Assignment 3: Solve Nonograms

AI-1 Systems Project (Summer Semester 2025)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic:	SAT Solvers
Due on:	August 14, 2025
Version from:	June 10, 2025
Author:	Jan Frederik Schaefer

Make sure you sign up before working on this assignment.[a]
Using someone else's solution code, even as inspiration, is not allowed!

[a] You can still decide to postpone the assignment. Signing up includes an elligibility check, which avoids situations where you invest work into an assignment that you are not supposed to take.

Summary Solve picture puzzles using a SAT solver. The puzzles are a generalization of **nonograms** [WN], which are NP-complete logic puzzles originating in Japan. Besides the traditional rectangular grids, we will also cover hexagonal grids. You will have to convert the puzzle to a SAT problem, solve it with a SAT solver, and then use the solution to the SAT problem to infer the correct coloring of the cells.

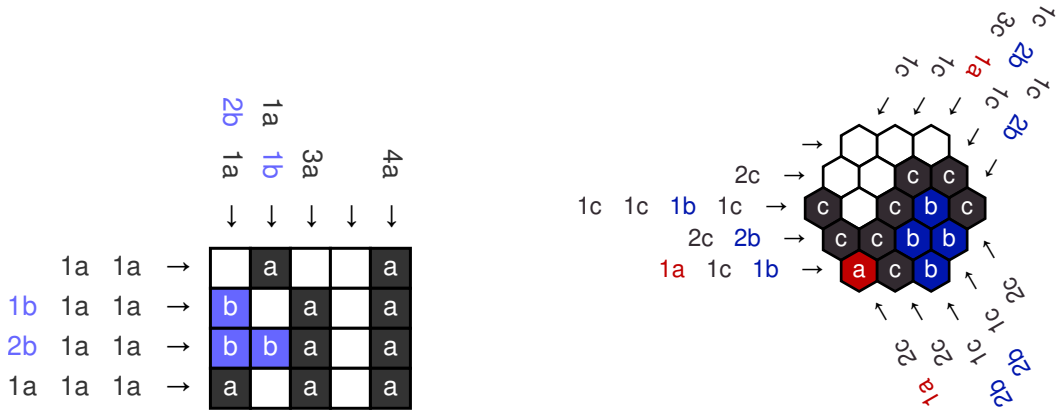
This assignment is more work than the other assignments and is therefore worth more points than the usual 100. Furthermore, you are supposed to also compare the efficiency of two different approaches (see Section 3). The assignment guide [AG] describes different approaches for this assignment (i.e. different ways to encode a nonogram as a SAT problem).

Objectives

1. Gain some experience with encoding a problem as a SAT problem,
2. understand that/why some encodings scale better than others.

Prerequisites and useful methods

1. SAT solving (as discussed in the AI lecture, though you should use an existing solver, i.e. the algorithms for SAT solving are not relevant),
2. Boolean/propositional logic (De Morgan's law, material implication, DNF, CNF, ...).



(a) Rectangular grid.

(b) Hexagonal grid.

Figure 1: Two example generalized nonograms (solved).

1 Puzzle rules

The goal of a (generalized) nonogram is to color the cells of a grid using clues. A **clue** is a sequence $n_1c_1 \ldots n_kc_k$, where n_i are numbers and c_i are color references. Each element n_ic_i indicates a continuous **block** of n_i cells of color c_i. The blocks occur in the same order as listed in the clue. Consecutive blocks of the same color must be separated by at least one empty cell.

As an example, let us take a look at the third row of the example nonogram in Figure 1a. We use letters (here `a` and `b`) to refer to colors. The clue `2b 1a 1a` for the third row indicates that it starts with a 2-cell block of color `b` (in this case blue), which is followed by two 1-cell blocks of color `a` (black). Since the last two blocks have the same color, they must be separated by at least one empty cell.

Incomplete clues It is possible that the length of a block is not specified. Instead, a "+" is used to indicate that the block is at least of length 1.

Simplification: To make the assignment a bit easier, we will only use nonograms with 1 or 2 colors.

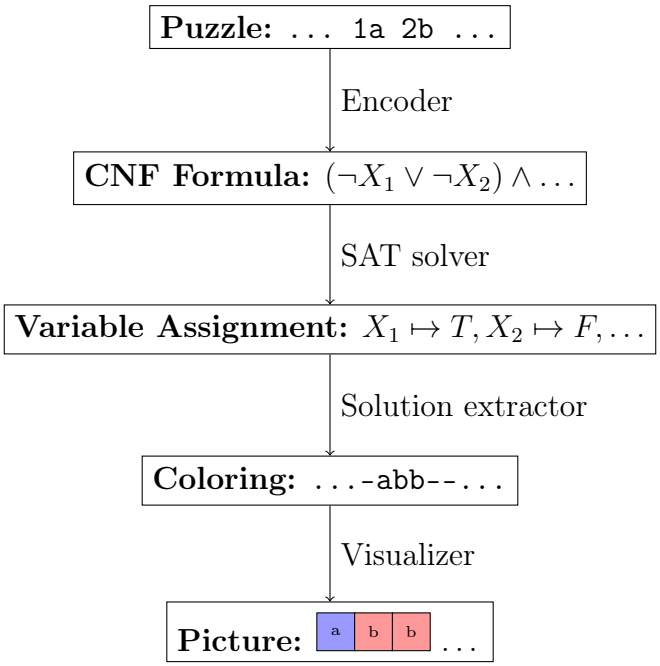


Figure 2: A sequence of steps to solve a nonogram using a SAT solver.

```
rect 4 5
#ffffff #333333 #6666ff
1a 1a
1b 1a 1a
2b 1a 1a
1b 1b 1a
3b
1a 1b
2a 1b

4a
```

```
hex 3
#ffffff #c10000 #0016ad #322a30

2c
1c 1c 1b 1c
2c 2b
1a 1c 1b
1a 2c
2c
2b 1c
2b 1c
2c
1c 2b
1c 2b 1c
3c 1a
1c
1c
```

Listing 1: Encoding of clues for the nonograms from Figure 1a (left) and Figure 1b (right).

2 Nonogram formats and SAT solvers

This section describes how generalized nonograms and their solutions should be represented. We also discuss the basics of using a SAT solver. Figure 2 sketches the pipeline of a solver: You start with an unsolved nonogram as an input (the format is specified in Section 2.1). The main challenge for you will be to translate it into a CNF formula that can be solved with a SAT solver. The assignment guide [AG] sketches different ways how this can be done. A SAT solver (Section 2.3) can provide a variable assignment that satisfies the formula. If you chose a good encoding, it should be easy to infer a solution for the nonogram (i.e. a coloring) from the variable assignment. Section 2.2 specifies a format for storing the solution so that it can be checked automatically. In the following sections, we will explore these steps in more detail.

2.1 Puzzle Format

Puzzles are represented as text files. First, the grid is described:

- Rectangular grids start with the line `rect <height> <width>`, where `<height>` is the number of rows and `<width>` is the number of columns.
- Hexagonal grids start with the line `hex <size>`, where `<size>` indicates the side length of the hexagon.

The second line lists the colors, starting with the background color, followed by the colors that are afterwards referred to as `a`, `b`, `c`, etc.

Each of the remaining lines corresponds to a clue. For a rectangular grid, the row clues are listed first, followed by the column clues. In a hexagonal grid, we have clues in three directions (see Figure 1b). The file lists them counter-clockwise, starting with the top-left corner (`1b 1b` in the example). Listing 1 shows the encodings of the example nonograms.

2.2 Storing, visualizing and checking the results

The solution of a nonogram should be represented as a text file. Each line describes the coloring of one row of cells. Uncolored cells are represented with a `-` and colored cells with the name of the color (`a`, `b`, ...). Listing 2 shows the solutions to the example nonograms from Figure 1.

The assignment repository [AR] contains a script that you can use to check if the solutions are correct. It also contains a script for visualizing the solution similar to the visualizations

```
-a--a
b-a-a
bba-a
b b-a
```

```
---
--cc
c-cbc
ccbb
acb
```

Listing 2: Solution files for the nonograms from Figure 1a (left) and Figure 1b (right).

```
p cnf 3 2
1 2 0
-1 3 0
```

```
-1 2 -3
```

Listing 3: Encoding of $(X_1 \vee X_2) \wedge (\neg X_1 \vee X_3)$ (left) and the solution $X_1 = F, X_2 = T, X_3 = F$ (right).

in Figure 1. More details are provided in the README of the assignment repository.

2.3 Using SAT solvers

To solve the nonograms, you should translate them into a SAT problem that can be solved by off-the-shelf SAT solvers.

We will represent SAT problems using (a subset of) the DIMACS format, which is commonly used for SAT competitions. Listing 3 shows the encoding of an example formula. The first line is always of the form `p cnf <nvars> <nclauses>`, where `<nvars>` is the number of variables used and `<nclauses>` is the number of clauses. Each subsequent line encodes one clause as a list of integers. Positive integers denote variables and negative integers their negations (i.e. the literal X_i is denoted by i and the literal $\neg X_i$ by $-i$). The end of a clause is marked by a 0. Problems in that format can be solved by many SAT solvers, including MiniSat [MS], which is relatively easy to use and install (at least on unix-like systems), and Kissat [KS], which has won recent SAT competitions (see e.g. [S22]). Students have also successfully used the Python library PySAT [PS] in the past, which was easier to install on Windows and is easier to use from Python code.

Variable assignments that satisfy the problem are also encoded as a list of integers, where positive integers indicate variables that set to true and negative integers indicate variables set to false (see Listing 3 for an example). Both MiniSat and Kissat produce solutions in that format.

3 Comparing different approaches

This assignment is a bit different from the other ones. Encoding the nonograms as a SAT problem is not easy, especially if you are not used to working with a SAT solver. In the past, students typically came up with a very inefficient encoding, which did not scale to larger problems, and various optimizations could not improve the efficiency on a fundamental level. To avoid this, we sketch different approaches how the nonograms can be encoded as a SAT problem in the assignment guide [AG]. We would like you to pick two different approaches – either from the assignment guide or your own – and compare them. If you come up with your own approach, please make sure that the approaches are different enough to allow for an interesting comparison. If you are unsure, please ask us! You are also welcome to tweak the suggested approaches in any way you see fit.

In the solution summary, estimate how well these two approaches would scale to larger nonograms. You do not have to indicate a very precise complexity class (that is a bit tricky in some cases, though it is great if you manage), but you should determine which one would scale better (and why). You should also implement both approaches (for rectangular single-color nonograms) and measure how efficient they are. The assignment repository [AR] contains nonograms of different sizes that you can use for the measurements. Include the results in your solution summary (e.g. as a plot).

As this evaluation makes the solution summary longer and more work, it will also be worth more points (see Section 5).

4 What to submit

At the deadline, we will download a snapshot of your repository. It should contain:

1. All your code for solving this assignment.
2. A `README.md` file explaining
   i. dependencies (programming language, version, external libraries and how to get them),
   ii. how to run your code to solve a nonogram,
   iii. the repository structure,
   iv. anything else we should know.
3. Solutions to all the test nonograms (`clues` directory in the assignment repository [AR]). That means that you should have a solution file `<name>.solution` as described in

Section 2.2 for every problem `<name>.clues`.

4. A solution summary (see [SoS] for more details – it should describe the main ideas, not document the code). In this assignment, it should also include estimates of how efficient approaches are and a plot with measurements for the efficiency of two different approaches (see Section 3 for details). A typical solution summary should be 1–3 pages long, but it is okay if your evaluation is much longer (it would just be more work from you than we expect).

5 Points

This assignment is worth 120 points. 90 points are awarded for the solutions to the nonograms in the assignment repository. Concretely, you will get $\lceil 18\sqrt{n} \rceil$ points where n is the number of correctly solved nonograms (there are 25 in total). The remaining 30 points are awarded for the quality of your submission, the README, and the solution summary.

If the grading scheme does not seem to work well, we might adjust it later on (likely in your favor).

References

[AG] *Guide for "Assignment 3: Solve Nonograms"*. URL: https://kwarc.info/teaching/AISysProj/SS25/assignment-1.3-guide.pdf.

[AR] *Repository for Assignment 3: Solve Nonograms*. URL: https://gitlab.rrze.fau.de/wrv/AISysProj/ss25/a1.3-solve-nonograms/assignment.

[KS] *Kissat SAT Solver*. URL: http://fmv.jku.at/kissat/ (visited on 01/18/2022).

[MS] *The MiniSat Page*. URL: http://minisat.se/ (visited on 01/18/2022).

[PS] *PySAT*. URL: https://pysathq.github.io/ (visited on 05/25/2024).

[S22] *SAT Competition 2022: Results*. URL: https://satcompetition.github.io/2022/results.html (visited on 07/21/2023).

[SoS] *Solution Summary*. URL: https://gitlab.rrze.fau.de/wrv/AISysProj/admin/general/-/blob/main/solution-summary.md.

[WN] *Nonogram*. URL: https://en.wikipedia.org/wiki/Nonogram (visited on 01/18/2022).