Exercise: Solving Mastermind with Propositional Logic

1. The Problem: Mastermind

In this game, there are two players:

- **Player 1** secretly arranges four colors in a sequence (positions 0–3).
- **Player 2** tries to guess this sequence.
- After each guess, Player 1 responds with a number: how many colors are in the correct position.

In our example:

- Player 2 makes a first guess. Player 1 says "2" ⇒ exactly two colors are in the right place 1.

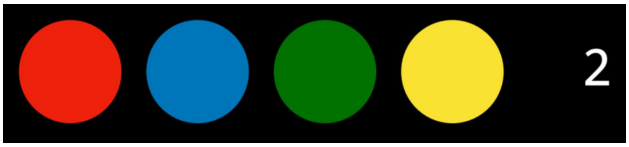

Figure 1: First round of Mastermind.

- Player 2 swaps two colors and guesses again. Player 1 says "0" ⇒ none are in the right place now 2.

This means the swapped colors must have been correct in the first guess.

We want to model this puzzle in propositional logic and use a **model checker** to find which propositions must be true.

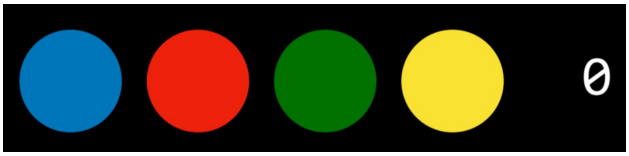

Figure 2: Second round of master mind.

2. The Logic Library

We will use a simple Python library that represents propositional logic sentences:

- `Symbol("A")` → atomic proposition A.
- `Not(X)` → logical negation $\neg X$.
- `And(X, Y, ...)` → conjunction $X \wedge Y \wedge \ldots$
- `Or(X, Y, ...)` → disjunction $X \vee Y \vee \ldots$
- `Implication(X, Y)` → implication $X \Rightarrow Y$.
- `Biconditional(X, Y)` → biconditional $X \Leftrightarrow Y$.

Each sentence has:

- `.evaluate(model)` → evaluates the truth given a model (a dictionary of truth values).
- `.formula()` → returns a human-readable string.
- `.symbols()` → returns the set of atomic symbols inside.

Finally, the function

`model_check(knowledge, query)`

checks whether the knowledge base entails the query, i.e. whether in every model where the KB is true, the query is also true.

3. Setting up the Exercise

We represent the problem with 16 atomic propositions:

- Four colors: red, blue, green, yellow.
- Four positions: 0–3.
- Propositions are named as `color+position`, e.g. `red0` means "red is in position 0".

4. What You Need to Do

Step 1: Encoding the Rules of the Game

1. Each color must appear in exactly one position.
2. Each position must contain exactly one color.

You will use combinations of `And`, `Or`, and `Not` to encode these rules into the knowledge base.

Step 2: Adding Information from the Guesses

- First guess: exactly 2 positions correct. Encode this as a constraint.
- Second guess: 0 positions correct. Encode this as another constraint.

Step 3: Running the Model Checker

For each symbol (e.g. `red0`, `blue2`, ...), run model checking to find out the correct solution to the game.

5. Expected Outcome

By carefully encoding the rules and the feedback, the model checker should be able to rule out impossible assignments and tell you which colors are forced into which positions. This shows how logic and model checking can be applied to reasoning problems like Mastermind.