On the Complexity of Computing Winning Strategies for Finite Poset Games

Michael Soltys and Craig Wilson

Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, Ontario, L8S 4K1, Canada
soltys@mcmaster.ca

Abstract. This paper is concerned with the complexity of computing winning strategies for poset games. While it is reasonably clear that such strategies can be computed in PSPACE, we give a simple proof of this fact by a reduction to the game of geography. We also show how to formalize the reasoning about poset games in Skelley's theory $\mathbf{W}_1^1$ for PSPACE reasoning. We conclude that $\mathbf{W}_1^1$ can use the "strategy stealing argument" to prove that in poset games with a supremum the first player always has a winning strategy.

Keywords: Finite games, posets, PSPACE, proof complexity.

1 Introduction

We study the complexity of computing winning strategies for poset games; these are two player games on finite partially ordered sets that were already studied in 1902 by [2]. The two players take turns selecting an element x from the poset and deleting all the elements y such that $x \leq y$. The player who has to play an empty set loses. Our running example of a poset game is Chomp, first proposed by Gale ([1]) in 1974. This is a curious game because it is very easy to give a proof of existence of a winning strategy for the first player, but no efficient method for computing the actual strategy exists.

Given a poset game, in order to give a complexity-theoretic framework, we consider the *language* (i.e., set) of those "board configurations" from which the first player has a winning strategy. Although it is reasonably clear that such a language is in PSPACE—as the answer can be determined by evaluating a quantified boolean formula—we give a simple and direct proof of this fact by reducing general poset games to the game of geography (known to be PSPACE complete from [7]).

We also show how to reason about poset games in Skelley's third-order logical theory $\mathbf{W}_1^1$ (introduced in [8]) for PSPACE reasoning. Since, as we show, $\mathbf{W}_1^1$ can formalize the "strategy-stealing argument," $\mathbf{W}_1^1$ proves that the first player has a winning strategy in the poset game Chomp.

More than anything, this is an invitation to consider the problem whether poset games are PSPACE complete, and show under what conditions computing a winning strategy can be done efficiently.

The paper is organized as follows: in section 2 we give some background on posets, poset games—and, in particular, the poset game Chomp—explain the "strategy-stealing argument," and remind the reader about PSPACE and completeness. In section 3 we give a reduction from poset games to geography and thereby show that computing a winning strategy for poset games can be done in PSPACE. In section 4 we show how to formalize the "strategy-stealing argument" in $\mathbf{W}_1^1$ and use it to show that $\mathbf{W}_1^1$ proves that in Chomp the first player has a winning strategy; this gives another proof (by a standard witnessing argument) that this winning strategy can be computed in PSPACE. We end with two sections discussing open problems and with acknowledgments.

2 Background

A partially ordered set (a *poset*) is a set U together with an ordering relation $\prec$ on its elements, where $\prec$ is a subset of $U \times U$. The relation $\prec$ must satisfy the following conditions: (1) if $a \prec b$, then $b \not\prec a$ (*anti-symmetry*), and (2) if $a \prec b$ and $b \prec c$, then $a \prec c$ (*transitivity*). Not all elements are necessarily comparable, in that there may be elements a, b such that $a \neq b$, where $a \not\prec b$ and $b \not\prec a$. When two elements are *incomparable*, we write $a||b$. We are going to use $\preceq$ in practice, where $a \preceq b \iff [a \prec b \vee a = b]$.

Given a poset $(U, \preceq)$, a *poset game* $(A, \preceq)$ on $(U, \preceq)$ is played as follows: at first $A := U$. Then, two players take turns making moves. On each move, a player picks an element $x \in A$, and removes all the elements $y \in A$ such that $x \preceq y$. More precisely, let $S_x = \{y \in A | x \preceq y\}$. Then, after choosing x we reduce the universe to be $A - S_x$. The player who is unable to move because on their turn $A = \emptyset$, loses.

Our example of a poset game is Chomp, first detailed by Gale in [1] (but there are other Chomp games such as Nim, Hackendot or Hackenbush—see [2–4]). A game of Chomp is played on a "chocolate bar" divided into individual squares, with the bottom-left square being "poisoned." This is usually represented by a grid of m rows and n columns, with the poisoned square residing at position $(1, 1)$ (see Fig. 1). Two players take turns breaking off pieces of the chocolate by selecting a square (i, j) from among the remaining squares and deleting all the squares (k, l) such that $i \leq k$ and $j \leq l$. The game ends when a player selects the poisoned square—the player who does so loses.

To transform Chomp to a poset game we delete the lower-left square, so now the player who ends up without any chocolate to chomp loses. Define

$$\mathrm{Bar}_{m,n} := \{(i, j) | 1 \leq i \leq m, 1 \leq j \leq n\} - \{(1, 1)\},$$
$$(i, j) \preceq (k, l) \iff [(i \leq k \wedge j < l) \vee (i < k \wedge j \leq l)].$$

Note that $(i, j) = (k, l)$ iff $i = k$ and $k = l$, and it can be checked that $\prec$ is a partial order. Thus, for any m, n we have the poset game $(\mathrm{Bar}_{m,n}, \preceq)$.

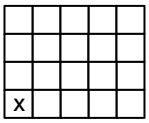

Fig. 1. A 4×5 Chomp grid. The 'X' denotes the poisoned square.

It is surprisingly easy to show, by a *strategy-stealing argument*, that player 1 has a winning strategy (no matter what the initial parameters m, n are, as long as it is not the case that $m = n = 1$). This argument works as follows: say that player 2 has a winning strategy (player 1 or 2 always wins, i.e., there can be no draw, and we show in theorem 1 that $\mathbf{W}_1^1$ can prove this). Then player 1 selects the top-left square (i.e., square (m, n)), and player 2 responds by selecting some square (i, j). But for all i, j it is always the case that $i \leq m$ and $j \leq n$. Therefore, player 1 could have chosen (i, j) to begin with; in either case, the outcome is the same board configuration, but the turns of the players have been reversed. So loser becomes winner and vice versa.

Note that the strategy-stealing argument can be applied to any poset game with a supremum (and where $|U| > 1$). But this argument is non-constructive, in the sense that it does not tell us what the strategy is. In the case of Chomp, it is an open question whether a winning strategy for player 1 can be computed in polynomial time. In section 3 we show, with a short and slick reduction to geography, that the existence of a winning strategy for player 1, for *any* poset game, can be computed in polynomial space. Therefore, using a standard reduction of a search problem to its decision problem, we can actually compute the strategy (if it exists) in polynomial space.

Recall that PSPACE is the class of languages decidable on a Turing machine in polynomial space in the length of the input. A language L is PSPACE complete if it is in PSPACE, and for every language L' in PSPACE, there exists a polynomial time function $f : \Sigma^* \longrightarrow \Sigma^*$, such that $x \in L' \iff f(x) \in L$. It is well known that many two-player games can be decided in PSPACE. See [5] or [6] for more background on PSPACE.

3 From poset games to geography

In this section we present a polynomial time reduction from poset games to geography and thereby show that poset games are in PSPACE. In order to study the complexity of poset games and their reductions, we assume that the posets are finite (i.e., U is finite). However, the reductions still works on infinite posets, and yields infinite instances of the game of geography.

The game of *geography* is a well known game in complexity, and it is often used as a paradigmatic example of a PSPACE complete language. As was mentioned in the introduction, its PSPACE completeness was shown for the first time in [7]—for a more recent presentation, see [5, Theorem 19.3] or [6, Theorem 8.14].

Geography is played as follows: a directed graph is given (it does not have to be acyclic), and a starting node s is specified. The first player selects s, then the second player selects an outgoing edge of s, and takes that edge to another node. The two players then traverse the graph, alternatively selecting outgoing edges of the current node. The player who is forced to revisit a node, or has no outgoing edges to select from, loses.

Define the following languages over $\{0,1\}$:

$$\textsc{Posetgame} = \{\langle (U, \preceq)\rangle : \text{player 1 has a winning strategy}\},$$
$$\textsc{Geography} = \{\langle (G = (V, E), s)\rangle : \text{player 1 has a winning strategy}\}.$$

Here $\langle (U, \preceq)\rangle$ and $\langle (G = (V, E), s)\rangle$ are the encodings of the poset $(U, \preceq)$ and the graph $(G = (V, E), s)$, respectively. To be more specific, let $\langle (U, \preceq)\rangle$ be a string over $\{0, 1\}$, of length $|U|^2$, consisting of the rows of a matrix with entries from $\{0, 1\}$, where entry (i, j) is a 1 iff $i \preceq j$. Let $\langle (G = (V, E), s)\rangle$ be the adjacency matrix of G, with the understanding that node s is always 1, also presented as a sequence of rows, giving a string of length $|V|^2$. (Note that in either case, a proper input is a string which is a perfect square, so the Turing machine can easily figure out the length of the rows.)

Since s (i.e., node 1) is the specified starting node for the game of geography on the graph G, the game starts by player 1 selecting node s, then player 2 selects an edge out of s to a new node n. This is in contrast to poset games, where any element $x \in U$ can be selected on the first turn.

We show that Posetgame can be reduced to Geography in logarithmic space (and hence in polynomial time), and conclude that Posetgame is in PSPACE. Note that Posetgame is a decision problem; the corresponding search problem is actually *finding* the winning strategy. This can be done with a standard (polytime) reduction of a search problem to its decision problem. Thus, computing winning strategies for poset games can also be done in PSPACE.

The following algorithm shows how to transform $\langle (U, \preceq)\rangle$ to $\langle (G = (V, E), s)\rangle$.

Reduction Algorithm: For every element $x \in U$ there corresponds a gadget of nodes $g(x)$ in G; see Fig. 2. The *gadget* is a subgraph consisting of the following nodes and edges:

$$g(x) := (\{x, x_1, x_2, x_3, x_4\}, \{(x, x_2), (x_2, x_3), (x_2, x_4), (x_4, x_1), (x_1, x)\}).$$

We call the nodes $\{x_1, x_2, x_3, x_4\}$ *auxiliary* nodes.

We say that there is a (directed) *connection* from gadget $g(y)$ to gadget $g(x)$, and write $[g(y), g(x)]$, if there are two (directed) edges from y_3 (in the gadget for y) to x and x_1 (in the gadget for x). Again, see Fig. 2.

We connect the gadgets as follows: if $x \prec y$ then we add the connection $[g(y), g(x)]$. If $x||y$ (i.e., x, y are incomparable), then we add both connections $[g(x), g(y)]$ and $[g(y), g(x)]$.

Finally, we add two more nodes, s, s', with an edge from s to s', and edges from s' to every non-auxiliary node. This ends the description of the reduction algorithm.

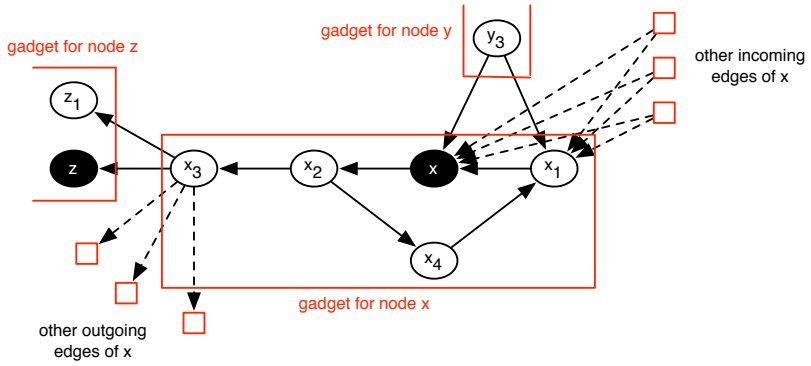

Fig. 2. Reduction gadget: each $x \in U$ is represented by five nodes in G.

The idea of the gadgets is to simulate the poset game by moving to gadget $g(x)$ whenever x was selected in U. The gadgets help us to keep track of A: every time player 2 makes a move to a gadget, player 1 can challenge that move. What is this challenge? It is player 1's claim that player 2 moved to a gadget $g(y)$ such that $x \prec y$ and x has already been played. We must also ensure that challenging a legal move, or even one's own move, yields no benefit to the challenger. These challenges ensure that players simulate the poset game correctly.

The nodes s, s' are required because in a poset game $(U, \preceq)$ we can select any $x \in U$ to start the game, whereas in geography the first player must select node s on their first turn. To emulate the free choice of the poset game, we have s as the initial node, which means player 2 must move to s', then player 1 can select any non-auxiliary node in V. It is clear that this simple construction can be carried out in logarithmic space: $|V| = 5|U| + 2$ and computing E from $\preceq$ is straightforward.

It is obvious that the geography game on G mirrors a poset game on $(U, \preceq)$ *if* it is played correctly, in the sense that we never go to a node that corresponds to an element in U that was removed at a previous step. As mentioned previously, the auxiliary nodes enable players to challenge each other's moves. We examine below what occurs during these challenges.

Legitimate Challenge: Here player 2 moves to a node y where $x \preceq y$ and node x was visited in the past. This means that player 2 moves to a $y \notin A$. Then player 1 challenges player 2 as follows: he moves from y to the challenging node x_1 of x. Player 2 is now forced to revisit x, and loses. This ensures that players can only move to legal nodes.

False Challenge: What if a player challenges a legal move? That is, if player 2 moved to node y which was in A? Then the following happens: player 1 challenges player 2 by moving to node x_1, and now player 2 moves to x and nothing happens as x has not been visited before. Then player 1 is forced to

move to x_2, and then player 2 moves to x_4, and player 1 is now forced to move to x_1, and thereby revisit a node, and player 1 loses. This not only shows that challenges to legal nodes yield no benefit, but also challenges to incomparable nodes yield no benefit.

Self-Challenge: The final type of challenge occurs when a player challenges their own move. Suppose player 2 arrives at node x without a challenge, i.e., directly through one of the incoming edges, then player 1 moves to x_2. Now player 2 has a choice to move to x_4, as if player 2 were going to do a counter-challenge. If indeed player 2 moves to x_4, then player 1 moves to x_1, and player 2 is forced to revisit node x, and player 2 loses. Thus, no gains are made if a player challenges their own move.

Therefore, if player 2 arrives at node x directly (without a challenge) he must continue (after the move of player 1) from x_2 to x_3, and then player 1 is free to select an outgoing edge from x_3 to another node z.

The above shows that there is a (correct) polynomial time reduction from Posetgame to Geography. As Geography itself is in PSPACE, this gives us that Posetgame $\in$ PSPACE as well. Of course, this was expected as the existence of a winning strategy for a given "board configuration" can be expressed as the satisfiability of a quantified Boolean formula. On the other hand, we have given a simple and direct proof by reducing to Geography.

The interesting open questions are whether the language Posetgame is PSPACE-complete, and to characterize the poset games for which a winning strategy can be computed efficiently. There are a few instances of starting configurations for which this can be done; for example, consider a chomp configuration in the shape of an "L." That is, we have the first column and last row, with their intersection containing the poisoned square. Then as long as the two arms of the "L" have a different length, the first player has the following (simple) winning strategy: the first player chomps the longer arm to be the same length as the shorter, and then copies symmetrically the moves of the second player on the opposite arm, to force the second player to be left with the poisoned square.

For further instances for which it is possible to compute a winning strategy see [18], where it is shown how to play when the chocolate bar has at most three rows (with some restrictions), and [21], where the *Poset Game Periodicity Theorem*, regarding certain winning configurations. It should be noted, however, that we seem to be very far from knowing a general strategy for winning Chomp. This is what makes Chomp so curious: it is easy to establish that player 1 *can* win; what is computationally difficult is to establish *how*.

4 Formalizing strategy stealing in $\mathbf{W}_1^1$

In this section we provide an alternative proof of the existence of a PSPACE winning strategy for player 1 in the game of Chomp: we formalize the strategy stealing argument using Skelley's theory $\mathbf{W}_1^1$ ([8]). In fact, our approach works for any poset game to which the strategy stealing argument may be applied, namely, any poset game with a supremum. We hope that a program consisting

in formalizing proofs of existence of winning strategies in weaker and weaker fragments of Bounded Arithmetic can yield better algorithms (i.e., algorithms of lower complexity) for computing winning strategies for poset games.

$\mathbf{W}_1^1$ is a logical theory that captures PSPACE reasoning. It extends $\mathbf{V}^1$ (which captures polynomial time, the class P) presented by Cook and Nguyen in [9], but $\mathbf{W}_1^1$ is designed to reason over sets of strings, which themselves encode sets of sets of elements. $\mathbf{W}_1^1$ is a three-sorted ("third-order") predicate calculus with free and bound variables of three sorts (the bound variables are given in parentheses): $a, b, c, \ldots$ $(x, y, z, \ldots)$ for the first sort, intended to denote natural numbers encoded in unary, $A, B, C, \ldots$ $(X, Y, Z, \ldots)$ for the second sort, intended to denote finite binary strings, and $\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$ $(\mathbb{X}, \mathbb{Y}, \mathbb{Z}, \ldots)$ for the third sort, intended to denote sets of strings, and in particular functions from strings to strings.

The third order language we use is $\mathcal{L}_A^3 = [0, 1, +, \cdot, |\cdot|_2, \in_2, \in_3, \leq, =]$ where $0, 1$ are numbers, $+, \cdot$ is addition and multiplication of numbers, $|\cdot|_2$ denotes length of strings (note that size of third sort objects is not there). We abbreviate $i \in_2 X$ and $X \in_3 \mathbb{X}$ as $X(i)$ and $\mathbb{X}(X)$, respectively. (we may omit the subscript $_2$ or $_3$ if it is clear from the context which type it is). Finally, $\leq, =$ are used to compare numbers. There is no equality symbol for the second and third sort; equality for these objects can be defined with the symbols already in $\mathcal{L}_A^3$; for example, equality of strings, $X = Y$, can be stated as $[|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i))]$.

Table 1. The set of axioms **2-BASIC**.

B1. $x + 1 \neq 0$	**B7.** $(x \leq y \wedge y \leq x) \rightarrow x = y$
B2. $(x + 1 = y + 1) \rightarrow x = y$	**B8.** $x \leq x + y$
B3. $x + 0 = x$	**B9.** $0 \leq x$
B4. $x + (y + 1) = (x + y) + 1$	**B10.** $x \leq y \vee y \leq x$
B5. $x \times 0 = 0$	**B11.** $x \leq y \leftrightarrow x < y + 1$
B6. $x \times (y + 1) = (x \times y) + x$	**B12.** $x \neq 0 \rightarrow \exists y \leq x(y + 1 = x)$
L1. $X(y) \rightarrow y < \|X\|$	**L2.** $y + 1 = \|X\| \rightarrow X(y)$

We let $\Sigma_i^{\mathbb{B}}$ be the set of formulas over $\mathcal{L}_A^3$ containing formulas with arbitrarily many bounded first and second-order quantifiers, and exactly i alternations of third-order quantifiers (when put in prenex form). The comprehension axiom schemes are the following:

$$\left(\exists Y \leq t\left(\overline{a}, \overline{A}\right)\right)\left(\forall z \leq s\left(\overline{a}, \overline{A}\right)\right)\left[\phi\left(\overline{a}, \overline{A}, \overline{\mathbb{A}}, z\right) \leftrightarrow Y\left(z\right)\right], \qquad \Sigma_0^{\mathbb{B}}\text{-2COMP}$$

$$\left(\exists \mathbb{Y}\right)\left(\forall Z \leq s\left(\overline{a}, \overline{A}\right)\right)\left[\phi\left(\overline{a}, \overline{A}, \overline{\mathbb{A}}, Z\right) \leftrightarrow \mathbb{Y}\left(Z\right)\right]. \qquad \Sigma_0^{\mathbb{B}}\text{-3COMP}$$

In each of these schemes $\phi \in \Sigma_0^{\mathbb{B}}$ is subject to the restriction that neither Y nor $\mathbb{Y}$ (as appropriate) occurs free in ϕ.

The theory $\mathbf{W}_1^1$ is axiomatized by B1–B12 and L1,L2, listed in table 1, as well as the two comprehension axiom schemes $\Sigma_0^{\mathbb{B}}$-2COMP and $\Sigma_0^{\mathbb{B}}$-3COMP. The

reader is directed once more to the source, [8], for details on $\mathbf{W}_1^1$; we are going to limit ourselves to the use of the witnessing theorem for $\mathbf{W}_1^1$ ([8, theorem 4]) in the following fashion. The third order elements can be used to represent functions: $\mathbb{S}(C_1) = C_2$ (formally, $\langle C_1, C_2\rangle \in \mathbb{S}$). If $\mathbf{W}_1^1 \vdash \exists\mathbb{S}\alpha$, where $\alpha \in \Sigma_0^{\mathbb{B}}$, and $\mathbb{S}$ represent a function, then we can compute $\mathbb{S}$ in polynomial space (in $|C_1|$).

Thus, our goal is now to show that $\mathbf{W}_1^1$ proves the existence of a winning strategy $\mathbb{S}$ for player 1 in chomp. We start with a neat way of encoding board configurations of chomp.

Following a suggestion of [10], a chomp configuration on an $n \times m$ board can be represented with a binary string $X \in \{0,1\}^{n+m}$, where there are exactly n ones and m zeros. In this scheme, a board configuration can be seen as a path from the upper-left corner to the lower-right corner (the poisoned square is in the lower-left corner), where we are only allowed to move right or down. Reading the string left to right, every time we see a 0 we move right, and every time we see a 1 we move down.

The original configuration according to this representation is `000000011111`, and the final configuration is the string with all the 1s moved to the left. Whoever moves the game into the final configuration loses. An example of two intermediate configurations is given in Fig. 3.

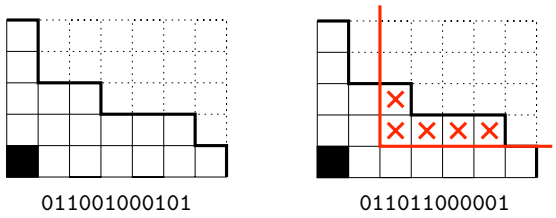

Fig. 3. The board on the right is the result of chomping off the squares with the X's from the board on the left. The corresponding string representations are given below each board.

Each move is a "chomp," which consists in picking a square, from among the surviving squares (i.e., to the left of the thick line in Fig. 3), and chomping off all the squares to the right and up. This means, that each move of chomp consists in moving one or more 1s, one or more positions to the left, without ever "overtaking" the 1s in front.

All this can be expressed in the language of $\mathbf{W}_1^1$. We are going to provide a formula $\Phi(X, n, m)$ which will assert that X is a valid chomp game on an $n \times m$ board. Here X is a long string, consisting of $n \cdot m$ many segments (the longest possible chomp game, where each player bites off just one square at a time) of length $(n+m)$ each. We denote the i-th segment by $X^{[i]}$ (this can be defined in $\mathbf{W}_1^1$; see [9]); Φ is the conjunction of three formulas: $\phi_{\text{init}}, \phi_{\text{final}}$, and ϕ_{move}.

ϕ_{init} asserts that $X^{[1]}$ is the initial configuration, i.e.,

$$\phi_{\text{init}}(X, n, m) := \forall i \leq (n+m)((i > m) \rightarrow X^{[1]}(i) = 1),$$

ϕ_{final} asserts that $X^{[n \cdot m]}$ is the final configuration:

$$\phi_{\text{final}}(X, n, m) := \forall i \leq (n+m)((i > n) \rightarrow X^{[n \cdot m]}(i) = 0),$$

and ϕ_{move} asserts that each segment of X can be obtained from a legal move applied to the previous segment (or simply by copying the last segment—this will be needed for games that last less than $n \cdot m$ many moves):

$$\phi_{\text{move}}(X, n, m) := \forall i < (n \cdot m)(X^{[i]} \text{ “yields” } X^{[i+1]}).$$

The work, of course, is in defining the “yields,” which we proceed to do next. Note that for the remainder of this paper we will “invert” our coordinate system for Chomp configurations, as now the “origin” $(1, 1)$ is the top-right square, while (m, n) is the bottom-right square.

First, notice that in our string representation of configurations, a 0 occurring to the left of at least a single 1 corresponds to a column of playable squares, the height of which is the number of 1s to the right of the 0. Thus, a square (j, k) can be played on $X^{[i]}$ if and only if, in reading $X^{[i]}$ from left to right, the k^{th} 0 is encountered before the j^{th} 1.

Using the function `numones` from [9] we can count the number of 1s in any sub-segment of a string X. Thus, we can define the following two functions: F_0 and F_1, where

$$\mathrm{F}_0(a, b, X) = c,$$

if the earliest position in X where we see b zeros, starting in position a and going right, is position c. Analogously, we define $\mathrm{F}_1(a, b, X) = c$ but we count ones instead. There are details to be taken care of, like how to define the function when there is no such c (e.g., if $b > |X|$). But taking care of those details is routine, as we have a powerful language at our disposal, and so we omit it.

Lemma 1. *A square (j, k) can be played on a configuration $X^{[i]}$ if and only if*

$$\mathrm{F}_0(1, k, X^{[i]}) < \mathrm{F}_1(1, j, X^{[i]}). \qquad (\psi_1)$$

Proof. We will argue by contradiction; suppose square (j, k) can be played on configuration $X^{[i]}$, but $\mathrm{F}_0(1, k, X^{[i]}) > \mathrm{F}_1(1, j, X^{[i]})$.

We cannot have $\mathrm{F}_0(1, k, X^{[i]}) = \mathrm{F}_1(1, j, X^{[i]})$ as a position in $X^{[i]}$ cannot be simultaneously 0 and 1. This means that the k^{th} 0 occurs after that j^{th} 1, which in turn means that row j terminated before column k. Thus, there are no squares beyond column $k-1$ in row j, so square (i, j) does not exist in $X^{[i]}$, and cannot be played. □

Now that we can identify playable squares, we need to determine the configurations which result from playing these squares. In general, playing a square (j, k) shifts at least a single 1 behind the 0 corresponding to the move, to delimit

the row being shortened or eliminated. If a move affects multiple rows, then multiple 1s will be shifted over. In order to figure out the substring of X which is affected by playing a square (j, k), we calculate two values p and q which mark the beginning and ending of the substring (inclusive):

$$p = \mathrm{F}_0(1, k-1, X^{[i]}) + 1, \tag{ψ_2}$$

$$q = \mathrm{F}_1(p, j, X^{[i]}). \tag{ψ_3}$$

Intuitively, $(p-1)$ marks the location of the $(k-1)^{\text{th}}$ 0, so p forms the left boundary of the substring, and q is the position of $X^{[i]}$ in which we have seen j 1s starting from p, so it designates the right boundary. Finally, to arrive at the new configuration $X^{[i+1]}$ we replace positions p through q in $X^{[i]}$ with $1^j 0^{(q-p-k+1)}$, i.e., the number of 1s corresponding to the number of rows affected, then the number of 0s indicating the new difference in row lengths.

In order to express this new substring as a formula we break it down into several sub-formulas. First, positions that are outside the range of p and q remain unchanged:

$$(r < p) \to (X^{[i+1]}(r) = X^{[i]}(r)), \tag{ψ_4}$$

$$(r > q) \to (X^{[i+1]}(r) = X^{[i]}(r)). \tag{ψ_5}$$

Next, for positions between p and q we assign values based on their placement relative to the 0 associated with the move. Positions before the 0 hold 1s affected by the move, while positions after contain 0s.

$$(r < p + j) \to (X^{[i+1]}(r) = 1), \tag{ψ_6}$$

$$(r \geq p + j) \to (X^{[i+1]}(r) = 0), \tag{ψ_7}$$

$$(p \leq r \leq q) \to (\psi_6 \wedge \psi_7). \tag{ψ_8}$$

We now put formulas ψ_4, ψ_5 and ψ_8 together to create one formula describing the changes between configurations $X^{[i]}$ and $X^{[i+1]}$:

$$(\forall r \leq |X|)(\psi_4 \wedge \psi_5 \wedge \psi_8). \tag{ψ_9}$$

Finally, we combine formulas ψ_2 and ψ_3 with existential quantification to describe the existence of valid p and q:

$$(\exists p < |X^{[i]}|)(\exists q \leq |X^{[i]}|)(\psi_2 \wedge \psi_3) \tag{ψ_{10}}$$

Thus, if we take the formula ψ_9 for describing legal changes between configurations and combine it with formula ψ_1 for the existence of a playable square, as well as ψ_{10} for the existence of values for p and q, we have the following formula for "yields":

$$(\exists j \leq |X^{[i]}|)(\exists k \leq |X^{[i]}|)\left[\psi_1 \wedge \psi_{10} \wedge \psi_9\right].$$

Having designed the formula $\Phi(X, n, m)$—and note that Φ is a $\Sigma_0^{\mathbb{B}}$—we can now state and prove in $\mathbf{W}_1^1$ properties of a winning strategy.

A winning strategy in chomp is just a function $\mathbb{S}$ which maps configurations to configurations. If a player *has* a winning strategy, and *plays by* $\mathbb{S}$, then that player wins. The fact that player 1 in chomp has a winning strategy can be stated as follows: $\forall X$, where $|X| \leq nm(n+m)$, and $\Phi(X, n, m)$ (i.e., X is a valid history of a chomp game), if $\forall i$, where $i < (nm/2)$, it is the case that $\mathbb{S}(X^{[2i+1]}) = X^{[2i+2]}$, then player 2 eats the poisoned square.

We want to say that given any configuration C as an initial configuration, either player 1 or player 2 has a winning strategy. This can be stated as follows:

$$\forall C \exists \mathbb{S}[\mathrm{Win}_{P_1}(\mathbb{S}, C) \vee \mathrm{Win}_{P_2}(\mathbb{S}, C)], \tag{1}$$

where $\mathrm{Win}_{P_i}(\mathbb{S}, C)$ ($i \in \{0, 1\}$, and $\bar{i} := 1 - i$) asserts that if player i plays by strategy $\mathbb{S}$, then player i wins.

We now define the formula Win_{P_i} as follows: $\forall Y$ (where Y is a sequence of moves of player $\bar{i}$), if it is player i's move on configuration C, player i plays $C' = \mathbb{S}(C)$, and the two players play a valid game (which can be ensured with the formula Φ), then player $\bar{i}$ will end up with the poisoned square. Note that (1) is a $\Sigma_1^{\mathbb{B}}$ formula since the "$\forall C$" is bounded by the size of the chomp grid—we omit the bound for clarity, and Win_{P_i} is a $\Sigma_0^{\mathbb{B}}$ formula.

Theorem 1. $\mathbf{W}_1^1 \vdash \forall C \exists \mathbb{S}[\mathrm{Win}_{P_1}(\mathbb{S}, C) \vee \mathrm{Win}_{P_2}(\mathbb{S}, C)]$.

Proof. We prove it by induction on $|C|$, where we let $|C|$ be the number of squares in configuration C (in particular, if C is just the poisoned square, then $|C| = 1$). Note that we do not have induction on the length of strings in $\mathbf{W}_1^1$ as is, however induction is derivable from the comprehension axioms $\Sigma_0^{\mathbb{B}}$-2COMP—see [9] where this is done in the context of the theory $\mathbf{V}^1$.

Basis case: if $|C| = 1$, then P_1 loses, so in particular $\exists \mathbb{S} \mathrm{Win}_{P_2}(\mathbb{S}, C)$ holds; in fact any $\mathbb{S}$ is a witness here.

For the induction step, suppose that the claim holds for all C such that $|C| \leq n$. Consider some C' such that $|C'| = n + 1$. We want to show that

$$\exists \mathbb{S}[\mathrm{Win}_{P_1}(\mathbb{S}, C') \vee \mathrm{Win}_{P_2}(\mathbb{S}, C')]. \tag{2}$$

Let C'' be the result of P_1 making a move; since P_1 must select some square, it follows that $|C''| < |C'|$, and so we can apply the induction hypothesis to it; in other words, $\exists \mathbb{S} \mathrm{Win}_{P_1}(\mathbb{S}, C'') \vee \exists \mathbb{S} \mathrm{Win}_{P_2}(\mathbb{S}, C'')$ (we used the fact here that $\exists x(\alpha \vee \beta)$ is equivalent to $\exists x \alpha \vee \exists x \beta$).

If no matter what first move P_1 makes (to obtain C'') it is always the case that $\exists \mathbb{S} \mathrm{Win}_{P_2}(\mathbb{S}, C'')$, then we can conclude that $\exists \mathbb{S} \mathrm{Win}_{P_2}(\mathbb{S}, C')$. If, on the other hand, for some move of P_1, it is the case that $\exists \mathbb{S} \mathrm{Win}_{P_1}(\mathbb{S}, C'')$, then define (using comprehension) $\mathbb{S}'$ to be the same as $\mathbb{S}$, except $\mathbb{S}'(C') = C''$, and we can conclude that $\mathrm{Win}_{P_1}(\mathbb{S}', C')$, and so $\exists \mathbb{S} \mathrm{Win}_{P_1}(\mathbb{S}, C')$. Therefore, in either case we obtain (2). □

Once we know that $\mathbf{W}_1^1$ proves (1), i.e., that $\mathbf{W}_1^1$ can show that either player 1 or player 2 has a winning strategy, then we can show that $\mathbf{W}_1^1$ proves that P_1 has a winning strategy for the full rectangle. It is in this theorem that we formalize the strategy-stealing argument in $\mathbf{W}_1^1$.

Theorem 2. $\mathbf{W}_1^1 \vdash$ *"C is full rectangle"* $\rightarrow \exists \mathbb{S} \mathrm{Win}_{P_1}(\mathbb{S}, C)$.

Proof. Suppose that C is a full rectangle (an $r \times c$ chomp grid). We know that either P_1 or P_2 has a winning strategy. Suppose that it is P_2; so P_2 (playing by some $\mathbb{S}$) will win no matter what first move P_1 makes. So let P_1 select the top-right square (i.e., the square (r, c)), and let C' be the resulting configuration (i.e., C' has all the squares except it has a dent in the top-right square).

Now P_2 makes a move according to $\mathbb{S}$, i.e., P_2 plays to obtain $C'' = \mathbb{S}(C')$. Now observe that whoever starts playing from configuration C'' loses, i.e., P_1 loses on C'', (by our assumptions). Also observe that P_1 could have played to obtain C'' directly, since no matter what C'' is, it must contain the top-right corner, as it is the supremum of the grid. Thus, by taking the strategy $\mathbb{S}'$ to be $\mathbb{S}(C) = C''$ and otherwise $\mathbb{S}' = \mathbb{S}$, P_1 can win (note that we have used comprehension to define $\mathbb{S}'$ from $\mathbb{S}$). Contradiction; so P_2 cannot have a winning strategy, and so P_1 must have a winning strategy. □

5 Open Problems

Theorem 2 may be redone for any poset game with a supremum. The question is: can we prove the existence of winning strategies for such games in a theory weaker than $\mathbf{W}_1^1$, and thereby conclude that the winning strategy can be computed in a subclass of PSPACE?

Another possible direction is to classify poset games, using formal proofs of existence of strategies, according to the complexity required for computing the winning strategy.

An interesting open question is whether poset games are PSPACE complete. It has been suggested by [11] that the game of Hex, which is known to be PSPACE complete, would be a good candidate to reduce to POSETGAME.

6 Acknowledgments

The results of this paper were first presented at the *Computability in Europe 2008* conference in Athens, Greece. We are very grateful to the anonymous referees of the conference who gave us invaluable comments regarding this paper.

References

1. Gale, D.: A Curious Nim-Type Game. The American Mathematical Monthly **81**(8) (October 1974) 876–879
2. Bouton, C.L.: Nim, A Game with a Complete Mathematical Theory. The Annals of Mathematics **vol. 3**(1/4) (1902) 35–39
3. Úlehla, J.: A complete analysis of Von Neumann's Hackendot. International Journal of Game Theory **9** (1980) 107–113
4. Berlekamp, E., Conway, J., Guy, R.: Winning Ways for Your Mathematical Plays. Second edn. Volume 1. A K Peters, Ltd. (2001)

5. Papadimitriou, C.H.: Computational Complexity. Addison Wesley Longman (1995)
6. Sipser, M.: Introduction to the Theory of Computation. Second edn. Thomson Course Technology (2006)
7. Schäfer, T.J.: Complexity of some two-person perfect-information games. J.CSS **16** (1978) 185–225
8. Skelley, A.: A Third-Order Bounded Arithmetic Theory for PSPACE. In: Computer Science Logic, 18th International Workshop. (2004) 340–354
9. Cook, S., Nguyen, P.: Foundations of Proof Complexity: Bounded Arithmetic and Propositional Translations. `www.cs.toronto.edu/~sacook/csc2429h/book` (2006)
10. Herman, G.: Private communication (2006) (McMaster University).
11. Cook, S.: Private communication (2008)
12. Friedman, E.J., Landsberg, A.S.: Nonlinear dynamics in combinatorial games: Renormalizing chomp. Chaos: An Interdisciplinary Journal of Nonlinear Science **17**(2) (2007)
13. Sun, X.: Improvements on chomp. Integers: Electronic Journal of Combinatorial Number Theory **2**(G01) (2002) 1–8
14. Conway, J.: On Numbers and Games. Second edn. A K Peters, Ltd. (2001)
15. Fraenkel, A.S., Yesha, Y.: The generalized Sprague-Grundy function and its invariance under certain mappings. J. Comb. Theory Ser. A **43**(2) (1986) 165–177
16. Backhouse, R., Michaelis, D.: Fixed-Point Characterisation of Winning Strategies in Impartial Games. In: Relational and Kleene-Algebraic Methods in Computer Science. Volume 7., Springer (2004) 34–47
17. Fraenkel, A.S.: Scenic Trails Ascending from Sea-Level Nim to Alpine Chess. In: Games of No Chance. Volume 29., MSRI (1996)
18. Zeilberger, D.: Three-Rowed CHOMP. Advanced Applied Math **26** (2001) 168–179
19. Brouwer, A.: Chomp. `http://www.win.tue.nl/~aeb/games/chomp.html` (2005)
20. Deuber, W., Thomassé, S.: Grundy Sets of Partial Orders. preprint (1996)
21. Byrnes, S.: Poset Game Periodicity. Integers: Electronic Journal of Combinatorial Number Theory **3**(G03) (November 2003)