulated by the series union construction of Definition 1.2. See below.)

The astute reader will notice that Kalinich's result appears to be weaker than the other two earlier results. In fact, the three results are actually incomparable with each other, because they make different assumptions about how poset games are represented as inputs. We say more about this below, but first we mention that Wagner's and Thierauf's results both hold even when restricted to Nim games with two stacks, and Kalinich's result holds restricted to N-free games. Modest as they are, these are currently the best lower bound we know of for N-free poset games.

Very recently, the complexity of PG was settled completely by Daniel Grier, an undergraduate at the University of South Carolina [13]. He showed that PG is **PSPACE**-complete via a polynomial reduction (henceforth, p-reduction) from Node Kayles. Here, it is not important how a game is represented as an input, so long as the encoding is reasonable. His proof shows that **PSPACE**-completeness is still true when restricted to three-level games, i.e., posets where every chain has size at most three (equivalently, posets that are partitionable into at most three antichains). The games used in the reduction are of course not N-free.

4.4 Representing posets as input

As we discussed above, for any of the various well-studied families of poset games (Chomp, Divisors, Nim, etc.), there is usually an obvious and natural way to represent a game as input. For example, an instance of Chomp can be given with just two positive integers, one positive integer for Divisors, and a finite list of positive integers for Nim, giving the heights of the stacks. When considering arbitrary finite posets, however, there is no single natural way to represent a poset as input, but rather a handful of possibilities, and these may affect the complexity of various types of poset games. We consider two broad genres of poset representation:

Explicit The poset is represented by an explicit data structure, including the set of points and the relations between them. In this representation, the size of the poset is always comparable to the size of the input.

Succinct (Implicit) The poset is represented by a Boolean circuit with two n-bit inputs. The inputs to the circuit uniquely represent the points of the poset, and the (1-bit) output gives the binary relation between these two inputs. In this representation, the size of the poset can be exponential in the size of the circuit.

Within each representational genre, we will consider three general approaches to encoding a poset P, in order from "easiest to work with" to "hardest to work with":

Partial Order (PO) P is given as a reflexive, transitive, directed acyclic graph, where there is an edge from x to y iff $x \leq y$.

Hasse Diagram (HD) P is given as a directed acyclic graph whose reflexive, transitive closure (i.e., reachability relation) is the ordering $\leq$. The graph then gives the Hasse diagram of P.

Arbitrary (binary) Relation (AR) An arbitrary directed graph (or arbitrary binary relation) is given, whose reflexive, transitive closure is then a pre-order whose induced partial order is P. (Equivalently, P is the set of strongly connected components, and $\leq$ is the reachability relation between these components.)

The first two (PO and HD) must involve promises that the input satisfies the corresponding constraint, so problems in these categories are posed as promise problems. Notice that the PO promise is stronger than the HD promise, which is stronger than the AR (vacuous) promise. So in either the Explicit or Succinct cases, the complexity of the corresponding problems increases monotonically as PO $\to$ HD $\to$ AR.

We will ignore some additional subtleties: In the explicit case, is the graph (or relation) given by an adjacency matrix or an array of edge lists? In the succinct case, should we be able to represent a poset whose size is not a power of 2? For example, should we insist on including a second circuit that tells us whether a given binary string represents a point in the poset? These questions can generally be finessed, and they do not affect any of the results.

4.5 The decision problems

The two genres and three approaches above can be combined to give six versions of the basic decision problem for arbitrary posets: the three explicit problems PG(Explicit, PO), PG(Explicit, HD), and PG(Explicit, AR); and the three succinct problems PG(Succinct, PO), PG(Succinct, HD), and PG(Succinct, AR). We will define just a couple of these, the others being defined analogously.

Definition 4.1. PG(Succinct, HD) is the following promise problem:

> **Input:** A Boolean circuit C with one output and two inputs of n bits each, for some n.
>
> **Promise:** G is acyclic, where G is the digraph on $\{0, 1\}^n$ whose edge relation is computed by C.

Question: Letting P be the poset given by the reachability relation on G, is P an $\exists$-game?

Definition 4.2. PG(Explicit, AR) is the following promise problem:

Input: A digraph G on n nodes.
Promise: None.
Question: Letting P be the poset given by the reachability relation on the strongly connected components of G, is P an $\exists$-game?

We also can denote subcategories of poset games the same way. For example, Nim(Explicit, HD) is the same as PG(Explicit, HD), but with the additional promise that the poset is a parallel union of chains; for any $k > 0$, Nim_k(Explicit, HD) is the same as Nim(Explicit, HD) but with the additional promise that there are at most k chains; N-Free(Succinct, PO) is the same as PG(Succinct, PO) with the additional promise that the poset is N-free.

4.6 The first results

Here are the first lower bounds known for poset games, given roughly in chronological order. The first four involve Nim; the first two of these consider explicit games, and the next two consider succinct games. None of these results is currently published; proof sketches can be found in the extended paper.

Theorem 4.3 (Wagner, 2009)**.** Nim_4(Explicit, HD) *is* **L**-*hard under* $\mathbf{AC}^0$ *reductions.*

The proof reduces from the promise problem ORD (order between vertices), which is known to be complete for **L** via quantifier-free projections [8, 15].

Theorem 4.4 (Thierauf, 2009)**.** Nim_2(Explicit, AR) *is* **NL**-*hard under* AC^0 *reductions.*

The proof reduces from the reachability problem for directed graphs, which is **NL**-complete under $\mathbf{AC}^0$-reductions.

The next result about succinct poset games is straightforward.

Theorem 4.5 (F, 2009)**.** Nim_2(Succinct, PO) *is* $\mathbf{co}C_=\mathbf{P}$-*hard under p-reductions.*

The idea here is that, for any $L \in \mathbf{coC_=P}$ and any input x, we produce two NIM stacks, and $x \in L$ if and only if they are of unequal length.

Theorem 4.6 (F, 2009)**.** Nim_6(Succinct, HD) *is* **PSPACE**-*hard under p-reductions.*

The proof uses a result of Cai & Furst [4] based on techniques of David Barrington on bounded-width branching programs. Recall that S_5 is the group of permutations of the set $\{1, 2, 3, 4, 5\}$. Their result is essentially as follows:

Theorem 4.7 (Cai & Furst)**.** *For any* **PSPACE** *language L, there exists a polynomial p and a polynomial-time computable (actually, log-space computable) function σ such that, for all strings x of length n and positive integers c (given in binary), $\sigma(x, c)$ is an element of S_5, and $x \in L$ if and only if the composition $\sigma(x, 1)\sigma(x, 2)\sigma(x, 2) \cdots \sigma(x, 2^{p(n)})$, applied left to right, fixes the element* 1.

The idea is that we connect the first five NIM stacks level-by-level via permutations in S_5, as well as adding a couple of widgets. If the product of all the permutions fixes 1, then we get five NIM stacks of equal length $N + 1$ and one NIM stack of length $N + 3$, which is an $\exists$-game by the Sprague-Grundy theorem. If 1 is not fixed, then we get four stacks of length $N+1$ and two of length $N+2$—a $\forall$-game by the same theorem.

Although the above results all mention Nim, the representations we use of a Nim game as a poset are not the natural one. Therefore, it is better to consider these as lower bounds on N-free poset games, which *are* naturally represented as posets.

The next results regard N-free games. They depend on Adam Kalinich's game outcome-flipping trick. The trick turns a poset game A into another poset game $\neg A$ with opposite outcome, starting with A and applying series and parallel union operations in a straightforward way. Here we describe a simplification of the trick due to Daniel Grier:

Given a poset A,

1. Let k be any (convenient) natural number such that $2^k \geq |A|$ (that is, A has at most 2^k elements).

2. Let $B := A/C_{2^k-1}$.

3. Let $C := B + C_{2^k}$.

4. Let $D := C/C_1$.

5. Finally, define $\neg A := D + A$.

Exercise 4.8. Check that: (1) if $g(A) \neq 0$, then $g(\neg A) = 0$; (2) if $g(A) = 0$, then $g(\neg A) = 2^{k+1}$. See the extended paper for a proof.

Observe that the size of $\neg A$ is linearly bounded in $|A|$. In fact, $|\neg A| \leq 6|A|$ if $A \neq \emptyset$.

Theorem 4.9 (Kalinich [16])**.** N-FREE(Explicit, PO) *is* $\mathbf{NC}^1$*-hard under* $\mathbf{AC}^0$ *reductions.*

Proof sketch. We reduce from the Circuit Value problem for $\mathbf{NC}^1$ circuits with a single output. Given an $\mathbf{NC}$ circuit C with a single output and whose inputs are constant Boolean values, we produce a poset game P so that P is an $\exists$-game if and only if $C = 1$. We can assume WLOG that all gates in C are either (binary) OR-gates or NOT-gates. Starting with the input nodes, we associate a poset P_n with every node n in C from bottom up so that the outcome of P_n matches the Boolean value at node n. P is then the poset associated with the output node of C. The association is as follows:

- If n is an input node, we set $P_n := \emptyset$ if $n = 0$; otherwise, if $n = 1$, we set $P_n := C_1$.
- If n is an OR-gate taking nodes ℓ and r as inputs, then we set $P_n := P_\ell / P_r$. (Recall Exercise 1.4.)
- If n is a NOT-gate taking node c as input, we set $P_n := \neg P_c$.

This transformation from C to P can be done in (uniform) $\mathbf{AC}^0$, producing a poset of polynomial size, provided C has $O(\log n)$ depth. □

The next theorem is not published elsewhere.

Theorem 4.10 (F, 2011)**.** N-FREE(Succinct, PO) *is* $\mathbf{PP}$*-hard under p-reductions.*

To prove this, we generalize the Kalinich/Grier construction a bit.

Definition 4.11. For any poset A and any integer $t > 0$, define

$$\text{Threshold}(A, t) := \frac{(A/C_{2^k - t}) + C_{2^k}}{C_t} + A \ ,$$

where k is any convenient natural number (the least, say) such that $2^k > \max(|A| - t, t - 1)$.

Note that $\neg A = \text{Threshold}(A, 1)$. It can be checked that

$$g(\text{Threshold}(A, t)) = \begin{cases} 2^{k+1} & \text{if } g(A) < t, \\ 0 & \text{if } g(A) \geq t. \end{cases} \tag{1}$$

We then use the Threshold$(\cdot, \cdot)$ operator to polynomially reduce any $\mathbf{PP}$ language to N-FREE(Succinct, PO).

4.7 A note on the complexity of the g-number

Of course, computing the g-number of an impartial game is at least as hard as computing its outcome, the latter just being a test of whether the g-number is zero. Is the reverse true, i.e., can we polynomial-time reduce computing the g-number to computing the outcome? For explicitly represented poset games, this is certainly true. Given an oracle S returning the outcome of any poset game, we get the g-number of a given poset game G as follows: query S with the games $G, G + C_1, G + C_2, \ldots, G + C_n$, where n is the number of options of G (recall that that C_i is a NIM stack of size i). By the Sprague-Grundy theorem (Theorem 2.27), all of these are $\exists$-games except $G + C_{g(G)}$, which is a $\forall$-game.

What about succinctly represented games? The approach above can't work, at least for poset games, because the poset has exponential size. Surprisingly, we can still reduce the g-number to the outcome for succinct poset games in polynomial time, using the threshold construction of Definition 4.11 combined with binary search. Given a succinctly represented poset P of size $\leq 2^n$, first query S with $\text{Threshold}(P, 2^{n-1})$. If S says that this is an $\exists$-game, then we have $g(P) < 2^{n-1}$; otherwise, $g(P) \geq 2^{n-1}$. Next, query S with $\text{Threshold}(P, 2^{n-2})$ in the former case and $\text{Threshold}(P, 3 \cdot 2^{n-2})$ in the latter case, and so on. Note that in this reduction, the queries are adaptive, whereas they are nonadaptive for explicitly represented games.

4.8 PSPACE-completeness

In this section we sketch the proofs of two recent **PSPACE**-completeness results for poset game. The first, by Daniel Grier, is that the outcome problem for general explicit (impartial) poset games is **PSPACE**-complete [13]. The second is a similar result about the complexity of black-white poset games [10].

Theorem 4.12 (Grier [13])**.** *Deciding the outcome of an arbitrary finite poset game is* **PSPACE**-*complete.*

Here we describe the reduction but do not prove correctness. See the extended paper or [13] for a full proof.

Proof sketch. Membership in **PSPACE** is clear. For **PSPACE**-hardness, we reduce from Node Kayles. Let $G = (V, E)$ (a simple undirected graph) be an arbitrary instance of Node Kayles. By altering the graph slightly if necessary without changing the outcome of the game, we can assume that $|E|$ is odd and that for every $v \in V$ there exists $e \in E$ not incident with v. We can do this by adding two disjoint cliques to G—either two K_2's or a K_2 and a K_4, whichever of these options results in an odd number of edges. We then construct the following three-level poset P from G:

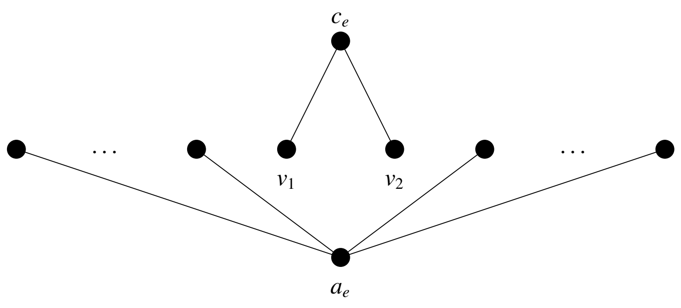


Figure 3: The $<$ relations in P obtained from the edge $e = \{v_1, v_2\}$ in G.

- The points of P are grouped into three disjoint antichains, A, B, and C, with A being the set of minimal points, C the maximal points, and B the points intermediate between A and C.
- For each edge $e \in E$ there correspond unique points $c_e \in C$ and $a_e \in A$, and vice versa.
- We let $B := V$.
- For each edge $e = \{v_1, v_2\}$ and $b \in B$, we have $b < c_e$ iff $b = v_1$ or $b = v_2$, and $a_e < b$ iff this is not the case, i.e., iff $b \neq v_1$ and $b \neq v_2$. This is illustrated in Figure 3.

This construction can clearly be done in polynomial time, given G. □

Finally, we turn to the complexity of black-white poset games. The next theorem is the first **PSPACE**-hardness result for a numeric game.

Theorem 4.13. *Determining the outcome of a black-white poset game is* **PSPACE**-*complete.*

Proof sketch. Membership in **PSPACE** is straightforward. For hardness, we reduce from TQBF. We present the reduction in detail and briefly describe optimal strategies for the winning players, but we do not show correctness. See the extended version for a more detailed sketch and [10] for a full proof.

Suppose we are given a fully-quantified boolean formula φ of the form $\exists x_1 \forall x_2 \exists x_3 \cdots \exists x_{2n-1} \forall x_{2n} \exists x_{2n+1} f(x_1, x_2, \ldots, x_{2n+1})$, where $f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is in cnf with clauses $c_1, \ldots, c_m$. We define a two-level black-white poset (game) X based on φ as follows:

- X is divided into sections. There is a section (called a *stack*) for each variable, a section for the clauses (the *clause section*), and a section for fine-tuning the balance of the game (*balance section*).

- The ith stack consists of a set of incomparable *waiting nodes* W_i above (i.e., greater than) a set of incomparable *choice nodes* C_i. We also have a pair of *anti-cheat nodes*, α_i and β_i, on all stacks except the last stack. For odd i, the choice nodes are white, the waiting nodes are black, and the anti-cheat nodes are black. The colors are reversed for even i.

- The set of choice nodes C_i, consists of eight nodes corresponding to all configurations of three bits (i.e., $000, 001, \ldots, 111$), which we call the *left bit*, *assignment bit* and *right bit* respectively.

- The number of waiting nodes is $|W_i| = (2n + 2 - i)M$, where M is the number of non-waiting nodes in the entire game. It is important that $|W_i| \geq |W_{i+1}| + M$.

- The anti-cheat node α_i is above nodes in C_i with right bit 0 and nodes in C_{i+1} with left bit 0. Similarly, β_i is above nodes in C_i with right bit 1 and nodes in C_{i+1} with left bit 1.

- The *clause section* contains a black *clause node* b_j for each clause c_j, in addition to a black *dummy node*. The clause nodes and dummy node are all above a single white *interrupt node*. The clause node b_j is above a choice node z in C_i if the assignment bit of z is 1 and x_i appears positively in c_j, or if the assignment bit of z is 0 and x_i appears negatively in c_j.

- The balance section or *balance game* is incomparable with the rest of the nodes. The game consists of eight black nodes below a white node, which is designed to have numerical value $-7\frac{1}{2}$. All nodes in this section are called *balance nodes*.

The number of nodes is polynomial in m and n, so the poset can be efficiently constructed from φ.

A sample construction is shown in Figure 4. The idea is that players take turns playing choice nodes, starting with White, and the assignment bits of the nodes they play constitute an assignment of the variables, $x_1, \ldots, x_{2n+1}$. The assignment destroys satisfied clause nodes, and it turns out that Black can win if there remains at least one clause node. The waiting nodes and anti-cheat nodes exist to ensure players take nodes in the correct order. The interrupt node and dummy node control how much of an advantage a clause node is worth (after the initial assignment), and the balance node ensures the clause node advantage can decide whether White or Black wins the game. One can show that White (i.e., Right) can force a win when playing first if and only if the formula is true. □

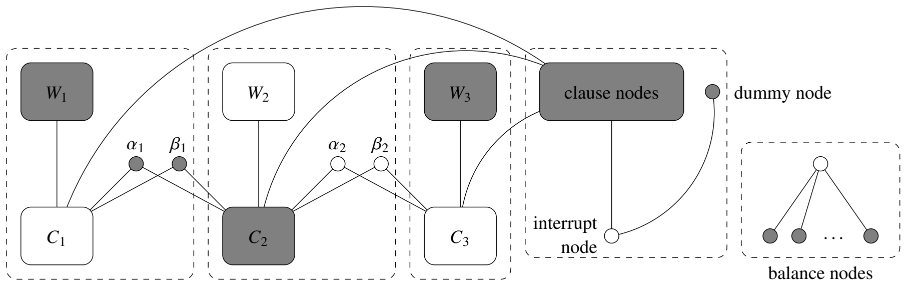


Figure 4: An example game with three variables ($n = 1$). Circles represent individual nodes, blobs represent sets of nodes, and χ is the set of clause nodes. An edge indicates that some node in the lower level is less than some node in the upper level. The dotted lines divide the nodes into sections (stacks, clause section and balance section).

5 Open questions

Are there interesting games whose complexity is complete for a subclass of **PSPACE**? The natural black-white version of GenCol is complete for the class $\mathbf{P}^{\mathbf{NP}[\log]}$, but the game itself and the reasons for its complexity are not so interesting. In this version, each uncolored node is reserved ("tinted") for being colored one or the other color, e.g., some node u can only be colored black, while some other node v can only be colored white. Then the outcome of this game depends only on which subgraph (the black-tinted nodes or the white-tinted nodes) contains a bigger independent set. Given two graphs G_1 and G_2, the problem of determining whether G_1 has a bigger independent set than G_2 is known to be complete for $\mathbf{P}^{\mathbf{NP}[\log]}$ [27].

Fix a natural number $k > 2$. For poset games of bounded width k, defined in Section 3.2.1, is there an algorithm running in time $o(n^k)$?

Grier's proof that the poset game decision problem is **PSPACE**-complete (Theorem 4.12) constructs posets having three levels, that is, whose maximum chain length is three. What about two-level games? Those having a single maximum or a single minimum element are easily solved. What is the complexity of those with more than one minimum and more than one maximum? Certain subfamilies of two-level posets have g-numbers that show regular patterns and are easily computed, or example, games where each element is above or below at most two elements, as well as "parity uniform" games [9]. Despite this, we conjecture that the class of all two-level poset games is **PSPACE**-complete, but are nowhere near a proof. Are there larger subfamilies of the two-level poset games that are in **P**?

A more open-ended goal is to apply the many results and techniques of combinatorial game theory, as we did in Theorem 4.13, to more families of games.

Finally, we mention a long-standing open problem about a specific infinite poset game: What is the outcome of the game $\mathbb{N}^3 - \{(0, 0, 0)\}$, where $(x_1, x_2, x_3) \leq (y_1, y_2, y_3)$ iff $x_i \leq y_i$ for all $i \in \{1, 2, 3\}$?

References

[1] E. R. Berlekamp, J. H. Conway, and R. Guy. *Winning Ways for your Mathematical Plays*. Academic Press, 1982.

[2] C. L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3:35–39, 1901-1902.

[3] S. Byrnes. Poset game periodicity. *INTEGERS: The Electronic Journal of Combinatorial Number Theory*, 3, 2003.

[4] Jin-Yi Cai and Merrick Furst. PSPACE survives constant-width bottlenecks. *Int. J. Found. Comput. Sci.*, 02(01):67, March 1991.

[5] J. H. Conway. *On Numbers and Games*. Academic Press, 1976.

[6] W. Deuber and S. Thomassé. Grundy sets of partial orders. www.mathematik.uni-bielefeld.de/sfb343/preprints/pr96123.ps.gz.

[7] S. Even and R. E. Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the ACM*, 23:710–719, 1976.

[8] Kousha Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54(3):400–411, 1997.

[9] S. A. Fenner, R. Gurjar, A. Korwar, and T. Thierauf. On two-level poset games. Technical Report TR13-019, Electronic Colloquium on Computational Complexity, 2013.

[10] S. A. Fenner, D. Grier, J. Meßner, L. Schaeffer, and T. Thierauf. Game values and computational complexity: An analysis via black-white combinatorial games. Technical Report TR15-021, Electronic Colloquium on Computational Complexity, February 2015.

[11] D. Gale. A curious nim-type game. *Amer. Math. Monthly*, 81:876–879, 1974.

[12] M. Garey and D. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[13] Daniel Grier. Deciding the winner of an arbitrary finite poset game is PSPACE-complete. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming*, volume 7965-7966 of *Lecture Notes in Computer Science*, pages 497–503. Springer-Verlag, 2013.

[14] P. M. Grundy. Mathematics and games. *Eureka*, 2:6–8, 1939.

[15] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66(3):549–566, 2003.

[16] A. O. Kalinich. Flipping the winner of a poset game. *Information Processing Letters*, 112(3):86–89, January 2012.

[17] Donald E. Knuth. *Surreal Numbers*. Addison-Wesley, 1974.

[18] J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory*, 13(3):297–305, 1972.

[19] David Lichtenstein and Michael Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, 1980.

[20] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8(2):151–174, 1979.

[21] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

[22] F. Schuh. Spel van delers (game of divisors). *Nieuw Tijdschrift voor Wiskunde*, 39:299, 2003.

[23] A. N. Siegel. *Combinatorial Game Theory*, volume 146 of *Graduate Studies in Mathematics*. American Mathematical Society, 2013.

[24] M. Sipser. *Introduction to the Theory of Computation (2nd Ed.)*. Course Technology, Inc., 2005.

[25] R. P. Sprague. Über mathematische Kampfspiele. *Tohoku Mathematical Journal*, 41:438–444, 1935-1936.

[26] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[27] H. Spakowski and J. Vogel. θ_p^2-completeness: A classical approach for new results. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FST TCS)*, number 1974 in Lecture Notes in Computer Science, pages 348–360, 2000.

[28] T. Thierauf, 2009. Private communication.

[29] J. Úlehla. A complete analysis of Von Neumann's Hackendot. *International Journal of Game Theory*, 9:107–113, 1980.

[30] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11:298–313, 1982.

[31] F. Wagner, 2009. Private communication.