The Fewest Clues Problem

Erik D. Demaine[1], Fermi Ma[2], Ariel Schvartzman[3], Erik Waingarten[4], and Scott Aaronson[5]

1 **MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St, Cambridge, MA 02139, USA**
edemaine@mit.edu
2 **Department of Computer Science, Princeton University, 35 Olden St, Princeton, NJ 08544, USA**
fermim@princeton.edu
3 **Department of Computer Science, Princeton University, 35 Olden St, Princeton, NJ 08544, USA**
acohenca@cs.princeton.edu
4 **Department of Computer Science, Columbia University 1214 Amsterdam Ave, New York, NY 10027, USA**
eaw@cs.columbia.edu
5 **MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St, Cambridge, MA 02139, USA**
aaronson@csail.mit.edu

Abstract

When analyzing the computational complexity of well-known puzzles, most papers consider the algorithmic challenge of *solving* a given instance of (a generalized form of) the puzzle. We take a different approach by analyzing the computational complexity of designing a "good" puzzle. We assume a puzzle maker designs part of an instance, but before publishing it, wants to ensure that the puzzle has a unique solution. Given a puzzle, we introduce the FCP (fewest clues problem) version of the problem:

> Given an instance to a puzzle, what is the minimum number of clues we must add in order to make the instance uniquely solvable?

We analyze this question for the Nikoli puzzles Sudoku, Shakashaka, and Akari. Solving these puzzles is NP-complete, and we show their FCP versions are Σ_2^P-complete. Along the way, we show that the FCP versions of 3SAT, 1-IN-3 SAT, TRIANGLE PARTITION, PLANAR 3SAT, and LATIN SQUARE are all Σ_2^P-complete. We show that even problems in P have difficult FCP versions, sometimes even Σ_2^P-complete, though *"closed under cluing"* problems are in the (presumably) smaller class NP; for example, FCP 2SAT is NP-complete.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases computational complexity, pencil-and-paper puzzles, hardness reductions

Digital Object Identifier 10.4230/LIPIcs.FUN.2016.12

1 Introduction

A natural aesthetic for designing pencil-and-paper puzzles is to provide as few starting hints as possible, yet have the resulting solution be unique. For example, in the well-known Sudoku puzzle, the starting configuration is a partially filled 9×9 grid where the goal is to fill in the remaining entries. Hard Sudoku puzzles tend to give very few numbers in the starting

configuration, but must still give enough numbers to ensure that the puzzle has a unique solution. The natural question here is, how small can we make the starting configuration?

In this paper, we formalize this question by the family of "Fewest Clues Problem" (FCP). Given an NP search problem where the certificate is written as a string, we ask: is there a setting of at most k characters of the certificate such that there exists exactly one way to complete the certificate (fill in the remaining characters)?

FCP seems superficially related to two other classes of problems, the "Another Solution Problem" (ASP) and counting (#) problems [7, 10, 11, 12, 13]. The ASP problem asks: given an instance as well as one solution to a search problem, is there another solution? Counting problems ask: given an instance, how many solutions are there? All three of these problem types are transformations of an original NP search problem.

ASP versions of NP-hard problems are in NP by definition, and interestingly, there are NP-hard problems whose ASP versions are NP-hard as well. The counting version of a problem can be significantly harder the original problem. It is well known that some problems in P (such as 2SAT) have #P-hard counting versions, and Toda's theorem states that such problems are as hard as the polynomial hierarchy [9].

Our results. This paper has three main contributions:

1. We introduce the FCP framework for analyzing puzzles, and we develop simple techniques to adapt existing hardness reductions to the FCP setting (Section 2). With these techniques, we show that the FCP versions of common NP-complete problems are Σ_2^P-complete. These include FCP 3SAT, FCP 1-in-3 SAT, FCP Triangle Partition, FCP Planar 3SAT, and FCP Latin Square (Section 3.2).
2. We show that the FCP versions of three common Nikoli puzzles (Sudoku, Shakashaka, and Akari) are Σ_2^P-complete by reducing from the above problems (Section 4). Figure 1 shows a chain of reductions which allow us to conclude these problems are Σ_2^P-complete.
3. We analyze how the computational complexity of problems in P changes when we consider their FCP versions (Section 5). For a class of problems *closed under cluing*, their FCP versions are in NP. In addition, we show FCP 2SAT, which naturally falls in the class of problems closed under cluing, is NP-complete. We give an example of a problem in P whose FCP version is Σ_2^P-complete.

2 Definitions and Overview

2.1 FCP Framework

▶ **Definition 1.** For each $A \in$ NP, we associate an NP relation, R_A the set of instance-certificate pairs.

For example, the NP relation for the Boolean satisfiability problem (SAT) consists of pairs (ϕ, x) where ϕ is a Boolean formula and x is an assignment to the variables where $\phi(x)$ evaluates to true. We assume that instances and certificates are strings from some alphabet Σ. From now on, when referring to some $A \in$ NP, we implicitly consider the NP relation, R_A.

If the string x is a solution to a SAT problem, we call a "partially filled" version of x (constructed by blocking out certain characters of x with a $\perp$ symbol) a *clue* for that problem. We define this notion for all NP problems as follows.

▶ **Definition 2.** For any instance I of a problem $A \in$ NP, a clue is a string $c = (c_i) \in (\Sigma \cup \{\perp\})^*$ if there exists some certificate string $y = (y_i) \in \Sigma^*$ with $(I, y) \in A$ such that if

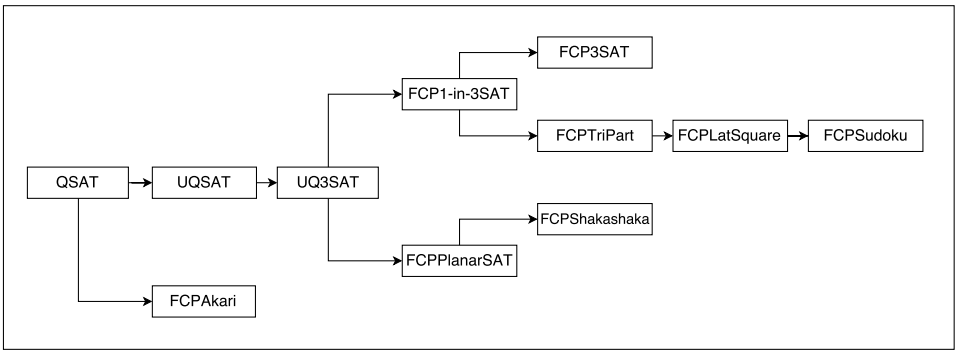


Figure 1 Chain of reductions from QSAT to FCP versions of satisfiability problems and FCP versions of pencil-and-paper puzzles.

$c_i \neq \perp$, then $c_i = y_i$. The size of a clue is the number of non-$\perp$ characters. We refer to this y as a solution which *satisfies* the clue c, and denote this as $c \subset y$.

▶ **Definition 3.** For $A \in$ NP, FCP A is the following decision problem:

Given $x \in \Sigma^*$ and an integer k, does there exist a clue c of size at most k with only one satisfying solution to x?

2.2 FCP Versions of Classic Problems

In Section 3.1, we show that the FCP version of any NP problem is in Σ_2^P. Our more compelling result is that the FCP versions of many NP-complete problems are themselves Σ_2^P-complete. We show this by using a variety of techniques that modify existing NP-completeness reductions to give Σ_2^P-completeness results for the corresponding FCP problems.

We first prove Σ_2^P-completeness of the FCP versions of many common SAT variants. This will be useful since many known NP-completeness reductions are from variants of SAT. We show that FCP 3SAT, FCP 1-in-3 SAT, and FCP Planar 3SAT are all Σ_2^P-complete. We obtain these hardness results with a chain of reductions, starting with the canonical Σ_2^P-complete problem, quantified satisfiability with one pair of alternating quantifiers, $\exists$ and $\forall$ (QSAT). The chain of reductions for all problems defined in this subsection and Section 2.3 are shown in Figure 1.

A slight issue with constructing this chain of reductions is that QSAT does not have any notion of uniqueness, which is essential in FCP problems. We overcome this by defining the unique quantified satisfiability problem (UQSAT), a problem we show is Σ_2^P-complete. Unfortunately, the reduction will introduce clauses which are longer than three literals. For our hardness reductions, we often want to work with 3-CNF formulae. Thus we introduce UQ3SAT, another provably Σ_2^P-complete problem.

For the following three problems, $\phi(x, y)$ is a Boolean function in CNF form on the sets of variables x and y.

QSAT: Given $\phi(x, y)$, does there exist an assignment of the variables in x, such that for all assignments of variables in y, $\phi(x, y) = 1$?

UQSAT: Given $\phi(x, y)$, does there exist an assignment of the variables in x, such that there exists a unique assignment of the variables in y with $\phi(x, y) = 1$?

UQ3SAT: Given $\phi(x, y)$ in 3-CNF form, does there exist an assignment of the variables in x, such that there exists a unique assignment of the variables in y with $\phi(x, y) = 1$?

The transition to FCP problems relies on the following observation: while UQ3SAT specifies which variables can be assigned, FCP 3SAT allows any k variables to be assigned.

We address this distinction by building *assign gadgets*; these gadgets are built so that specifying some part of them fixes in place other aspects of the reduction. These assign gadgets also have the property of having inherent ambiguity. Thus, any clue must specify something within the assign gadget to resolve this ambiguity, or else a unique solution will not be possible. If we place exactly k assign gadgets, we will need a clue of size at least k. This technique will be useful for reducing from non-FCP problems to FCP problems.

For the following three problems, $\phi(x)$ is a Boolean formula in 3-CNF form.

FCP 1-IN-3 SAT: Given $\phi(x)$ and a number k, does there exist a partial assignment of at most k variables such that the remaining formula ϕ' has only one satisfying assignment, where each clause has exactly one true literal?

FCP PLANAR 3SAT: Given a planar $\phi(x)$ and a number k, does there exist a partial assignment of at most k variables such that the remaining formula ϕ' has only one satisfying assignment?

FCP 3SAT: Given $\phi(x)$ and a number k, does there exist a partial assignment of at most k variables such that the remaining formula ϕ' has only one satisfying assignment?

We use assign gadgets to show FCP 1-in-3 SAT is Σ_2^P-complete. The same technique will show that FCP Planar 3SAT is also Σ_2^P-complete.

In some cases, an NP-hardness reduction from problem A to problem B may already preserve clue structure without needing any modifications. If FCP A is Σ_2^P-hard, then the same reduction implies that FCP B is Σ_2^P-hard. This method will show that FCP 3SAT is Σ_2^P-complete.

2.3 FCP Versions of NP-hard Puzzles

In this section, we introduce a number of NP-hard pencil-and-paper puzzles. We modify NP-hardness reductions for these problems to show that their FCP versions are Σ_2^P-hard. These problems were chosen because their NP-hardness reductions mostly preserve clue structure. Figure 1 summarizes the chain of reductions.

FCP LATIN SQUARE: Given a partially filled Latin Square and a number k, do there exist k additional squares to fill in such that the remaining puzzle has a unique solution?

FCP SUDOKU: Given a partially filled Sudoku board and a number k, do there exist k additional squares to fill in such that the remaining puzzle has a unique solution?

FCP SHAKASHAKA: Given a Shakashaka board and a number k, do there exist k clues for the board such that the remaining board has a unique solution?

FCP AKARI: Given an Akari board and a number k, do there exist k clues to the board such that the remaining board has a unique solution?

There is a fundamental difference between Shakashaka and Akari, and Sudoku. In the former, there is a clear distinction between problem instance and solution, whereas in the latter, this distinction is less clear. In particular, filling in a clue in Sudoku and Latin Squares simply produces a new instance of the problem, whereas filing in clues for Shakashaka and Akari do not create new problem instances.

2.4 FCP Versions of Easy Problems

It is also interesting to consider the FCP transformation applied to problems in P. Here, we do not find many general hardness bounds; instead, the results are problem-dependent.

We provide an upper bound for the class of problems in P where filling in a clue results in another instance of the problem. The FCP versions of such problems are always in NP. As an example of such a problem, consider 2SAT.

> FCP 2SAT: Given a Boolean formula $\phi(x)$ written in 2-CNF form, and a number k, does there exist a partial assignment of at most k variables such that the remaining formula ϕ' has only one satisfying assignment?

In this problem, filling in a clue means assigning truth values to certain variables of ϕ, which results in another instance of 2SAT. Thus, our upper bound will imply that FCP 2SAT is in NP. In fact, this bound is tight as we show that FCP 2SAT is NP-complete by reduction from Minimum Independent Dominating Set.

However, for problems in P that do not have this property, the hardness of their FCP versions cannot be so easily characterized. This happens because we can modify hard problems by planting solutions to make them "easy", while maintaining the hardness of their FCP versions. We give a simple example of a problem in P whose FCP version is Σ_2^P-hard.

3 The FCP Transformation

3.1 Upper Bounds

▶ **Theorem 4.** *If $L \in$ NP, then FCP $L \in \Sigma_2^P$.*

Proof. We write FCP L as an instance of QSAT, a problem in Σ_2^P. The original problem of deciding membership in L can be written as

$$x \in L \leftrightarrow \exists y : A(x, y) = 1,$$

where A is a polynomial time algorithm and $|y|$ is polynomial in $|x|$.

FCP L asks: for a given instance x and positive integer k, does there exist a clue c of size at most k such that there is only one solution y with $c \subset y$ and $(x, y) \in R$? We write this in logical notation as

$$(x, k) \in \text{FCP } L \leftrightarrow \exists c, y : \forall y' : A'(x, c, y, y') = 1.$$

A' simply checks that $c \subset y$, c is of size at most k (c is a valid clue of the correct size), $c \subset y' \neq y$ (y' is another possible solution), $A(x, y) = 1$ (y is a solution), and $A(x, y') = 0$ (all other possible solutions do not work). ◀

3.2 Lower Bounds

In this subsection we present a chain of Σ_2^P-hardness reductions starting with UQSAT and UQ3SAT. These results allow us to transition to Σ_2^P-hardness proofs of various FCP problems.

▶ **Lemma 5.** UQSAT *is Σ_2^P-hard.*

Proof. We reduce from QSAT. We define a function f from instances of QSAT to instances of UQSAT which maps $\phi(x, y)$ to $\phi'(x, y, z)$, where z is one additional variable. We show

$$\exists x : \forall y : \phi(x, y) = 1 \leftrightarrow \exists x : \phi'(x, y, z) \text{ has a unique solution.}$$

Let $\phi'(x, y, z) = (\overline{z} \wedge \bigwedge_i \overline{y_i}) \vee \overline{\phi(x, y)}$.

Note that for each assignment of x, $\phi'(x, 0, 0) = 1$. If there exists some x for which $\phi(x, y) = 1$ always, then for this x, $\overline{\phi(x, y)} = 0$ always. For this x, the only way to satisfy ϕ' is to set z and each y_i to 0. It remains to rewrite ϕ' as a CNF formula.

The given $\phi(x, y)$ is in CNF form, so we write $\overline{\phi(x, y)}$ in DNF form using De Morgan's laws. For the ith clause, we introduce the variable c_i to represent whether that clause is satisfied. If there are n clauses, $\overline{\phi(x, y)}$ becomes $(c_1 \vee c_2 \vee \cdots \vee c_n)$. We write the jth literal in the ith clause as $l_{i,j}$, and introduce A as one additional variable. Then the following formula is logically equivalent to ϕ':

$$\phi''(x, y, z, \{c_i\}, A) = (\overline{A} \vee \overline{z}) \wedge \bigwedge_i (\overline{A} \vee \overline{y_i}) \wedge \bigwedge_i \bigwedge_j (\overline{c_i} \vee l_{i,j}) \wedge (A \vee \bigvee_i c_i).$$

Then the following two claims complete the proof.

Claim. Suppose for some assignment x, $\phi(x, y) = 1$ for all y. Then $\phi''(x, y, z, \{c_i\}, A)$ has a unique satisfying assignment.

Since $\phi(x, y) = 1$, $\overline{\phi(x, y)} = 0$. This implies that each $l_{i,j} = 0$ and therefore each $c_i = 0$. This forces $A = 1$, $z = 0$, and $y_i = 0$ for all i.

Claim. If for each x there exists some y where $\phi(x, y) = 0$, then after fixing x, $\phi''(x, y, z, \{c_i\}, A)$ has more than one satisfying assignment.

Setting each $y_i = 0$, $z = 0$, $c_i = 0$ for each i and $A = 1$ gives one satisfying assignment. If there is a setting of the y_i's which makes some of the clauses true, then for the true clauses set $c_i = 1$. Then set $A = 0$ and z to either 0 or 1. This gives additional satisfying assignments for ϕ''. ◀

▶ **Lemma 6.** UQ3SAT *is* Σ_2^P*-hard.*

Proof. The reduction is from UQSAT. Given $\phi(x, y)$, an instance of UQSAT, it remains to transform $\phi(x, y)$ into 3-CNF form. For clauses in $\phi(x, y)$ of length $m > 3$, we imagine a binary tree of height $\log m$ where leaves correspond to clause literals. For all interior nodes of the tree, we introduce a new variable to act as the OR of its two children. If an interior node has children i_1 and i_2, we introduce the variable o along with the following clauses:

$$(\overline{i_1} \vee \overline{i_2} \vee o), \quad (\overline{i_1} \vee i_2 \vee o), \quad (i_1 \vee \overline{i_2} \vee o), \quad (i_1 \vee i_2 \vee \overline{o}).$$

These clauses ensure that o is the OR of i_1 and i_2. Additionally, we add the variable at the root of the tree as a new clause of size 1 to capture the original clause exactly.

To handle clauses with fewer than three literals, we pad the clause with the additional variable a along with the clause $(\overline{a} \vee \overline{a} \vee \overline{a})$ to ensure that a is set to false. ◀

▶ **Lemma 7.** FCP 1-IN-3 SAT *is* Σ_2^P*-complete.*

Proof. The reduction is from UQ3SAT. We use the reduction from Proposition 3.3 in [6] which is a parsimonious reduction from PLANAR 3SAT to POSITIVE PLANAR 1-IN-3 SAT. For this problem, we disregard the planarity and note that the reduction works to map non-planar instances of 3SAT to 1-IN-3 SAT.

We modify the reduction slightly by constructing the assign gadget from Figure 2 for each variable in x. This assign gadget is designed so that any clue to the problem must resolve an ambiguity within the gadget, forcing every clue to assign a value to each variable in x. Therefore, even though FCP 1-IN-3 SAT can give clues for any variable, because of the assign gadgets, clues will only correspond to variables with existential quantifiers in UQ3SAT. ◀

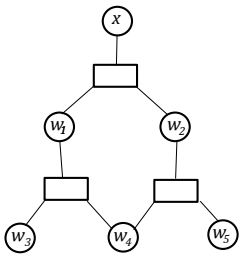


Figure 2 Clue gadget for 1-IN-3 SAT. Each box represents a 1-IN-3 SAT clause with variables corresponding to the three lines connected to the box. x is set to true by setting w_4 to true. x is set to false by setting w_1 or w_2 to true (but not both). If none of the w_i in the clue gadget is assigned, there exists more than one satisfying assignment.

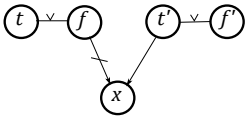


Figure 3 FCP PLANAR 3SAT assign gadget for variable x. The above diagram is written with 4 clauses: $(t \vee f) \wedge (\overline{f} \vee \overline{x}) \wedge (\overline{t'} \vee x) \wedge (t' \vee f')$. The arrows indicate "causal" relationships (if t' is set to true, then x must be assigned to true; if f is set to true, then x must be assigned to false). In order to assign x to true, set f' to false. In order to assign x to false, set t to false. If neither of the t, t', f, or f' variables are set, then the solution is not unique.

▶ **Lemma 8.** *FCP* 3SAT *is* Σ_2^P*-complete.*

Proof. We reduce from FCP 1-IN-3 SAT. For each clause of the form $(l_1 \vee l_2 \vee l_3)$ in the FCP 1-IN-3 SAT instance, we construct the following clauses:

$$(l_1 \vee l_2 \vee l_3), \quad (l_1 \vee \overline{l_2} \vee \overline{l_3}), \quad (l_2 \vee \overline{l_1} \vee \overline{l_3}), \quad (l_3 \vee \overline{l_1} \vee \overline{l_2}), \quad (\overline{l_1} \vee \overline{l_2} \vee \overline{l_3}).$$

Note that if $(l_1 \vee l_2 \vee l_3)$ is true in the 1-IN-3 SAT configuration, all of the above clauses are satisfied. Conversely, if all of the above clauses are satisfied, then the clause $(l_1 \vee l_2 \vee l_3)$ has exactly one variable set to true. ◀

▶ **Lemma 9.** *FCP* PLANAR 3SAT *is* Σ_2^P*-complete.*

Proof. The reduction is from UQ3SAT. We use the reduction from [4] showing PLANAR 3SAT is NP-complete. The only change is that we introduce clauses to form an assign gadget as shown in Figure 3. The assign gadget guarantees variables with existential quantifiers in UQ3SAT are given clues. ◀

4 FCP Versions of NP-hard Puzzles

In this subsection we show that the FCP versions of popular puzzles are Σ_2^P-complete. It is worth noting that all problems considered in this section are in NP, so their FCP versions are in Σ_2^P due to Theorem 4. Therefore, we focus on obtaining Σ_2^P-hardness reductions.

4.1 Triangle Partition

The Triangle Partition problem is the following:

Given an undirected graph $G = (V, E)$, can we partition E into disjoint triangles?

Holyer [3] showed this problem is NP-complete. We use elements of Holyer's reduction to show FCP Triangle Partition is Σ_2^P-complete. We use the notation as well as elements from the reduction in [3].

▶ **Theorem 10.** *FCP Triangle Partition is Σ_2^P-complete.*

Proof. In order to show FCP Triangle Partition is Σ_2^P-hard, we reduce from FCP 1-IN-3 SAT. We use Holyer's reduction, which reduces 3SAT to Triangle Partition [3]. For the FCP setting, we switch from 3 SAT to 1-IN-3 SAT so that it is not advantageous to provide clues within clause gadgets. The other modification we make is that instead of a single two-way join between literals and variables, we make two two-way joins in other to enforce the variable matches the literal exactly. Since the reduction is from 3SAT in [3], and the three-way join enforces a 1-IN-3 SAT constraint, [3] cannot map the literals and variables directly. Since we are reducing from 1-IN-3 SAT constraints, we can make the literals and variable gadgets be the exact same partition.

This completes the proof, as now a clue on variables in the 1-IN-3 SAT instance corresponds exactly to a clue of the same size on the variable gadgets in the Triangle Partition instance. ◀

4.2 Latin Squares

The Latin Squares problem is the following:

> Given an $n \times n$ grid, with some entries filled in with the numbers 1 to n, can we fill in the remainder of the grid with numbers from 1 to n so that no row or column repeats a number?

Colbourn proved this problem was NP-complete by showing it is equivalent to the Triangle Partition problem restricted to tripartite graphs [1]. We show FCP Latin Squares is Σ_2^P-complete.

▶ **Theorem 11.** *FCP Latin Squares is Σ_2^P-complete.*

Proof. We show hardness via a reduction from FCP Triangle Partition of Tripartite Graphs which is Σ_2^P-complete following Theorem 10 and [1]. The reduction is identical to the one in [1]. The mapping sends Latin Squares entries of the form $L(i, j) = m$ to triangles with vertices (i, j, m) on different parts of the partition, where $L(i, j)$ is the entry on the i-th row and j-th column of the Latin Square L. The mapping between entries and triangles is bijective, implying that a clue of size k to the Latin Squares problem will produce a clue of size k in the Triangle Partition problem. ◀

4.3 Sudoku

The Sudoku problem is as follows:

> Given an $n^2 \times n^2$ grid, divided into n^2 subgrids of size $n \times n$, can we fill in the grid with numbers from $[n^2]$ so that no row, column, or any of the n^2 subgrids contain any repeated numbers?

We will show the FCP problem is Σ_2^P-complete by reduction from FCP Latin Squares. The Latin Square is embedded into the Sudoku puzzle. The columns of the Latin Square are mapped to the parts of columns of the Sudoku. In the mapping, the values are scaled by n.

▶ **Lemma 12.** *FCP Sudoku is Σ_2^P-complete.*

Proof. We note the same reduction as [8] will serve as a reduction from FCP Latin Square to FCP Sudoku. Each entry in the Sudoku puzzle is mapped to an entry in the Latin Squares problem, after diving by n. Therefore, a clue of size k in the constructed Sudoku puzzles can be mapped to a clue of size k in the Latin Squares problem. Figure 4 shows the case when $n = 3$. ◀

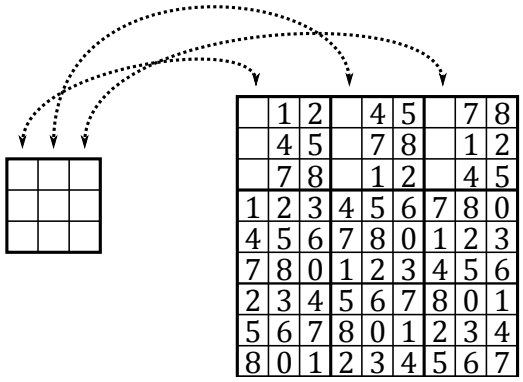


Figure 4 Reduction from Latin Square to Sudoku for $n = 3$ from [8].

4.4 Shakashaka

Shakashaka is a pencil-and-paper puzzle published by Nikoli. A Shakashaka puzzle consists of a rectangular grid of black and white squares. Black squares either contain an integer from $\{0, 1, 2, 3, 4\}$ or can be left blank. Each white square is filled in with a black triangle or is left blank. The filled-in squares become half-black with right isosceles triangles in any of four possible directions (referred to as b/w squares in [2]). The resulting configuration must satisfy the following constraints: each black square with a number c must have c adjacent black isosceles triangles, and the remaining white area must be partitioned into (possibly rotated) rectangles.

The problem is shown to be NP-complete in [2]. The same reduction will show that FCP Shakashaka is Σ_2^P-complete.

▶ **Lemma 13.** *FCP Shakashaka is Σ_2^P-complete.*

Proof. We use the reduction from [2] to reduce from FCP Planar 3SAT. It remains to show that all clue structure of the original problem is preserved in the reduction. Shakashaka entries are comprised of assignments (either to one of the four possible triangle orientations or to be left blank) to the white squares. The key observation is that any assignment to a white square in the reduction can be implied by an assignment in the variable gadget of [2]. Therefore, we can assume without loss of generality that each assignment in an entry is in a unique variable gadget. Thus, the entries correspond exactly to assignments on variables in FCP Planar 3SAT, completing the reduction. ◀

4.5 Akari

Akari, or Light Up, is a pencil-and-paper puzzle consisting of a grid with light and dark squares. Dark squares may be labelled with integers from $\{0, 1, 2, 3, 4\}$. The player places lightbulbs on light squares which illuminate all light squares in the four directions around the lightbulb (although light stops upon a dark square). Equivalently, a square is lit if there exists at least one light bulb connected to it by a straight line of light squares. The goal is to place lightbulbs so that every light square is lit, no two light bulbs illuminate each other, and every numbered dark square has exactly that many light bulbs adjacent to it.

This problem is known to be NP-complete. We show that its FCP version is Σ_2-complete.

▶ **Lemma 14.** *FCP Akari is Σ_2^P-complete.*

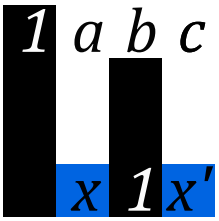


Figure 5 Akari entry gadget. If x or x' is lit, then there is ambiguity regarding placing the light in b or c, in the case of x, and a or b in the case of x'. Lighting up a means x' must be lit, and lighting up c means x must be lit.

Proof. In order to show FCP Akari is Σ_2^P-complete, we reduce from Σ_2 Circuit SAT, using the reduction from [5]. We add the following assign gadgets to a wire to assign the necessary variables. The gadget is shown in Figure 5. Variables with existential quantifiers will need to receive clues to resolve ambiguity. Therefore, clues in Akari will only be given to variables corresponding to existential quantifiers in Σ_2 Circuit SAT. ◀

5 FCP Versions of Easy Problems

In this section, we consider the complexity of the FCP versions of various problems in P.

5.1 Problems Closed under Cluing

▶ **Definition 15.** We call a problem *closed under cluing* if plugging any partial solution into a problem instance results in another instance of the same problem.

For example, 2SAT is closed under cluing since plugging in values for some of the variables will result in another instance of 2SAT.

▶ **Proposition 16.** *If A is a problem closed under cluing in P, then* FCP $A \in$ NP.

Proof. We give the following nondeterministic polynomial time algorithm. FCP A asks us to find a clue set of size at most k. First, nondeterministically pick a clue set of size k. Then for each string character in the solution that has not been assigned a value, try all possible settings of that character and accept if only one setting of that character produces a solvable problem. This step relies on closedness, as we use the fact that the problems that result from setting certain characters of the solution are still solvable in polynomial time.

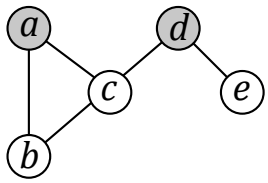
If for any string character we find that multiple settings give a solvable problem, then our clue set does not give a unique solution. If no setting gives a solvable problem, then our clue set is not valid since no solution is possible. ◀

5.2 FCP 2SAT

Proposition 16 states that FCP 2SAT $\in$ NP. Here, we show that this problem is in fact NP-complete.

▶ **Theorem 17.** *FCP* 2SAT *is NP-complete.*

We reduce from the NP-hard problem, Minimum Independent Dominating Set (MIDS), which asks whether a graph has an independent dominating set of size at most k.



$$(\neg a \vee \neg b) \wedge (\neg a \vee \neg c) \wedge (\neg b \vee \neg c) \wedge (\neg c \vee \neg d) \wedge (\neg d \vee \neg e)$$

Figure 6 Reduction from Minimum Independent Dominating Set to FCP 2SAT.

Given a graph $G = (V, E)$, does there exist $S \subset V$, with $|S| \leq k$ such that, for all $v \in V$, either $v \in S$ or $(u, v) \in E$, $u \in S$, and $u, v \in S$ implies that there is no edge between u and v.

We transform G into a formula ϕ. For each edge (u, v), we add the constraint $(\neg u \vee \neg v)$ to the 2SAT formula. The following two lemmas prove that this reduction is correct.

▶ **Lemma 18.** *Suppose G contains an independent dominating set of size at most k, then there exists a clue of the* 2SAT *formula containing at most k variables.*

Proof. Suppose $S \subset V$ is an independent dominating set of size k. Then for each $s \in S$, assign s to true. We claim the formula is uniquely satisfiable.

If x is some variable in ϕ, then x corresponds to some node in G. Since S is a dominating set, let $s \in S$ be a neighbor of x. Then $(\neg x \vee \neg s)$ is a clause in ϕ. For ϕ to be satisfied, x must be false.

Since S is independent, we have no false clauses. ◀

▶ **Lemma 19.** *Suppose ϕ contains a clue of size k. Then G contains an independent dominating set of size at most k.*

Proof. Let C_T and C_F be the set of variables set to true and false, respectively, in the clue for ϕ. Without loss of generality, we may force C_F to be empty. For any variable x in C_F, consider a clause $(\neg x \vee \neg y)$ in ϕ. We can give a clue of identical size by removing x from C_F and including y in C_T.

Also, we note that all variables assigned to true in the satisfying assignment must be in C_T. If a variable x set to true is not in C_T, uniqueness is violated since setting x to false gives another satisfying assignment.

Let $S = C_T$. S must be an independent set because ϕ is satisfiable. In addition, if $v \in G - S$, then since the satisfying assignment to ϕ is unique, v contains a neighbor whose variable is set to true. So S is dominating. ◀

5.3 FCP versions of other easy problems

Consider the following language:

$$L = \{\phi' = (\phi \wedge \overline{z}) \vee z | \phi \text{ is a Boolean formula and } z \text{ does not appear in } \phi\}.$$

Note that if we ask whether ϕ' is satisfiable, the answer is trivially yes. So $L \in$ P. FCP L asks whether there exists a setting of k variables to ϕ' which makes it uniquely satisfiable.

▶ **Proposition 20.** *FCP L is Σ_2^P-complete.*

Proof. We reduce from FCP SAT. If ϕ has a clue of size at most k with a unique solution, then $\phi' = (\phi \wedge \overline{z}) \vee z$ has a partial assignment of size at most $k+1$ with a unique solution. We simply add the assignment for variable z.

If ϕ has no satisfying assignment, then ϕ' requires at least $k+1$ assigned variables for a partial assignment with a unique solution. Let x be a particular satisfying assignment to ϕ' that requires more than k assigned variables. If ϕ has a partial satisfying assignment $\mathbf{x}$ and $z \in \mathbf{x}$, then $\mathbf{x}$ needs at least n assigned variables to be specified. If $\overline{z} \in \mathbf{x}$, then we must specify $\overline{z}$ in the clue, and by assumption we must specify more than k variables. Therefore, if ϕ needs a clue of size greater than k, ϕ' will need a clue of size greater than $k+1$. ◀

Acknowledgements. This work began as a final project for MIT's Algorithmic Lower Bounds class (6.890) taught by Erik Demaine in Fall 2014. Thanks to Sarah Eisenstat and Jayson Lynch for valuable discussion and insight.

References

1 Charles J Colbourn. The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984.

2 Erik D Demaine, Yoshio Okamoto, Ryuhei Uehara, and Uno Yushi. Computational complexity and an integer programming model of shakashaka. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 97(6):1213–1219, 2014.

3 Ian Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

4 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

5 Brandon McPhail. Light up is NP-complete. *Unpublished manuscript*, 2005.

6 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *CoRR*, abs/cs/0601002, 2006. URL: http://arxiv.org/abs/cs/0601002.

7 Takahiro Seta. The complexities of puzzles, cross sum and their another solution problems (ASP). Senior thesis, Univ. of Tokyo, Dept. of Information Science, Tokyo, Japan, Feb 2001. URL: http://www-imai.is.s.u-tokyo.ac.jp/~seta/paper/senior_thesis/seniorthesis.ps.

8 Yato Takayuki and Seta Takahiro. Complexity and completeness of finding another solution and its application to puzzles. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(5):1052–1060, 2003.

9 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, October 1991. doi:10.1137/0220053.

10 Nobuhisa Ueda and Tadaaki Nagao. NP-completeness results for nonogram via parsimonious reductions. *preprint*, 1996.

11 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

12 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.

13 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(C):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.