Group 48: Modelling Connect Four

Karl Dorogy (20168380)

Tom Lin (20148517)

Douglas Chafee (20144996)

Matthew Krpac (20188368)

Course Modelling Project Documentation:

CISC 204/CMPE 204

Logic for Computing Science

December 8th, 2020

Project Summary:

- Connect Four is a game where the color black or red, each representing a single player, must connect four or more of their color pieces consecutively by dropping units down columns in turns; eventually forming a line of length 4 or greater in the finite playing area of a 6x7 grid, whether it be horizontally, vertically, or diagonally.

- Our model depicts the ongoing as well as end states of a Connect Four board with the intent of discerning if either black or red has won in a winning state, or if the generate board/state is a no winner/stalemate state. Our model also depicts the number of possible winning states for each color, all colors, and no colors which would be a no winner/stalemate state.

- In order to achieve a winning state in our model, either a red or black player must connect four of their pieces consecutively in a line, whether it be via a single row, column, or diagonal. A winning state in our model also has the possibility to be any combination of the above winning states.

- In order to achieve a no winner/stalemate state in our model, both the red and black players must not connect four of their pieces consecutively in a single row, column, diagonal, or any combination of previous single lines.

- Within our model, we restrict a Connect Four board to the finite playing area of a six row by seven columns board (which can be decreased or increased in size in our code by changing the values that are assigned to rowNum and columnNum).

- In reality, the physically upright nature of a Connect Four board causes pieces to fall to the lowest available position, thus, to ensure the relevance of our model we made this an additional core tenant and constraint that has to be considered for each and every piece that wants to be placed in our model of Connect Four.

- In tandem with the abstraction of gravity in our model, we have established and added many more additional constraints throughout modelling the game, Connect Four, in which would assist in creating valid Connect Four board states (to a certain extent) to be then further examined, explored, or logically assessed for winning states as well as stalemate states shown below in our documentation.

Propositions

There is a total of 14 propositions in our model of Connect Four and 3 proposition used for counting being described below**:**

(B) - $blackBoard_{(i,\ j)}$ = True when the slot/position at row i and column j in a 6 by 7 board is occupied by a single black piece.

(R) - $redBoard_{(i,,\ j)}$ = True when the slot/position at row i and column j in a 6 by 7 board is occupied by a single red piece.

(E) – $emptyBoard_{(i,\ j)}$ = True when the slot/position at row I and column j, in a 6 by 7 board is not occupied by a single red piece or a single black piece.

(H) - $blackRow_{(i,\ j)}$ = True when there are 4 consecutive black pieces in a row in/at row i and starting in/at column j. This implies that there are 3 consecutive black pieces to the right of position (i ,j) on the same row.

(H) - $redRow_{(i,\ j)}$ = True when there are 4 consecutive black pieces in a row in/at row i and starting in/at column j. This implies that there are 3 consecutive black pieces to the right of position (i ,j) on the same row.

(C) - blackColumn$_{(i, j)}$ = True when there are 4 consecutive black pieces in a column in/at column j and starting in/at row i. This implies that there are 3 consecutive black pieces below position (i ,j) in the same column.

(C) - redColumn$_{(i, j)}$ = True when there are 4 consecutive red pieces in a column in/at column j and starting in/at row i. This implies that there are 3 consecutive red pieces below position (i ,j) in the same column.

(D) - leftBlackDiagonal$_{(i, j)}$ = True when there are 4 consecutive black pieces in a left upward facing diagonal starting in/at row i and in/at column j. This implies that there are 3 consecutive black pieces starting at position (i ,j) and moving down one row and to the right one column after each piece in the same diagonal channel.

(D) - leftRedDiagonal$_{(i, j)}$ = True when there are 4 consecutive red pieces in a left upward facing diagonal starting in/at row i and in/at column j. This implies that there are 3 consecutive red pieces starting at position (i ,j) and moving down one row and to the right one column after each piece in the same diagonal channel.

(D) - RightBlackDiagonal$_{(i, j)}$ = True when there are 4 consecutive black pieces in a right upward facing diagonal starting in/at row i and in/at column j. This implies that there are 3 consecutive black pieces starting at position (i ,j) and moving down one row and to the left one column after each piece in the same diagonal channel.

(D) - RightRedDiagonal$_{(i, j)}$ = True when there are 4 consecutive red pieces in a right upward facing diagonal starting in/at row i and in/at column j. This implies that there are 3 consecutive red pieces starting at position (i ,j) and moving down one row and to the left one column after each piece in the same diagonal channel.

(S_B) - BlackWin = True when any blackRow, blackColumn, leftBlackDiagonal, or rightBlackDiagonal are true, otherwise false.

(S_R) - RedWin = True when any redRow, redColumn, leftRedDiagonal, or rightRedDiagonal are true, otherwise false.

(T) - NoWin = True when there is neither a BlackWin or RedWin, basically implying that there isn't a single case where 4 consecutive piece for either colors, black or red, are true.

blackPartialCount = The partial count of red pieces at position (i, j)

redPartialCount = The partial count of red pieces at position (i, j)

totalCount = the max total black count of pieces possible in a 6 row 7 column board which is 54 pieces.

Constraints:

All constraints in our model of Connect Four described below:

This constraint describes the abstract concept of gravity which ensures that all the pieces on the board must have pieces below them to the floor. This works in tandem with other constraints (i.e the empty slots constraints for row, columns, and diagonals) in order to make sure the game will end in a valid way/move.

$\neg E_{(x,\ y)} \rightarrow \neg E_{(x+1\ ,\ y)}$

These two constraints describe a winning row for each color, black and red, by holding true if and only if a row consists of four consecutive pieces of the same color. We have to generalize the constraint here, replacing the specific blackRow for black pieces and redRow for red pieces at position (x,y) to H, which is just a general winning row at the position (x,y).

$H \leftrightarrow B_{(x,\ y)} \wedge B_{(x,\ y+1)} \wedge B_{(x,y+2)} \wedge B_{(x,\ y+3)}$

$H \leftrightarrow R_{(x,\ y)} \wedge R_{(x,\ y+1)} \wedge R_{(x,\ y+2)} \wedge R_{(x,\ y+3)}$

This constraint makes sure there is at least one empty slot above one of the winning row positions. Paired with our gravity constraint this provides a viable route via an entire empty column above one of the winning row positions in which the winning piece of that row would use to complete the winning row.

$$H \rightarrow E_{(x-1,\, y)} \vee E_{(x-1,\, y+1)} \vee E_{(x-1,\, y+2)} \vee E_{(x-1,\, y+3)}$$

These two constraints describe a winning column for each color, black and red, by holding true if and only if a column consists of four consecutive pieces of the same color. We have to generalize the constraint here, replacing the specific blackColumn for black pieces and redColumn for red pieces at position (x,y) in our code to C, which is just a general winning column at the position (x,y).

$$C \leftrightarrow (B_{(x,\, y)} \wedge B_{(x+1,\, y)} \wedge B_{(x+2,\, y)} \wedge B_{(x+3,\, y)})$$

$$C \leftrightarrow (R_{(x,\, y)} \wedge R_{(x+1,\, y)} \wedge R_{(x+2,\, y)} \wedge R_{(x+3,\, y)})$$

This constraint makes sure there is an empty slot above the winning column positions, in which working with our gravity constraint provides a viable route via an entire empty column above the winning column positions in which the winning piece would have then used to complete the winning column.

$$C \rightarrow E_{(x-1,y)}$$

These four constraints describe a winning diagonal for each color, black and red, by holding true if and only if a diagonal consists of four or more consecutive pieces of the same color. We have to generalize the constraint here, replacing the specific leftBlackDiagonal and RightBlackDiagonal for black pieces and or leftRedDiagonal and RightRedDiagonal for red pieces at position (x,y) in our code to D, which is just a general winning diagonal at the position (x,y).

Left Upward Facing Diagonals:

$$D \leftrightarrow B_{(x, y)} \wedge B_{(x+1, y+1)} \wedge B_{(x+2, y+2)} \wedge B_{(x+3, y+3)}$$

$$D \leftrightarrow R_{(x, y)} \wedge R_{(x+1, y+1)} \wedge R_{(x+2, y+2)} \wedge R_{(x+3, y+3)}$$

Right Upward Facing Diagonals:

$$D \leftrightarrow B_{(x, y)} \wedge B_{(x+1, y-1)} \wedge B_{(x+2, y-2)} \wedge B_{(x+3, y-3)}$$

$$D \leftrightarrow R_{(x, y)} \wedge R_{(x+1, y-1)} \wedge R_{(x+2, y-2)} \wedge R_{(x+3, y-3)}$$

This constraint makes sure there is at least one empty slot above one of the winning diagonals positions, in which working with our gravity constraint provides a viable route via an entire empty column above one of the winning diagonal positions in which the winning piece of that diagonal would have then used to complete the winning diagonal.

Left Upward Facing Diagonals:

$$D \rightarrow E_{(x-1, y)} \vee E_{(x, y+1)} \vee E_{(x+1, y+2)} \vee E_{(x+2, y+3)}$$

Right Upward Facing Diagonals:

$$D \rightarrow E_{(x-1, y)} \vee E_{(x, y-1)} \vee E_{(x+1, y-2)} \vee E_{(x+2, y-3)}$$

This constraint checks that there is a black win if and only if there is a black row, column, or diagonal. Also allows there to be combinations of rows, columns, or diagonals.

$$S_B \leftrightarrow H_B \vee C_B \vee D_B$$

This constraint checks that there is a red win if and only if there is a red row, column, or diagonal. Also allows there to be combinations of rows, columns, or diagonals.

$$S_R \leftrightarrow H_R \vee C_R \vee D_R$$

This constraint makes sure that there is a tie/stalemate if and only if there are no rows, columns, or diagonals for each color in the board.

$$T \rightarrow (\neg H_B \wedge \neg C_B \wedge \neg D_B) \wedge (\neg H_R \wedge \neg C_R \wedge \neg D_R)$$

This constraint makes sure an empty slot at the position (x,y) implies that there is no red piece or black piece at the position (x,y) as well.

$$E \rightarrow \neg R \wedge \neg B$$

This constraint makes sure a red piece at the position (x,y) implies that there is no empty piece or black piece at the position (x,y) and there are pieces under the red piece, abiding to the gravity constraint stated earlier.

$$R \rightarrow \neg E \wedge \neg B \wedge G$$

This constraint makes sure a black piece at the position (x,y) implies that there is no empty piece or red piece at the position (x,y) and there are pieces under the black piece, abiding to the gravity constraint stated earlier.

$$B \rightarrow \neg E \wedge \neg R \wedge G$$

This constraint makes sure that for a single slot to exist on the board at position(x.y), there must be an empty slot, or a single red piece, or a single black piece at that position specific position.

$$E \vee R \vee B$$

This constraint states that if red wins, then black does not win and game does not tie.

$$S_R \rightarrow \neg S_B \wedge \neg T$$

This constraint states that if black wins, then red does not win and the game does not tie.

$$S_B \rightarrow \neg S_R \wedge \neg T$$

This constraint states that if the game ties, then red does not win and black does not win.

$$T \rightarrow \neg S_R \wedge \neg S_R$$

This constraint makes sure that there can only be one winning row. This means that if row i is true, meaning row i has 4 or more pieces of the same color consecutively, then all other rows besides row i must be false essentially.

Example: If row 1 is true, then not row 0, row 2, or row 3, or row 4, or row 5 in the case of a 6 row by 7 column board.

$$H_1 \rightarrow \neg(H_0 \vee H_2 \vee H_3 \ldots \vee H_x)$$

This constraint makes sure that there can only be one winning column. This means that if column j is true, meaning column j has 4 pieces of the same color consecutively, then all other columns besides column j must be false essentially.

Example: If column 2 is true, then not column 0, or not column 1, or column 3, or column 4, or column 5, or column 6 in the case of a 6 row by 7 column board.

$$C_2 \rightarrow \neg(C_0 \vee C_1 \vee C_3 \ldots \vee C_x)$$

These two constraints make sure that there can only be one winning diagonal channel for a left and right diagonal. This means that if left diagonal (i, j) is true, meaning diagonal (i, j) has 4 or more pieces of the same color consecutively in a upward left facing direction, then all other diagonals that face in an upward left facing direction besides diagonal (i ,j) must be false essentially. The same can be said for right upward facing diagonals.

Example: If left diagonal 1 is true, then not left diagonal 0, or left diagonal 2, or left diagonal 3, or left diagonal 4 in the case of a 6 by 7 board since there are only 4 possible diagonal channels in a board of that size.

$$D1 \rightarrow \neg(D_0 \vee D_2 \vee D_3 \ldots \vee D_x)$$

This constraint limits the position where a specific color's column can be and or positioned at on a connect four board in reference to the positions of the other winning rows or diagonals of the same color, since it is possible for a player to win in a single move also creating a winning state combing a winning column with a winning row, diagonals, or both at the same time.

Example: If column (i, j), then rows or diagonals not around position (i, j) cannot be true.

$$C_{(i, j)} \rightarrow \neg (H_{(i, j)} \vee D_{(i, j)})$$

These last couple constraints are for counting the number of piece that each color have for a game of Connect Four, making sure by the end that they are equal in pieces indicating whoever won played first move of the game.

Checks that total count is true if and only if partial counts are equal to the full count:

- E.add_constraint(iff(totalCount[c], partialCount[rowNum- 1][columnNum - 1][c]))

Checks that the number of pieces you have can't be greater than the number of pieces that you've already seen via the index/position of the board:

- E.add_constraint(~partialCount[i][j][c])

These two checks that only a piece/count can only be a black piece or red piece :

- E.add_constraint(iff(partialCount[0][0][0], ~boardColor[0][0]))
- E.add_constraint(iff(partialCount[0][0][0], boardColor[0][0]))

This last constraint checks and looks at the other color pieces counts to decide the current color piece to be placed using these below:

- E.add_constraint(iff(partialCount[i][j][0], partialCount[(i-1) if (j==0) else i][(columnNum-1) if (j==0) else (j-1)][0] & ~boardColor[i][j]))

- E.add_constraint(iff(partialCount[i][j][c], increased | stay_same))

Model Exploration

Steps/Thought Process of Adding Constraints for Model:

1) Add/create variables and variable boards for a single black, red, or empty piece represented in the picture below of black peices:

```
[Black(0,0), Black(0,1), Black(0,2), Black(0,3), Black(0,4), Black(0,5), Black(0,6)]
[Black(1,0), Black(1,1), Black(1,2), Black(1,3), Black(1,4), Black(1,5), Black(1,6)]
[Black(2,0), Black(2,1), Black(2,2), Black(2,3), Black(2,4), Black(2,5), Black(2,6)]
[Black(3,0), Black(3,1), Black(3,2), Black(3,3), Black(3,4), Black(3,5), Black(3,6)]
[Black(4,0), Black(4,1), Black(4,2), Black(4,3), Black(4,4), Black(4,5), Black(4,6)]
[Black(5,0), Black(5,1), Black(5,2), Black(5,3), Black(5,4), Black(5,5), Black(5,6)]
```

2) Add/created variables and variable boards for all the possible row arrangements for a row represented in the picture below:

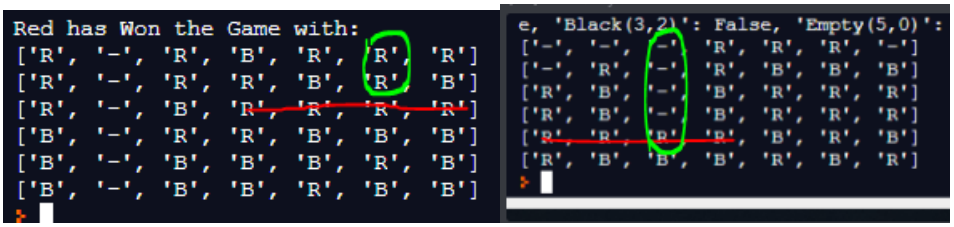
```
[BlackWinningRow(0,0), BlackWinningRow(0,1), BlackWinningRow(0,2), BlackWinningRow(0,3)]
[BlackWinningRow(1,0), BlackWinningRow(1,1), BlackWinningRow(1,2), BlackWinningRow(1,3)]
[BlackWinningRow(2,0), BlackWinningRow(2,1), BlackWinningRow(2,2), BlackWinningRow(2,3)]
[BlackWinningRow(3,0), BlackWinningRow(3,1), BlackWinningRow(3,2), BlackWinningRow(3,3)]
[BlackWinningRow(4,0), BlackWinningRow(4,1), BlackWinningRow(4,2), BlackWinningRow(4,3)]
[BlackWinningRow(5,0), BlackWinningRow(5,1), BlackWinningRow(5,2), BlackWinningRow(5,3)]
```

Similar boards had to be then had to be created for columns, left diagonals, and right diagonals but in different dimensions since there are different number of possible arrangements for each type of line in a finite playing area shown below:

```
python run.py
[BlackWinningColumn(0,0), BlackWinningColumn(0,1), BlackWinningColumn(0,2), BlackWinningColumn(0,3), BlackWinningColumn(0,4), BlackWinningColumn(0,5), BlackWinningColumn(0,6)]
[BlackWinningColumn(1,0), BlackWinningColumn(1,1), BlackWinningColumn(1,2), BlackWinningColumn(1,3), BlackWinningColumn(1,4), BlackWinningColumn(1,5), BlackWinningColumn(1,6)]
[BlackWinningColumn(2,0), BlackWinningColumn(2,1), BlackWinningColumn(2,2), BlackWinningColumn(2,3), BlackWinningColumn(2,4), BlackWinningColumn(2,5), BlackWinningColumn(2,6)]
```

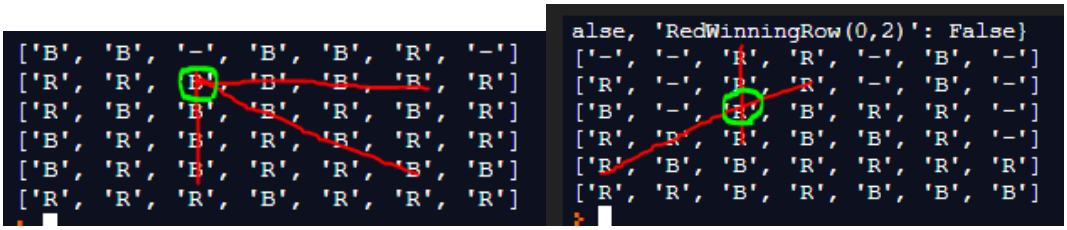
```
[LeftBlackWinningDiagonal(0,0), LeftBlackWinningDiagonal(0,1), LeftBlackWinningDiagonal(0,2), LeftBlackWinningDiagonal(0,3)]
[LeftBlackWinningDiagonal(1,0), LeftBlackWinningDiagonal(1,1), LeftBlackWinningDiagonal(1,2), LeftBlackWinningDiagonal(1,3)]
[LeftBlackWinningDiagonal(2,0), LeftBlackWinningDiagonal(2,1), LeftBlackWinningDiagonal(2,2), LeftBlackWinningDiagonal(2,3)]
```

3) Noticed that for a board to be valid to the rules of the game, Connect Four, then we must have a viable “route” or “column” for which that last piece must have been placed to complete a winning row, column, or diagonal shown below in our screenshots of the before (left) and after (right) pictures:



This constraint was also then extended and used for columns and diagonals in their respective boards and lead to the gravity constraint implying that an occupied slot has occupied slots below it to the floor, which creates this “empty” column/route.

4) Realized that there could be combinations of wining rows, diagonals, or columns so in addition to the previous constraint, we had to explore more constraints to limit the winning states of a column in correlation to winning rows and diagonals, since in order to have a winning column combined with either a winning row or diagonal, the last single piece must be placed correctly so that it completes the winning column as well as the winning row and or diagonal.



5) We realized that toward the end of modelling our Connect four model, order to get a valid connect 4 board abiding to the rule of turns, we needed to count each individual piece of the board/state and determine whether black or red should have placed the next piece. This means that the total # of black pieces should be equal to the red pieces when black wins and vice versa when red wins. This was needed because if these constraints weren't in place we would be

exploring the number of states in which diagonals, rows or columns could be arranged in a finite 6 by 7 board.

Finally, we explored our model of Connect Four in three different ways:

- The first two ways our model of Connect Four was explored is how we explored both the number of ways and how/what defines a black win and red win. This was done by defining all the characteristics of a winning state for the color black and red that would be then iterated through, checking the board for a winning row, column or diagonal of black pieces or red pieces. Once a generated state has been determined to be winning, it then checks which color won, that being black or red using our previous declared constraints. We also explore further in these two ways by then also counting the number of ways there can be for a winning state of color, black or red, in which given our constraints that we built, we could then add and or remove constraints to explore the different/varying number of ways a winning state can be created for either color.

- Our third way of model exploration for our model of Connect Four was how we explored both the number of ways and how/what defines a tie, or stalemate, or no winner state where neither the colors black nor red has won yet. This was done by defining all the characteristics of a stalemate state for the colors black and red that would be then also be iterated through, checking the board cannot be for a winning row, column or diagonal of black pieces or red pieces. Once the generated state has been determined to be a tie, using the negate constraints determined for each color, it was then further explored by also counting the number of ways there can be a tie, stalemate, or no winner state, in which again given our constraints that we built, we could then add and or remove constraints to explore the

different/varying number of ways a tie/stalemate state can be created from each color.

First-Order Extension:

Exploring our model of Connect Four and extending it to the predicate logic setting could be described using some of our previously stated constraints. We could extend our model using predicate logic by using the constraint that checks for multiple winning rows in a single winning state of a Connect Four game. Instead of checking every possible row and its 4 possible consecutive pieces arrangements individually, like our current constraints, It can be said that if there exists a single winning row for example, then for all other rows cannot be single winning row (true) except for the initial single row that won. This same extension of predicate logic for our model could also be applied to the constraints for columns and diagonals as well.

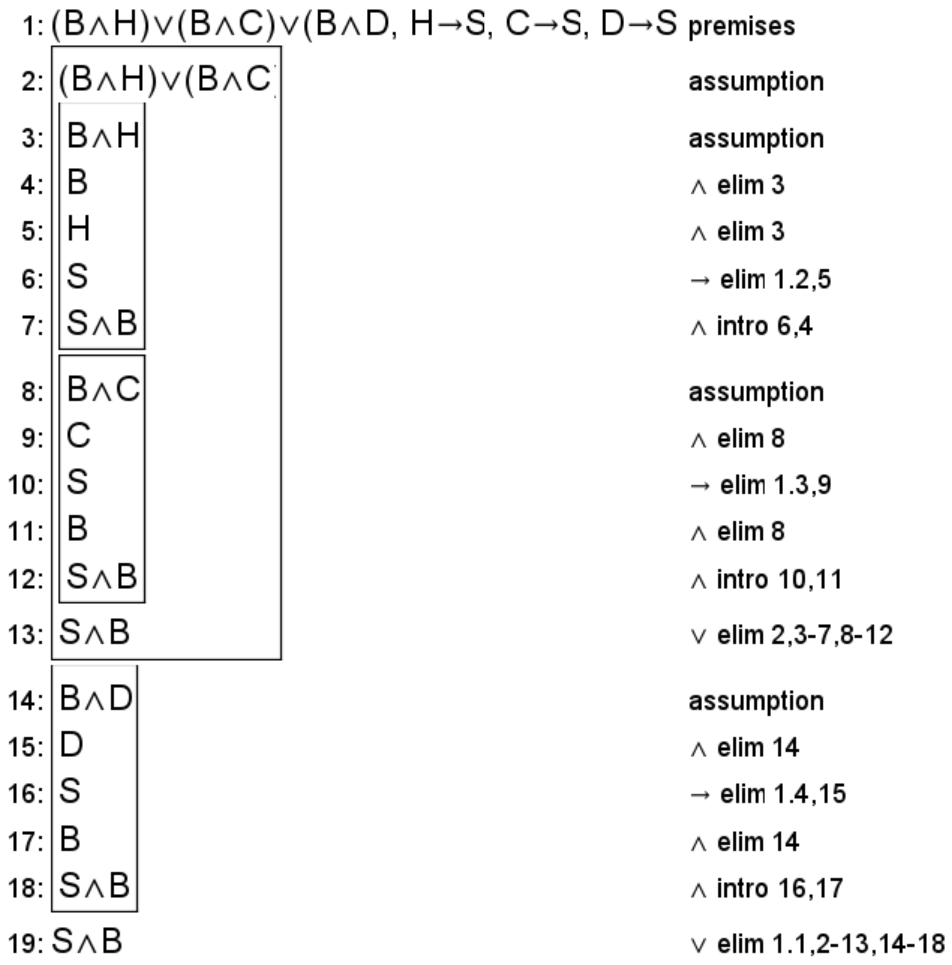
Continuing our predicate logic exploration for our model of Connect Four, we could also extend its logic and say that for all black pieces, there must be the same number as for all red pieces in a game of Connect Four. This logic could also be said in existential form saying that if there exist a single black piece on the board, then there exists a single red piece for that single black piece on the board.

Jape Proofs:

B = Black, R = Red, H = Row, C = Column, D = Diagonal, S = Win, T = Tie

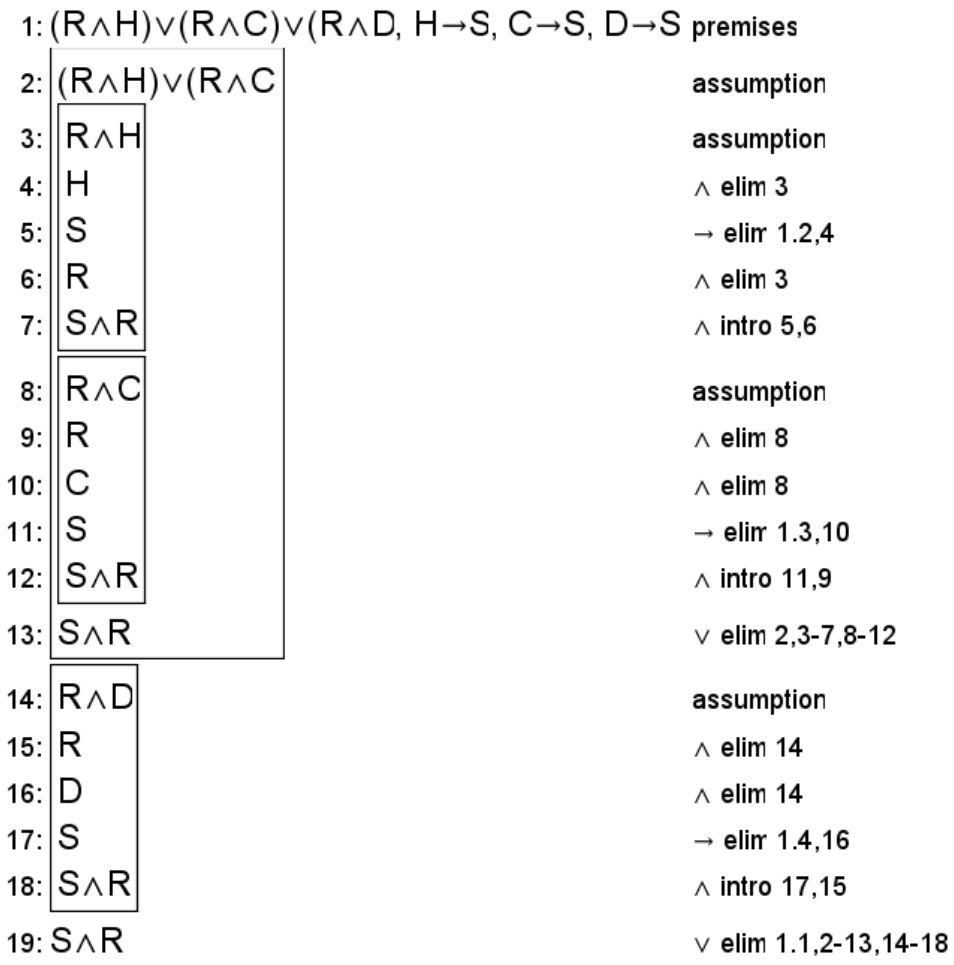
Black has won with a row, column, or diagonal:

(B∧H)∨(B∧C)∨(B∧D), H→S, C→S, D→S ⊢ S∧B



Red has won with a row, column, or diagonal:

(R∧H)∨(R∧C)∨(R∧D), H→S, C→S, D→S ⊢ S∧R



A tie has occurred:

1:	$T \rightarrow ((\neg B \wedge \neg H) \wedge (\neg B \wedge \neg C) \wedge (\neg B \wedge \neg D)) \wedge ((\neg R \wedge \neg H) \wedge (\neg R \wedge \neg C) \wedge (\neg R \wedge \neg D))$	premise
2:	$S \rightarrow (C \vee D \vee H)$, $T \vee S$, $T \wedge \neg S$, $C \rightarrow S$, $H \rightarrow S$, $D \rightarrow S$	premises
3:	$\neg S$	$\wedge$ elim 2.3
4:	T	$\wedge$ elim 2.3
5:	$((\neg B \wedge \neg H) \wedge (\neg B \wedge \neg C) \wedge (\neg B \wedge \neg D)) \wedge ((\neg R \wedge \neg H) \wedge (\neg R \wedge \neg C) \wedge (\neg R \wedge \neg D)$	$\rightarrow$ elim 1,4
6:	$(\neg B \wedge \neg H) \wedge (\neg B \wedge \neg C) \wedge (\neg B \wedge \neg D) \wedge (\neg R \wedge \neg H) \wedge (\neg R \wedge \neg C) \wedge (\neg R \wedge \neg D$	assumption
7:	$\neg S$	hyp 3
8:	$(\neg B \wedge \neg H) \wedge (\neg B \wedge \neg C) \wedge (\neg B \wedge \neg D) \wedge (\neg R \wedge \neg H) \wedge (\neg R \wedge \neg C) \wedge (\neg R \wedge \neg D) \rightarrow \neg S$	$\rightarrow$ intro 6-7