Strategy-Stealing is Non-Constructive

Greg Bodwin[*1] and Ofer Grossman[†2]

[1]Georgia Tech
[2]MIT

Abstract

In many combinatorial games, one can prove that the first player wins under best play using a simple but non-constructive argument called *strategy-stealing*. This work is about the complexity behind these proofs: how hard is it to actually find a winning move in a game, when you know by strategy-stealing that one exists? We prove that this problem is PSPACE-Complete already for *Minimum Poset Games* and *Symmetric Maker-Maker Games*, which are simple classes of games that capture two of the main types of strategy-stealing arguments in the current literature.
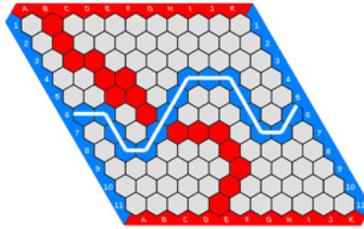
Figure 1: A Hex board with a winning configuration for Blue [hex].

1 Introduction

Theoretical Computer Science includes a rich theory of the complexity class TFNP, defined as the set of NP search problems where a solution always exists. The interesting subclasses of TFNP are based on simple yet non-constructive existence proofs for these solutions. For example: given a circuit

$$C : \{0,1\}^n \to \{0,1\}^{n-1},$$

one sees immediately by the Pigeonhole Principle that there exist distinct inputs x_1, x_2 with matching output $C(x_1) = C(x_2)$. But can one find such a pair of inputs computationally? This problem is complete for a complexity class PWPP $\subseteq$ TFNP, and a similar story holds for various other problems with other non-constructive proofs of solution existence.

A major motivation for TFNP as an object of study is that it gives satisfying formalizations of the natural question of whether a type of proof is constructive ("is the Pigeonhole Principle constructive?" roughly corresponds to "is P = PWPP?"). But really, some non-constructive proof methods in mathematics do not correspond to NP search problems at all. Thus, we argue, a valuable direction for research in the spirit of TFNP is to look outside TFNP itself to analyze the constructiveness of proofs in other complexity classes. This paper is about one such instance: *strategy-stealing proofs*, which are fundamental existence results in combinatorial game theory that naturally lie in PSPACE.

1.1 Combinatorial Games and Strategy-Stealing

A *combinatorial game* is a finite two-player game of perfect information. The players take turns choosing *moves* that manipulate a game board by some predefined rules, eventually reaching one out of a set of *terminal states* which determine the outcome of the game. Examples of combinatorial games include chess, go, tic-tac-toe, and some others that we will describe in detail shortly.

In general, deciding which player has a winning strategy in a combinatorial game is computationally hard. However, certain classes of games admit slick proofs that a certain player wins under best play (and thus determining who has a winning strategy in these games is not computationally hard). A famous example is the game *Hex*, in which two players named Red and Blue alternately color in hexagons in a symmetric board (pictured in Figure 1); Red wins if there is a continuous path of red hexagons connecting the top and bottom, and Blue wins if there is a continuous path of blue hexagons connecting the left and right. The *Hex Theorem* states that exactly one of the two players will achieve a winning configuration once all hexagons have been colored, so there are no draws. It was observed by Nash [Nas52] that:

Theorem 1.1 ([Nas52])**.** *The first player has a winning strategy in the game of Hex.*

Proof Sketch. Suppose for contradiction that the second player has a winning strategy. The first player can then make an arbitrary first move and then "steal" the winning strategy of the second player. That is, he will now pretend he is the second player, and play according to the second player's winning strategy. This

will lead to a win for the first player anyways since their arbitrary initial move can only help them achieve a winning configuration. □

This has been dubbed the first *strategy-stealing* proof, referring to a now-broad collection of proofs that assume for contradiction that the second player can win, then repurpose the winning strategy to create a win for the first player. Another illustrative example is the game *Chomp*. Here, the game board is an $m \times n$ chocolate bar in which the top right square has been poisoned. The players alternately choose an uneaten square, and then eat that square and all remaining squares down and to the left. A player loses if they eat the poisoned square.
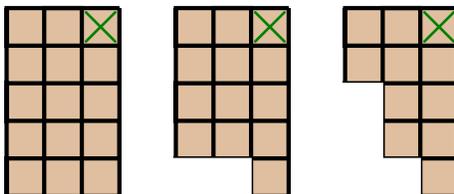

Figure 2: A valid two-move sequence from the starting position in 5×3 Chomp.

The following strategy stealing argument applies to Chomp:

Theorem 1.2 (Folklore). *The first player has a winning strategy in the game of Chomp.*

Proof Sketch. Consider the possible first move where the first player chomps off only the bottom-left-most square. There are two cases. Maybe the second player does not have a winning response, in which case the game is a win for the first player. Alternately, suppose the second player has a winning response by chomping off an $a \times b$ block. Since this block necessarily contains the bottom-left-most square, the board state is the same as if an $a \times b$ block had been chomped off with the first move of the game. It follows that this $a \times b$ chomp, instead, would have been a winning first move for the first player. □

Both of the proofs above *seem* non-constructive, in the sense that they do not yield an actual winning first move. Our central research question is whether this is inherent:

Are strategy-stealing proofs constructive?

To tackle this problem, we consider games that admit strategy stealing proofs, and we investigate the computational hardness of finding winning moves in such games. With this view, one can see that strategy-stealing proofs can essentially be *arbitrarily* non-constructive: for any combinatorial game X with two players P1 and P2, we can define a game X' in which the first player can decide whether he wishes to play as P1 or P2 in game X.[1] Then finding a winning move for P1 is the same as determining the winner of X, which in general is computationally hard. (We discuss this point in a little more detail in the conclusion.)

Thus, a more interesting direction is not to proceed in the maximally general case, but rather to investigate whether hardness persists in special classes of games to which strategy-stealing applies. In particular, we will study games which (to our eye) are the minimal natural classes captured by the two strategy-stealing arguments given above.

---

[1]Notice the strategy stealing argument that the first player has a winning strategy in the game X': suppose otherwise. Then we know if the first player chooses to play as P1, the second player has a winning strategy, so P2 has a winning strategy in X. But then the first player can choose to play as P2 in X' and use the winning strategy for P2 in X.

1.2 Our Results

To capture "Hex-type strategy stealing," we consider the well-studied class of *symmetric Maker-Maker games*:

Definition 1.3 (Symmetric Maker-Maker Game[2]). In a *Maker-Maker Game*, two players alternately claim elements of a finite universe U. There are families of *winning sets* $W_1, W_2 \subseteq P(U)$; the first player wins as soon as they claim all the elements of any winning set $S \in W_1$, the second player wins as soon as they claim all the elements of any winning set $S \in W_2$, and the game is a draw if all of U is claimed without either player winning. The game is *symmetric* if W_1, W_2 are isomorphic, i.e., there is a permutation π of U and a bijection $\phi : W_1 \to W_2$ such that for all $S_1 = \{s_1, \ldots, s_k\} \in W_1$, we have $\phi(S_1) = \{\pi(s_1), \ldots, \pi(s_k)\}$.

Hex is a symmetric Maker-Maker (SMM) game, and indeed the proof of Theorem 1.1 generalizes immediately to imply that any SMM game is not a win for the second player. There are many other examples of SMM games, which will be surveyed later. In general an SMM game can be a draw under best play, but some games like Hex are *draw-free* and so a first-player win is the only remaining possibility. We associate a computational problem to these games as follows:

Definition 1.4 (SMMMove). The problem SMMMove is defined as follows:

- Input: Circuits C_1, C_2, both with input wires labelled $x_1, \ldots, x_n$. C_1 and C_2 are the same up to relabelling of the wires. Call $X = \{x_1, \ldots, x_n\}$. The Maker-Maker game associated with this input is where W_1 contains any set of inputs $Y \subseteq X$ for which C_1 evaluates to true when the inputs in Y are set to true and $X \setminus Y$ to false. W_2 is defined similarly with respect to C_2.

- Output: any optimal first move for the first player in the associated game.

We prove:

Theorem 1.5. SMMMove *is* PSPACE-*hard,[3] even under the additional promise that the input defines a draw-free game with $W_1 = W_2$.*

Thus, Hex-type strategy stealing is a formally non-constructive style of proof, and additional draw-freeness results like the Hex Theorem do not generally help. To capture "Chomp-type strategy stealing," we consider:

Definition 1.6 (Minimum Poset Games). In a *poset game*, two players alternately choose remaining elements of a poset P, removing the chosen element and all lesser elements at each step. A player loses if it is their turn but the poset is empty. The game is *minimum* if P has a minimum element (i.e., $m \in P$ that is comparable to and less than every other element in P).

(For both these types of games, we refer to [Bec08, HKSS14] for some of their history and prior work.) Chomp is a minimum poset game, where the associated poset holds the squares of the chocolate bar, with the poisoned square removed, and squares are compared by the usual poset relation on $\mathbb{Z}^2$ (note that the bottom-left-most square is a minimum element). Theorem 1.2 generalizes to show that any minimum poset game is a win for the first player. Other examples of poset games, which may or may not have a minimum, include Nim, Hackendot, certain cases of Hackenbush, and many others. Computationally, we have:

Definition 1.7 (MPMove). The problem MPMove is defined as follows:

- Input: a poset P (with elements and relations between them enumerated explicitly) with a minimum element.

- Output: any winning move for the first player in the poset game defined by P.

---

[2]This is a generalization of the usual definition: in the literature, a "Maker-Maker game" often implies $W_1 = W_2$.

[3]A straightforward algorithm solves SMMMove in polynomial space, so in some sense it is complete (ignoring subtleties in the terminology), but we will not discuss these easy upper bounds in this paper. A similar comment holds for MPMove below.

Theorem 1.8. MPMOVE *is* PSPACE-*hard.*

From a technical standpoint, both Theorems 1.5 and 1.8 are proved roughly as follows. We start with a theorem in prior work stating that it is PSPACE-hard to decide the winner in a related class of games: Hex from an arbitrary starting position [Rei81], or a certain class of poset games [Gri13]. We then apply transformations that introduce the necessary strategy-stealing properties to these games while arguing that the winner in the original game is implicitly encoded by the first player's winning move(s). In the case of minimum poset games, this is an easy extension of the theorem in [Gri13]; we include this mostly to illustrate our conceptual goal of computationally formalizing non-constructiveness. For SMM games, the transformation is nontrivial and requires significant new ideas. Thus, the SMM result constitutes our main technical contribution.

1.3 Related Work

As mentioned, this paper is conceptually related to the study of the complexity class TFNP, defined in [MP91] and including notable subclasses PPAD, PPA, CLS, PPP, PWPP, PLS, among others. These classes all hold search problems in NP that admit proofs that a solution always exists. Our work is related in that our goal is to prove hardness of searching for a winning moves in games, when there are strategy-stealing proofs that one always exists. The key difference is that our problems are not in NP; there is not generally a short certificate that an optimal move is indeed the first one in some optimal strategy.

In [Bys04], the author proves that it is PSPACE-hard to decide whether an SMM game is a win for the first player or a draw. Our work differs in that (1) to minimally generalize Hex we restrict attention to *draw-free* games, in which this decision problem is trivial, and (2) we are interested in constructively finding a winning move rather than deciding existence. Similarly related is [Rei81], in which it is proved to be PSPACE-hard to decide whether Hex from a partially-filled board is a win for the first or second player. This can be viewed as a Maker-Maker game, but since the board is partially filled, it is not generally a symmetric Maker-Maker game and thus strategy-stealing does not apply.

There is a rich and developed theory of Maker-Maker games, poset games, and variants, most of which focuses on understanding these games under best play (rather than computational aspects of playing the games). See books [Bec08, HKSS14] for more information.

2 Non-Constructiveness of Strategy-Stealing

We will now prove our main results.

2.1 Symmetric Maker-Maker Games

Our first topic will be Symmetric Maker-Maker games, and eventually a proof of Theorem 1.5.

Examples. We first survey some famous examples of SMM games in the literature.

- Hex is an SMM game, as discussed above, which is draw-free and has non-equal winning sets.

- In the (n, k)-*Clique game*, the game board is a complete graph on n nodes and the players take turns claiming its edges. The first player to claim all edges in a k-clique wins. This is a symmetric Maker-Maker game, even with identical winning sets $W_1 = W_2$ (i.e., the underlying permutation is the identity). An interesting property of this game is that, for all k, if n is sufficiently large then the game is draw-free. This follows from *Ramsey's Theorem*, which states that any 2-coloring of the edges of the complete graph has a monochromatic clique of size $\Omega(\log n)$. Hence, for large enough n, strategy-stealing implies specifically that the first player has a winning strategy. For work on the Clique game and some natural variants, see e.g., [Bec81, Geb12, ES73, Bec02, Kus17].

- Tic-Tac-Toe is an SMM game, where the winning sets are the 8 possible "lines" in the 3×3 grid. This game is not draw-free.

- In (k, d) Tic-Tac-Toe, the game board is the elements of the k^d hypercube $(\{1, 2, \ldots, k\}^d)$, and the winning sets $W_1 = W_2$ are the k-element subsets which are colinear in the hypercube. The *Hales-Jewett Theorem* [HJ09] implies that for every k, if d is sufficiently large then the game is draw-free. For work on this game, see e.g., [Bec08, GH02].

- In the (n, k) *Arithmetic Progression game*, the universe is the set of integers $\{1, \ldots, n\}$, and the winning sets are any k elements that form an arithmetic progression (i.e., the difference between successive integers is equal). *Van-der-Waerden's Theorem* [vdW27] implies that for every k, if n is sufficiently large then the game is draw-free. The Arithmetic Progression game has been studied e.g., in [KRSS17, Bec81, Kus17].

All of these games are SMM and hence admit strategy-stealing proofs that the second player does not win under best play. The problem of finding an optimal first move for the first player can thus be captured as a special case of SMMMove. All of these games are draw-free in the appropriate range of parameters (except standard Tic-Tac-Toe), and thus here they even fit the promise that the input to SMMMove defines a draw-free game. Of course, these special cases need not be as hard as SMMMove: for example, it is trivial to find a winning first move in the (n, k)-Clique game, since by symmetry of the game board all first moves are equivalent. (Perhaps a more interesting version of the Clique game computational problem is to determine the first player's optimal move on their second turn, since the game necessarily still retains the symmetry needed for a strategy-stealing argument after each player claims only one edge.)

Hardness for SMMMove. We now show computational hardness for SMMMove. We first outline the proof ideas, and then provide a full proof. Our starting point is the following result from prior work:

Definition 2.1 (DecisionHex). The problem DecisionHex is defined as follows:

- Input: a partially-filled Hex board Q

- Output: does Red have a winning strategy in the Hex game starting from Q (assuming it is currently Red's turn to play)?

Theorem 2.2 ([Rei81]). DecisionHex *is* PSPACE-*complete.*

Our goal is to reduce DecisionHex to SMMMove. First, let us remark on why Theorem 2.2 does not *directly* give hardness for SMMMove, given that Hex is an SMM game. The result that DecisionHex is hard means that there exist families of positions from which deciding the winner is hard. However, these positions are not generally symmetric, so the game starting from these positions is not SMM. Additionally, a talented player *playing from the starting position* could still potentially be able to win the game while avoiding these hard settings of the game board, thus winning without ever really encountering a PSPACE-complete problem.

So, we are given a partially filled (possibly asymmetric) board Q on input to DecisionHex, representing a Hex game between Red and Blue where it is Red's turn to move, and our goal is to create a new SMM draw-free game between players First and Second that captures the structure of Q in some useful way. To build intuition, let us start with a first (incorrect) attempt at such a game G. Suppose we add two new elements to the universe called r and b. The idea will be that claiming r is choosing to play as Red in the Hex game defined by Q, and claiming b is choosing to playing as Blue. More formally, the winning sets $W_1 = W_2$ of the new SMM game would be:

- r and any set of hexagons that complete a win for Red in Q,

- b and any set of hexagons that complete a win for Blue in Q, and

- $\{r, b\}$.

If SMMMOVE(G) $= r$, this solves Q: the second player must claim b with their next move to block the winning set $\{r, b\}$, and then the game reduces to Q itself where First plays as Red and Second plays as Blue. Thus, if r is a winning move for First, then Q is a win for Red. Unfortunately, the other cases of the proof break down. For example: suppose the position on the board Q is such that whoever has the next move wins (so Q is a win for Red). Then it is winning for First to claim *either* r or b with their first move, meaning the output of SMMMOVE(G) is not very informative. Our fix is, intuitively, to amplify the game to avoid the possibility that the game winner depends on the turn order.

Proof of Theorem 1.5. We will prove that SMMMOVE is hard by reducing DECISIONHEX to it. Let Q an instance of DECISIONHEX. We will construct a Symmetric Maker-Maker game G which is draw-free and $W_1 = W_2$, such that finding a winning move in G allows us to find who has a winning strategy in Q.

The universe of our new game G will contain *two* identical copies Q_1, Q_2 of the input to DECISIONHEX, as well as new elements r, b like before. The winning sets $W_1 = W_2$ in G are (see Figure 3):

- r and any set of hexagons that completes a win for Red in *either* Q_1 or Q_2,

- b and any set of hexagons that completes a win for Blue in *both* Q_1 and Q_2, and

- $\{r, b\}$.
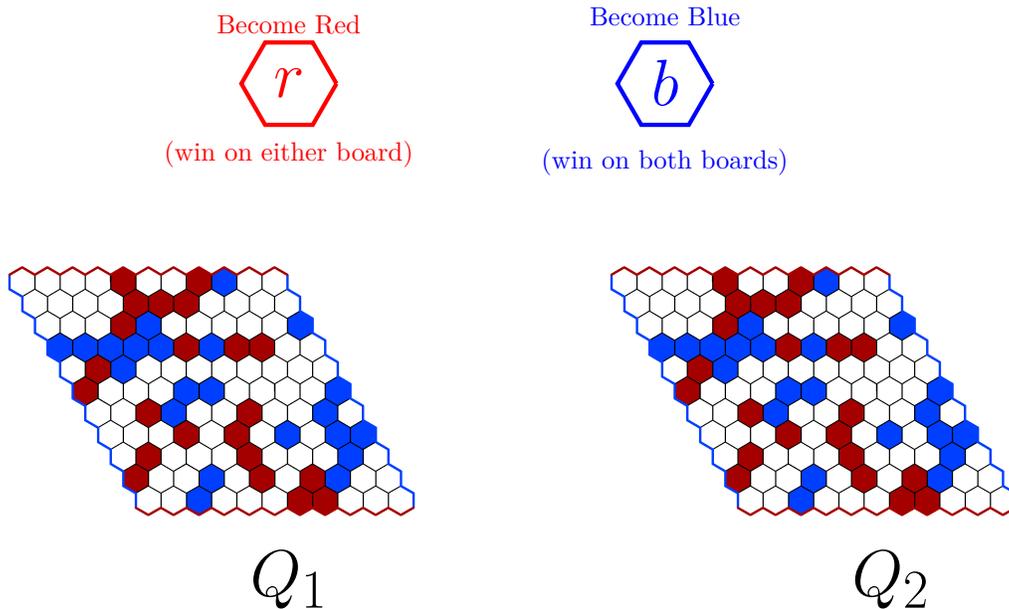
Figure 3: The universe used in our definition of an SMM game G.

It is immediate that G is an SMM game, since the winning sets are identical. Additionally, we have:

Lemma 2.3. *G is draw-free.*

Proof. Let S be any subset of the universe in G. We will show that either S or its complement S^C contains a winning set. First, if $r, b \in S$ then $\{r, b\} \subseteq S$, or if $r, b \notin S$ then $\{r, b\} \subseteq S^C$. So the nontrivial case is when S contains exactly one of r, b; let us assume without loss of generality that $r \in S, b \notin S$ (else switch the roles of S and S^C). For either board Q_i, by the Hex Theorem and the fact that the union of S_i, S_i^C covers the board, exactly one of the following two statements hold:

1. The elements of S on the board Q_i (call this S_i), combined with the elements on Q_i which are initially marked red, form a winning configuration for red on Q_i.

2. The elements of S^C on the board Q_i (call this set S_i^C), combined with the elements on Q_i which are initially marked blue, form a winning configuration for blue on Q_i.

Therefore, we conclude that in each board Q_i, either the elements of S complete a win for Red (we call Q_i a "red" board in this case), or the elements of S^C complete a win for Blue (we call Q_i a "blue" board in this case). If at least one of Q_1, Q_2 is red, then S contains r and a Red winning set. If both Q_1, Q_2 are blue, then S^C contains b and a Blue winning set on each board. In either case the lemma holds. □

Our goal is now to show that the winning move(s) for First in G completely determine the winner of Q. We consider two cases:

Lemma 2.4. *If Blue has a winning strategy in Q, then the unique winning move for First in G is to claim b.*

Proof. We first show that claiming b is a winning move for First. In response, Second is forced to claim r to block the winning set $\{r, b\}$. First then claims an arbitrary hexagon, and then each time Second claims a hexagon on Q_1 or Q_2, First claims a hexagon on the same board to execute a winning strategy for Blue. Thus, First will have b and also a winning set for Blue on both boards, meaning First wins in G. (Note that Second will be unable to ever obtain a red winning set on either board, since it is not possible for both sides to obtain a winning configuration on any individual Hex board.)

We then show that, if First does not claim b with their first move, then it is a winning response for Second to claim b. Here we consider two cases. If First claims r, then after Second claims b, in each subsequent turn, each time First claims a hexagon on Q_1 or Q_2, Second can claim a hexagon on the same board to execute a winning strategy for Blue on that board, thus obtaining a winning set for Blue on both boards and hence winning in G. In the other case, if First claims a hexagon in (say) Q_1 with their first move, then after Second claims b, First must immediately claim r to block the winning set $\{r, b\}$. Second then claims a hexagon on Q_1 and from here this case reduces to the first one. □

Lemma 2.5. *If Red has a winning strategy in Q, then it is not a winning move for First in G to claim b.*

Proof. Suppose that First claims b. The winning response for Second is to claim r. Without loss of generality, First then claims a hexagon in Q_1. Second then decides to permanently ignore Q_1 and focus entirely on Q_2, claiming exclusively hexagons in Q_2 for the rest of the game. Since Second is the first to move on Q_2, they can execute a winning strategy for Red on Q_2. Thus Second will eventually claim r and a winning set for Red on Q_2, meaning that Second wins in G. (Note, again, that First cannot possibly obtain a winning set in the meantime, since they cannot possibly hold a winning set for Blue on Q_2.) □

We now put the pieces together: after constructing the game G as described above, from Lemmas 2.4 and 2.5 we have

$$\text{SMMMOVE}(G) = b \qquad \text{if and only if} \qquad \neg\text{DECISIONHEX}(Q).$$

Since DECISIONHEX is PSPACE-complete, it follows that SMMMOVE is PSPACE-hard. □

2.2 Minimum Poset Games

Next, we prove Theorem 1.8. Our starting point is:

Definition 2.6 (DECISIONPOSET)**.** The problem DECISIONPOSET is defined as follows:

- Input: a poset P, described by explicitly listing its elements and the relations between them.

- Output: is the poset game (see Definition 1.6) associated to P a win for the first player under best play?

Theorem 2.7 ([Gri13])**.** DECISIONPOSET *is* PSPACE-*complete.*

We then argue:

Proof of Theorem 1.8. Given a poset game defined by P, generate a new poset P' by adding a new element m, defined to be less than every other element in P. We now argue that $\text{MPMOVE}(P') = m$ if and only if the original poset game defined by P was a win for the second player:

- Suppose P is a win for the first player. If in P' the first player claims m with their first move, then the game becomes equivalent to P with the turn order reversed. Thus claiming m is a losing move for the first player, and so $\text{MPMOVE}(P') \neq m$.

- Suppose P is a win for the second player. If in P' the first player claims m with their first move, then again the game is equivalent to P with the turn order reversed, so the first player has a winning strategy. This means we *can* have $\text{MPMOVE}(P') = m$, but since MPMOVE might return any winning move, we also need to rule out the possibility that any other move is winning. For this, we observe that any other first move in P' necessarily removes m and at least one other element from P', thus giving a position that can possibly be obtained after one move in P. Since P is a win for the second player, any such position must be losing for the player who creates it, and thus we have $\text{MPMOVE}(P') \notin P' \setminus \{m\}$, so $\text{MPMOVE}(P') = m$.

This completes the reduction from DECISIONPOSET to MPMOVE, and thus MPMOVE is PSPACE-hard. $\square$

3 Open Questions

We conclude by listing some conceptual open questions left by this work.

TFPSPACE. Can we similarly analyze the computational properties of other interesting non-constructive proof techniques that lie outside of NP? Is there a satisfying theory of TFPSPACE, in analogy with TFNP?

Bounded Computational Power. The existence of a winning strategy in a game does not necessarily shed much light on how *computationally bounded* players would play the game. To illustrate, consider the following game: player 1 declares a circuit C of their choice. Then, player 2 wins if they can declare an input x such that $C(x) = 1$. Then, player 1 then wins if *they* can declare an input x with $C(x) = 1$. If both players fail to declare such an input x, then player 2 wins. Here, there is clearly a winning strategy for the second player: if there exists an x such that $C(x) = 1$, then declare that x and win immediately; if there is no such x then player 2 also wins. However, if the players are represented by Turing machines that can only run for a polynomial amount of time, then the game is (probably) a win for player 1: for example, player 1 can pick a one way function f, compute it on some random x' of his choice to get output y, and then have the circuit C output 1 on all x such that $f(x) = y$. Under standard cryptography assumptions, player 2 will be unable to find such an x, and then player 1 will win in the next turn by declaring the x used to create the circuit.

Interestingly, for draw-free SMM games, such situations will not arise: even for computationally bounded players, it is preferable to play first, since playing an extra move is never disadvantageous. In contrast, this is not clearly true for poset games, where the wrong first move can possibly throw the game. Hence, this might be an interesting avenue to separate the computational properties of these two strategy-stealing arguments (since they are both PSPACE-hard under "best play," i.e., unbounded computational power). More generally, it would be interesting to further understand and formalize the effects of bounded computational power on various existence proofs for winning strategies in combinatorial game theory.

Generalized Strategy-Stealing. Many strategy-stealing arguments can be viewed as a reduction of the game tree to a tautology. To illustrate, the game tree of Chomp may be phrased as follows. Let X be the subgame tree from the starting chocolate bar with the bottom-left-most square removed. The proof of Theorem 1.2 essentially observes that the Chomp game tree is equal to X or $\neg X$, which is true as a formula (meaning a win for the first player) regardless of the value of X (see Figure 4).

Starting Position

(chomp bot-left square only)

(all other possible first moves)

X

X
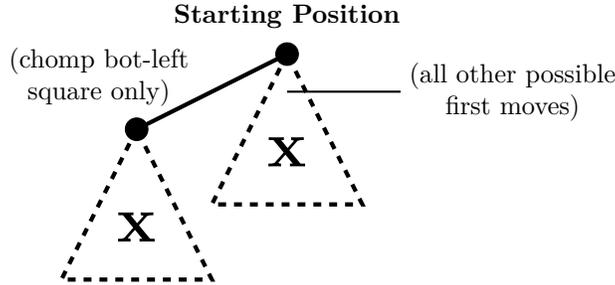
Figure 4: Due to the symmetry in the game tree of Chomp illustrated here, the value of the game tree can be expressed as X or $\neg X$, where X is the game tree after a 1×1 square has been chomped.

Naturally, a reduction of the game tree to *any* tautology implies a first-player win, including more complicated tautologies in which multiple variables are assigned to multiple subgames. We might call this type of argument *generalized strategy-stealing*, as it extends the usual proofs in the literature that use only X or $\neg X$. For any given tautology it is easy enough to invent an artificial game that admits a generalized strategy-stealing proof via that tautology. However, it would be interesting to find a "natural" game that admits a generalized strategy-stealing proof, using a tautology formally distinct from X or $\neg X$.

Hardness for Specific Games. While we have proved hardness for finding winning moves in game classes that include Hex and Chomp, our results do not imply hardness for Hex and Chomp specifically. In particular, it would be interesting to determine whether or not the following problem is in FP: given game board dimensions for Hex or Chomp (or basically any other game mentioned in this paper), written in unary, output a winning move for the first player. This problem is in PSPACE, but it will not be readily possible to prove it PSPACE-hard for the following reason: it is known that no unary language can be NP-complete unless P = NP; thus, a unary language complete for PSPACE would imply that P = NP or NP $\neq$ PSPACE, which is not known and would constitute a breakthrough in complexity theory. Thus it is unclear what hardness notion should be used to approach this question.

Other Notions of Constructiveness. We have proved that it is generally hard to find a winning move in a game, even when strategy-stealing arguments apply. Finding a good first move is one natural formalization of "constructiveness" in PSPACE, but there are others. For example, here is an open question that we have not addressed: for (say) the game Hex, does there necessarily exist a polynomial-size circuit that plays the game optimally, even if it is computationally hard to find the circuit?

References

[Bec81] József Beck. Van der Waerden and Ramsey type games. *Combinatorica*, 1(2):103–116, 1981.

[Bec02] József Beck. Positional games and the second moment method. *Combinatorica*, 22(2):169–216, 2002.

[Bec08] József Beck. *Combinatorial Games: Tic-Tac-Toe Theory*, volume 114. Cambridge University Press, 2008.

[Bys04] Jesper Makholm Byskov. Maker-maker and maker-breaker games are PSPACE-complete. *BRICS Report Series*, 11(14), 2004.

[ES73] Paul Erdös and John L Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.

[Geb12]   Heidi Gebauer. On the clique-game. *European Journal of Combinatorics*, 33(1):8–19, 2012.

[GH02]   Solomon W Golomb and Alfred W Hales. Hypercube tic-tac-toe. *More Games of No Chance*, 42:167–180, 2002.

[Gri13]   Daniel Grier. Deciding the winner of an arbitrary finite poset game is PSPACE-Complete. In *International Colloquium on Automata, Languages, and Programming*, pages 497–503. Springer, 2013.

[hex]   Wining situation on a Hex Board. https://en.wikipedia.org/wiki/Hex_(board_game)#/media/File:Hex-board-11x11-(2).jpg.

[HJ09]   Alfred W Hales and Robert I Jewett. Regularity and positional games. In *Classic Papers in Combinatorics*, pages 320–327. Springer, 2009.

[HKSS14] Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. *Positional Games*. Springer, 2014.

[KRSS17] Christopher Kusch, Juanjo Rué, Christoph Spiegel, and Tibor Szabó. Random strategies are nearly optimal for generalized Van der Waerden games. *Electronic Notes in Discrete Mathematics*, 61:789–795, 2017.

[Kus17]   Christopher Kusch. *Problems in Positional Games and Extremal Combinatorics*. PhD thesis, 2017.

[MP91]   Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[Nas52]   John F Nash. Some games and machines for playing them. 1952.

[Rei81]   Stefan Reisch. Hex ist PSPACE-Vollständig. *Acta Informatica*, 15(2):167–191, 1981.

[vdW27]   Bartel van der Waerden. Beweis einer baudetschen vermutung. *Nieuw Arch. Wisk.*, 19:212–216, 1927.