Daniel Dickson
Fiona LeClair-Robertson
Aniket Mukherjee
Shauna Tuinstra

CISC 204
Fall 2021

Mastermind

A Logical Modelling Project

by Group 6

Premise

Our project explores the game of **Mastermind**, which is a 2 player code breaking game where each player assumes the role of either the Codemaster or the Codebreaker

CODEMASTER: The Codemaster must create a code consisting of 4 coloured pegs and give their opponent feedback on each guess; this feedback can be in the form of a purple peg (meaning the guess is the exact same as the code) or a white peg (meaning the colour is somewhere in the code, but in a different slot than the ussr's guess)

CODEBREAKER: The Codebreaker must determine the code by creating guesses in accordance with the feedback given to them by the Codemaster

The goal of our model is to determine, given a limited number of user guesses, whether or not the code can be determined logically in the next guess.

It will calculate the likelihood of each peg in the code being a certain colour out of 4 options: red, blue, green, or yellow

Modifications

To make our project easier for us to model with the time given, we added in some minor adjustments from the normal rules of the game:

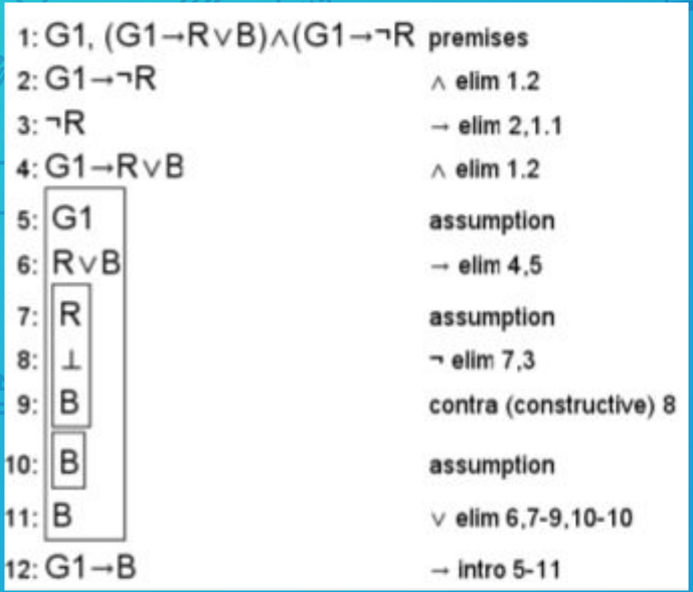
- There are only 4 colours; usually there may be 6 or more colours to choose for a guess
- Purple and white pegs will reveal what slot they are assigned to; usually these are given to the user without making them aware of what position they are referring to
- The user has 7 chances to break the code; this was a compromise from the usual 10 turns since we feel the above changes may make the game slightly easier to win

Propositions

- G_n R_i represents the red value of the i^th peg in the user's i^th guess
  - Each peg is represented by four propositions: one for the red value, one for blue, one for green, and one for yellow
  - A colour proposition being true, indicates that the peg is that colour
  - For example, G_2B_3 represents the blue value of the third peg in the user's second guess. If true, the peg is blue.
- C R_i: Similarly to the representation of the guess pegs, code pegs are represented with four colour propositions.
  - I.e. C R_i indicates the red colour proposition for the i^th peg in the guess.
- P□i: represents a purple feedback peg for the i^th peg in the k^th user guess, true if a purple peg is given for that peg and false if not
- W□i: represents a white feedback peg for the i^th peg in the k^th user guess, true if a white peg is given for that peg and false if not
- R, B, E, A represent the respective colours that a peg may be
  - Green is E because G is guess and R is red
  - **JAPE would not let us use Y**, so we used A when necessary for yellow

Constraints

- Each player only has at most 7 chances to correctly guess the code before they must forfeit
- Any peg in a guess or final code can only be one colour
  - $G_i r_x \rightarrow \neg G_i b_x \wedge \neg G_i a_x \wedge \neg G_i e_x$, $G_i b_x \rightarrow \neg G_i r_x \wedge \neg G_i a_x \wedge \neg G_i e_x$, . . .
- Any purple peg assigned to slot n in the guess means that the peg in slot n can only be that colour that was guessed
  - $P_{11} \rightarrow (Cr_1 \wedge G_1 r_1 \wedge \neg G_1 b_1 \wedge \neg G_1 e_1 \wedge \neg G_1 a_1) \vee (Cb_1 \wedge G_1 b_1 \wedge \neg G_1 r_1 \wedge \neg G_1 e_1 \wedge \neg G_1 a_1) \vee (Ce_1 \wedge G_1 e_1 \wedge \neg G_1 r_1 \wedge \neg G_1 b_1 \wedge \neg G_1 a_1) \vee (Ca_1 \wedge G_1 a_1 \wedge \neg G_1 r_1 \wedge \neg G_1 e_1 \wedge \neg G_1 b_1)$
- Any white peg assigned to slot n in the guess means that the peg in slot n can not be that colour, but at least one of the other pegs is that colour
- A player can win if and only if their guess exactly matches the code (i.e. 4 purple pegs given to their guess)



```
1: G1, (G1→R∨B)∧(G1→¬R     premises
2: G1→¬R                   ∧ elim 1.2
3: ¬R                      → elim 2,1.1
4: G1→R∨B                  ∧ elim 1.2
5:  | G1                   assumption
6:  | R∨B                  → elim 4,5
7:  | | R                  assumption
8:  | | ⊥                  ¬ elim 7,3
9:  | | B                  contra (constructive) 8
10: | | B                  assumption
11: | B                    ∨ elim 6,7-9,10-10
12: G1→B                   → intro 5-11
```

Proofs

- Broke premise into two implications
- Eliminated implication using premise G_1
- Backwards implication on conclusion, opened assumption box G_1 … B
- Eliminated implication using premise G_1; R ∨ B in box
- ∨ elim; B … B auto-completes, R … B is solved using negation elim and contra construction via ¬R

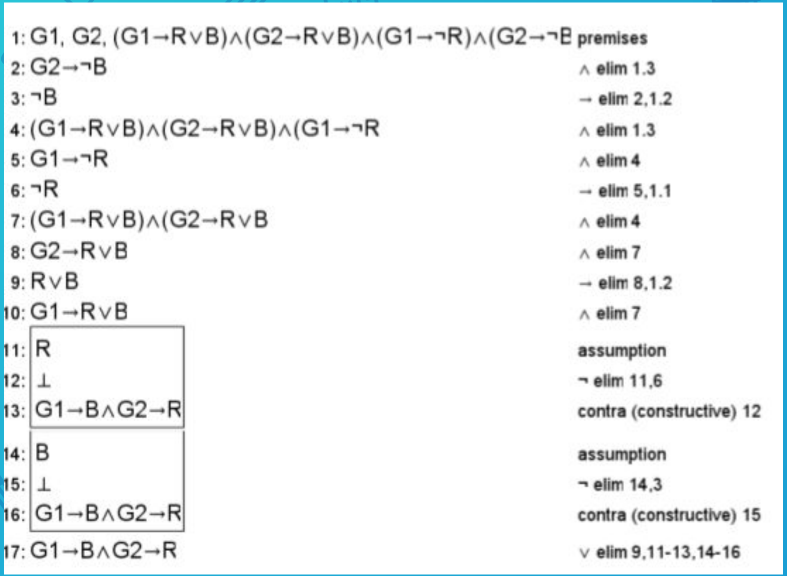
Premises:

- We examine only one peg from the guess
- We know the peg may be either red or blue
- We know the peg can not be red

Conclusion:

- The peg must be blue

NOTE: For our first two proofs, we decided to only consider two colours, to make it easier to construct and prove the sequents.



1:	$G1, G2, (G1 \to R \vee B) \wedge (G2 \to R \vee B) \wedge (G1 \to \neg R) \wedge (G2 \to \neg B$	premises
2:	$G2 \to \neg B$	$\wedge$ elim 1.3
3:	$\neg B$	$\to$ elim 2,1.2
4:	$(G1 \to R \vee B) \wedge (G2 \to R \vee B) \wedge (G1 \to \neg R$	$\wedge$ elim 1.3
5:	$G1 \to \neg R$	$\wedge$ elim 4
6:	$\neg R$	$\to$ elim 5,1.1
7:	$(G1 \to R \vee B) \wedge (G2 \to R \vee B$	$\wedge$ elim 4
8:	$G2 \to R \vee B$	$\wedge$ elim 7
9:	$R \vee B$	$\to$ elim 8,1.2
10:	$G1 \to R \vee B$	$\wedge$ elim 7
11:	R	assumption
12:	$\perp$	$\neg$ elim 11,6
13:	$G1 \to B \wedge G2 \to R$	contra (constructive) 12
14:	B	assumption
15:	$\perp$	$\neg$ elim 14,3
16:	$G1 \to B \wedge G2 \to R$	contra (constructive) 15
17:	$G1 \to B \wedge G2 \to R$	$\vee$ elim 9,11-13,14-16

Proofs

- Broke premise into separated implications
- Eliminated implications using premises G_1 and G_2
- $\vee$ elim; R … conclusion and B … conclusion
- Assumptions solved using negation elim and contra construction via ¬R and ¬B, respectively

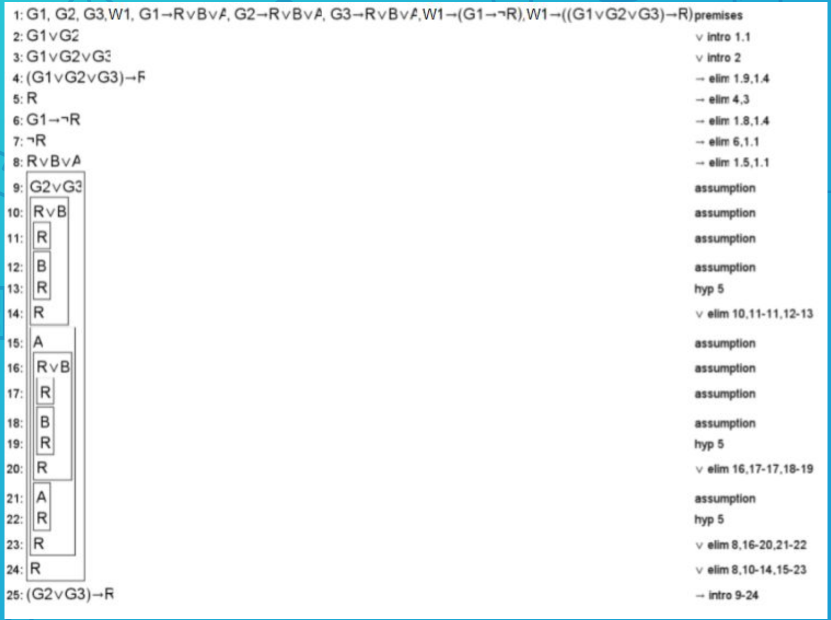
Premises:

- We examine two pegs from the guess
- We know the pegs may be either red or blue
- We know the first peg can not be red
- We know the second peg can not be blue

Conclusion:

- The first peg must be blue and the second peg must be red

Proofs



```
 1: G1, G2, G3,W1, G1→R∨B∨A, G2→R∨B∨A, G3→R∨B∨A,W1→(G1→¬R),W1→((G1∨G2∨G3)→R) premises
 2: G1∨G2                       ∨ intro 1.1
 3: G1∨G2∨G3                    ∨ intro 2
 4: (G1∨G2∨G3)→R                → elim 1.9,1.4
 5: R                           → elim 4,3
 6: G1→¬R                       → elim 1.8,1.4
 7: ¬R                          → elim 6,1.1
 8: R∨B∨A                       → elim 1.5,1.1
 9: | G2∨G3                     assumption
10: | | R∨B                     assumption
11: | | | R                     assumption
12: | | | B                     assumption
13: | | | R                     hyp 5
14: | | R                       ∨ elim 10,11-11,12-13
15: | | A                       assumption
16: | | | R∨B                   assumption
17: | | | | R                   assumption
18: | | | | B                   assumption
19: | | | | R                   hyp 5
20: | | | R                     ∨ elim 16,17-17,18-19
21: | | | A                     assumption
22: | | | R                     hyp 5
23: | | R                       ∨ elim 8,16-20,21-22
24: | R                         ∨ elim 8,10-14,15-23
25: (G2∨G3)→R                   → intro 9-24
```

Premises:

- We examine three pegs from the guess
- We know the pegs may be one of either red, blue, or yellow
- We know the first peg can not be red
- The second & third pegs have no further limitations on colour
- Peg 1 was given a white peg

- Implication intro to conclusion; opened assumption box $G_2 \vee G_3$ … R
- Implication elim using G_1 to separate $R \vee B \vee A$
- $\vee$ elim; opened assumption boxes $R \vee B$ … R and A … R
- Split first box using $\vee$ elim; R … R auto-completes, B … R still open
- W_1 used for implication elim and free $G_1 \vee G_2 \vee G_3$
- G_1 used for $\vee$ intro inventing right side to create something in similar form to $G_1 \vee G_2 \vee G_3$
- Implication elim to free R from $G_1 \vee G_2 \vee G_3 \rightarrow R$, solving remaining assumption boxes

Conclusion:

- At least one of peg 2 or 3 must be red

NOTE: A is meant to represent the colour yellow in our sequent, JAPE didn't like Y as a variable.

Our Program

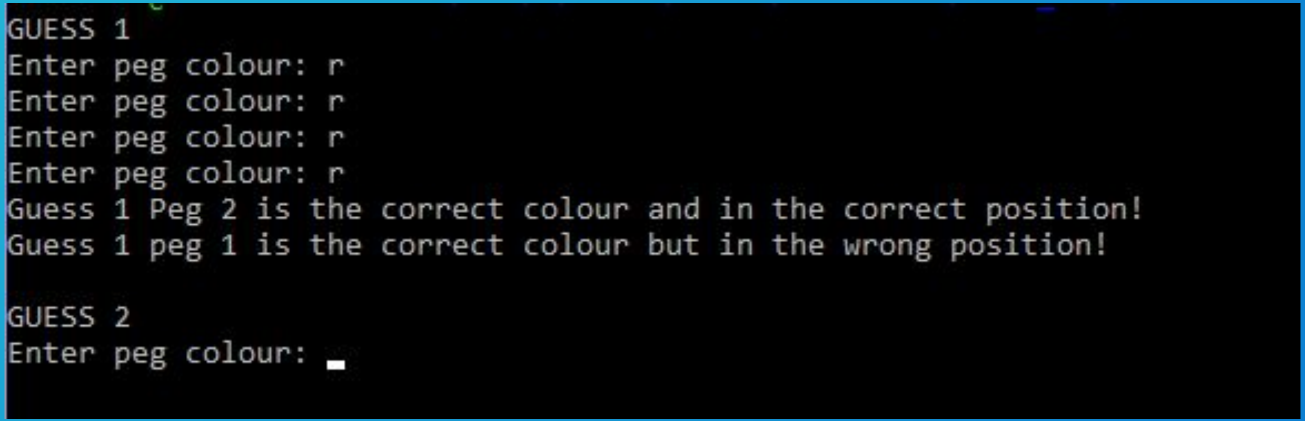
- Our code combines mastermind game logic and user input with bauhaus propositions and constraints to create a game state of guesses, feedback pegs and a secret code.
- The user creates a guess by inputting a sequence of 4 characters corresponding to peg colours (R, G, B, or Y). Each guess is then stored in two ways: a list of characters representing colours, and as a 2D array of bauhaus peg colour propositions.
- The code is stored as a list of randomly generated characters from the set ['R', 'G', 'B', 'Y']. The necessary bauhaus code peg colour propositions are created but not explicitly set with true or false values. So, the model solution consists of a setting of the code colours that satisfies the feedback constraints given for all of the guesses.
- Each guess is then passed to functions that determine if white or purple feedback pegs should be given.
    - Purple peg propositions are created for each peg position in the guess. The proposition is set to true if the requirements for a purple peg being given are met. Then the constraints for a purple peg are added (i.e. the code peg in that position must match the guess peg in colour)

Our Program (Continued)

- White peg propositions are then created for the guess. White pegs are not given for pegs that were already given a purple peg. (i.e. say peg 2 in the guess is correct and a purple peg is given. If peg 4 is the same colour as peg 2, it is not given a white peg.)
- The constraints are then also added: the code peg in the same position is not the colour of the guess peg, and at least one of the other code pegs must be that colour.
- After the user has inputted all of their allotted guesses (or solved the code), the theory is compiled.
- For each peg in the code, the probability of that peg being each colour is output.
    - For example, say a purple peg was given for guess peg 2 which was guessed to be green. The output for peg 2 has probability 1 for green and 0 for all other colours.

Screenshots of Program

Inputting Guess:

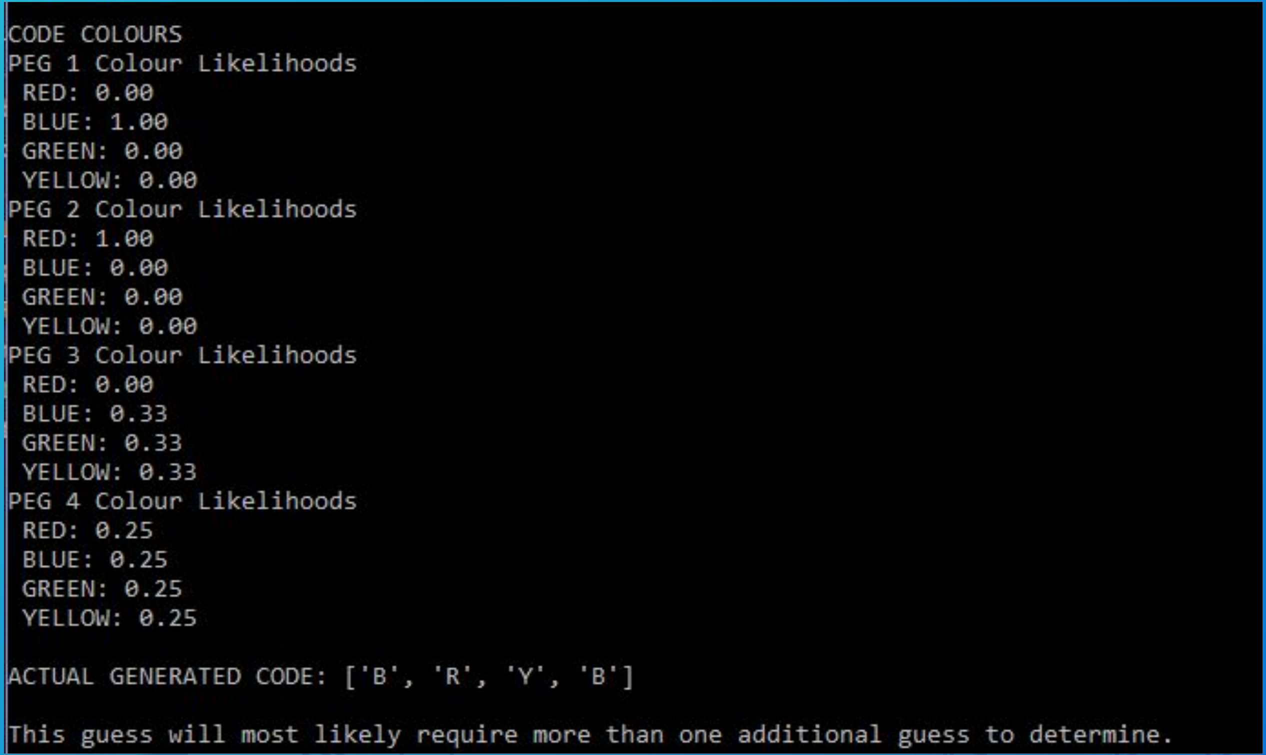


```
GUESS 1
Enter peg colour: r
Enter peg colour: r
Enter peg colour: r
Enter peg colour: r
Guess 1 Peg 2 is the correct colour and in the correct position!
Guess 1 peg 1 is the correct colour but in the wrong position!

GUESS 2
Enter peg colour: _
```

Screenshots of Program

Model Output:



```
CODE COLOURS
PEG 1 Colour Likelihoods
 RED: 0.00
 BLUE: 1.00
 GREEN: 0.00
 YELLOW: 0.00
PEG 2 Colour Likelihoods
 RED: 1.00
 BLUE: 0.00
 GREEN: 0.00
 YELLOW: 0.00
PEG 3 Colour Likelihoods
 RED: 0.00
 BLUE: 0.33
 GREEN: 0.33
 YELLOW: 0.33
PEG 4 Colour Likelihoods
 RED: 0.25
 BLUE: 0.25
 GREEN: 0.25
 YELLOW: 0.25

ACTUAL GENERATED CODE: ['B', 'R', 'Y', 'B']

This guess will most likely require more than one additional guess to determine.
```

First-Order Extension

- We can extend our model to a predicate logic setting by looking at the white pegs, which signify that a peg in the guess is the right color but in the wrong position. With this, we can say that for all possible guesses, there exists a guess that contains at least one instance of that color within the code made by the code maker.

- To apply this in the context of predicate logic, we can include universal and existential quantifiers to our propositions and constraints.

- Propositions:
  - C(x): the code has been guessed
  - R(x): the peg at position *x* is red
  - E(x): the peg at position *x* is green
  - B(x): the peg at position *x* is blue
  - A(x): the peg at position *x* is yellow
  - P(x): a purple peg was given for peg *x*
  - W(x): a white peg was given for peg *x*
  - T(x): the sequence of colours *x* is a guess
  - M(x): The peg at x matches the peg at position x in the code

Constraints of our First Order Extension

- Constraints:
  - $\exists x R(x) \rightarrow (\neg G(x) \wedge \neg B(x) \wedge \neg Y(x))$
    - Basically just saying if one colour peg is in a position there can't be a peg of a different colour occupying that same position
  - $C(x) \rightarrow \exists x(T(x) \wedge \forall y M(y))$
    - If the code has been guessed, that implies that there exists a position such that the sequence of colors is a guess and for all positions y, there the peg at y matches the same position in the code
  - $P(x) \rightarrow M(x)$
    - If a purple peg was given at x then it is implied that the peg at x matches the peg at position x in the code
  - $M(x)$