Electronic Preprint for Journal of Information Processing Vol.23 No.3

Regular Paper

You Should Be Scared of German Ghost

Erik D. Demaine[1,a)] Fermi Ma[1,b)] Matthew Susskind[1,c)] Erik Waingarten[1,d)]

Received: August 1, 2014, Accepted: January 7, 2015

Abstract: Ghost is a popular word game played by two or more players. Players take turns adding a letter to the end of a growing word fragment, trying not to be the first to complete a valid word. We show that the game, when played on a regular language, is PSPACE-hard, and extend the result to four variants of the game. In addition, we take advantage of a quirk of the German language — that German words can be concatenated together to form longer words — to give a fun extension of our proof of PSPACE-hardness to subsets of the German language.

Keywords: algorithmic combinatorial game theory, mathematical games and puzzles, computational complexity

1. Introduction

Ghost is a written or spoken word game, often played with no materials or preparation. As with many spoken word games, it is commonly played on long road trips, and its difficulty (and entertainment value) relies heavily on a player's skill and vocabulary. The game is quite popular, having been referenced in several American television shows, such as a 1962 episode of *Car 54, Where Are You?* and a 2003 episode of *Buffy The Vampire Slayer* [13]. It has also appeared in the January 10, 1974 *Doonesbury* comic strip [12], Chapter 7 of *The Long Walk* by Richard Bachman (a pseudonym for Stephen King, 1979) [4], and *The Deer Park* by Norman Mailer (1955) [6].

The dictionary is generally assumed to be English, and with a perfect knowledge of vocabulary the game is solved: the game tree of all words in the dictionary can be searched for a winning strategy in polynomial time with a minimax algorithm. Randall Munroe of *xkcd* used this algorithm to find winning strategies for Ghost played with the Ubuntu dictionary [3], [7]. More generally, the problem is uninteresting from an algorithmic standpoint when the input size is equal to the dictionary size.

This motivates our analysis of Ghost with dictionaries that can be expressed compactly. In particular, we focus on Ghost played over regular languages, as the equivalence with regular expressions allows such languages to be expressed with a small input size.

We analyze Ghost as well as four popular variants—Superghost, Superduperghost, Xghost, and Spook—as defined in Section 2. In Section 3, we show that the classic version of the game played on regular languages is PSPACE-hard and is in EXPSPACE. In Section 4, we extend the PSPACE-hardness and EXPSPACE result to Superghost, Superduperghost, and Xghost. We then use a different approach to show that Spook is PSPACE-hard, though it remains open what complexity class Spook is in. In Section 5, we consider the game played over subsets of the German language and show that it is PSPACE-hard. We abuse the fact that, in German, valid words are often formed by merely concatenating other words together, a feature that drastically impacts play of games such as Ghost. We conclude with open problems in Section 6.

[1] MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA
[a)] edemaine@mit.edu
[b)] fermima@mit.edu
[c)] msuss@mit.edu
[d)] eaw@mit.edu

2. Game Definitions

2.1 Classic Game Rules

The classic game is played with two or more players in a round format. A round begins with one player naming a letter of the alphabet to start a word. Gameplay continues with players taking turns adding letters to the end of the growing fragment, under the condition that at all times the fragment must be the valid prefix of some word.

If a player suspects that the most recent move violates this condition, s/he can challenge the player who played last to name the full word. If the challenged player can name the word, s/he wins and the challenger loses the round. Otherwise, the challenged player loses the round. If no challenges occur, play continues until one player has spelled out a valid word, at which point that player loses the round. When a player loses a round, s/he picks up a letter of the word G-H-O-S-T, and a player loses the game after picking up five letters (as in the basketball game of H-O-R-S-E). The last player to be eliminated is the winner. Often, words with four letters or fewer are ignored.

2.2 Our Assumptions

We restrict our analysis of the game to the case where the dictionary is a regular language. As mentioned earlier, regular expressions allow us to consider large (possibly infinite) languages that can be specified by a short input.

Given this restriction, we can assume that each player has perfect knowledge of the game dictionary; given the regular expression, a player can check efficiently whether a string is a valid word or the prefix of a valid word. This eliminates the need for challenges in our analysis.

We will focus our attention on the two-player version of the game, and for simplicity we will assume that only one round of the game is played.

2.3 Variants

We consider all variants of Ghost mentioned on Wikipedia [13]. For clarity, we will sometimes refer to the game played under the original set of rules as *classic Ghost*.

- **Superghost**: This is an extension of classic Ghost where players now have the ability to add letters to the front of the word fragment [2], [10].
- **Superduperghost**: This is an extension of Superghost where players may now reverse the word fragment before playing a letter [1].
- **Xghost**: This is an extension of classic Ghost where players are now allowed to add letters before or after any other letter in the word fragment [2].
- **Spook**: This variant works like classic Ghost, but the order of the letters in the word fragment is independent of the order in which letters were played. Thus, players add letters to a set of letters, trying to avoid forming a set such that a permutation of those letters would form a valid word. Just as a word fragment in Ghost must be a valid prefix of some word, the sets in Spook must be a valid subset of the letters of some word.

3. Hardness of Classic Ghost

In this section, we analyze Ghost played over a regular language. Formally, the problem can be defined as follows:

Instance: A regular expression R.

Question: Players 1 and 2 take turns playing Ghost with the language generated by R. If player 1 starts, does s/he have a winning strategy?

Theorem 1. *The problem of determining whether player 1 has a winning strategy in Ghost played over regular languages is in EXPSPACE.*

Proof. First, we argue that we only need to analyze deterministic winning strategies. These are strategies that, given any move by the opposing player, specify exactly one move. Suppose we have a winning nondeterministic strategy. Every time the strategy specifies multiple winning moves (moves that are part of a winning strategy), we can simply play the character that comes first in the lexicographic order of the choices. This new strategy is deterministic and is as good as the nondeterministic one.

Let the regular expression have length n. We can build a deterministic finite automata (DFA) with $2^{p(n)}$ states that recognizes the corresponding language, where $p(n)$ is a polynomial in n [9]. We claim that a player playing with a winning strategy makes at most $2^{p(n)}$ moves.

To see why, suppose the winning strategy takes more than $2^{p(n)}$ moves. Then in playing the game, there exists a state of the DFA that is traversed twice (by the pigeonhole principle), and thus the gameplay goes through at least one loop. Suppose the loop length is even, and we are at the stage where the players have gone through the loop once. The losing player (the one without the winning strategy) can then play the same moves that caused the players to go through the loop the first time, and since the winning player's strategy is deterministic, they will in fact go through the loop. The losing player can do this infinitely many times to avoid losing, which is a contradiction.

Now suppose the loop length is odd. Once the loop is traversed once, the positions of the players are switched. In particular, the losing player is now at the exact spot the winning player was in before they went through the loop. By assumption, there was a winning strategy at that spot, but this means the losing player now has a winning strategy, which is a contradiction. This means that if there exists a winning strategy, every game played using this winning strategy takes at most exponentially many turns. So the specification of the winning strategy is at most exponentially long.

This bound on the winning strategy implies that we can solve the game by expressing it as an exponentially-long fully-quantified boolean formula where variables represent choices for the two players. The formula simply needs to check whether pairs of letters are said correctly and whether or not player 2 was the first to complete a word [9]. If the fully-quantified boolean formula is true, then it specifies a winning strategy for player 1. If the formula is not true, then player 1 does not have an at most exponentially long winning strategy, and thus has no winning strategy. □

Theorem 2. *Determining whether player 1 has a winning strategy when Ghost is played over regular languages is PSPACE-hard.*

Proof overview. The proof is by a reduction from Generalized Geography, a problem known to be PSPACE-hard [9]. Recall that the problem is defined as follows:

Instance: A directed graph G with a token placed on an initial node s.

Question: Players 1 and 2 take turns moving the token from the current node to an adjacent node along directed edges in G, where player 1 starts with the token at s. A player loses when s/he is unable to move the token to a node that did not have the token previously. Does player 1 have a winning strategy?

We will use the graph G to build a nondeterministic finite automaton (NFA) that will give a regular expression to play Ghost with. This construction will give an equivalence between games played in Generalized Geography and Ghost, so a winning strategy in one will correspond to a winning strategy in the other.

Proof. We give each edge in G a distinct, arbitrary label. We then build the NFA by "redrawing" the graph G, with its labels, as a diagram for the NFA. This means that vertices of G are drawn as states of the NFA, and the directed edges become NFA transitions which inherit labels from the corresponding edges in G.

We then extend the construction: for each node $x \in G$, we make another copy of G called G_x, and make it part of the NFA. For any directed edge (x, a) pointing away from x with label l, we draw a transition from x to a' in G_x with label l, where the prime denotes a corresponding vertex in G_x. In addition, we make $x' \in G_x$ the only accept state among the states in G_x. If $x \in G$ has no outgoing edges, the corresponding x state in the NFA will have a transition, labeled with all possible labels, to an accept state y.

An example of the construction is given in **Fig. 1** and **Fig. 2**. In Fig. 2, the solid transitions correspond to the transitions from G and the dashed transitions correspond to the transitions to the copies of G.

From this NFA, we can generate an equivalent regular expression in polynomial time [9].

We claim that a winning strategy for Generalized Geography corresponds to a winning strategy for Ghost, played on the language generated by the corresponding NFA. Each move a player makes in Generalized Geography corresponds to reading a label in the NFA. Once a player reaches a node in Generalized Geography that has no usable outgoing edges (and has thus won), the opposing player in the corresponding Ghost game is forced to move to an accept state in the following move, which spells out a word and loses the game for that player.

For the other direction of the equivalence, a winning strategy in Ghost (played on the language of the NFA that corresponds to the instance of Generalized Geography) gives a winning strategy in Generalized Geography. The winning strategy in Ghost played on the NFA is to follow valid edges and to avoid ever getting to the accept state of some G_x, which corresponds to going back to the same vertex in Generalized Geography. □

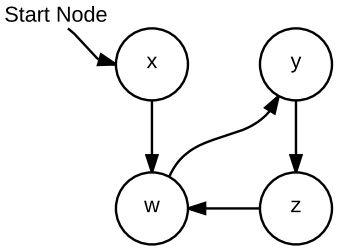

Fig. 1 Instance of Generalized Geography.

4. Variants

Membership in EXPSPACE as well as PSPACE-hardness of Ghost played on regular languages extends to Superghost, Superduperghost, and Xghost, played on regular languages. Recall that these variants merely extend the rules of the classic game: Superghost allows players to also add letters to the beginning of the string, Superduperghost extends Superghost so that players can reverse a word before playing a letter, and Xghost allows players to play a letter anywhere in the word fragment (but does not allow word reversal). We show that for each of these three variants, membership in EXPSPACE follows easily from the fact that Ghost played on regular languages is in EXPSPACE. We then demonstrate ways to simulate classic Ghost played on a regular language within each of these variants, from which PSPACE-hardness will follow.

Recall that in Spook, players add letters to a set of letters, trying to avoid forming a set such that there is a permutation of those letters that forms a valid word. We will show PSPACE-hardness by reduction from Generalized Geography. However, we are unable to show membership in EXPSPACE for Spook.

Corollary 1. *The problem of determining whether player 1 has a winning strategy in Superghost, Superduperghost, and Xghost played on regular languages is in EXPSPACE.*

Proof. We note that we can still reduce the problem to the satisfiability of a fully-quantified boolean formula. We use the same formula as in the proof of Theorem 1, except that now at each player's turn, there are additional variables specifying which position the letter was played at, and/or whether the word fragment was reversed, depending on the variant under consideration. The number of turns is exponential for a similar reason that the number of turns was exponential in classic Ghost. The additional capabilities in the variants add shortcut edges in the DFA, and going

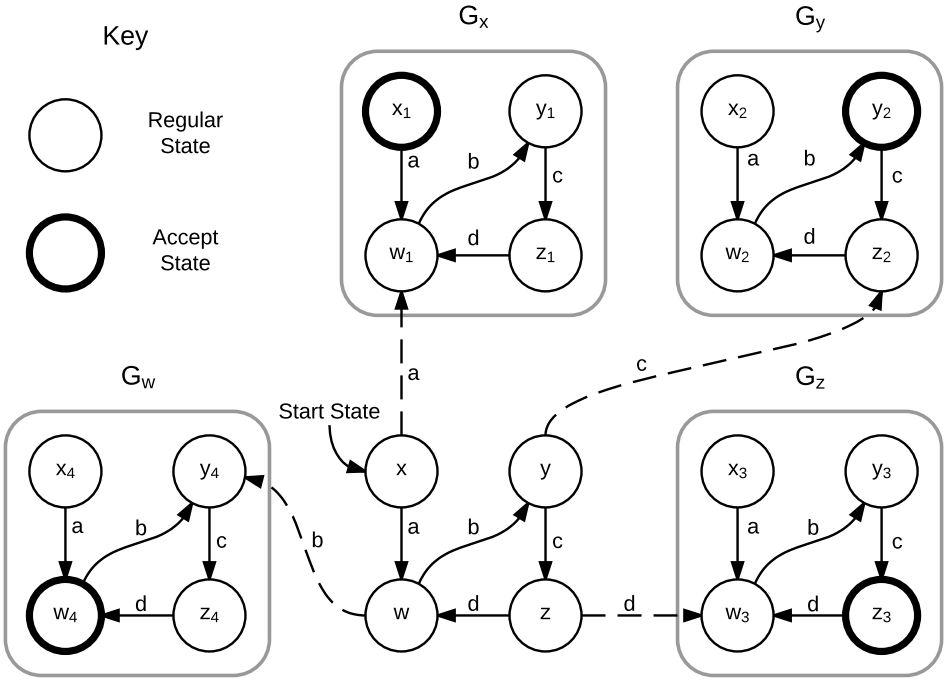

Fig. 2 Corresponding NFA.

through a loop in the DFA is never in a player's winning strategy. There is a polynomially-sized multiplicative overhead for the formula length, which means the formula is still exponentially-sized. □

Corollary 2. *Determining whether player 1 has a winning strategy when Superghost is played over regular languages is PSPACE-hard.*

Proof. We simulate playing Ghost over a given regular language with corresponding regular expression R. Given R, we generate a regular expression R_1 to play Superghost on.

We form R_1 by taking R and prepending the new symbol # twice at the front. This way, no words have # in the middle, so a player cannot place a letter at the begining of the word. Because we have added two symbols at the begining, we have maintained the parity of the game. Note that this forces the # symbol to be played twice at the start of every game.

Playing Superghost on R_1 simulates playing Ghost on R, so the proof follows. □

Corollary 3. *Determining whether player 1 has a winning strategy when Superduperghost is played over regular languages is PSPACE-hard.*

Proof. The construction here is identical to the one in the proof of Corollary 2; if we ever reverse a string, then the two # symbols will be at the end and we will not have a valid prefix.

This effectively blocks the ability to reverse a word, so playing Superduperghost on R_1 simulates playing Ghost on R. □

Corollary 4. *Determining whether player 1 has a winning strategy when Xghost is played over regular languages is PSPACE-hard.*

Proof. The technique follows the strategy used in the proof of Corollary 2. Given a regular expression R, we generate a regular expression R_2 to play Xghost on.

We introduce three new symbols, @, %, and &. To construct R_2, we replace each letter a in R with the three concatenated letters $\%a\&$. In addition, prepend @ to the beginning of the expression. For example, if $R = a(b^* \cup cd)$, then

$$R_2 = @(\%a\&)\Big((\%b\&)^* \cup (\%c\&)(\%d\&)\Big)$$

The @ guarantees that we never prepend anything to the beginning of the word and the % and & guarantee that there are never any letters added in the middle. Also, because each letter is replaced by three letters, we maintain the parity of the game.

In summary, this construction replaces each letter of R with three letters in R_2 in a way that forces players to obey the rules of classic Ghost while playing Xghost. □

Our hardness result for Spook uses a wholly different technique. The fact that order does not matter in Spook offers players significant freedom, which makes it difficult to simulate a classic game of Ghost over a regular language. We note that the following proof, with slight modifications, will work to show PSPACE-hardness for Superghost, Superduperghost, and Xghost. However, the previous results generalize cleanly to subsets of the German language, as we will show in Section 5.

Theorem 3. *Determining whether player 1 has a winning strategy when Spook is played over regular languages is PSPACE-hard.*

Proof overview. The proof is by a reduction from Generalized Geography. Given an instance of the problem, (G, s), we will build a regular expression whose language is exactly the set of words that represent an *incorrect* play history for Generalized Geography. The idea is that the set of letters will always be a subset of a set of letters that can be permuted to form a word in the language. Because the language contains all incorrect play histories of Generalized Geography, the gameplay of Spook will involve players forming *valid* play histories of Generalized Geography. Only when a player is stuck in Generalized Geography will s/he be forced to form a set that is not a valid play history, losing the game of Spook.

Proof. Given $G = (V, E)$ where $|V| = n$, we will have a letter v_i for each $v \in V$ and $i \in [n]$. The presence of the letter v_i in the word fragment will indicate that the token in Generalized Geography was moved to v at turn i. Let $\Sigma = \{v_i | v \in V, i \in [n]\}$. The regular expression will contain three parts:

(1)

$$R_a = \bigcup_{v \in V} \bigcup_{i,j \in [n]} (v_i v_j \Sigma^*)$$

This checks that each node in V is only visited once.

(2)

$$R_b = \bigcup_{v,w \in V, (v,w) \notin E} \bigcup_{i \in [n-1]} (v_i w_{i+1} \Sigma^*)$$

This checks that only nodes with edges leading to them are visited.

(3)

$$R_c = \bigcup_{v \in V} \bigcup_{i \in [n] \setminus \{1\}} (v_i (\Sigma - \cup_{x \in V} x_{i-1})^*)$$

This checks that nodes are visited in the correct order.

Thus, the expression

$$R_a \cup R_b \cup R_c$$

generates the language of all invalid play histories of Generalized Geography.

This will ensure that the letters chosen in a game of Spook can be arranged in a line that represents a valid play history of Generalized Geography. If player 1 can force a win in the Spook game, this corresponds to player 2 being stuck in the corresponding Generalized Geography game. Note that the expression is polynomial in length and can be written down in polynomial time. The proof follows. □

5. Playing on Subsets of German

In this section, we see how our analysis can be extended to a real human language. As discussed in Section 2, a real language is assumed to be specified with its exact dictionary, and thus we still avoid that case here. However, we can make claims about subsets of human languages specified by regular expressions, and we focus on the German language in particular because of its rules regarding word concatenation.

5.1 Concatenating Words

The idea behind the proof of Theorem 1 can be used to show that Ghost, played on subsets of the German language specified by regular expressions, is PSPACE-hard. We focus on the German language because it allows for compound words formed by concatenating shorter words. For example, "wort," "band," and "teil" are all valid German nouns, and thus concatenations such as "wortteil" and "bandteil" are also valid nouns [11].

This concatenation process often leads to nonsensical words, but for the sake of mathematical analysis, we assume a relaxed form of the standard concatenation rules of German. This allows us to describe subsets of the German language with regular expressions by having the characters of the regular expression be complete German words.

For example, the regular expression

$$\text{teil}(\text{wort} \cup \text{band})\text{teil}$$

describes the regular language {teilwortteil, teilbandteil}.

5.2 PSPACE-hardness

We consider the following special case of Ghost, called German Ghost:

Instance: A regular expression R where the language generated by R consists of valid German words.

Question: Players 1 and 2 take turns playing Ghost with the language generated by R. If player 1 starts, does s/he have a winning strategy?

Theorem 4. *The problem of determining whether player 1 has a winning strategy in Ghost played over subsets of the German language is PSPACE-hard.*

Proof. We use a reduction similar to the one in the proof of Theorem 2. However, here we consider Planar Generalized Geography, which is still PSPACE-hard [5]. Given an instance of Planar Generalized Geography, we can four-color the graph in polynomial time [8]. Each edge is then labeled with the color of the vertex it points to. Given this labeling of the graph, we build a corresponding NFA following the procedure in the proof of Theorem 2. The only change we make is that the labels in the graph are mapped to German nouns of odd length that start with different letters. For the sake of concreteness, we choose the nouns "junge" (boy), "mädchen" (girl), "tag" (day), and "nacht" (night). Thus, all edges that are colored one color might give transitions labeled with "tag," for example.

Because we refer to concatenated nouns as valid words, the subset of words accepted by the NFA will be grammatically valid German words. Because the words are all of odd length, when the word finishes, the next player can pick the next word. This corresponds to picking an edge in the NFA, except each transition may take multiple turns. Because the first letter of all the words chosen above are different, starting a word specifies the whole word. This construction builds an NFA that accepts a subset of the words in the German language. □

Corollary 5. *Superghost, Superduperghost, and Xghost are all PSPACE-hard when played on subsets of the German language.*

Proof. The reduction in the proof of Theorem 4 can be easily modified to work with the reductions given for Superghost (Corollary 2) and Superduperghost (Corollary 3). The new symbol # just becomes another German word of odd length with a distinct first letter; for example, "sonne" (sun) will work.

The reduction for Xghost (Corollary 4) is the same as the one for classic Ghost, because if a letter is not added to the end of a word, the word is no longer a valid prefix. □

6. Conclusion

We determined that the problem of finding a winning strategy for Ghost and the variants Superghost, Superduperghost, and Xghost when played over regular languages is in EXPSPACE and is PSPACE-hard. We used the same reduction structure to show that finding a winning strategy for Ghost and these three variants, when played over subsets of the German language, is also PSPACE-hard.

We also considered a fourth variant called Spook, in which the order of the letters does not matter. We showed finding a winning strategy in Spook played over regular languages is PSPACE-hard, but we have not determined which complexity class Spook is a member of.

Many related questions remain open:

(1) Are the problems of finding winning strategies for Ghost, Superghost, Superduperghost, and Xghost, played over regular languages, in PSPACE?

(2) Is the problem of finding a winning strategy for Spook played over regular languages in PSPACE? Is it even decidable? Can we bound the number of turns in the winning strategy?

(3) It would be interesting to consider the same games played over other classes of languages. For example, does playing Ghost over a context-free language make finding a winning strategy harder? Is it in PSPACE?

(4) Finally, we were able to encode some subsets of the German language in terms of regular expressions. We would like to somehow encode subsets of the English language and show that finding a winning strategy in English is PSPACE-hard. It seems like such a goal would require a radically different approach, as word concatenation will not be available.

Acknowledgments This paper began as a final project for the MIT class "The Mathematics of Toys and Games", taught by Jayson Lynch and Robert Sloan and supervised by Erik D. Demaine. We thank them for helpful dicussions on this topic.

We would also like to thank the anonymous reviewers for their helpful suggestions and comments.

References

[1] DeKoven, B.: The game of GHOST, available from ⟨http://www.deepfun.com/2010/04/game-of-ghost.html⟩ (2010).
[2] Fader, A.: GHOST - The addictive word game, available from ⟨http://andre.facadecomputer.com/ghost/⟩.
[3] Frank, A.: Ghostbusters, *Word Ways*, Vol.20, No.4, p.4 (1987).
[4] King, S. and Bachman, R.: *The Long Walk*, Penguin Group US (1999).
[5] Lichtenstein, D. and Sipser, M.: GO Is Polynomial-Space Hard, *J. ACM*, Vol.27, No.2, pp.393–401 (online), DOI: 10.1145/322186.322201 (1980).
[6] Mailer, N.: *The Deer Park*, Putnam (1955).
[7] Munroe, R.: Ghost, available from ⟨http://blog.xkcd.com/2007/12/31/ghost/⟩ (2007).
[8] Robertson, N., Sanders, D.P., Seymour, P. and Thomas, R.: Efficiently

four-coloring planar graphs, *Proc. 28th Annual ACM Symposium on Theory of Computing*, pp.571–575, ACM (1996).
[9] Sipser, M.: *Introduction to the Theory of Computation*, Cengage Learning, 3rd edition (2012).
[10] Thurber, J.: Do You Want To Make Something Out of it?, Or, If you Put An "O" On "Understo", You'll Ruin My "Thunderstorm", *The New Yorker* (1959), available from ⟨http://archives.newyorker.com/?i=1951-09-29#folio=026⟩.
[11] Translation Dictionary: German, available from ⟨http://www.translationdirectory.com/article693.htm⟩ (2009).
[12] Trudeau, G.: Doonesbury Comic Strip, January 10, 1974 on GoComics.com, available from ⟨http://www.gocomics.com/doonesbury/1974/01/10⟩ (1974).
[13] Wikipedia: Ghost (game) — Wikipedia, The Free Encyclopedia, available from ⟨http://en.wikipedia.org/w/index.php?title=Ghost_(game)⟩ (2014).

Erik D. Demaine received a B.Sc. degree from Dalhousie University in 1995, and M.Math. and Ph.D. degrees from the University of Waterloo in 1996 and 2001, respectively. Since 2001, he has been a professor in computer science at the Massachusetts Institute of Technology. His research interests range throughout algorithms, from data structures for improving web searches to the geometry of understanding how proteins fold to the computational difficulty of playing games. In 2003, he received a MacArthur Fellowship as a "computational geometer tackling and solving difficult problems related to folding and bending—moving readily between the theoretical and the playful, with a keen eye to revealing the former in the latter". He cowrote a book about the theory of folding, together with Joseph O'Rourke (*Geometric Folding Algorithms*, 2007), and a book about the computational complexity of games, together with Robert Hearn (*Games, Puzzles, and Computation*, 2009).

Fermi Ma is an undergraduate in Mathematics at the Massachusetts Institute of Technology, working with Professor Erik D. Demaine. His research interests include computational complexity of games, algorithms, and data structures.

Matthew Susskind is an undergraduate student at the Massachusetts Institute of Technology, where he studies Computer Science. His research interests include theory of computation, machine learning, and game design.

Erik Waingarten is an undergraduate student at the Massachussetts Institute of Technology where he studies Mathematics. He is a member of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) where he works with Professor Erik Demaine. His research interests include theory of computation, algorithms and data structures.