GROUP 32: Flow Free

Matthew Vandergrift (19mwv1)

Logan Cantin (19lkc2)

Jagrit Rai (19jr28)

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

December 14, 2021

1 Abstract

1.1 Flow Free Mobile Game

Flow Free is a popular mobile puzzle game. The game presents you with a square grid which contains coloured dots called endpoints. The goal of the game is to connect the endpoints with a combination of vertical and horizontal lines. When two endpoints of the same colour are connected it is called a flow. The rules are that flows may not intersect and must cover the whole grid. When all endpoints are connected and the grid is covered, the level is complete and you have won.

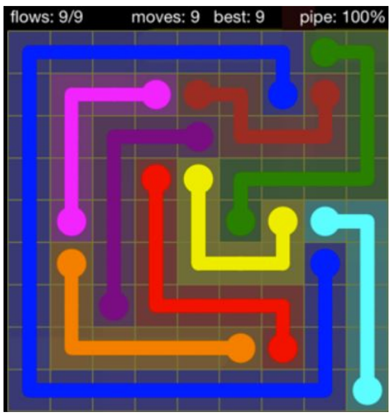


Figure 1: An example of a solved level of Flow Free

1.2 Purpose of our Model

Given a board with dimensions 5x5 and a certain arrangement of endpoints, a model of the game flow will consist of a series of flows that connect all endpoints and fill up the whole game board. Figure 2 shows an unsolved level and its solution. If no model exists then the given board and endpoints represent an unsolvable level. The goal of our project is to find a model for any given level or determine that the level is unsolvable.

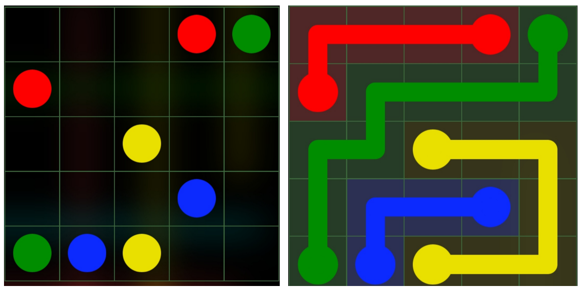


Figure 2: An unsolved level (left) and the solution (right)

2 Propositions

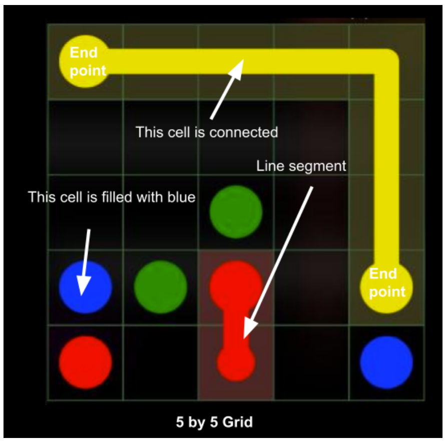
We begin by explaining the notation used to define our propositions and constraints. A **cell** is a square on the grid. A cell is **adjacent** to the cell above, below, and to the left and right of it, but not those which are diagonal to it (i.e. two cells are adjacent if their Manhattan distance is 1). **n** is the length and height of the board in terms of number of cells, so if a board is 5 by 5 it means that it is 5 cells high, 5 cells wide and that n=5. To represent a position on the board we use **x,y**, which are positive integers less than or equal to n and greater than 0. The position 0,0 would be in the top left and n,n would be the bottom right. Additionally a **flow** is a series of line segments that form a path from one endpoint to another. Lastly **c** represents a colour.

We define our propositions below:

- An **endpoint** is a coloured circle on the grid which the player is trying to connect a flow to. Let, $E(c, x, y)$ be an endpoint of colour c at the point (x, y).
  - The proposition $E(c, x, y)$ is true when there is an endpoint of colour c at the point (x, y), and false otherwise.
- A **line segment** is a line that connects two adjacent cells. Let, $L(c, x_1, y_1, x_2, y_2)$ be a line segment of colour c from (x_1, y_1) to (x_2, y_2).
  - $L(c, x_1, y_1, x_2, y_2)$ will be true when a line segment of colour C joins the two adjacent cells (x_1, y_1) and (x_2, y_2), and false otherwise.
- A cell is **filled** with a colour c when there is a line segment of colour c or an endpoint of colour c at that cell. Let $F(c, x, y)$ be a cell at (x, y) which is filled with the colour c.
  - $F(c, x, y)$ is true when the cell at (x, y) has either a line segment of colour of c to/from it or when there is an endpoint of colour c at (x, y), and false otherwise.
- A cell is **connected** if it is connected via line segments to an endpoint. In other words, if there is a path from an endpoint to the cell, then it is connected.
  - $\mathbb{C}(x, y)$ is true if the cell at (x, y) is connected to an endpoint, and false otherwise.

Figure 3 shows an example of each of the terms that were defined above.

Figure 3: Visualization of all the terms that were defined above



3 Constraints

- The game of flow consists of connecting coloured end points with a line of the same colour. Therefore, for each colour C on the board, there are exactly two endpoints of that colour. If a cell is filled with a colour this means that this colour exists on the board, therefore a filled proposition implies the existence of exactly two endpoints of the colour of the filled proposition.
  - $F(c,x,y) \rightarrow (\neg E(c,1,1) \wedge \neg E(c,1,2) \wedge \dots E(c,x_i,y_i) \wedge E(c,x_j,y_j) \wedge \dots \wedge \neg E(c,x_n,x_{n-1}) \wedge \neg E(c,x_n,x_n))$, where (x_i,y_i) and (x_j,y_j) are the cells where the endpoints are located.
- Each cell on the board must be filled with exactly one colour. This is one of the main rules of the game.
  - $\neg F(c_1,x,y) \wedge \neg F(c_2,x,y) \wedge \dots \wedge F(c_j,x,y) \wedge \dots \wedge \neg F(c_n,x,y)$ for all cells (x,y) on the board, where C_j is the colour that is filled at the point (x,y).
- An endpoint of colour C on a given cell implies that that cell must be filled with colour C.
  - $E(c,x,y) \rightarrow F(c,x,y)$
- Since each endpoint has exactly one flow connected to it, there must be exactly one line segment connected to each endpoint.
  - $E(c,x,y) \rightarrow [[L(C,x,y,x+1,y) \wedge \neg L(c,x,y,x-1,y) \wedge \neg L(c,x,y,x,y+1) \wedge \neg L(c,x,y,x,y-1)] \vee [\neg L(c,x,y,x+1,y) \wedge L(c,x,y,x-1,y) \wedge \neg L(c,x,y,x,y+1) \wedge \neg L(c,x,y,x,y-1)] \vee [\neg L(c,x,y,x+1,y) \wedge \neg L(c,x,y,x-1,y) \wedge L(c,x,y,x,y+1) \wedge \neg L(c,x,y,x,y-1)] \vee [\neg L(c,x,y,x+1,y) \wedge \neg L(c,x,y,x-1,y) \wedge \neg L(c,x,y,x,y+1) \wedge L(c,x,y,x,y-1)]]$

    To briefly explain the propositional logic, the consequent is a series of disjunctions and in each disjunction there is every possible line segment from a cell to another cell, and in each one only one of the line segment propositions is true.

- Given that the board must be full, if a cell does contain an endpoint then there must be a line segment going to and coming from that cell. To be precise, there are exactly two line segments connected to a cell that is not an endpoint.

  - $\neg E(c,x,y) \rightarrow [(L(c,x,y,x+1,y) \wedge L(c,x,y,x-1,y) \wedge \neg L(c,x,y,x,y-1) \wedge \neg L(c,x,y,x+1,y+1)] \vee [\neg L(c,x,y,x-1,y) \wedge L(c,x,y,x+1,y) \wedge L(c,x,y,x,y-1) \wedge \neg L(c,x,y,x+1,y+1)] \vee [\neg L(c,x,y,x,y-1) \wedge \neg L(c,x,y,x-1,y) \wedge L(c,x,y,x+1,y) \wedge L(c,x,y,x,y+1)] \vee [L(c,x,y,x,y-1) \wedge \neg L(c,x,y,x-1,y) \wedge L(c,x,y,x+1,y) \wedge L(c,x,y,x,y+1)] \vee [L(c,x,y,x,y-1) \wedge \neg L(c,x,y,x-1,y) \wedge L(c,x,y,x+1,y) \wedge \neg L(c,x,y,x,y+1)]$

    To explain this in brief, the antecedent: ¬E(C, x, y) means a cell is not an endpoint. The consequent has a series of disjunctions which contains all the possible line segments that could go from this cell to another. In each disjunction two of these lines are true and the rest are false. We have a series of disjunctions because any single one of these pairings could possibly be true.

- The start and end points of a line segment must have the same colour as the line segment.

  - $L(c,x_1,y_1,x_2,y_2) \rightarrow F(c,x_1,y_1) \wedge F(c,x_2,y_2)$ where x_1, y_1 is adjacent to x_2, y_2

    Note: x_1, y_1 are adjacent because line segments occur only between adjacent cells.

- All cells must be connected. This constraint prevents having disconnected loops and will be explored further in the Model Exploration section.

  - $\mathbb{C}(x,y) \wedge \mathbb{C}(x_2,y_2) \wedge \mathbb{C}(x_3,y_3) \wedge \cdots \wedge \mathbb{C}(x_n,y_n)$

    Note that $(x,y),(x_2,x_3),\ldots(x_n,y_n)$ are all cells on the board.

- Any cell containing an endpoint is connected. This is trivially true.

  - $E(c,x,y) \rightarrow \mathbb{C}(x,y)$, in other words, an endpoint of any colour at cell (x,y) implies that the cell (x,y) is connected.

- If there is a line segment between two cells and a cell under either of the ends of the line segment is connected, then both ends of the line segment are connected. For example, say that the left side of a line segment is connected (i.e. the left side has a path to an endpoint). The right side is connected to the left side via the line segment, so the right side is also connected to an endpoint, by extension.

  - $\big((L(c_1,x_1,y_1,x_2,y_2) \vee \cdots \vee L(c_k,x_1,y_1,x_2,y_2)) \wedge (\mathbb{C}(x_1,y_1) \vee \mathbb{C}(x_2,y_2))\big) \rightarrow (\mathbb{C}(x_1,y_1) \wedge \mathbb{C}(x_2,y_2))$ for all colours $c_1, c_2, ..., c_k$ and all adjacent cells (x_1,y_1) and (x_2,y_2).

4 Model Exploration

4.1 Disconnected loop error

The first iteration of our model did not contain the "connected" propositions and relevant constraints. We thought that if every endpoint had exactly one line segment and every non-endpoint cell had exactly two line segments coming out of it, then the board would be solved. During the testing, we encountered an odd bug: sometimes, the model would return a solution that contained a loop that was not connected to any endpoints. See Figure 4 for an example of this error. Technically, this disconnected loop does satisfy our constraints, since each cell in the loop has exactly 2 line segments coming out of it.

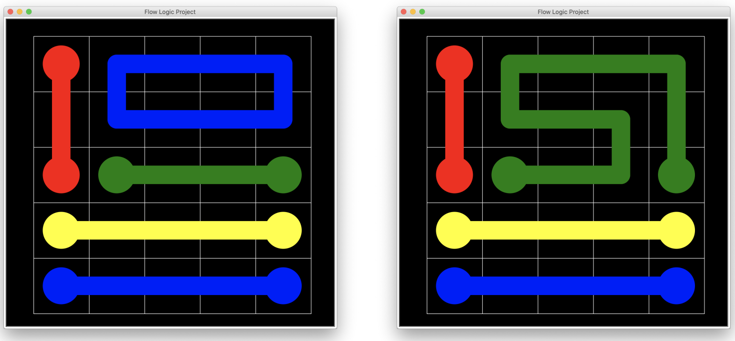


Figure 4: Disconnected loop (left) and a valid solution for the same board (right)

To solve this problem, we implemented the "connected" proposition. This proposition along with its relevant constraints ensure that disconnected loops do not occur on the board.

4.2 Exactly k constraint

One of the challenges that we encountered was implementing an "exactly k" constraint. In other words, a constraint that exactly k of a set of propositions is true. We needed to use this constraint to ensure that there are exactly 2 endpoints of each color and that there are exactly 2 line segments coming out of each non-endpoint cell. There is no built-in function that can add this constraint. We explored the possibility of using an "at most k" constraint and the negation of an "at most $k-1$" constraint. However, bauhaus did not seem to support adding the negation of the built-in SAT constraints, so we implemented the exactly k constraint manually.

Our implementation consisted of iterating over all combinations of k propositions, and using a logical "AND" on the k propositions and the negation of all the other propositions. We then used a logical "OR" on all of the formulas previously mentioned. This method works, but it significantly increases the complexity of the constraints. The size of the constraint grows large very quickly as the number of propositions grows, which makes it harder for the SAT solver to find a solution.

4.3 Length of time to find a solution

Another challenge we encountered was the length of time it takes for the SAT solver to solve a level of Flow Free. The current version of the model takes 4 minutes to solve a 5x5 board. In contrast, solving the board by hand takes around 10-30s for an average player. This significant increase in time can be attributed to the advantage that humans have over the SAT solver. As humans, we have experience working with the game, and we can make educated guesses about where the flows will go.[1] In addition, we can solve some of the flows that have only one possible path, which makes solving the rest of the board becomes easier as there are less possible spaces for the other flows to go. The SAT solver does not know any of these features about the game and can't use any of these strategies, which is why it takes so much longer.

We were interested in measuring how the run time of the model changed with board size. We ran a 6x6 board to see how it would compare to a 5x5 board, and it ended up taking 36 minutes and 30 seconds. This makes our model impractical for usage on any size board greater than 5x5.

4.4 GUI Visualization

The visualization brought about many issues during its implementation, the very first of which was the way we wanted to approach visualizing our data in the first place. In our original draft, the data was displayed by printing a 2d array, which, while sufficient for an incomplete draft, was not up to the standards we held for a complete product. We wanted to replicate the game we were experimenting with, thus we decided on creating some kind of GUI with visuals that were reminiscent of the original game. Some time was spent considering using tkinter, a more complex GUI library, to achieve this. However, we realized that the user doesn't actually have to interact with the window; the program does all the work for us. All that we needed to do was draw the completed board. Upon this realization, we decided to use the turtle library, which resulted in an incredibly accurate recreation of the original game's design.

4.5 User Input

A more insignificant issue we faced in our project was the way user input is handled. Since the decision was made to draw out the board rather than making an actual interactive GUI, users must input different boards by using command line arguments to pass a file to the program. While this is a very tenuous issue, it is still inconvenient to users, as it is not a very intuitive way of inputting the information. The files must be formatted in a very specific manner and the program can only accept certain colours to be used. However, the formatting for the board files is very simple to understand, and by using command line arguments it offers the chance for simple user input, rather than forcing a specific filename to be hardcoded into source code. So, although the user cannot directly interact with the visualization window, there is still the opportunity for user interaction.

[1]Thanks to our TA, Victoria, for explaining to us why the SAT solver takes so much longer to come to a solution than humans.

5 First-Order Extension

Our propositions in predicate logic will closely resemble the propositions in propositional logic for two reasons; first off we used the notation Proposition(c,x,y) to explain that a proposition would have many versions for different values of c, x, and y, and in predicate logic this notation is used to state that a proposition will be defined for the objects c, x, and y. The second reason they will look similar is that the logical setting is exactly the same. In both logical systems we are modelling an n by n solution to a level of flow. We will start by defining the *propositions* which model the different 'types' of objects in our logical setting,

- Coord(a) where a is a co-ordinate on the board. (a is any integer less than or equal to n where n is the size of the board)
- Colour(c) where c is a colour that is present somewhere on the board. Example colours are red, blue, green, and yellow.

Next we will define our propositions which closely resemble the propositions in propositional logic. They are the more complex propositions that describe the different elements which make up a model(a solved level of flow). These propositions are as follows,

- E(c,x,y), represents an endpoint with the colour c at the point x, y.
- L(c,x_1,y_1,x_2, y_2) represents a line segment from the point x_1,y_1 to the point x_2, y_2
- F(c,x,y) represents a filled cell at x,y (i.e a cell that has some element of colour in it either a line segment, or endpoint).
- $\mathbb{C}$(c,x,y) represents a connected cell at x,y with colour c.

These are all the propositions we need for the extension into predicate logic. Now we will convert our constraints into predicate logic. We will start with the *constraints* that state that the arguments given to to the complex propositions must be the right type,

- $\forall c.\forall x.\forall y.(E(c,x,y) \rightarrow (\text{Colour}(c) \wedge \text{Coord}(x) \wedge \text{Coord}(y))$. This constraint states that the arguments to the endpoint proposition [E(c,x,y)] must be a colour for c, and co-ordinates for x and y.
- $\forall c.\forall x.\forall y.(F(c,x,y) \rightarrow (\text{Colour}(c) \wedge \text{Coord}(x) \wedge \text{Coord}(y))$ This constraint states that arguments to the filled proposition [F(c,x,y)] must be a colour for c, and co-ordinates for x and y.
- $\forall c.\forall x.\forall y.\forall x_2.\forall y_2.(L(c,x,y,x_2,y_2) \rightarrow \text{Colour}(c) \wedge \text{Coord}(x) \wedge \text{Coord}(y) \wedge \text{Coord}(x_2) \wedge \text{Coord}(y_2) \wedge (x_1 \neq x_2) \wedge (y_1 \neq y_2))$. This constraint states that arguments to the line segment proposition [L(c,x, y,x_2,y_2)] must be a colour for c, and co-ordinates for x, y, x_2 and y_2, additionally we include the condition that $x_1 \neq x_2 \wedge y_1 \neq y_2$ simply because a line segment cannot go from one point to the exact same point.
- $\forall x.\forall y.(\mathbb{C}(x,y) \rightarrow (\text{Coord(x)} \wedge \text{Coord}(y))$ This constraint simply states that the arguments to the connected proposition must be two co-ordinates.

Next we will define the constraints in predicate logic that we have already modelled in propositional logic. We will keep in mind the constraints for the types of arguments that each proposition can have, these constraints make the other constraints simpler because we do not need to define what types the arguments are for each constraint. The constraints are modelled as follows, note that since some are quite verbose explanation of the logic may follow;

- We know that for a solution to be valid every cell must be filled with *exactly* one colour in predicate logic we can write; $\forall x.\forall y.\exists c.(F(c,x,y) \land \neg\exists c_2.F(c_2,x,y) \land c_1 \neq c_2))$.

  Explanation of Logic: Since each cell can only have one colour we state there exists a colour (c) for which a cell is filled with c, but also that there does **not** exist a different colour (c_2) that fills the same cell.

- Every endpoint must have the same colour as the cell where it is located so in predicate logic we can write; $\forall x.\forall y.\forall c.(E(c,x,y) \rightarrow F(c,x,y))$

- The start and end points of a line segment must have the same colour as the line segment, this is true for every colour, and every line segment therefore in predicate logic we can write; $\forall c.\forall x.\forall y.\forall x_2.\forall y_2.(L(c,x_1,y_1,x_2,x_2) \rightarrow F(c,x_1,y_1) \land F(c,x_2,y_2))$

- Every cell must be filled by one and only one colour. In predicate logic we can write; $\forall x.\forall y.(\exists c_1.F(c_1,x,y) \land \neg\exists c_2.(F(c_2,x,y) \land (c_1 = c_2)))$

  Explanation of logic: for every pair of coordinates there is one colour for which x,y is filled, and there does exist a different colour such that x,y is filled with that colour.

- Each endpoint must have exactly one line segment connected to it. In predicate logic; $\forall x.\forall y.\forall c.(E(c,x,y) \rightarrow \exists x_2.\exists y_2.L(c,x,y,x_2,y_2) \land \neg\exists x_3.\neg\exists y_3.(L(c,x,y,x_3,y_3) \land x_2 \neq x_3 \land y_2 \neq y_3)))$

  Explanation of logic: For all coordinates and colours the existence of an endpoint at x,y of a colour c implies there exists *only* one pair or co-ordinates for which there is a line segment connected to it. We express that there is only one pair of co-ordinates by stating that there does not exist a different pair of coordinates for which there is a line segment connected to the endpoint.

- If any cell is filled with a colour there must be exactly two endpoints which share that colour. $\forall c.\forall x.\forall y.(F(c,x,y) \rightarrow (\exists x_2.\exists y_2.E(c,x_2,y_2) \land \exists x_3.\exists x_3.E(c,x_3,y_3) \land (x_2 \neq x_3) \land (y_2 \neq y_3) \land \neg\exists x4.\neg\exists y_4.(E(c,x_4,y_4) \land ((x_4 \neq x_3) \lor (x_4 \neq x_2)) \land ((y_4 \neq y_3) \lor (y_4 \neq y_2)))$

  Explanation of the logic, for every position on the board for every colour if that position on the board is filled with a colour then two endpoints are implied. These endpoints are unique which is expressed by $x_2 \neq x_3$, additionally it also implies that there does exist a third endpoint of that colour c, which we denote by saying does not exist the pair x_4,y_4 which they are different than x_2, y_2, or x_3, y_3.

- We know that every cell that is not an endpoint must have exactly two line segments connected to it. In predicate logic we write; $\forall x.\forall y.\forall c.(\neg E(c, x, y) \rightarrow (\exists x_2.\exists y_2.\exists x_3.\exists y_3.(L(c, x, y, x_2, y_2) \wedge L(c, x, y, x_3, y_3) \wedge (x_2 \neq y_2) \wedge (x_3 \neq y_3)))$

  Explanation of Logic: For all x,y and colours on the board if a cell at x,y is not an endpoint then two unique sets of coordinates must exist. One unique x, y for one line segment and another for another line segment, we show that they are unique by stating that $x_2 \neq x_3$ etc.

- We know that every cell has to be connected.Instead of a series of conjunctions of connected propositions like in propositional logic, in predicate logic we can simply write, $\forall x.\forall y.\mathbb{C}(x, y)$

- We know that an endpoint at cell x,y implies that the cell at x,y is connected. In predicate logic we write; $\forall x.\forall y.\forall c.(E(c, x, y) \rightarrow \mathbb{C}(x, y))$

- As is explained in the constraints section of the report, if a line segment is between two cells and one or both of those cells are connected then both cells in that line segment are by definition connected.
  $\forall c.\forall x_1.\forall y_1.\forall x_2 \forall y_2.(L(c, x_1, y_1, x_2, y_2) \wedge (\mathbb{C}(x_1, y_1) \vee \mathbb{C}(x_2, y_2))) \rightarrow (\mathbb{C}(x_1, y_1) \wedge \mathbb{C}(x_2, y_2)))$

6 JAPE Proofs

6.1 Base Encoding

One simplification has been made to our notation to make the JAPE proofs less verbose and allow them to better highlight the novel proofs we have created. Instead of denoting each cell by a pair of co-ordinates with two integers x, y according to a Cartesian coordinated system in JAPE we simply give each cell on the board a unique integer identifier. For each proof which cell has which number is explained.

In order to use JAPE we denote our propositions in a slightly different way, Line Segments are denoted by $Sxyc$, where x and y are the cell numbers of the start and end points of the line segment and c is the color of the line segment. End points are denoted by Pxc where x is the cell number of where the endpoint is located, and c is the colour of the endpoint. We can use Fxc to represent the filled proposition, this means Fxc is a filled cell at cell number x with colour c.

- Jape Proof #1: A 2x2 board with endpoints on the diagonals can not be solved. This is fairly unintuitive until you visualize a 2x2 board, so a depiction of such a board, along with the unique numbering of each cell is presented in Figure 5, and as is clearly visible there does not exist a way to connect these dots that does not leave one cell empty. We will use JAPE to prove this.

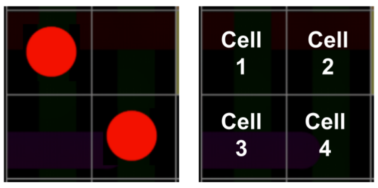


Figure 5: Unsolvable 2x2 board, along with the cell numbering

In this case we have one endpoint at x=0 of colour 1 and one endpoint at x=4 of colour 1. We then encode the constraint that every endpoint implies *exactly* one line segment, $P1 \rightarrow ((S12 \wedge \neg S13) \vee (S13 \wedge \neg S12))$ and $P4 \rightarrow ((S42 \wedge \neg S43) \vee (S43 \wedge \neg S42))$Now we must encode the constraint that each non endpoint cell has two lines. To encode this in JAPE we note that we only have two non endpoint cells cell 3 and cell 2. For each of these there are two possible lines for example, for cell 3 there is a line from 3 to 1 and 3 to 4. Since there are two possible lines for each non endpoint cell we say that a non endpoint implies that both of the possible lines exist. In Jape we write, $\neg P21 \rightarrow (S121 \wedge S241)$ and $\neg P31 \rightarrow (S341 \wedge S131)$. Additionally we will encode that each colour must have exactly two endpoints. We simply write that a filled cell of colour 1 implies two end-points of that colour; $F41 \rightarrow P11 \wedge P41 \wedge \neg P31 \wedge \neg P21$ and $F11 \rightarrow P11 \wedge P41 \wedge \neg P31 \wedge \neg P21$. With these constraints and our base encoding of the propositions, we can prove this board is unsolvable in JAPE in Figure 6

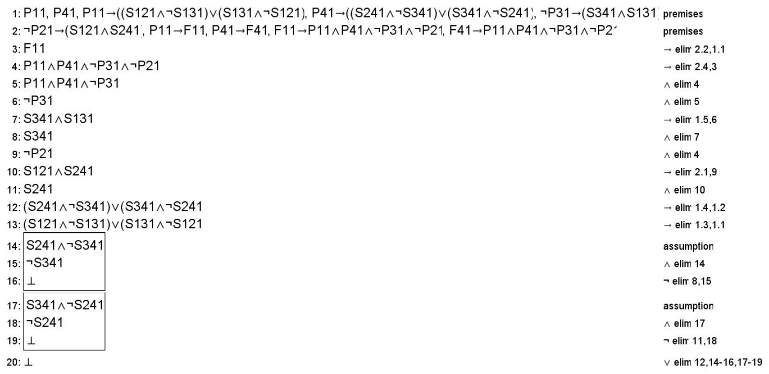

```
 1: P11, P41, P11→((S121∧¬S131)∨(S131∧¬S121), P41→((S241∧¬S341)∨(S341∧¬S241), ¬P31→(S341∧S131   premises
 2: ¬P21→(S121∧S241, P11→F11, P41→F41, F11→P11∧P41∧¬P31∧¬P21, F41→P11∧P41∧¬P31∧¬P2           premises
 3: F11                                                                                        → elim 2.2,1.1
 4: P11∧P41∧¬P31∧¬P21                                                                          → elim 2.4,3
 5: P11∧P41∧¬P31                                                                               ∧ elim 4
 6: ¬P31                                                                                       ∧ elim 5
 7: S341∧S131                                                                                  → elim 1.5,6
 8: S341                                                                                       ∧ elim 7
 9: ¬P21                                                                                       ∧ elim 4
10: S121∧S241                                                                                  → elim 2.1,9
11: S241                                                                                       ∧ elim 10
12: (S241∧¬S341)∨(S341∧¬S241                                                                   → elim 1.4,1.2
13: (S121∧¬S131)∨(S131∧¬S121                                                                   → elim 1.3,1.1
14: │S241∧¬S341│                                                                               assumption
15: │¬S341     │                                                                               ∧ elim 14
16: │⊥          │                                                                               ¬ elim 8,15
17: │S341∧¬S241│                                                                               assumption
18: │¬S241     │                                                                               ∧ elim 17
19: │⊥          │                                                                               ¬ elim 11,18
20: ⊥                                                                                          ∨ elim 12,14-16,17-19
```

Figure 6: JAPE proof of unsolvable board

- Jape Proof #2: It is not possible to have a flow that contains more than one color. Intuitively, this happens because line segments force their color onto the cells underneath them. Since cells can only have one color, line segments of different colors would force the cell that they meet at to be different colors, which would cause a contradiction.

  In the Jape proof below, there is a blue horizontal flow starting at cell 0 and going to cell 3. The middle line segment (from cell 1 to 2) is missing. We say that the color of the line segment from cell 1 to 2 is either a blue, green, or red, and we prove that it must be the blue line segment. The other premises ensure that line segments force their color onto the cells below them and that cells only have exactly one color. See Figure 7 for the setting of this proof and Figure 8 for the proof. *Note* that in this proof there is a slight change in notation where Cxc is used for the filled proposition, x is a cell number and c is the colour that cells that cell number.

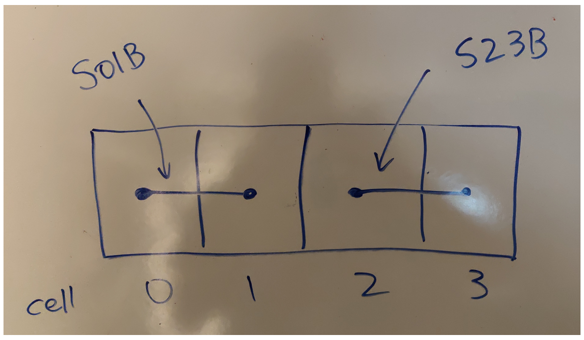
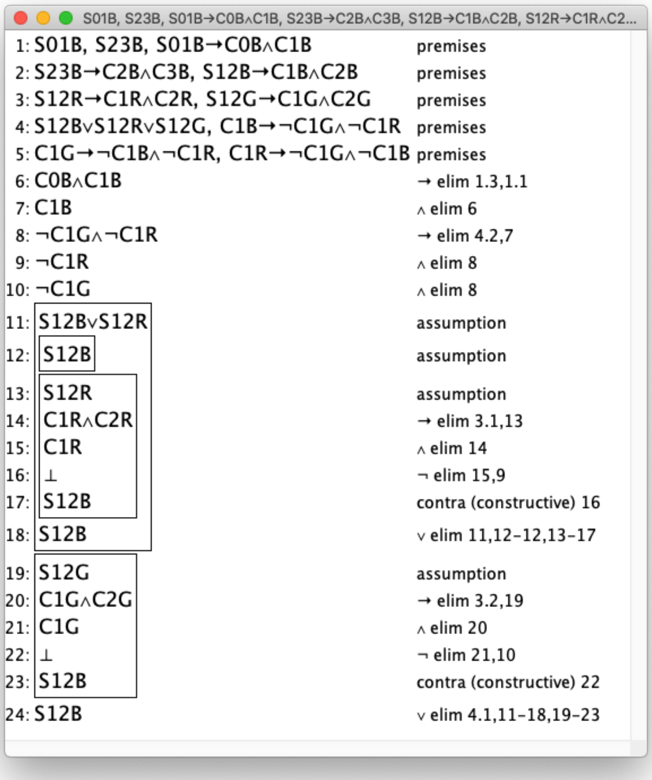


Figure 7: Setting for jape proof #2



```
S01B, S23B, S01B→C0B∧C1B, S23B→C2B∧C3B, S12B→C1B∧C2B, S12R→C1R∧C2...
 1: S01B, S23B, S01B→C0B∧C1B                  premises
 2: S23B→C2B∧C3B, S12B→C1B∧C2B                premises
 3: S12R→C1R∧C2R, S12G→C1G∧C2G                premises
 4: S12B∨S12R∨S12G, C1B→¬C1G∧¬C1R             premises
 5: C1G→¬C1B∧¬C1R, C1R→¬C1G∧¬C1B              premises
 6: C0B∧C1B                                   → elim 1.3,1.1
 7: C1B                                       ∧ elim 6
 8: ¬C1G∧¬C1R                                 → elim 4.2,7
 9: ¬C1R                                      ∧ elim 8
10: ¬C1G                                      ∧ elim 8
11: S12B∨S12R                                 assumption
12:   S12B                                    assumption
13:   S12R                                    assumption
14:   C1R∧C2R                                 → elim 3.1,13
15:   C1R                                     ∧ elim 14
16:   ⊥                                       ¬ elim 15,9
17:   S12B                                    contra (constructive) 16
18: S12B                                      ∨ elim 11,12-12,13-17
19: S12G                                      assumption
20: C1G∧C2G                                   → elim 3.2,19
21: C1G                                       ∧ elim 20
22: ⊥                                         ¬ elim 21,10
23: S12B                                      contra (constructive) 22
24: S12B                                      ∨ elim 4.1,11-18,19-23
```

Figure 8: JAPE proof that all flows must be contiguous

- Jape Proof #3: A 2x2 board with adjacent endpoints of two different colours cannot be solved. This is a fairly direct proof, as one of our constraints states that there must be exactly two endpoints of a single colour. However, while this constraint seems simple in plain English, it is slightly more complex to encode in Jape. The way we approach this is to state that if an endpoint at cell 1 of colour 1 exists, then cell 2 contains an endpoint of colour 1. Additionally, if an endpoint of colour 1 exists at cell 2, then cell 2 does not contain an endpoint of colour 2, and it does not contain an endpoint of colour 3 or colour 4, assuming of course there are only 4 colours available.

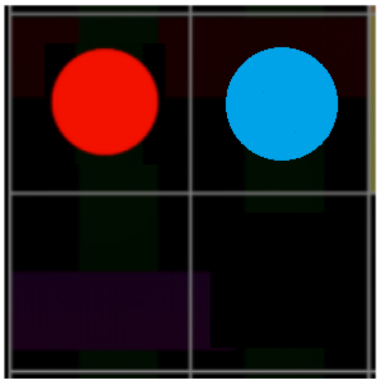
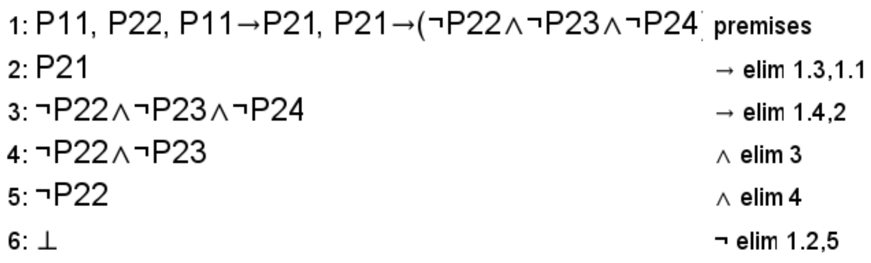


Figure 9: 2x2 board with different coloured endpoints



```
1: P11, P22, P11→P21, P21→(¬P22∧¬P23∧¬P24   premises
2: P21                                       → elim 1.3,1.1
3: ¬P22∧¬P23∧¬P24                            → elim 1.4,2
4: ¬P22∧¬P23                                 ∧ elim 3
5: ¬P22                                      ∧ elim 4
6: ⊥                                         ¬ elim 1.2,5
```

Figure 10: JAPE proof of 2x2 board with different coloured endpoints