Picture-hanging puzzles

Picture-Hanging Puzzles*

Erik D. Demaine† Martin L. Demaine† Yair N. Minsky‡ Joseph S. B. Mitchell§
Ronald L. Rivest† Mihai Pătraşcu¶

Abstract

We show how to hang a picture by wrapping rope around n nails, making a polynomial number of twists, such that the picture falls whenever any k out of the n nails get removed, and the picture remains hanging when fewer than k nails get removed. This construction makes for some fun mathematical magic performances. More generally, we characterize the possible Boolean functions characterizing when the picture falls in terms of which nails get removed as all monotone Boolean functions. This construction requires an exponential number of twists in the worst case, but exponential complexity is almost always necessary for general functions.

1 Introduction

If you hang a picture with string looped around two nails, and then remove one of the nails, the picture still hangs around the other nail. Right? This conclusion is correct if you hang the picture around the two nails in the obvious way shown in Figure 1(a). An intriguing puzzle, originally posed by A. Spivak in 1997 [Spi97], asks for a different hanging of the picture with the property that removing *either* nail causes the picture to fall. Figure 1(b) shows a solution to this puzzle.

This puzzle has since circulated around the puzzle community. Michael Hardy from Harvard posed the puzzle to Marilyn vos Savant (famous for her claimed ability to answer any riddle), and the puzzle and solution appeared in her column [vS01]. Torsten Sillke [Sil01] distributed the puzzle, in particular to Ed Pegg Jr., and mentioned a connection to Borromean rings and Brunnian links described in Section 3.1. This connection provides a solution to a more general form of the puzzle, which we call 1-*out-of-n*: hang a picture on n nails so that removing any one nail fells the picture. Pegg's MathPuzzle.com [Peg02] has facilitated a discussion between Sillke, Neil Fitzgerald, and Chris Lusby Taylor. Fitzgerald pointed out a connection to group theory, described in Section 3.2, which provides a direct solution to the 1-out-of-n puzzle. Taylor pointed out a more efficient solution to the same puzzle. All of this work is detailed and carefully analyzed in Section 3.

We consider a more general form of the puzzle where we want the removal of certain subsets of nails to fell the picture. We show that any such puzzle has a solution: for any collection of subsets of nails, we can construct a picture hanging that falls when any entire subset of nails gets

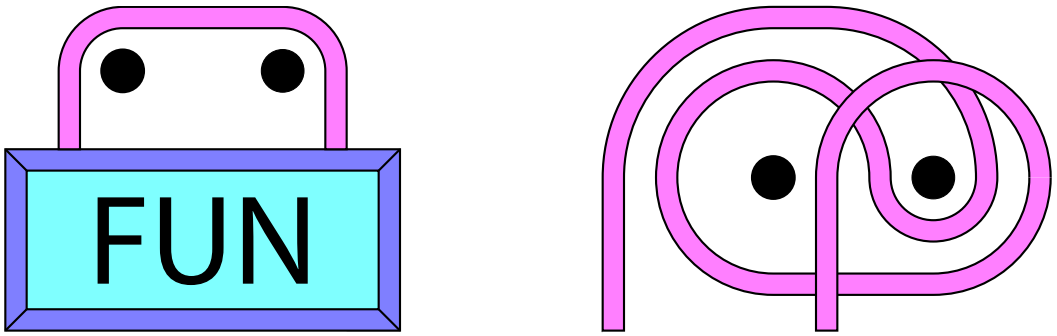
*A preliminary version of this paper appeared in *Proceedings of the 6th International Conference on Fun with Algorithms*, Venice, 2012.

†MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {edemaine,mdemaine,rivest}@mit.edu

‡Department of Mathematics, Yale University, 10 Hillhouse Ave., New Haven, CT 06520, USA, yair.minsky@yale.edu

§Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600, USA, jsbm@ams.sunysb.edu. Partially supported by NSF grant CCF-1018388.

¶AT&T Labs—Research, 180 Park Ave., Florham Park, NJ 07932, mip@alum.mit.edu



(a) A normal hanging. (b) Solution to the two-nail puzzle.

Figure 1: Two ways to hang a picture by looping around two nails.

removed, but remains hanging when every subset still has at least one unremoved nail. This result generalizes picture-hanging puzzles to the maximum extent possible.

Unfortunately, our construction makes an exponential number of twists around the n nails. Indeed, we show that this is necessary, for most general settings of the problem. Fortunately, we find polynomial constructions for the 1-out-of-n puzzle, as well as the *k-out-of-n* generalization where the picture falls only after removing (any) k out of the n nails. More generally, we show that any monotone Boolean function in the complexity class mNC^1 (monotone logarithmic-depth bounded-fanin circuits) has a polynomial-length solution, which can also be found by a polynomial-time algorithm.

These generalizations make for fun puzzles as well as magic performances. Section 2 gives several puzzles accessible to the public that become increasingly easier to solve while reading through this paper. These constructions have been featured as a kind of mathematical magic trick during several of the first authors' talks (first his FUN 2004 plenary talk): the magician wraps large rope around various volunteers' outstretched arms (which act as the "nails"), spectators choose which arms to remove from the construction, and the magician simply "applies infinite gravity" (untangles and pulls on the ends of the rope) to cause the rope to mathemagically fall to the ground. Figure 2 shows some examples.

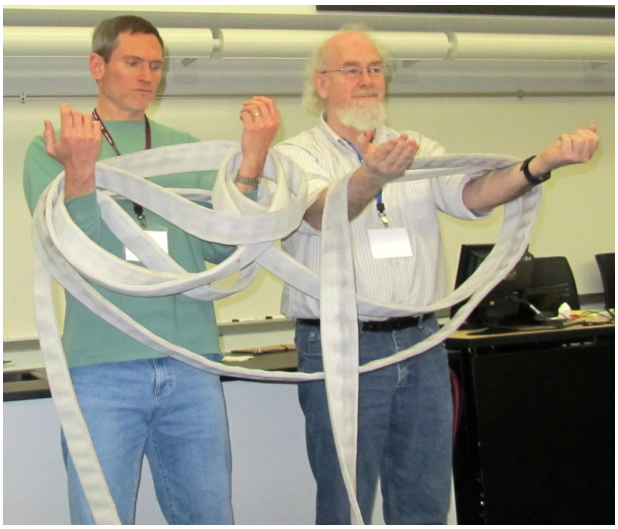
Our work interrelates puzzles, magic, topology, Borromean rings, Brunnian links, group theory, free groups, monotone Boolean function theory, circuit complexity, AKS sorting networks, combinatorics, and algorithms.

A related result constructs interlocked 2D polygons that separate (fall apart) when certain subsets of polygons are removed, again according to an arbitrary monotone Boolean function [DDU10]. That result is essentially a geometric analog of the topological results presented here, although most of the challenges and remaining open questions differ substantially.

2 Puzzles

To whet the appetite of puzzle aficionados, we present a sequence of picture-hanging puzzles ranging from simple to more interesting extensions, some of which require rather involved constructions. We have tested our solutions with 38-inch lanyard wrapped around fingers, and found that this length suffices for Puzzles 1, 2, 3, 6, 7, and 8, but for the other puzzles you would need a longer cord or string. In public performances with large rope wrapped around volunteers' arms, the first author typically performs Puzzles 1, 4, 2, 6, and 8.

(a) A solution to Puzzle 1 implemented by wrapping rope around children's arms for the Porter Public Lecture during the Joint Mathematics Meetings in Boston, Massachusetts in January 2012.



(b) A solution to Puzzle 8 implemented by wrapping fire hose from the local fire department, when the first author forgot to bring his rope for a Polyá Lecture in Menomonie, Wisconsin in April 2011.

Figure 2: Picture-hanging puzzles performed as mathematical magic tricks by the first author.

Puzzle 1 (1-out-of-3) Hang a picture on three nails so that removing any one nail fells the picture.

Puzzle 2 (2-out-of-3) Hang a picture on three nails so that removing any two nails fells the picture, but removing any one nail leaves the picture hanging.

Puzzle 3 (1+2-out-of-3) Hang a picture on three nails arranged along a horizontal line so that removing the leftmost nail fells the picture, as does removing the rightmost two nails, but removing one of the two rightmost nails leaves the picture hanging.

Puzzle 4 (1-out-of-4) Hang a picture on four nails so that removing any one nail fells the picture.

Puzzle 5 (2-out-of-4) Hang a picture on four nails so that removing any two nails fells the picture, but removing any one nail leaves the picture hanging.

Puzzle 6 (3-out-of-4) Hang a picture on four nails so that removing any three nails fells the picture, but removing just one or two nails leaves the picture hanging.

Puzzle 7 (2+2-out-of-2+2) Hang a picture on two red nails and two blue nails so that removing both red nails fells the picture, as does removing both blue nails, but removing one nail of each color leaves the picture hanging.

Puzzle 8 (1+2-out-of-2+2) Hang a picture on two red nails and two blue nails so that removing any one red nail fells the picture, as does removing both blue nails, but removing just one blue nail leaves the picture hanging.

Puzzle 9 (1+3-out-of-3+3) Hang a picture on three red nails and three blue nails so that removing any one red nail fells the picture, as does removing all three blue nails, but removing just one or two blue nails leaves the picture hanging.

Puzzle 10 (1+2-out-of-3+3) Hang a picture on three red nails and three blue nails so that removing any one red nail fells the picture, as does removing any two of the blue nails, but removing just one blue nail leaves the picture hanging.

Puzzle 11 (1+1-out-of-2+2+2) Hang a picture on two red nails, two green nails, and two blue nails so that removing two nails of different colors (one red and one green, or one red and one blue, or one green and one blue) fells the picture, but removing two nails of the same color leaves the picture hanging.

3 Basic Theory: 1-out-of-n

We start our mathematical and algorithmic study of picture-hanging puzzles with the simplest generalization, called 1-*out-of-n*, where the goal is to hang a picture on n nails such that removing any one nail fells the picture. This generalization is what has been studied in the past. Our contribution is to give a thorough complexity analysis of the resulting solutions, the best of which Theorem 1 summarizes below.

Then, in Section 3.4, we give a slight generalization to handle colored nails, which is enough to solve many of the puzzles listed above.

3.1 Connection to Borromean and Brunnian Links

According to Torsten Sillke [Sil01], Werner Schwärzler observed that the Borromean rings provide a solution to the two-nail picture-hanging problem, and that generalized forms of Borromean rings provide solutions to more general picture-hanging problems. This section describes those connections.

The classic *Borromean rings* are three loops that are inseparable—in topology terms, *nontrivially linked*—but such that no two of the rings are themselves linked. Figure 3 shows the standard drawing as interwoven circles, used by the Italian Renaissance family Borromeo as their family crest.

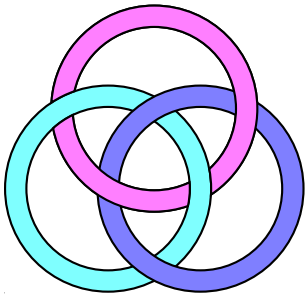
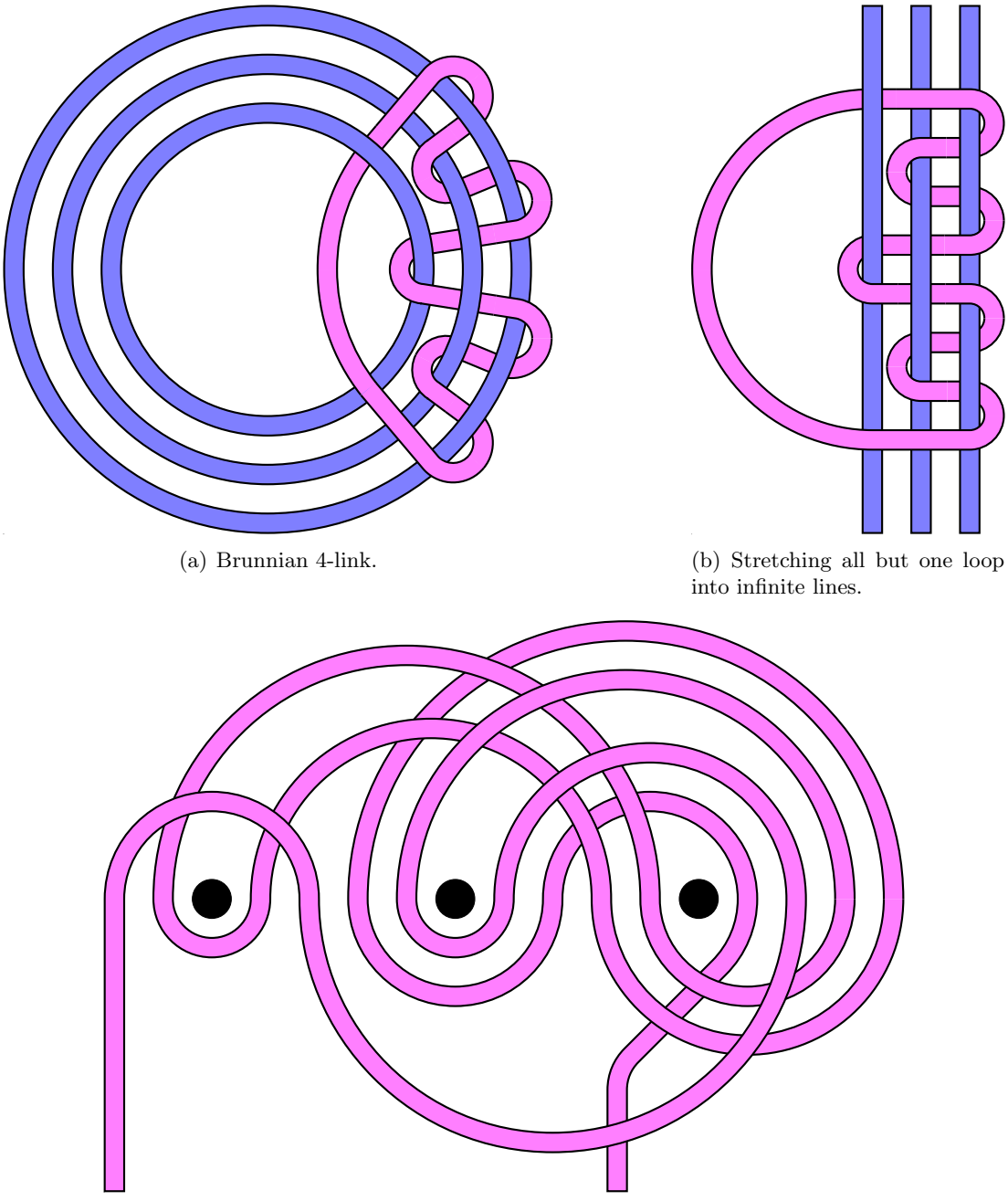


Figure 3: The Borromean rings.

The property of Borromean rings sounds similar to the picture-hanging puzzle: the three loops are linked, but removing any one loop unlinks them. Indeed, by stretching one loop to bring a point to infinity, and straightening out the loop, we can view a loop as an infinite line—or nail—that penetrates the entire construction. Applying this topology-preserving transformation to two out of the three loops, we convert any Borromean-ring construction into a solution to the two-nail picture-hanging puzzle. Conversely, any solution to the two-nail picture-hanging puzzle can be converted into a Borromean-ring construction by viewing the nails as infinite lines piercing the loop of rope and converting these lines to large loops.

Knot theorists have studied two generalizations to the Borromean rings. The first generalization, a *Borromean link*, is a collection of n loops that are linked but such that no two of the loops are linked. This property seems less useful for an n-nail picture-hanging puzzle, because it guarantees only that removing $n-2$ of the nails fells the picture; removing between 1 and $n-3$ of the nails might fell the picture or might not, depending on the particular Borromean link at hand. The

second generalization, a *Brunnian link*, is a collection of n loops that are linked but such that the removal of any loop unlinks the rest. This property is exactly what we need for the n-nail picture-hanging puzzle where removing any one of the n nails fells the picture. Figure 4 shows an example of transforming a Brunnian link into a picture-hanging puzzle.



(a) Brunnian 4-link.

(b) Stretching all but one loop into infinite lines.

(c) Picture-hanging equivalent from over-under pattern of (b).

Figure 4: Transforming a Brunnian n-link into a 1-out-of-n picture-hanging puzzle, for $n = 4$.

Hermann Brunn [Bru92] introduced Brunnian links in 1892, about 25 years after the first mathematical study of Borromean links [Tai76]. Brunn gave a construction for a Brunnian link of

n loops for every $n \geq 3$. See [Rol76] for a more accessible description of this construction. Using the reduction described above, we obtain a solution to the 1-out-of-n picture-hanging puzzle for any $n \geq 2$. The only negative aspect of this solution is that its "size" (combinatorial complexity) grows exponentially with n; we will see a better solution in Section 3.3.

Theodore Stanford [Sta] characterizes a generalized form of Brunnian links, where the removal of arbitrary subsets of loops causes the link to trivialize (fall apart). This problem is subtly different from picture hanging (and indeed, for years, we thought that it had already solved our problem): like Borromean links, it does not require the link remain nontrivial until one of the subsets gets entirely removed. In particular, the trivial link is considered a "solution", no matter what subsets get specified. Conceivably, Stanford's characterization can be used to obtain a solution with this property, but it is not obvious how to do so.

3.2 Connection to Free Group

This section describes a more general framework to study picture-hanging puzzles in general. The framework is based on group theory and comes naturally from algebraic topology. To the best of our knowledge, this connection was first observed by Neil Fitzgerald [Peg02]. Although we do not justify here why the group-theoretic representation is accurate, this is an easy exercise for those familiar with algebraic topology.

A powerful way to abstract a weaving of the rope around n nails uses what is called the *free group on n generators*. Specifically, we define $2n$ symbols:

$$x_1,\ x_1^{-1},\ x_2,\ x_2^{-1},\ \ldots,\ x_n,\ x_n^{-1}.$$

Each x_i symbol represents wrapping the rope around the ith nail clockwise, and each x_i^{-1} symbol represents wrapping the rope around the ith nail counterclockwise. Now a weaving of the rope can be represented by a sequence of these symbols. For example, the solution to the two-nail picture-hanging puzzle shown in Figure 5 can be written $x_1x_2x_1^{-1}x_2^{-1}$ because, starting from the left, it first turns clockwise around the first (left) nail, then turns clockwise around the second (right) nail, then turns counterclockwise around the first nail, and finally turns counterclockwise around the second nail.

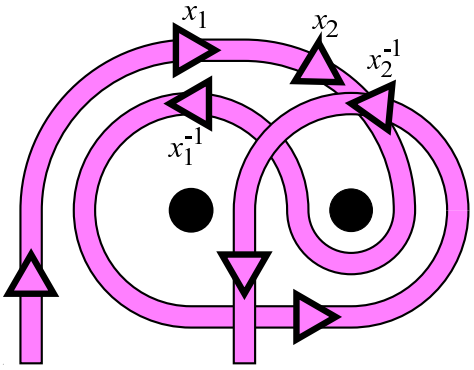


Figure 5: Understanding the algebraic notation for Figure 1(b).

In this representation, removing the ith nail corresponds to dropping all occurrences of x_i and x_i^{-1} in the sequence. Now we can see why Figure 5 disentangles when we remove either nail. For example, removing the first nail leaves just $x_2x_2^{-1}$, i.e., turning clockwise around the second nail and then immediately undoing that turn by turning counterclockwise around the same nail. In general x_i and x_i^{-1} cancel, so all occurrences of $x_ix_i^{-1}$ and $x_i^{-1}x_i$ can be dropped. (The free group specifies that these cancellations are *all* the simplifications that can be made.) Thus, the original weaving $x_1x_2x_1^{-1}x_2^{-1}$ is nontrivially linked with the nails because nothing simplifies; but if we remove either nail, everything cancels and we are left with the empty sequence, which represents the trivial weaving that is not linked with the nails (i.e., the picture falls).

In group theory, the expression $x_1x_2x_1^{-1}x_2^{-1}$ is called the *commutator* of x_1 and x_2, and is written $[x_1, x_2]$. The commutator is a useful tool for solving more general picture-hanging puzzles.

Terminology. In general, define a *picture hanging on n nails* to be a word (sequence of symbols) in the free group on n generators. We refer to the number of symbols in the word as the *length* of the hanging, as it approximates the needed length of the string or cord. The special identity word 1 represents the *fallen* state. *Removing* the ith nail corresponds to removing all occurrences of x_i and x_i^{-1}, which may or may not cause the hanging to fall.

3.3 1-out-of-n

We can use this free-group representation to solve the 1-out-of-n picture-hanging puzzle.

Theorem 1 *For any $n \geq 1$, there is a picture hanging on n nails of length at most $2n^2$ that falls upon the removal of any one nail. For each $i = 1, 2, \ldots, n$, symbols x_i and x_i^{-1} appear at most $2n$ times.*

Exponential construction. We start with a simpler, less-efficient construction given by Neil Fitzgerald [Peg02].[1] The idea is to generalize the weaving $x_1x_2x_1^{-1}x_2^{-1}$ by replacing each x_i with an inductive solution to a smaller version of the problem. In other words, we start with the solution for $n = 2$:

$$S_2 = [x_1, x_2] = x_1x_2x_1^{-1}x_2^{-1}.$$

Now from this solution S_2 we build a solution for $n = 3$ by using the same pattern but involving copies of S_2 in place of one of the x_i's:

$$\begin{aligned} S_3 &= [S_2, x_3] \\ &= S_2x_3S_2^{-1}x_3^{-1} \\ &= (x_1x_2x_1^{-1}x_2^{-1})x_3(x_1x_2x_1^{-1}x_2^{-1})^{-1}x_3^{-1} \\ &= x_1x_2x_1^{-1}x_2^{-1}x_3x_2x_1x_2^{-1}x_1^{-1}x_3^{-1}. \end{aligned}$$

Here we are using the algebraic rules $(xy)^{-1} = y^{-1}x^{-1}$ and $(x^{-1})^{-1} = x$. Figure 6 shows the actual picture-hanging solution corresponding to this sequence.

Granted, this $n = 3$ solution S_3 is a bit complicated, but we can nonetheless verify that it satisfies the desired properties. First, as written, nothing cancels, so it holds the picture without removing any nails. Second, if we remove the first nail x_1, then each of the two copies of S_2 disappear because of an $x_2x_2^{-1}$ cancellation, so we are left with just $x_3x_3^{-1}$ which also disappears. Similarly, if we remove the second nail x_2, again the copies of S_2 collapse and so the entire expression disappears. Finally, if we remove the third nail x_3, then we are left with

$$S_2S_2^{-1} = x_1x_2x_1^{-1}x_2^{-1}x_2x_1x_2^{-1}x_1^{-1},$$

in which again everything cancels. (Cancellation is particularly clear on the left-hand side of this equation, but the low-level cancellation in terms of x_i's is explicit on the right-hand side.) Therefore, no matter which nail we remove, the picture falls.

Naturally, this construction generalizes to all n by defining $S_n = [S_{n-1}, x_n] = S_{n-1}x_nS_{n-1}^{-1}x_n^{-1}$. For example,

$$\begin{aligned} S_4 &= [S_3, x_4] \\ &= S_3x_4S_3^{-1}x_4^{-1} \\ &= x_1x_2x_1^{-1}x_2^{-1}x_3x_2x_1x_2^{-1}x_1^{-1}x_3^{-1}x_4x_3x_1x_2x_1^{-1}x_2^{-1}x_3^{-1}x_2x_1x_2^{-1}x_1^{-1}x_4^{-1}. \end{aligned}$$

[1] This construction also turns out to be essentially the same as the solution that comes out of the Brunnian-link construction described in Section 3.1.

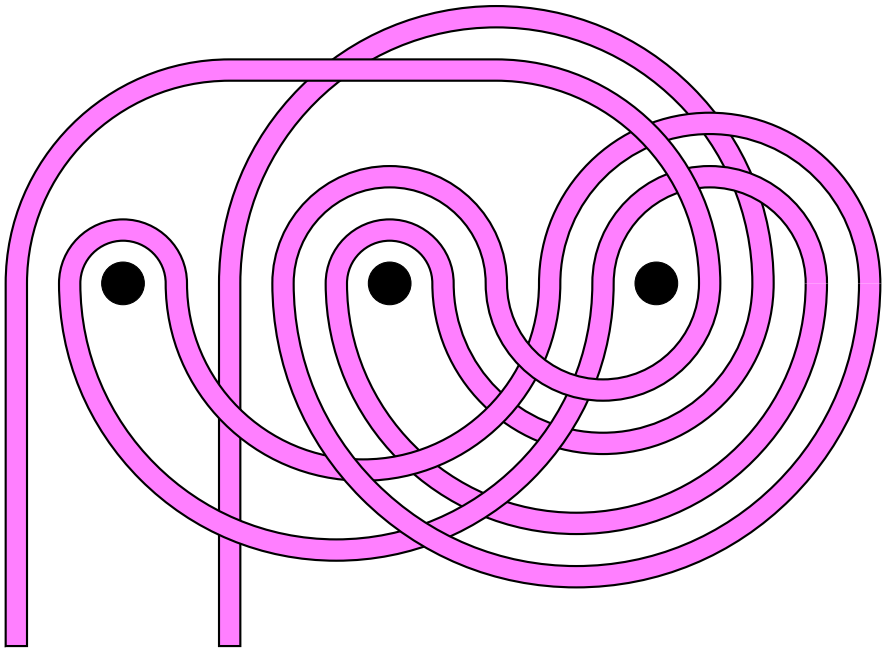


Figure 6: Hanging a picture on three nails so that removing any one nail fells the picture.

By induction, if we remove any of the first three nails, the two copies of S_3 disappear, leaving us with $x_4 x_4^{-1}$ which cancels. And if we remove the fourth nail x_4, we are left with $S_3 S_3^{-1}$ which cancels.

The problem with this construction, which we start to see with the full expansion of S_4, is that the length of the sequence S_n grows exponentially with n. More precisely, the number of symbols in S_n is $2^n + 2^{n-1} - 2$. To see why this count is correct, first check that S_2 has length $4 = 2^2 + 2^1 - 2$. Then, if we suppose inductively that S_{n-1} has length $2^{n-1} + 2^{n-2} - 2$, we can conclude that S_n has twice that length plus 2 for the occurrences of x_n and x_n^{-1}, for a total of

$$\begin{aligned} & 2(2^{n-1} + 2^{n-2} - 2) + 2 \\ = \; & 2^n + 2^{n-1} - 4 + 2 \\ = \; & 2^n + 2^{n-1} - 2, \end{aligned}$$

as claimed.

Polynomial construction. Fortunately, there is a more efficient construction that solves the 1-out-of-n picture-hanging puzzle. This construction was designed by Chris Lusby Taylor [Peg02]. The idea is to recursively build S_n in a more balanced way, in terms of $S_{n/2}$ for the first half of the nails and $S_{n/2}$ for the second half of the nails, instead of one S_{n-1} and a single variable. To enable this construction, we need to consider a more general problem involving the nails from i through j for various i and j. At the simplest level we have a single nail:

$$E(i : i) = x_i.$$

At the next simplest level we have two nails as before:

$$E(i : i+1) = [x_i, x_{i+1}] = x_i x_{i+1} x_i^{-1} x_{i+1}^{-1}.$$

Then for an arbitrary interval $i : j$, we build $E(i : j)$ out of a recursive copy of E applied to the first half of the interval and a recursive copy of E applied to the second half of the interval:

$$E(i : j) \quad = \quad \left[E\left(i : \left\lfloor \tfrac{i+j}{2} \right\rfloor\right), E\left(\left\lfloor \tfrac{i+j}{2} \right\rfloor + 1 : j\right)\right].$$

For $n = 3$, this construction does not save anything, because splitting an interval of length three in half leaves one piece of length two and one piece of length one. But for $n = 4$ we gain some efficiency:

$$\begin{aligned} E(1:4) &= [E(1:2), E(3:4)] \\ &= E(1:2)\ E(3:4)\ E(1:2)^{-1}\ E(3:4)^{-1} \\ &= (x_1x_2x_1^{-1}x_2^{-1})(x_3x_4x_3^{-1}x_4^{-1})(x_1x_2x_1^{-1}x_2^{-1})^{-1}(x_3x_4x_3^{-1}x_4^{-1})^{-1} \\ &= x_1x_2x_1^{-1}x_2^{-1}x_3x_4x_3^{-1}x_4^{-1}x_2x_1x_2^{-1}x_1^{-1}x_4x_3x_4^{-1}x_3^{-1}. \end{aligned}$$

This sequence has 16 symbols compared to the 22 from $S(4)$ above.

While this savings may not seem significant, the savings becomes substantially more impressive as n grows. If n is a power of two, then $E(1 : n)$ has length n^2, because it consists of two copies of $E(1 : n/2)$ and two copies of $E(n/2 + 1 : n)$ and because $4(n/2)^2 = n^2$. Furthermore, in this case, symbols x_i and x_i^{-1} appear exactly n times in $E(1 : n)$ because by induction they appear exactly $n/2$ times in exactly one of $E(1 : n/2)$ and $E(n/2 + 1 : n)$.

If n is not a power of two, we at least have that $E(1 : n)$ has length at most $(2n)^2 = 4n^2$, because $E(1 : n)$ only increases if we round up to the next power of two. The integer sequence formed by the length of $E(1 : n)$ with $n = 1, 2, 3, \ldots$ is in fact in Neil Sloane's Encyclopedia [Slo02]. Ellul, Krawetz, Shallit, and Wang [EKSW05] proved that, if n is b larger than the previous power of two, 2^a, then the length of $E(1 : n)$ is precisely $(2^a)^2 + b(2^{a+2} - 2^a)$. This formula is always at most $2n^2$. Furthermore, symbols x_i and x_i^{-1} appear at most $2n$ times in $E(1 : n)$ because each recursion doubles the number of appearances, and there are precisely $\lceil \log_2 n \rceil \leq \log_2 n + 1$ recursions, so the number of appearances is at most $2^{\log_2 n + 1} = 2n$.

This completes the proof of Theorem 1.

3.4 Disjoint Subsets of Nails

One way to state the most general form of a picture-hanging puzzle is the following: given arbitrary subsets S_1, S_2, ..., S_k of $\{1, 2, \ldots, n\}$, hang a picture on n nails such that removing all the nails in S_i fells the picture, for any i between 1 and k, but removing a set of nails that does not include an entire S_i leaves the picture hanging. For example, the 1-out-of-n puzzle is the special case of $S_i = \{i\}$ for $i = 1, 2, \ldots, n$. All of the puzzles posed in Section 2 can be represented as particular instances of this general puzzle.

As a warmup to this general form of the puzzle, we first illustrate how the theory we have developed so far easily solves the special case in which the subsets $S_1, S_2, \ldots, S_k$ are pairwise disjoint. This corresponds to the pegs being divided into different color classes, and the picture falling precisely when an entire color class has been removed. Many of the puzzles posed in Section 2 fall into this class.

Theorem 2 *For any partition of $\{1, 2, \ldots, n\}$ into disjoint subsets $S_1, S_2, \ldots, S_k$, there is a picture hanging on n nails of length at most $2kn$ that falls when removing all nails in S_i, for any i between 1 and k, but does not fall when keeping at least one nail from each S_i.*

The idea is to replace each subset S_i of nails with a "supernail" and then apply the 1-out-of-n solution to the supernails. Whenever the solution says to wrap clockwise around a supernail, we wrap clockwise around each of the nails in the supernail in a particular order; when we wrap counterclockwise around the supernail, we wrap counterclockwise around each of the nails in the reverse order.

More precisely, we represent each subset $S_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,r_i}\}$ by the sequence $w_i = x_{i,1}x_{i,2}\cdots x_{i,r_i}$. This sequence w_i collapses to nothing precisely when that subset has been satisfied, i.e., all of the $x_{i,j}$ nails constituting S_i have been removed. Our goal is therefore to combine $w_1, w_2, \ldots, w_k$ so that the entire sequence collapses precisely when any of the w_i's collapses.

Next we combine $w_1, w_2, \ldots, w_k$ using the $E(1:k)$ construction, where each x_i in $E(1:k)$ is replaced by a w_i. In other words, we define $W(i:i) = w_i$ and

$$W(i:j) \quad = \quad \left[W\left(i : \lfloor \tfrac{1}{2}(i+j) \rfloor\right), W\left(\lfloor \tfrac{1}{2}(i+j) \rfloor + 1 : j\right)\right].$$

The resulting sequence $W(1:n)$ collapses whenever any of the w_i's collapses, because by induction the left or right half containing w_i collapses, leaving the other half next to its inverse, which collapses.

The last requirement of the solution is that it does not collapse if every w_i remains intact. This property follows because no two of the w_i's share a letter. Thus, any two subconstructions $W(i:j)$ and $W(j+1:k)$ do not share a letter. Therefore, none of the sequence concatenations we do in the construction of $W(1:n)$ could have accidental cancellation if every w_i keeps at least one letter.

The length of $W(1:n)$ is at most $2k$ times the total length of the w_i's, because Theorem 1 guarantees that each w_i appears at most $2k$ times (counting the negated form w_i^{-1}). The total length of the w_i's is exactly n because the subsets S_i's form a partition of $\{1, 2, \ldots, n\}$. Therefore the total length is at most $2kn$, which in particular is at most $2n^2$, completing the proof of Theorem 2.

4 General Theory

This section develops a general theory for solving the most general form of the picture-hanging puzzle. Section 3.4 above described one statement of this general form, using subsets, but this turns out to be an inefficient way to represent even relatively simple problems. For example, the k-out-of-n puzzle has $\binom{n}{k}$ subsets of nails that fell the picture, which is exponential for k between εn and $(1-\varepsilon)n$. We therefore turn to a more general representation, called "monotone Boolean functions". Although our general solution remains exponential in the worst case, we show in Section 4.4 how this representation allows us to achieve a polynomial solution for k-out-of-n in particular.

4.1 Connection to Monotone Boolean Functions

For a given picture hanging p on n nails, define the *fall function* $f_p(r_1, r_2, \ldots, r_n)$, where each r_i is a Boolean value (true/1 or false/0), to be a Boolean value specifying whether the hanging p falls after removing all x_i's corresponding to true r_i's. For example, a solution p to the 1-out-of-n puzzle has the fall function "is any r_i set to true?", because setting any r_i to true (i.e., removing any x_i) causes the construction p to fall. In logic, we would write

$$f_p(r_1, r_2, \ldots, r_n) = r_1 \vee r_2 \vee \cdots \vee r_n, \tag{1}$$

where $\vee$ represents OR (logical disjunction).

The most general form of picture-hanging puzzle on n nails is the following: given a desired fall function $f(r_1, r_2, \ldots, r_n)$, find a picture hanging p with that fall function, i.e., with $f_p = f$. For example, the function in Equation (1) is a specification of the 1-out-of-n problem.

Not all such puzzles can be solved, however. Every fall function must satisfy a simple property called *monotonicity*: if $r_1 \leq r'_1$, $r_2 \leq r'_2$, ..., and $r_n \leq r'_n$, then $f(r_1, r_2, \ldots, r_n) \leq f(r'_1, r'_2, \ldots, r'_n)$. Here we view the truth values as 0 (false) and 1 (true), so that false $<$ true. This condition just says that, if the hanging falls when removing certain nails given by the r_i's, and we remove even additional nails as given by the r'_i's, then the hanging still falls. A picture hanging cannot "unfall" from removing nails, so monotonicity is a necessary condition on fall functions. For example, it is impossible to find a picture hanging that falls from removing any one nail but not from removing more nails.

Monotone Boolean functions are well-studied in combinatorics (through Dedekind's Problem), computational complexity, and computational learning theory, among other fields. It is well-known that they are exactly the functions formed by combining the variables $r_1, r_2, \ldots, r_n$ with the operators AND ($\wedge$) and OR ($\vee$). (In particular, NOT is forbidden.) We can leverage this existing theory about monotone Boolean functions to obtain powerful results about picture hanging.

4.2 Arbitrary Monotone Boolean Functions

In particular, we establish that monotone Boolean functions are exactly the fall functions of picture hangings. We have already argued that every fall function is monotone; the interesting part here is that every monotone Boolean function can be represented as the fall function of a picture hanging. Our construction is exponential in the worst case, but as we will see, is efficient in many interesting cases.

Theorem 3 *Every monotone Boolean function f on n variables is the fall function f_p of a picture hanging p on n nails. If the function f can be computed by a depth-d circuit of two-input* AND *and* OR *gates, then we can construct p to have length c^d for a constant c. We can compute such p in time linear in the length of p. In particular, for functions f representable by a depth-$O(\log n)$ circuit of two-input* AND *and* OR *gates (the complexity class* mNC^1*), there is a polynomial-length picture hanging.*

Our approach to proving this theorem is to simulate AND and OR gates in a way that allows us to combine them into larger and larger circuits. The most intuitive version of the construction is when the function f is represented as a monotone Boolean *formula* (as opposed to circuit), which can be parsed into a tree with the r_i's at the leaves and the value of f at the root. As base cases, we can represent the formula r_i by the picture hanging x_i (or x_i^{-1}), which falls precisely when the ith nail gets removed. We show next that, given picture hangings p and q representing two monotone Boolean functions f and g, we can construct picture hangings AND(p, q) and OR(p, q) representing $f \wedge g$ and $f \vee g$, respectively. While most intuitively applied up a tree representing a formula, the same construction applies to a directed acyclic graph representing a general circuit.

AND. Our AND and OR constructions build on two known lemmas from monotone function theory. We start with AND:

Lemma 4 [A. I. Mal'tsev] [Mak85, Lemma 3] *For any two words p, q in the free group on $x_1, x_2, \ldots, x_n$, the equation*

$$p^2 x_1 p^2 x_1^{-1} = (q x_2 q x_2^{-1})^2$$

is equivalent to the conjunction $(p = 1) \wedge (q = 1)$.

Using commutator notation, the equation becomes $[p, x_1] = [q, x_2]^2$. Because the free group is a group, we can right-multiply the equation by $[q, x_2]^{-2}$ to obtain the equivalent equation

$$[p, x_1] \cdot [q, x_2]^{-2} = 1.$$

Lemma 4 states that this equation holds if and only if $p = 1$ and $q = 1$. Recall that 1 is the fallen state of picture hangings. Thus, the left-hand side

$$\textsc{and}(p, q) = [p, x_1] \cdot [q, x_2]^{-2} = p x_1 p^{-1} x_1^{-1} x_2 q x_2^{-1} q^{-1} x_2 q x_2^{-1} q^{-1} \tag{2}$$

falls if and only if both p and q fall. This construction is our desired AND.

OR. We now turn to the OR construction:

Lemma 5 [G. A. Gurevich] [Mak85, Lemma 4] *For any two words* p, q *in the free group on* $x_1, x_2, \ldots, x_n$*, the conjunction of the four equations*

$$(p x_1^s p x_1^{-s})(q x_2^t q x_2^{-t}) = (q x_2^t q x_2^{-t})(p x_1^s p x_1^{-s}) \quad \textit{for all } s, t = \pm 1$$

is equivalent to the disjunction $p = 1 \vee q = 1$.

Using commutator notation, the equations become

$$[p, x_1^s] \cdot [q, x_2^t] = [q, x_2^t] \cdot [p, x_1^s] \quad \text{for all } s, t = \pm 1.$$

Right-multiplying by the inverse of the right-hand side (again), the equations are equivalent to

$$[p, x_1^s] \cdot [q, x_2^t] \cdot [p, x_1^s]^{-1} \cdot [q, x_2^t]^{-1} = 1 \quad \text{for all } s, t = \pm 1.$$

Again using commutator notation, the equations become

$$\big[[p, x_1^s], [q, x_2^t]\big] = 1 \quad \text{for all } s, t = \pm 1.$$

Lemma 5 states that these equations all hold if and only if $x = 1$ or $y = 1$. To obtain the conjunction of the four equations, we apply the AND construction above:

$$\begin{aligned} \textsc{or}(p, q) = \textsc{and}\Big(&\textsc{and}\Big(\big[[p, x_1], [q, x_2]\big], \big[[p, x_1], [q, x_2^{-1}]\big]\Big), \\ &\textsc{and}\Big(\big[[p, x_1^{-1}], [q, x_2]\big], \big[[p, x_1^{-1}], [q, x_2^{-1}]\big]\Big)\Big). \end{aligned} \tag{3}$$

Thus $\textsc{or}(p, q)$ falls if and only if either p or q falls. This construction is our desired OR. The OR formula expands to 144 p and q terms, and 474 x_1 and x_2 terms, for a total of 618 terms.

Analysis. Now we argue that a circuit of depth d results in a picture hanging of length at most c^d for a constant c. The output of the circuit is the output of some gate, either AND or OR, which has two inputs. Each input can be viewed as the output of a subcircuit of the overall circuit, with smaller depth $d-1$. The two subcircuits may overlap (or even be identical), but we treat them as separate by duplicating any shared gates. By induction on depth, these subcircuits can be converted into picture hangings p and q of length at most c^{d-1}. We combine these picture hangings via $\text{AND}(p,q)$ or $\text{OR}(p,q)$, according to the output gate type, to obtain our desired picture hanging. The resulting length is at most the maximum length of p and q, which is at most c^{d-1}, times the number of terms in Equations (2) and (3) defining AND and OR. Thus, setting $c = 618$ suffices.

In the base case, the depth-0 circuit has no gates and simply takes the value of a variable r_i, and we use the picture hanging x_i, which has length $1 = c^0$ as needed.

This argument gives a 618^d upper bound on the size of the constructed picture hanging. In fact, only 144 of the 618 terms in (3) are recursive (p or q), so the upper bound is 144^d plus lower-order terms.

Running time. To compute the picture hanging resulting from this construction in linear time, we simply need a data structure for representing a picture hanging that supports concatenation and inversion in constant time. One such data structure is a doubly linked list, with a bit in each node to indicate when the orientation and inverted state flips. Once we construct the data structure representing the final picture hanging, a single pass through the list can flatten into a typical representation of a picture hanging as a word in the free group.

This argument completes the proof of Theorem 3.

4.3 Worst-Case Optimality

Unfortunately, most monotone Boolean functions require exponential-length picture hangings:

Theorem 6 *Almost all monotone Boolean functions require length-$\Omega(2^n/(n \log n))$ picture hangings.*

This theorem follows from a counting argument, specifically, contrasting the large number of monotone Boolean functions with the relatively small number of picture hangings of a given length.

First we demonstrate a large number of monotone Boolean functions. (This argument is standard.) The vectors $(r_1, r_2, \ldots, r_n)$ with exactly $n/2$ 1's (and $n/2$ 0's) can all have their function values set independently. There are $\binom{n}{n/2}$ such vectors. Thus there are at least $2^{\binom{n}{n/2}}$ monotone Boolean functions on n variables.

Next we observe that the number of picture hangings of length ℓ is at most $(2n)^\ell$, because there are at most $2n$ choices for each symbol in the word. (The correct number of choices is $2n-1$, except for the first, to avoid cancelation.) The number of picture hangings of length at most ℓ is $\sum_{i=1}^{\ell}(2n)^i < 2(2n)^\ell$.

To represent all monotone Boolean functions, we must have

$$2(2n)^\ell \geq 2^{\binom{n}{n/2}}.$$

Taking $\log_2$ of both sides, we must have

$$1 + \ell(1 + \log_2 n) \geq \binom{n}{n/2}.$$

Asymptotically, $\binom{n}{n/2} \sim 2^n \sqrt{\frac{2}{\pi n}}$. Thus we must have $\ell \sim 2^n \sqrt{\frac{2}{\pi n \log_2 n}}$. A standard "almost every" argument completes the proof of Theorem 6.

4.4 *k*-out-of-*n*

One example of a picture-hanging puzzle that can be solved more efficiently (at least in theory) is the k-out-of-n puzzle:

Theorem 7 *For any $n \geq k \geq 1$, there is a picture hanging on n nails, of length $n^{c'}$ for a constant c', that falls upon the removal of any k of the nails.*

We simply argue that the monotone Boolean function "are at least k of the r_i's true?" is in the complexity class mNC^1, that is, can be represented by a logarithmic-depth binary circuit. The idea is to *sort* the r_i values, again viewing Boolean values as 0 (false) and 1 (true). The result of this sorting is a sequence of j 0's followed by a sequence of $n - j$ 1's. Our goal is to determine whether $n - j \geq k$. To do so, we would simply look at the $(n - k + 1)$st item in the sorted order: if it is 1, then there at least k 1's, and otherwise, there are fewer.

Our construction thus starts from a logarithmic-depth *sorting network*, as first achieved by Ajtai, Komlós, and Szemerédi [AKS83] and improved by Mike Paterson [Pat90]. A sorting network consists of a circuit of *comparators*. Each comparator has two inputs and two outputs, outputting the smaller input on top and the larger input on bottom (thus sorting the inputs).

Such a sorting network can be converted into monotone Boolean formulas, as illustrated in Figure 7. If a comparator has two inputs p and q, then the top (minimum) output is $p \wedge q$, and the bottom (maximum) output is $p \vee q$. Applying this rule to every comparator, we end up with a monotone Boolean formula (or, more efficiently, a monotone Boolean circuit) representing each of the items in the sorted output. The solution to the k-out-of-n sorting puzzle is simply the $(n - k + 1)$st of these circuits.

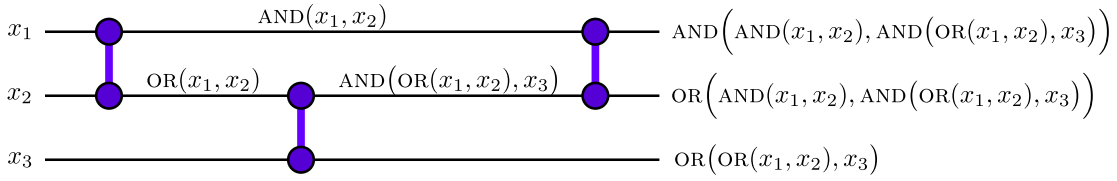


Figure 7: Converting a sorting network (with three elements and three comparators) into a sequence of (three) monotone Boolean formulas.

Because the sorting network has logarithmic depth, so does the monotone Boolean circuit; the depths are actually identical. The best known upper bound on the depth of sorting networks is under $6{,}100 \log_2 n$. Applying Theorem 3, we obtain a picture hanging of length $c^{6{,}100 \log_2 n} = n^{6{,}100 \log_2 c}$. Using the $c \leq 618$ upper bound, we obtain an upper bound of $c' \leq 56{,}556$. Using the $c \leq 144 + o(1)$ upper bound, we obtain an upper bound of $c' \leq 43{,}737 + o(1)$.

So, while this construction is polynomial, it is a rather large polynomial. For small values of n, we can use known small sorting networks to obtain somewhat reasonable constructions.

5 Spectating Is Hard

Imagine we turn the tables and, instead of considering the magician's challenge in hanging a picture on n nails with certain properties, we consider the spectator's challenge of choosing which nails to remove. A natural objective, if the spectator is shy and wants to get off stage as quickly as possible, is to remove as few nails as possible in order to make the picture fall. Unfortunately for the spectator, for a given picture hanging, this problem is NP-complete and hard to approximate:

Theorem 8 *For a given picture hanging on n nails, it is NP-complete to decide whether there are k nails whose removal fells the picture, and it is hard to approximate the minimum number of nails within an $\varepsilon \log n$ factor for some $\varepsilon > 0$.*

The decision problem is in NP: given the free-group representation of a picture hanging as input, and which k nails to remove as certificate, we can verify that the word cancels after removing the necessary nails.

For hardness, we reduce from the Set Cover problem: given a universe $U = \{u_1, u_2, \ldots, u_m\}$ of m elements, and a collection of n subsets, $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where each $S_i \subseteq U$, find a smallest subcollection $\mathcal{S}' \subseteq \mathcal{S}$ whose union $\bigcup_{S \in \mathcal{S}'} S = U$. This problem is NP-hard to approximate within an $\varepsilon' \log n$ factor.

Now, given an instance of Set Cover, we create a picture hanging on n nails as follows. Each x_i corresponds to a set S_i. Let E_j denote the efficient 1-out-of-n construction from Section 3.3 applied to the x_i's corresponding to S_j's that contain u_j. We construct E_j for each j between 1 and m, and then combine them with a balanced tree of AND's according to Equation (2). This picture hanging falls precisely when every element has at least one containing set chosen for removal, which means that the sets were all covered. The objective is the same for the two problems, and the problem size increased by only a polynomial factor, by Theorems 1 and 3.

This argument completes the proof of Theorem 8.

We can similarly argue that it is NP-hard for the attention-hoarding spectator who aims to *maximize* the number of nails to remove before felling the picture hanging. By the same reduction, this problem becomes finding a set of elements that hit every set in the collection $\mathcal{S}$, which is the Hitting Set problem. Reversing the roles of elements and sets, we have the identical Set Cover problem. Inapproximability no longer follows because the objectives are reversed.

6 Open Problems

Several interesting open questions remain about the optimality of our constructions. Does the 1-out-of-n picture hanging puzzle require a solution of length $\Omega(n^2)$? What is the complexity of finding the shortest picture hanging for a given monotone Boolean function? For the spectator, is there an $O(\log n)$-approximation algorithm for removing the fewest nails to fell the picture hanging?

Acknowledgments

We thank Jason Cantarella for helpful early discussions, and Kim Whittlesey for pointing out reference [Sta].

References

[AKS83] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatoria*, 3(1):1–19, 1983.

[Bru92] H. Brunn. Über Verkettung. In *Sitzungsbericht der Bayerischen Akademie der Wissenschaft Mathematisch Naturwissenschaftliche Abteilung*, volume 22, pages 77–99, 1892.

[DDU10] Erik D. Demaine, Martin L. Demaine, and Ryuhei Uehara. Any monotone Boolean function can be realized by interlocked polygons. In *Proceedings of the 22nd Canadian Conference on Computational Geometry*, pages 139–142, Winnipeg, Canada, August 2010.

[EKSW05] Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Ming-wei Wang. Regular expressions: new results and open problems. *Journal of Automata, Languages and Combinatorics*, 9(2–3):233–256, September 2005.

[Mak85] G. S. Makanin. Decidability of the universal and positive theories of a free group. *Mathematics of the USSR-Izvestiya*, 25(1):75–88, 1985. Translated from Russian edition, 48(4), 1984.

[Pat90] M. S. Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5:75–92, 1990.

[Peg02] Ed Pegg Jr. Picture hanging, 2002. http://www.mathpuzzle.com/hangingpicture.htm.

[Rol76] Dale Rolfsen. *Knots and Links*. Publish or Perish, Inc., Houston, Texas, 1976.

[Sil01] Torsten Sillke. Strange painting, 2001. http://www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/quantum/B201.

[Slo02] Neil J. A. Sloane. Sequence A073121. In *On-Line Encyclopedia of Integer Sequences*. August 2002. http://www.research.att.com/projects/OEIS?Anum=A073121.

[Spi97] A. Spivak. Brainteasers B 201: Strange painting. *Quantum*, page 13, May/June 1997. Solution on page 60, figure 5.

[Sta] Theodore Stanford. Brunnian braids and some of their generalizations. *Bulletin of the London Mathematical Society*. To appear. arXiv:math.GT/9907072, http://arXiv.org/abs/math/9907072.

[Tai76] P. G. Tait. On knots. *Transactions of the Royal Society of Edinburgh*, 28:145–190, 1876.

[vS01] Marilyn vos Savant. Ask Marilyn. *PARADE*, 2001. Puzzle posed in June 10 issue and solved in June 17 issue.

A Puzzle Solutions

This appendix gives solutions to the puzzles from Section 2 in the free-group notation defined in Section 3.2. These solutions are based on the basic theory described in Section 3, though they do not all fall under the specific statement of Theorem 2. The solutions are not unique; shorter solutions may well exist.

Puzzle 1: Figure 6 shows $S_3 = x_1x_2x_3x_2^{-1}x_3^{-1}x_1^{-1}x_3x_2x_3^{-1}x_2^{-1}$.

Puzzle 2: $x_1x_2x_3x_1^{-1}x_2^{-1}x_3^{-1}$.

Puzzle 3: $[x_1, x_2x_3] = x_1x_2x_3x_1^{-1}x_3^{-1}x_2^{-1}$, where the nails are ordered from left to right.

Puzzle 4: $E(1:4) = x_1x_2x_1^{-1}x_2^{-1}x_3x_4x_3^{-1}x_4^{-1}x_2x_1x_2^{-1}x_1^{-1}x_4x_3x_4^{-1}x_3^{-1}$.

Puzzle 5: $[[x_1x_2, [x_1x_3, x_1x_4]], [x_2x_3, [x_2x_4, x_3x_4]]$ $= x_1x_2x_1x_3x_1x_4x_3^{-1}x_1^{-1}x_4^{-1}x_1^{-1}x_2^{-1}x_1^{-1}x_1x_4$ $x_1x_3x_4^{-1}x_1^{-1}x_3^{-1}x_1^{-1}x_2x_3x_2x_4x_3x_4x_4^{-1}x_2^{-1}x_4^{-1}x_3^{-1}x_3^{-1}x_2^{-1}x_3x_4x_2x_4x_4^{-1}x_3^{-1}x_4^{-1}x_2^{-1}x_1x_3x_1x_4x_3^{-1}x_1^{-1}x_4^{-1}$ $x_1^{-1}x_1x_2x_1x_4x_1x_3x_4^{-1}x_1^{-1}x_3^{-1}x_1^{-1}x_2^{-1}x_1^{-1}x_2x_4x_3x_4x_4^{-1}x_2^{-1}x_4^{-1}x_3^{-1}x_2x_3x_3x_4x_2x_4x_4^{-1}x_3^{-1}x_4^{-1}x_2^{-1}x_3^{-1}x_2^{-1}$.

Puzzle 6: $x_1x_2x_3x_4x_1^{-1}x_2^{-1}x_3^{-1}x_4^{-1}$.

Puzzle 7: $[x_1x_2, x_3x_4] = x_1x_2x_3x_4x_2^{-1}x_1^{-1}x_4^{-1}x_3^{-1}$, where x_1 and x_2 are red and x_3 and x_4 are blue.

Puzzle 8: $[S_2, x_3x_4] = x_1x_2x_1^{-1}x_2^{-1}x_3x_4x_2x_1x_2^{-1}x_1^{-1}x_4^{-1}x_3^{-1}$, where x_1 and x_2 are red and x_3 and x_4 are blue.

Puzzle 9: $[S_3, x_4x_5x_6] = x_1x_2x_3x_2^{-1}x_3^{-1}x_1^{-1}x_3x_2x_3^{-1}x_2^{-1}x_4x_5x_6x_2x_3x_2^{-1}x_3^{-1}x_1x_3x_2x_3^{-1}x_2^{-1}x_1^{-1}$ $x_6^{-1}x_5^{-1}x_4^{-1}$, where x_1, x_2, and x_3 are red and x_4, x_5, and x_6 are blue.

Puzzle 10: $[S_3, x_4x_5x_6x_4^{-1}x_5^{-1}x_6^{-1} = x_1x_2x_3x_2^{-1}x_3^{-1}x_1^{-1}x_3x_2x_3^{-1}x_2^{-1}x_4x_5x_6x_4^{-1}x_5^{-1}x_6^{-1}x_2x_3x_2^{-1}$ $x_3^{-1}x_1x_3x_2x_3^{-1}x_2^{-1}x_1^{-1}x_6x_5x_4x_6^{-1}x_5^{-1}x_4^{-1}$, where x_1, x_2, and x_3 are red and x_4, x_5, and x_6 are blue.

Puzzle 11: $[[[x_1x_3,[x_2x_4,x_1x_5]],[x_3x_6,[x_1x_4,x_2x_3]]],[[x_1x_6,[x_2x_5,x_4x_6]],[x_3x_5,[x_2x_6,x_4x_5]]]] =$
$x_1x_3x_2x_4x_1x_5x_4^{-1}x_2^{-1}x_5^{-1}x_1^{-1}x_3^{-1}x_1^{-1}x_1x_5x_2x_4x_5^{-1}x_1^{-1}x_4^{-1}x_2^{-1}x_3x_6x_1x_4x_2x_3x_4^{-1}x_1^{-1}x_3^{-1}x_2^{-1}x_6^{-1}x_3^{-1}x_2x_3$
$x_1x_4x_3^{-1}x_2^{-1}x_4^{-1}x_1^{-1}x_2x_4x_1x_5x_4^{-1}x_2^{-1}x_5^{-1}x_1^{-1}x_1x_3x_1x_5x_2x_4x_5^{-1}x_1^{-1}x_4^{-1}x_2^{-1}x_3^{-1}x_1^{-1}x_1x_4x_2x_3x_4^{-1}x_1^{-1}x_3^{-1}$
$x_2^{-1}x_3x_6x_2x_3x_1x_4x_3^{-1}x_2^{-1}x_4^{-1}x_1^{-1}x_6^{-1}x_3^{-1}x_1x_6x_2x_5x_4x_6x_5^{-1}x_2^{-1}x_6^{-1}x_4^{-1}x_6^{-1}x_1^{-1}x_4x_6x_2x_5x_6^{-1}x_4^{-1}x_5^{-1}x_2^{-1}$
$x_3x_5x_2x_6x_4x_5x_6^{-1}x_2^{-1}x_5^{-1}x_4^{-1}x_5^{-1}x_3^{-1}x_4x_5x_2x_6x_5^{-1}x_4^{-1}x_6^{-1}x_2^{-1}x_2x_5x_4x_6x_5^{-1}x_2^{-1}x_6^{-1}x_4^{-1}x_1x_6x_4x_6x_2x_5x_6^{-1}$
$x_4^{-1}x_5^{-1}x_2^{-1}x_6^{-1}x_1^{-1}x_2x_6x_4x_5x_6^{-1}x_2^{-1}x_5^{-1}x_4^{-1}x_3x_5x_4x_5x_2x_6x_5^{-1}x_4^{-1}x_6^{-1}x_2^{-1}x_5^{-1}x_3^{-1}x_3x_6x_1x_4x_2x_3x_4^{-1}x_1^{-1}$
$x_3^{-1}x_2^{-1}x_6^{-1}x_3^{-1}x_2x_3x_1x_4x_3^{-1}x_2^{-1}x_4^{-1}x_1^{-1}x_1x_3x_2x_4x_1x_5x_4^{-1}x_2^{-1}x_5^{-1}x_1^{-1}x_3^{-1}x_1^{-1}x_1x_5x_2x_4x_5^{-1}x_1^{-1}x_4^{-1}x_2^{-1}$
$x_1x_4x_2x_3x_4^{-1}x_1^{-1}x_3^{-1}x_2^{-1}x_3x_6x_2x_3x_1x_4x_3^{-1}x_2^{-1}x_4^{-1}x_1^{-1}x_6^{-1}x_3^{-1}x_2x_4x_1x_5x_4^{-1}x_2^{-1}x_5^{-1}x_1^{-1}x_1x_3x_1x_5x_2x_4x_5^{-1}$
$x_1^{-1}x_4^{-1}x_2^{-1}x_3^{-1}x_1^{-1}x_3x_5x_2x_6x_4x_5x_6^{-1}x_2^{-1}x_5^{-1}x_4^{-1}x_5^{-1}x_3^{-1}x_4x_5x_2x_6x_5^{-1}x_4^{-1}x_6^{-1}x_2^{-1}x_1x_6x_2x_5x_4x_6x_5^{-1}x_2^{-1}$
$x_6^{-1}x_4^{-1}x_6^{-1}x_1^{-1}x_4x_6x_2x_5x_6^{-1}x_4^{-1}x_5^{-1}x_2^{-1}x_2x_6x_4x_5x_6^{-1}x_2^{-1}x_5^{-1}x_4^{-1}x_3x_5x_4x_5x_2x_6x_5^{-1}x_4^{-1}x_6^{-1}x_2^{-1}x_5^{-1}x_3^{-1}$
$x_2x_5x_4x_6x_5^{-1}x_2^{-1}x_6^{-1}x_4^{-1}x_1x_6x_4x_6x_2x_5x_6^{-1}x_4^{-1}x_5^{-1}x_2^{-1}x_6^{-1}x_1^{-1}$, where x_1 and x_2 are red, x_3 and x_4 are green, and x_5 and x_6 are blue.