Queen's UNIVERSITY

GRP_4: Theseus and the Minotaur

Lola Assad - 20lsa2

Sierra Cabrera - 18scc13

Grace Codrington - 21gcmc1

Selena Zou - 20sz88

Course Modelling Project

CISC/CMPE 204

Logic for Computing Science

December 2, 2022

Abstract

The board game this project attempts to model is *Theseus and the Minotaur*, a single-player game developed by Robert Abbott in 1998. The player plays Theseus, while a computer algorithm plays the Minotaur. The player's goal is to get Theseus to the exit square before the Minotaur can catch and eat him; Theseus, the Minotaur, and the exit square have their positions randomized prior to the start of the game. The catch is that the Minotaur can make two moves for each move Theseus makes, making him faster. However, the Minotaur's algorithm is inefficient and can be foolish: he will always move horizontally if he can get closer to Theseus this way, and only moves vertically if the previous condition evaluates to false; if neither horizontal nor vertical moves get him closer to Theseus, he skips his turn. This, in combination with the randomized hedges which prevent both Theseus and the Minotaur from moving between certain squares, means that with clever play, Theseus can sometimes manage to trap the Minotaur behind a wall and safely make his way to the exit.

We will limit our project scope to a maze of 6x6 dimensions, with a randomized number of hedges between 15 and 40, a randomized number of moves between 0 and 15 allotted for Theseus (his moves running out without his making a successful escape counts as a loss), a randomized exit square, and randomized starting positions for Theseus and the Minotaur.

In our model exploration, the problem will be encoded into logic through an initial encoding of Theseus and the Minotaur's starting positions, as well as the location of the exit square and the hedges in the maze. Then the Minotaur's moves and Theseus' possible moves are encoded. If a move to a given square allows the Minotaur to catch Theseus, that square is not safe for Theseus. In addition, if a series of moves culminates in the Minotaur catching Theseus, the sequence of squares in that series are not safe. A legal move consists of moving one square up, one square down, one right, one left, or skipping one's turn and remaining in the same spot. While the Minotaur will always have only one move strictly defined by his constraints, Theseus may have up to five options (up, down, left, right, and staying in place), the consequences of each of which will be assessed to either affirm or negate them in the model. As we seek only to determine whether a given maze is escapable for Theseus, the results of our model exploration will only tell us whether there exists a safe path for Theseus from his starting square to the exit, and not what this path is (if it exists) or how many there are.

Propositions

r corresponds to the row and c corresponds to the column. Starting at 0 for the left most column and top row. Going up to 5 for the right most column and the bottom row.

$TH_{r,c}$ - corresponds to a position in the grid/maze and is true if there is a horizontal hedge at the top edge of the square.

$BH_{r,c}$ - corresponds to a position in the grid/maze and is true if there is a

horizontal hedge at the bottom edge of the square.

$RH_{r,c,}$ - corresponds to a position in the grid/maze and is true if there is a vertical hedge at the right edge of the square.

$LH_{r,c,}$ - corresponds to a position in the grid/maze and is true if there is a vertical hedge at the left edge of the square.

$T_{r,c}$ - corresponds to Theseus' position in the maze and is true if it is Theseus' current position.

$M_{r,c}$ - corresponds to the Minotaur's position in the maze and is true if it is the Minotaur's current position.

$S_{r,c}$ - corresponds to a given position in the grid/maze and is true if given position is the exit/safe square.

$E_{r,c}$ - corresponds to a given position in the grid/maze and is true if Theseus and the Minotaur are both on that square. AKA Theseus has been eaten.

$TM_{r,c}$ - corresponds to a position in the grid/maze and is true if Theseus will choose to move to this position.

$MM_{r,c}$ - corresponds to a given position in the grid/maze and is true if Minotaur will choose to move to this position.

$TP_{r,c}$ - corresponds to a position in the grid/maze and is true if that position is in the list of possible positions in the maze Theseus can move to.

$MP_{r,c}$ - corresponds to a position in the grid/maze and is true if Minotaur can move to that space.

$MT_{r,c}$ - corresponds to a position in the grid/maze and is true if moving to that space will ensure the Minotaur gets closer to Theseus.

N - is true if Theseus has moves left in his list of possible moves.

A - is true if Theseus has turns left.

W - is true if Theseus has won the game.

L - is true if Theseus has has lost the game.

Constraints

Hedge Relationships Example: If there is a hedge at the top of a given square, there is also a hedge at the bottom of the square above it:

$$TH_{r,c} \rightarrow BH_{r-1,c}$$

Theseus Moving Outside Grid Example: Theseus cannot move outside borders.

$$T_{r,5} \rightarrow \neg TP_{r,c+1}$$

Theseus and Hedge Example: Theseus cannot move across a hedge. If Theseus is on the given square, and there is a hedge on the right of the given square, Theseus cannot move right.

$$T_{r,c} \wedge RH_{r,c} \rightarrow \neg TP_{r,c+1}$$

Minotaur and Hedge Example: Minotaur cannot move across a hedge. If Minotaur is on the given square, and there is a hedge at the bottom of the given square, Minotaur cannot move down.

$$M_{r,c} \wedge BH_{r,c} \rightarrow \neg MP_{r+1,c}$$

Theseus Can Move: Theseus can only move if he has turns left and he has moves left.

$$A \wedge \neg N \rightarrow TC$$

Theseus Will Escape: If Theseus can move, he can move to a given square, moving to that square won't get him eaten, and that square is the exit square, Theseus will move to that square.

$$TC \wedge TP_{r,c} \wedge \neg E_{r,c} \wedge S_{r,c} \rightarrow TM_{r,c}$$

Minotaur Will Move: The Minotaur will only move if he can move to a given square, and if moving to that square gets him closer to Theseus.

$$MP_{r,c} \wedge MT_{r,c} \rightarrow MM_{r,c}$$

How Will Minotaur Move Example: If the Minotaur can move vertically and moving vertically will get him closer to Theseus, and he cannot move horizontally or moving horizontally won't get him closer, he will move vertically.

$$((\neg MP_{r,c+1} \vee \neg MP_{r,c-1}) \vee (\neg MT_{r,c+1} \vee \neg MT_{r,c-1})) \wedge ((MP_{r+1,c} \vee MP_{r-1,c}) \wedge (MT_{r+1,c} \vee MT_{r-1,c})) \rightarrow MM_{r+1,c} \vee MM_{r-1,c}$$

Losing Conditions: If Theseus is out of turns, or Theseus has been eaten, or his list of possible moves is empty, then Theseus has lost the game.

$$N \vee E_{r,c} \vee \neg TP_{r,c} \rightarrow L$$

Winning Conditions: If Theseus is on the same square as the exit/goal square, and the Minotaur is not on the exit/goal square, Theseus has won the game.

$$T_{r,c} \wedge S_{r,c} \wedge \neg M_{r,c} \rightarrow W$$

Model Exploration

We began by trying to determine some reasonable constraints for our implementation of the Theseus and the Minotaur game. In traditional versions, Theseus is often given unlimited turns and the game only ends when he reaches the exit square or the Minotaur catches him. As we determined that unlimited moves would result in a complexity too high for the scope of this project, we had to either fix the number of moves at an arbitrary limit, or set the number of

moves randomly at the beginning of each game. Since the hedges (mini-walls between squares of the maze) are set randomly and we wanted to explore – rather qualitatively – how varying hedge configurations and move limits would affect outcomes of a particular maze configuration, we elected to randomize the number of moves. For this same reason, we also decided to randomize the starting positions of Theseus and the Minotaur, as well as the position of the exit square. This represented a departure from traditional implementations of the game, which have placed the exit square on a border square, but this afforded more possibilities when exploring our model. In addition, while we initially considered using a 7x7 maze, we eventually opted to use a 6x6 maze after determining that it would provide sufficient complexity. As a matter of fact, the creator of the game mentioned that certain configurations of the maze could take up to 92 moves to solve![1]

Most other constraints (i.e., rules of the game) we inherited from the game itself.

The first phase of our code implementation process consisted mainly of trying to determine how best to represent a specific maze configuration, as well as how to implement constraints on moves for Theseus and the Minotaur. (The latter topic will be discussed in more detail shortly due to its complexity, especially as it relates to Theseus.) Initially, we had decided to attempt to represent the maze configuration as a series of arrays, each one containing Boolean data specific to one characteristic of the maze for each square. In other words, we had separate arrays for vertical hedges, horizontal hedges, Theseus' position, the Minotaur's position, and the exit square. However, we soon overhauled this implementation following receipt of peer feedback, which pointed out that our implementation could be much less code-intensive and easier to understand if we applied object-oriented programming (OOP). As such, we developed an OOP-based approach to representing the maze configuration; instead of using many arrays, we were able to pack data on all necessary variables into a single two-dimensional array of objects each representing a specific maze square. Then, on each subsequent "turn," we simply had to iterate over one array instead of many, saving us time and potential bugs. This is the representation present in our final implementation.

With respect to implementing constraints on Theseus' and the Minotaur's moves, constraints on the Minotaur's moves were relatively straightforward to implement, as he only has one possible move, which is determined by his algorithm, on each move. Determining Theseus' move was more complicated. We originally planned to look for the move that brought him closest to the exit square and was "safe" (i.e., out of the Minotaur's range) for at least two moves. However, this did not agree well with our objective, which was to determine whether a given maze was solvable – for instance, consider the case where Theseus may have to move away from the exit square in order to trap the Minotaur behind a wall and buy time to reach the exit without being caught (demonstrative example below). Therefore, we adapted our approach to determining a *list*

[1] See *Further Notes on the Theseus and the Minotaur Mazes*, by Robert Abbott: http://www.logicmazes.com/notes.html

of moves that would be safe for Theseus for one turn, then examining each in turn to determine if any of them permitted Theseus to escape in the allotted turns. An iterative approach was initially considered. However, since the examination of possible moves results in unsuitable moves being struck from the list and thus makes the problem size smaller on each iteration, we eventually settled on using recursion to determine whether Theseus could escape a given maze. This recursive function (is_winnable() in our code) was the source of the majority of our bugs due to erroneous return statements, missing base cases leading to stack overflow, and other logical errors. However, once completed, this implementation represented a much more complete and generalizable representation than one only exploring moves that brought Theseus directly closer to the exit, or a hard-coded implementation.

This mixture of object-oriented and procedural programming produced a working program that could determine whether Theseus could escape a given maze. However, two problems remained: the first, that our implementation failed to use the logic libraries bauhaus and nnf; and the second, that the program was liable to run extremely slowly (or not at all) for move limits over 10. The first problem stemmed mainly from our planning of this project using informal, colloquial logic (i.e., the typical rambling pseudocode one might see from a programmer in a hurry to get things done), rather than a formalized understanding of the language of natural deduction which underlay our propositions and constraints. Once we formalized our propositions and constraints, we were able to implement our model exploration using the logic libraries and a SAT solver. In fact, upon comparison with our original working code (which we used for a sanity check, as the mazes can be difficult for us to solve too!), the SAT-solver-based code ran considerably faster and we were able to increase the maximum number of moves being tested to 15.

Due to the complexity of the original Theseus and the Minotaur game, our final model seeks only to obtain information on the escapability of the maze; that is, whether Theseus can win. (Possible extensions of this model could very well consider the likelihood of Theseus winning, a possible pathway for Theseus' win, or even determining the most efficient course of action for Theseus to win.) At the present time, information derived from our model centres around the concept of the model, T, being satisfiable. In our implementation, no safe squares being found for Theseus means that a constraint is added such that Theseus' current position, the Minotaur's current position, and the configuration of the exit square and hedges in the maze imply that Theseus' position cannot be his current square, whereas a square leading to escape for Theseus means that a constraint is added such that the current configuration of the maze (Theseus, Minotaur, exit, and hedge positions) implies that Theseus must be on that square for his new position. In short, then, our model is satisfiable if and only if Theseus can escape. This means that an output of "Satisfiable: True" means that Theseus has a path to escape without being caught by the Minotaur. Conversely, an output of "Satisfiable: False" indicates that Theseus has no safe path to the exit of the maze. This is easily understood by realizing that when we add a constraint declaring that the current configuration of the maze means that Theseus cannot be on his current square, as no safe squares have been found, the model immediately contains an inherent contradiction, and thus

no solution can be found. A few examples follow below, including one that we found interesting due to its hedge configuration preventing the Minotaur from making any effective moves (see example 3 for details).

Example Theory 1: Theseus Wins
In this example theory, Theseus is positioned right next to the exit with the Minotaur several squares away. As he is allotted three moves for the purposes of demonstration, he should be able to escape as long as there are no hedges barring him from accessing the exit square within three moves. (When running example_theory_1, if there turn out to be hedges preventing Theseus from escaping, run the program again. The hedges are randomized each time.) When we run the code, we find the following output:

```
Starting board for example theory 1:
['*', '*', '*', '*', '*', 'M']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', 'T', 'E', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*']

Vertical hedges:
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'T', 'T', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']

Horizontal hedges:
['F', 'F', 'T', 'F', 'F', 'F']
['T', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'T', 'F', 'F', 'F']
['F', 'F', 'T', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'T', 'F', 'F', 'F']

Number of rounds:  3
Theseus escapes!
Satisfiable: True
```

As expected, we find that our model is satisfiable and therefore there exists a safe pathway from Theseus' starting square to the exit. (As mentioned before, future explorations of this model in greater depth could consider pathfinding applications to determine, for example, the most efficient pathway for Theseus.)

Example Theory 2: Theseus Loses
In our second example theory, Theseus is positioned farther away from the exit than the Minotaur. Combined with the Minotaur's algorithm always moving him closer to Theseus, the Minotaur taking two turns for each of Theseus' one, and Theseus being allotted only eight moves, means that Theseus almost universally loses in random runs of this theory. (He can possibly be saved by fortunately-placed hedges; however, this is also unlikely, given the small leeway between his allotted number of moves and the shortest distance from his starting square to the exit, which is 6.) The output we generate from this model exploration follows:

```
Starting board for example theory 2:
['*', '*', 'T', '*', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', 'E']
['*', '*', '*', '*', '*', '*']
['*', '*', 'M', '*', '*', '*']

Vertical hedges:
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'F']

Horizontal hedges:
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'T']
['F', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'T']
['T', 'F', 'F', 'F', 'F', 'T']
['F', 'T', 'F', 'F', 'F', 'F']

Number of rounds:  8
Theseus got eaten by the Minotaur :(
Satisfiable: False
```

Here we find that our model is apparently unsatisfiable, which is what we expect for a board configuration that (almost – depending on the hedges) certainly will cause Theseus to be caught by the Minotaur before he can make his way to an exit.

Example Theory 3: An Intriguingly Satisfiable Model
This third example theory was actually generated by a random run of the draft of Example Theory 2, which initially randomized the number of moves Theseus was allotted in addition to randomizing the hedges; the result was a satisfiable model under starting conditions that *usually* force an unsatisfiable one:

```
Starting board:
['*', '*', 'T', '*', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', 'E']
['*', '*', '*', '*', '*', '*']
['*', '*', 'M', '*', '*', '*']

Vertical hedges:
['F', 'F', 'T', 'F', 'F', 'F']
['T', 'T', 'T', 'F', 'F', 'F']
['T', 'F', 'F', 'F', 'T', 'T']
['T', 'T', 'F', 'T', 'T', 'T']
['T', 'F', 'F', 'F', 'F', 'T']
['F', 'T', 'T', 'T', 'F', 'F']

Horizontal hedges:
['T', 'F', 'F', 'F', 'F', 'F']
['F', 'F', 'F', 'F', 'F', 'T']
['F', 'F', 'F', 'F', 'F', 'F']
['T', 'T', 'F', 'T', 'T', 'T']
['T', 'T', 'F', 'T', 'F', 'F']
['F', 'F', 'T', 'F', 'T', 'F']

Number of rounds:  14
Theseus escapes!
Satisfiable: True
```

Observe that here, the Minotaur starts on a square with a hedge on top, so he cannot move up, and a hedge to the right, so he cannot move right. Also observe that Theseus must move down and to the right to achieve his goal of accessing the exit square. So he must first move closer to the Minotaur if he wants to

escape. Ordinarily, this would be a death sentence, as the Minotaur moves twice for every move Theseus makes. However, in this specific configuration of the maze, the Minotaur is blocked from moving upwards or to the right to catch Theseus by the hedges above and to the right of him. As his algorithm does not permit him to first move *away* from Theseus in order to escape the confines of the hedges around him, he has no valid moves and can only skip his turn on each turn. (In this case, the algorithm turns out to not even matter, since he also has a hedge to his left, thus completely preventing him from moving; but if we were to remove the hedge to his left, he would not be able to catch Theseus anyway even though he moves faster – his algorithm will never allow him to make a move that takes him farther from Theseus, even if it would benefit him later.) This example serves to show that the outcome (i.e., satisfiability) of a given maze configuration is not only governed by the positions of Theseus, the Minotaur, the exit square, and the number of moves but also by the position of the hedges! If Theseus can manage to receive or create a configuration that effectively traps the Minotaur behind a hedge, then he can manage to win even if the Minotaur is closer to the exit square or faster than him. As such, with so many moving parts built into the laws of the game itself, potential models and explorations of this seemingly-simple board game can quickly become very complex.

First-Order Extension

To extend the model to a predicate logic setting, our team would first assign all elements of the logic puzzle to a propositional variable (ie. the game board, Theseus, the Minotaur, etc. would all be given predicate variables), then the game's progression would be modeled through constraints (written as formulas using the variables) including universal quantifiers (the quantifiers would simplify the game as "for all" would act as a catch-all for all possible moves, and "there exists" would check all moves to check if a favorable case exists). Each proposition would then be formulated as a verifiable formula to implement each possible case of the game.
Propositions:

A square in the grid/maze is given by x.

TH(x) means that the square has a top hedge.

BH(x) means that the square has a bottom hedge.

RH(x) means that the square has a right hedge.

LH(x) means that the square has a left hedge.

T(x) means that Theseus is currently on this square.

M(x) means that the Minotaur is currently on this square.

S(x) means that this square is the safe spot in the maze.

E(x) means that Theseus and the Minotaur are both on this square and that Theseus has been eaten.

Constraints:

Theseus and the Minotaur are on the same square. This means that this square has the property that Theseus is eaten.

$$\forall x.(T(x) \wedge M(x) \rightarrow E(x))$$

There is only one safe spot in the maze.

$$\exists x.S(x)$$

If Theseus is on the safe spot in the maze, then he wins the game.

$$\exists x.(T(x) \wedge S(x)) \rightarrow W$$

Jape Proofs

When implementing our Jape proofs we had to change the variables used for our propositions since Jape does not allow us to use every letter in the alphabet. We have defined the propositions for our proofs with new variables that can be used in Japе.

P = top hedge ($TH_{r,c}$ proposition)
Q = bottom hedge ($BH_{r,c}$ proposition)
R = right hedge ($RH_{r,c}$ proposition)
S = left hedge ($LH_{r,c}$ proposition)
A = possibly moving up ($TP_{r-1,c}$ or $MP_{r-1,c}$proposition)
B = possibly moving right ($TP_{r,c+1}$ or $MP_{r,c+1}$ proposition)
C = possibly moving down ($TP_{r+1,c}$ or $MP_{r+1,c}$ proposition)
D = possibly moving left ($TP_{r,c-1}$ or $MP_{r,c-1}$ proposition)
T = Theseus' current position ($T_{r,c}$ proposition)
E = the Minotaur's current position ($M_{r,c}$ proposition)
F = a spot Theseus chooses to move to which allows him to get closer to the safe spot ($TM_{r,c}$ proposition)
G = the possible move gets the Minotaur closer to Theseus ($MT_{r,c}$ proposition)

Proof 1: There is a top hedge and a left hedge on the square that Theseus is currently on. Theseus can't move across a hedge, so we can deduce that Theseus can't move up or to the left.

$$T, P, S, (T \wedge P) \rightarrow \neg A, (T \wedge S) \rightarrow \neg D \vdash \neg A \wedge \neg D$$

1: **T, P, S, T∧P→¬A, T∧S→¬D** premises
2: **T∧P** ∧ intro 1.1,1.2
3: **¬A** → elim 1.4,2
4: **T∧S** ∧ intro 1.1,1.3
5: **¬D** → elim 1.5,4
6: **¬A∧¬D** ∧ intro 3,5

Figure 1: Our Jape proof corresponding to Proof 1.

Proof 2: Theseus is in the square to the right of the safe spot. There are no hedges around Theseus and he can move up, down, left, or right but he should only move right to get him to the safe spot. We can deduce that Theseus should not move up, should move right, should not move down, and should not move left.

$$T, F, A \vee B \vee C \vee D, A \to \neg F, B \to F, C \to \neg F, D \to \neg F \vdash \neg A \wedge B \wedge \neg C \wedge \neg D$$

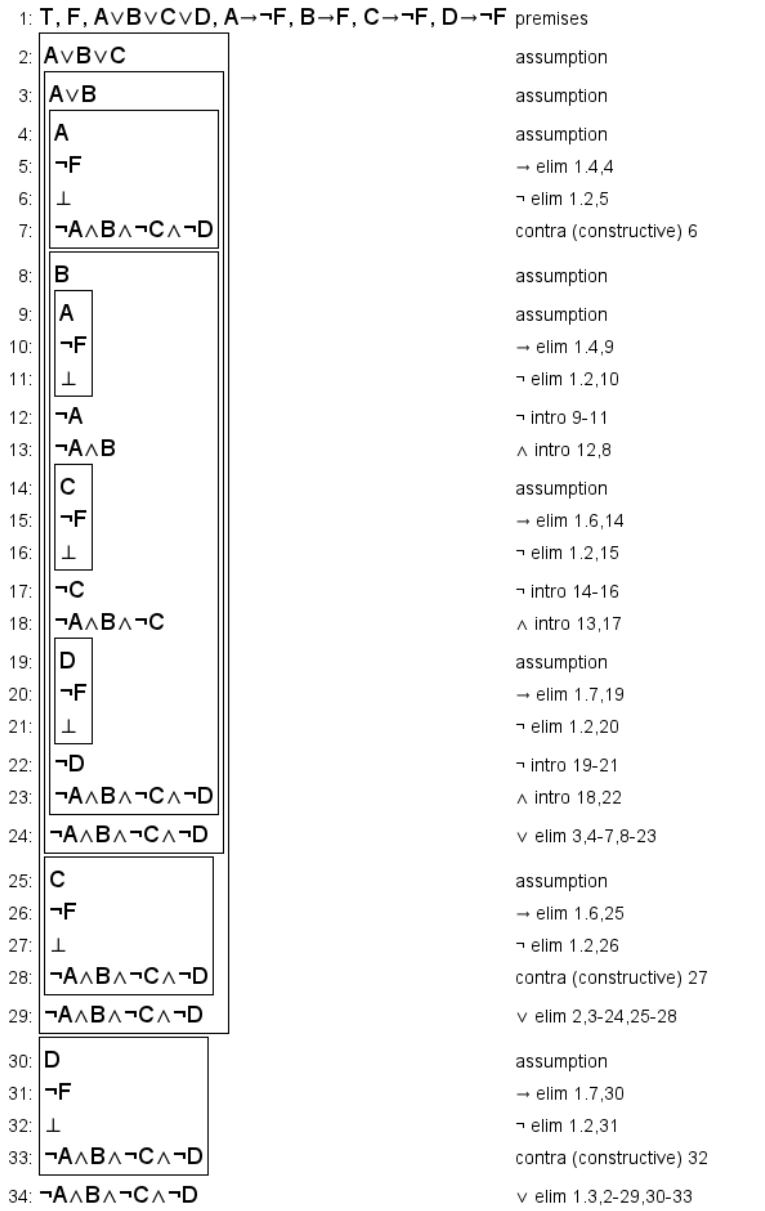

Figure 2: Our Jape proof corresponding to Proof 2.

Proof 3: The Minotaur is closer to Theseus moving horizontally, but there is a hedge to the left and to the right of the Minotaur. The Minotaur will have to move up to get closer to Theseus. We can deduce that the Minotaur can move up, can't move right, can't move down, and can't move left.

$$E, S, R, G, A \vee B \vee C \vee D, E \wedge S \rightarrow \neg D, E \wedge R \rightarrow \neg B, A \rightarrow G, C \rightarrow \neg G \vdash A \wedge \neg B \wedge \neg C \wedge \neg D$$

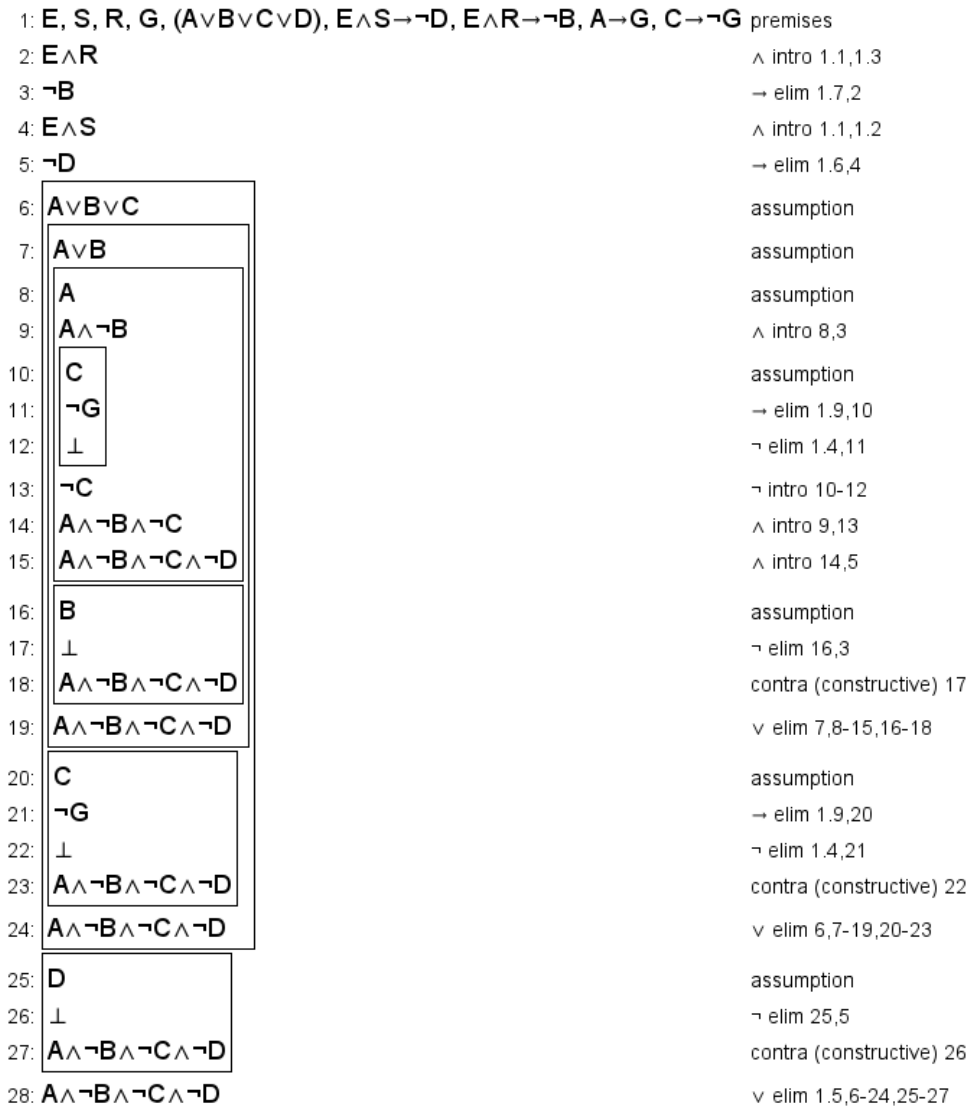

Figure 3: Our Jape proof corresponding to Proof 3.