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 the final constraints. If one of the garments is missing, we make sure that they are put in a **NOT** boolean operator to take their absence into account otherwise, if present, we simply create an **OR** operator of an **AND** over a pair comprising a garment g and its color c that we are given and we evaluate it with respect to our constraints. So, basically, we would either obtain, over an input $\langle g, c\rangle$, if absent, $\neg g$ or $(g \wedge c)$ otherwise.

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

Other than the typical issues that may arise when programming (i.e. bugs, code design, etc...), the main issue that we encountered ended up being caused by the fact that the problem assigned to us was loosely defined. This meant that we had freedom to pick on how to develop it but that, also, we had to make sure to keep it complete while not overcomplicating it.

As we had described in **Section 2**, we decided to make the size of the problem relatively small by limiting the possible colors and garments to a set of ten elements each. Along with the fact that, in reality, oftentimes we would not expect to have to dress a mannequin with thousands of garments and colours, we also thought that it would be easier to define our own sets and constraints over them rather than to allow variable inputs.

The main issue that would have come with a variable input where you would allow the input to decide how many garments and colours would be available (along with the subsequent possible pairs) would have been to define the constraints over the sets of colours and garments.

For instance, if we take our own sets, we have 10 garments and 10 colours and we have added constraints with the knowledge of those garments and colours.
If for example we had x garments and y colours where $x, y \in \mathbb{N}$, we then would first of all have $x \cdot y$ possible pairs and, furthermore, defining the constraints over them would be incredibly tough and we might have to take a randomized approach where, given x garments and y colours and a set S containing a pair of garments and colours $\langle g, c \rangle$, we take a random garment g and we put a constraint over it with respect to another garment g', same approach for the colours.

Therefore, given the fact that a variable input size with respect the number of garments and colours (and not to the number of pairs), we have delimited the problem to two given sets of size 10 each where we clearly define our constraints.

5 Conclusions