Any Monotone Boolean Function Can Be Realized by Interlocked Polygons

Any Monotone Boolean Function Can Be Realized by Interlocked Polygons

Erik D. Demaine* Martin L. Demaine* Ryuhei Uehara†

Abstract

We show how to construct interlocked collections of simple polygons in the plane that fall apart upon removing certain combinations of pieces. Precisely, interior-disjoint simple planar polygons are *interlocked* if no subset can be separated arbitrarily far from the rest, moving each polygon as a rigid object as in a sliding-block puzzle. Removing a subset S of these polygons might keep them interlocked or *free* the polygons, allowing them to separate. Clearly freeing removal sets satisfy monotonicity: if $S \subseteq S'$ and removing S frees the polygons, then so does S'. In this paper, we show that any monotone Boolean function f on n variables can be described by $m > n$ interlocked polygons: n of the m polygons represent the n variables, and removing a subset of these n polygons frees the remaining polygons if and only if f is 1 when the corresponding variables are 1.

1 Introduction

Since Sam Loyd invented the famous 15 puzzle, sliding-block puzzles have played an important role in mathematical recreations. There are many variations of sliding-block puzzles (Figure 1), and they have been investigated widely; see, for example, [1]. Recently, a new framework involving games on graphs led to establishing PSPACE-completeness of many sliding-block and related puzzles [4]. In most of these puzzles, for a given initial state, we aim at finding a way to its goal state. Sometimes, the difficulty of the puzzles is changed if we change the initial state. For example, the 15 puzzle becomes much easier if we remove more than one of the sixteen possible pieces. In the puzzles in Figure 1, if we remove more pieces, the problems (getting out a specified car or disassembling the puzzle) become easier. More generally, in a sliding-block puzzle, removing pieces makes for an easier puzzle. In this paper, we investigate such monotonicity of sliding-block puzzles.

Suppose we have a collection of n simple polygons in the plane, none of which overlap each other (except along their boundaries). We call the polygons *interlocked* if no subset can be separated arbitrarily far from the rest. Otherwise, we call the polygons *free*. (Our results hold equally well if we define "free" to mean that every polygon can separate arbitrarily far from every other polygon; we will guarantee full interlocking or full freedom.)

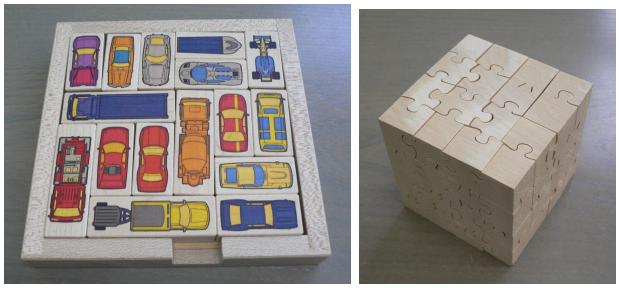

Figure 1: Typical sliding-block puzzles in 2D and 3D.

If we remove a subset S of the polygons, the remaining polygons might be interlocked or free. Define $\delta(S) = 1$ if the remaining polygons become free after removing S, and $\delta(S) = 0$ if they remain interlocked. Clearly f is monotone: if $S \subseteq S'$, then $\delta(S) \leq \delta(S')$. (Removing more makes the polygons less interlocked.)

A natural question arises: which monotone Boolean functions f can be described by n interlocked polygons? Figure 2 shows some simple examples. For a specified n and k, this technique can design n interlocked polygons such that removing any k of them makes them all free.

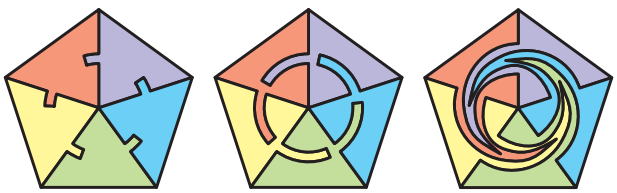

Figure 2: Interlocked polygons freed by removing any one, two, or three of the polygons (from left to right).

Monotone Boolean functions have long been investigated in computer science, especially in the context of the lower bound of circuit complexity; see, e.g., [5, Chapter 14.4]. In general, monotone Boolean functions are formed by AND and OR operations (without NOT). Thus we need to build AND and OR gates using interlocked polygons. Doing so, however, seems difficult

*Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, MA 02139, USA. {edemaine, mdemaine}@mit.edu

†School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan. uehara@jaist.ac.jp

without some extra pieces.

Thus our problem becomes the following: can any monotone Boolean functions f on n variables be described by $m > n$ interlocked polygons, where f operates on a particular subset of n polygons? In this paper, we give an affirmative answer to this question:

Theorem 1 *For any given monotone Boolean function f on n variables $x_1, x_2, \ldots, x_n$, there is a collection S of $m > n$ simple polygons such that (1) S is interlocked, (2) a subset $S' \subset S$ of n simple polygons correspond to the variables, and (3) $\delta(S'') = 1$ for $S'' \subseteq S'$ (removing the pieces in $S'' \subseteq S'$ frees the remaining polygons) if and only if $f(x_1, x_2, \ldots, x_n) = 1$, where x_i indicates whether $x_i \in S''$.*

This problem was originally motivated by analogy to a corresponding topological problem: design a braid of n strands, or a link of n components, that trivializes (and therefore the parts separate freely) only when removing certain subsets of the parts. Again any monotone Boolean function is possible [6, 2], and furthermore without the need for extra pieces.

2 Construction

The proof of the main theorem by the construction of the desired interlocked polygons for any monotone Boolean function f on n variables $x_1, x_2, \ldots, x_n$. Figure 3 gives an outline of the construction.

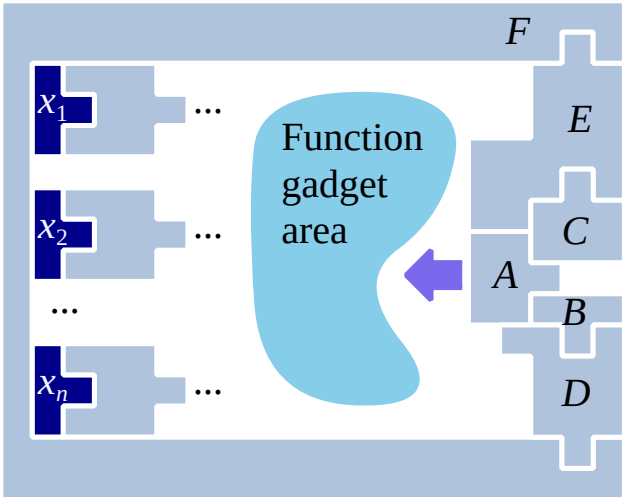

Figure 3: Outline of the construction.

The polygons B, C, D, E, and F form an *outer frame*. When polygon A is at the initial position as in the figure, these polygons are interlocked. However, once we slide A left, B can be slid up and right, and all of C, D and E can be removed in this order; then all remaining polygons can be moved out from F on the right side, so all polygons become free.

We construct all other gadgets inside of this frame. Each of the variables $x_1, x_2, \ldots, x_n$ corresponds to a polygon in S on the left of the frame, as depicted in Figure 3. If we remove the polygon x_i (we sometimes identify a variable/operator and the corresponding gadget), which is equivalent to setting $x_i = 1$, then the input wire can slide one unit to the left. Thus a true input wire effectively *pulls* the polygons one unit length.

To complete the construction of the gadgets that realize a monotone Boolean function f within the frame, we have to design six kinds of gadgets: (0) wire to transfer a signal, (1) AND gate for the conjunction of two signals, (2) OR gate for the disjunction of two signals, (3) SPLIT to duplicate a signal, (4) TURN to route a signal, and (5) CROSS to enable nonplanar routes. Then it is easy to assemble these gadgets to construct the output of any monotone Boolean function on the n inputs. (Technically, SPLIT is not necessary to achieve all monotone Boolean functions, but it enables efficient construction of monotone circuits with fanout.)

The WIRE, AND, and SPLIT gadgets are relatively simple to realize. In Figure 4(a), the operation (x_1 and x_2) is computed by the AND block, and the signal is sent to right by the WIRE blocks, and split by the SPLIT block. (In fact, SPLIT is essentially the same as AND, but backwards.)

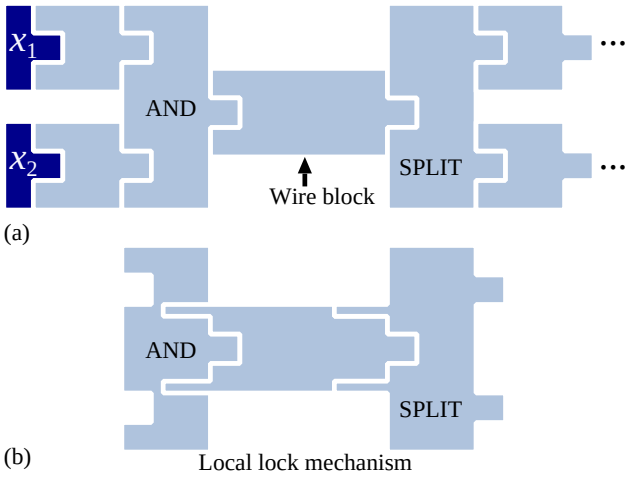

Figure 4: The gadgets for WIRE, AND, and SPLIT.

In each move, each block moves one unit length. One may wonder what happens if some blocks are moved out from our assumption. For example, the wire block between the x_1 and AND blocks in Figure 4(a) can be moved up after removing x_1. But we can avoid these unexpected cases: the blocks can be locked locally if we attach some long arms as shown in Figure 4(b). To simplify, we omit the arms in the other figures.

The OR gadget is more complicated, as shown in Figure 5. The initial position is depicted in Figure 5(a). If one of x_1 and x_2 moves one unit from the OR block,

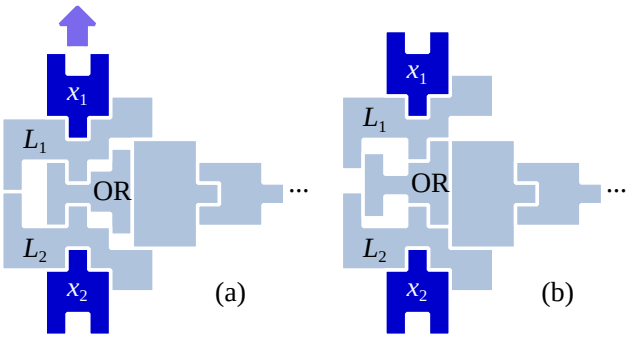

Figure 5: OR gadget.

then the OR block can move left as in Figure 5(b). (In the figure, x_1 moves up.) We note that, even if both x_1 and x_2 move far from the OR block, the OR block is still locked in with the L_1 and L_2 blocks. Otherwise, moving just x_1 might have some unintended influence on the variable x_2 and vice versa; intuitively, this mechanism prevents the signal from x_1 to bounce back into x_2 through this gadget.

Figure 6 shows the TURN gadget. The initial position is depicted in Figure 6(a). When the left block moves left, the blocks can move as in Figure 5(b).

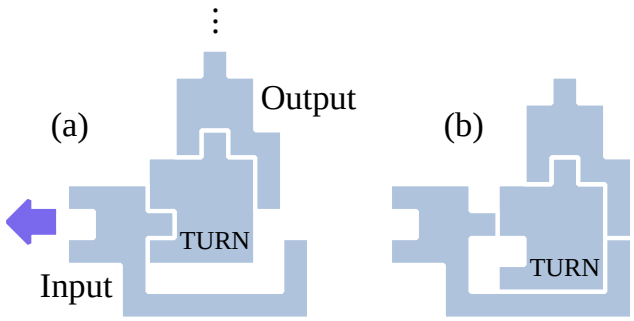

Figure 6: TURN gadget.

In many reductions, crossing wires is the key gadget; see, e.g., [4]. In this problem, the crossover gadget took a long time to find, but in the end is simple, as shown in Figure 7.

Using these gadgets, we can construct polygons to compute $f(x_1, x_2, \ldots, x_n)$, and connect the final output to the polygon A in Figure 3. Figure 8 shows a simple example of the construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$ (This example can be simplified logically, but serves for illustrative purposes.) Some blocks are stretched in a trivial way to adjust their size.

How many pieces does the construction use? Suppose we are given f as a monotone Boolean circuit with $w \geq n$ wires (edges). We start by converting the circuit to have bounded fan-in and fan-out, consisting of two-input one-output AND and OR gates and

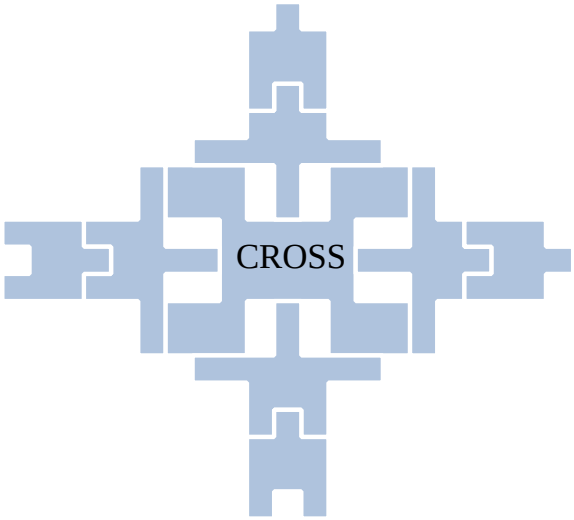

Figure 7: CROSS gadget.

one-input two-output SPLIT gates. This conversion increases the number of gates to $\Theta(w)$ by the Handshaking Lemma. Next we view the circuit as a maximum-degree-3 graph, and make the graph planar by adding degree-4 "crossover" vertices. This planarization increases the number of gates to at most $O(w^2)$. Now we use orthogonal graph drawing algorithms [3] to embed each edge as an orthogonal path with a constant number of bends, which become turn gadgets. All other vertices (AND, OR, SPLIT, and crossovers) become the corresponding gadgets. Each gadget consists of $O(1)$ pieces, so the total number of pieces is $O(w^2)$. The pieces can be drawn on a grid $O(1)$ times finer than the orthogonal graph drawing, so it is $O(w^2) \times O(w^2)$.

Proof of Theorem 1. Consider the construction desrcibed above for a monotone Boolean function f on n variables $x_1, x_2, \ldots, x_n$. Let α be any assignment of $x_1, x_2, \ldots, x_n$, and $S(\alpha) \subseteq S$ be the set of polygons x_i in S for which $\alpha(x_i) = 1$. When we remove the polygons in $S(\alpha)$ from S, the shift of wire gadgets propagates as described above. Then the remaining interlocked polygons are free if and only if $f(\alpha(x_1), \alpha(x_2), \ldots, \alpha(x_n)) = 1$. It is easy to see that the other conditions in the main theorem are satisfied by the reduction. □

3 Concluding remarks

In Figure 2, there is no space between the interlocked polygons. Although we can pad some extra polygons in Figure 8, some spaces around the gadgets and inside the gadgets cannot be padded while preserving the motion of the polygons. In particular, the spaces in the CROSS gadget seem to be essential. It would be interesting to determine what monotone Boolean functions can be

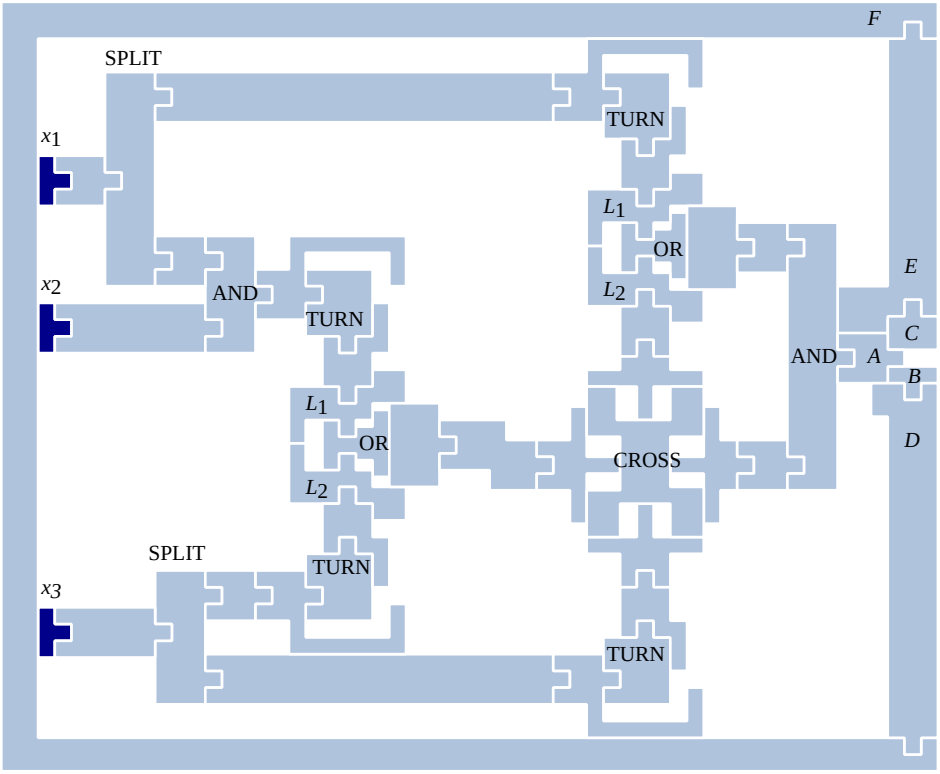

Figure 8: A construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$.

represented by interlocked polygons without any empty space (before polygon removal).

In our reduction, some gadgets can be glued together into one piece. Typically, a sequence of wire blocks can be replaced by one long block, and this long block can be glued to the output of the last gadget. These extra blocks serve their function, but they do not represent variables. It is an intriguing open problem whether we can construct any monotone Boolean function on all $m = n$ of the pieces, as in Figure 2. What if we allow $m = O(n)$ or $m = n^{O(1)}$ pieces?

We can also set a weaker goal for this problem. Suppose we are given a function on n pieces, and a "robustness" $k > 0$. Then we can ask to construct $m = O(n+k)$ polygons, n of which are special and $m - n$ of which are extra, such that even if we remove up to k extra pieces, freedom is determined by which special pieces we remove. Wires can be made robust in this way by an idea similar to Figure 2. However, the CROSS gadget seems to be difficult to make robust.

Acknowledgments

This work was initiated at the 25th Bellairs Winter Workshop on Computational Geometry, co-organized by Erik Demaine and Godfried Toussaint, held on February 6–12, 2010, in Holetown, Barbados. We thank the other participants of that workshop—Greg Aloupis, Brad Ballinger, Nadia Benbernou, Prosenjit Bose, David Charlton, Sébastien Collette, Mirela Damian, Karim Douïeb, Robin Flatland, Ferran Hurtado, John Iacono, Krishnam Raju Jampani, Anna Lubiw, Vera Sacristan, Vida Dujmović, Stefan Langerman, Pat Morin, Diane Souvaine—for providing a stimulating research environment. We thank the anonymous referees for helpful comments.

References

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays*, 2nd edition, volumes 1–4. A K Peters Ltd., 2001–2003.

[2] E. D. Demaine, M. L. Demaine, Y. N. Minsky, and J. S. B. Mitchell. Picture-hanging puzzles. Manuscript, 2004.

[3] M. Eiglsperger, S. P. Fekete, and G. W. Klau. Orthogonal graph drawing. In *Drawing Graphs: Method and Models*, LNCS 2025, chapter 6, pages 121–171, 2001.

[4] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters Ltd., 2009.

[5] J. Leeuwen. *Handbook of Theoretical Computer Science*. Elsevier Science Publishers, 1990.

[6] T. Stanford. Brunnian braids and some of their generalizations. arXiv:math/9907072v1 [math.GT], 1999. http://arXiv.org/abs/math/9907072