Theory of Computation
Assignment 6: SAT Encoding

Brites Marto Andrea Le Thuong
Rodolfo Masera Tommaso Taillefert Stefano

1 Installation and Instructions

2 Problem Design and Interpretation

First and foremost, this problem is very loosely defined which means that it was mostly up to us to come up with constraints or with a problem size over the specified input of a pair of garments and colours: $\langle g, c \rangle$ where $g \in G$ and $c \in C$ where G and C are sets that include garments and colours respectively, both of size 10, and we can define them as follows:

$$G = \{\text{pants, shirt, hat, jacket, sweater, gloves, shoes, tie, scarf, shorts}\}$$
$$S = \{\text{red, yellow, orange, green, blue, purple, brown, pink, white, black}\}$$

We chose a small size for the sake of simplicity of the project and for a more realistic aspect. We will go over this part in more detail in **Section 4**.
Given these two sets, we devised constraints over them that will always be added. They are hardcoded as they specify, for instance, which garments (or colours) should or should not go together. We define these constraints as follows in a boolean way:

Boolean Constraints
$\neg$(yellow $\wedge$ white)
$\neg$(blue $\wedge$ purple)
$\neg$(blue $\wedge$ black)
$\neg$(red $\wedge$ green)
$\neg$(red $\wedge$ orange)
$\neg$(green $\wedge$ pink)
$\neg$(green $\wedge$ orange)
$\neg$(pants $\wedge$ shorts)
$\neg$(shorts $\wedge$ jacket)
scarf $\rightarrow$ jacket
gloves $\rightarrow$ jacket
tie $\rightarrow$ shirt

Finally, we go over the given input file and we determine

3 Implementation

3.1 SAT Solver

The main core of the project is the solver program (`main.py`), written entirely in Python. We used the `z3-solver` library (`https://github.com/Z3Prover/z3`) to get most of the job done since it provided us with the functionalities we needed and was very straightforward to use. We also used `pandas` to read the input file. No, we did not ask a bunch of fluffy animals to read out the text, it's a Python library. Come on.
Our program first puts the input pairs of garments and colors into a set, to filter out duplicates. Then, we build the constraints list as detailed in section 2 and we check whether the model is satisfiable or not. If it's the case, we print out the solution.

3.2 User Interface

We needed to quickly develop a solid interface; the perfect candidate for this, given our skills, was a webpage. Therefore, given that the rest of the code was already in Python, we resorted to `Flask`, which is a simple yet powerful library to make web applications.
The script (`app/app.py`) simply renders an HTML page with two inputs to compose the mannequin and a canvas to represent the solution, which is obtained by calling our solver with the given input data.

3.3 Deployment

Since we built a web interface, we thought it would be a good idea to deploy our project as a fully functional website. For this reason, we purchased the domain `bestsatsolver.xyz`, packed our app in a Docker container and put it on a small server of ours. The whole process was pretty straightforward, given that our project has a very simple structure and doesn't depend on other services like databases or external APIs.

4 Issues Encountered

5 Conclusions