**Logic and Model Checking** Fall 2025

# Project 1: SAT

*Lecturer: Jaime Ramos* *Department of Mathematics, Técnico*

## 1 Resolution

Consider the formula

$$(\neg p_1 \vee p_3 \vee p_4 \vee p_5) \wedge (\neg p_3 \vee p_4 \vee p_5) \wedge (\neg p_1 \vee p_3 \vee \neg p_4) \wedge (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee \neg p_5) \wedge (\neg p_3 \vee \neg p_4 \vee p_5)$$

Apply the CDCL algorithm to this formula to decide if it satisfiable or not. For each conflict, draw the implication graph and learn the conflict clause corresponding to the principal cut. The literals should be decided in the order of their subscripts, that is, $p_1$, $p_2$, $p_3$, $p_4$, $p_5$ and choosing always the positive case first.

## 2 Mastermind

The objective of Mastermind is to guess a secret code consisting of a series of 4 coloured pegs. Each guest results in feedback narrowing down the possibilities of the code. Feedback is given by 0, 1, 2 or 3 feedback pegs, that can be either black or white. A black feedback peg means that a code peg is of the right colour and is in the right position. A white feedback peg means that there is a code peg of the right colour in the wrong position. No feedback peg indicates that there is a colour in the guess that does not appear in the secret code.

As an example, consider the following secret code

If we attempt to guess the code by trying the guess below, we would receive the feedback information on the right:

The black feedback peg tells us that we guessed one colour right and in the right position. The two white feedback pegs tell us that we guessed two colours right but in the wrong position. The missing fourth feedback peg tells us that we got one colour wrong. That is all the feedback that we receive. We does not know which pegs are right and which pegs are wrong.

In this project, colours are replaced by digits ranging from 0 to 9, and the size of the secret code and, consequently, of the guesses is not limited to 4.

### 2.1 Almost Mastermind

We start by considering a variant of the game where only the information concerning the black feedback pegs is provided. That is, in this case, for each guess we make we are given the number of right digits in the right positions. A board is composed by a list of guesses and corresponding feedback information. An example of a board is the following sequence of guesses and feedback:

90342 ; 2
70794 ; 0
39458 ; 2

34109 ; 1
51545 ; 2
12531 ; 1

This means that sequence 90342 has 2 digits in the right position with respect to the secret code, sequence 70794 has no digit in the right position with respect to the secret code, and so forth. A secret code satisfying the conditions given by this board is 39542 and this is the only code that meets these conditions.

Encode the problem of, given a board, determining the solution of this variant of Mastermind as a SAT problem. Based on this encoding, and using Z3, define the following functions in Python:

1. `mastermind1(B)` – function that given a board `B`, determines a solution, if there is one;
2. `unique_solution1(B)` – function that given a board `B`, determines if the given board has a unique solution.

A board is represented in Python by a list of lists and natural numbers, in alternating positions. Each of these lists corresponds to a guess and each natural number after a list corresponds to the number of correct numbers in correct positions for that guess. For example, the previous board can be encoded in Python as

```
[
  [9,0,3,4,2], 2,
  [7,0,7,9,4], 0,
  [3,9,4,5,8], 2,
  [3,4,1,0,9], 1,
  [5,1,5,4,5], 2,
  [1,2,5,3,1], 1
]
```

Finally, given the board

5616185650518293 ; 2
3847439647293047 ; 1
5855462940810587 ; 3
9742855507068353 ; 3
4296849643607543 ; 3
3174248439465858 ; 1
4513559094146117 ; 2
7890971548908067 ; 3
8157356344118483 ; 1
2615250744386899 ; 2
8690095851526254 ; 3
6375711915077050 ; 1
6913859173121360 ; 1
6442889055042768 ; 2
2321386104303845 ; 0
2326509471271448 ; 2
5251583379644322 ; 2
1748270476758276 ; 3
4895722652190306 ; 1
3041631117224635 ; 3
1841236454324589 ; 3
2659862637316867 ; 2

find the unique 16-digit secret code.

### 2.2 Mastermind

Now, we consider full Mastermind, with codes of arbitrary size. In this case, a board is again composed by a list of guesses and feedback for each guess, but in this cases, the feedback is composed by the number of black pegs and by the number of white pegs (in this order). An example of a board is the following sequence of guesses and feedback:

1122 ; 0 ; 1
3441 ; 1 ; 1
5351 ; 1 ; 1
6631 ; 0 ; 1
2453 ; 1 ; 3

This means that sequence 1122 has no digit in the right position, but has one correct digit in the wrong position, sequence 3441 has one digit in the right position and one correct digit but in the wrong position, and so forth for the other guesses. A secret code meeting these conditions is 5243. Is this the only secret code that meets these conditions?

Capitalizing on the encoding of simplified version of Mastermind, encode the problem of determining the solution a Mastermind board as a SAT problem. Based on this encoding, and using Z3, define, the following functions in Pyhton:

1. `mastermind2(B)` – function that given a board `B`, determines a solution, if there is one;
2. `unique_solution2(B)` – function that given a board `B`, determines if the given board has a unique solution.

In this case, a board is represented in Python by a list of lists and two natural numbers, in alternating positions. Each of these lists corresponds to a guess and the firsth natural number after a list corresponds to the number of correct numbers in correct positions for that guess, and the second natural number corresponds to the number of correct digits in wrong positions. For example, the previous board can be encoded in Python as

```
[
  [1,1,2,2],0,1,
  [3,4,4,1],1,1,
  [5,3,5,1],1,1,
  [6,6,3,1],0,1,
  [2,4,5,3],1,3
]
```

## 3 Submission

The project is to be submitted in the *Fenix* platform by a member of the group (after the group has been registered). The submission should consist of a **single compressed folder** containing:

- a report on the project (in pdf), that should include the answer to the first exercise, the encoding of each problem as a SAT problem, and execution examples;
- a Jupyter notebook (or equivalent) with the implementation of the requested functions.

Submission deadline: **October 18, 2025, at 23:59.**