Monsters and Munchkins

Anna Yang, Harris Bubalo,
Jamie Lin, Michael Huang

Abstract - Logic puzzles have garnered the attention of people of all backgrounds and experience levels, due to their thought-provoking nature and limited set of truths and a given question. Among them include the classic river crossing logic puzzle "Missionaries and Cannibals" which has become well-known in artificial intelligence. This article focuses on another variation of the game, "Monsters and Munchkins", and we demonstrate the satisfiability of our own algorithm, implemented and modeled in ACL2s. In addition, we broaden our solver by exposing possible instances of the game that fail to terminate and showing how our general algorithm catches those unsatisfiable cases.

Index terms **- satisfiability, termination, generalization**

I. Introduction

The relationship between logic and games is prevalent in reasoning, argumentation, education, and many other fields of study. From Aristotle's writings to textbooks, games and game theories have shifted from being seen as frivolous to purposeful. Games have been increasingly popular in formal settings. For example, they have been used professionally to give semantics to infinitary languages and provide discovery techniques that can be used in logical papers, as well as as used as a tool for analysing structures of debates [1]. Specifically, a game's unique ability to model relationships and structures of complex systems while allowing players to manipulate these systems encourage deeper understanding and thinking of aspects of the world. This type of participation has great potential to spark ideas of new technologies and ideas. Not only can games be academic, the challenge is also fun- which in Koster's theory of fun, is essential in pushing a player to continuously develop their understanding [2].

Puzzles are a one-person game genre that are a popular way of testing a person's ingenuity and knowledge. Solvers are expected to put pieces together in a logical way, following specific rules to arrive at the correct solution of a puzzle. Oftentimes, a sequence of moves or steps from the initial state of the game to a fixed goal are expected to be deduced by the player. These problems are solved based on reachability meaning there are only two outcomes - winning or losing. There are usually many possible scenarios or even methods of solving the problem, but reachability or termination is the winning condition. In this paper, we will be focusing on perfect-information games which is where the player has knowledge of all rules and parts of the game.

A river crossing puzzle is a type of "transport" perfect-information puzzle game where the objective is to carry items from one side of the river to the other. Usually, this involves certain restrictions such as how many items the boat can hold and requirements for which items can be left on either side of the bank at the same time. A more advanced requirement would be to complete this objective in the fewest trips. We will be solving this problem: monsters and munchkins which is based on these problems.

Specifically, the first part of our paper will be working with the traditional monsters and munchkins problem. The original problem involves 3 monsters and 3 munchkins on the left side of a river bank, and the goal is to get them all to the right

side. To accomplish this, a boat with a capacity of 2 is available for transporting the beings. A “move” in the game is whenever the boat crosses the river. With this, the only restrictions are that at no time in the game can there be more monsters than munchkins on either side of the river or in the boat (unless there are no munchkins on that side or in the boat), and the boat must be carrying at least one being, either monster or munchkin, in order to cross to the other side.

This version of the game can be generalized by changing the amount of starting monsters and munchkins. However, with 4 or more pairings, the problem becomes unsolvable if the boat capacity remains at 2. Increasing the capacity to 3 enables solutions for up to 5 pairings. Games with a boat capacity of 4 or more are solvable for any number of pairs. For our purposes then, we’d like to prove in ACL2s that, for any boat capacity ≥ 4, the game is solvable for any number of monster and munchkin pairings. We want to show that if there is a solution, we can compute it and return the specific steps, and if there is no solution, our algorithm will relay that.

To show this, we will be using a variety of methods together. First, we can break this up into several easier sub-problems depending on the maximum size of the boat. Then, we will be looking to find patterns of similar game states. Additionally, we will be checking situations that do not have a solution. Building off of the traditional version of monsters and munchkins, we will also be generalizing the problem. This involves having a starting state with an arbitrary number of monsters and munchkins on both the left and right bank.

A. Contributions

Our contribution is as follows. We will present an algorithm that solves the problem after giving it starting conditions. Through a combination of trial and error and deductive reasoning, we will draw out patterns that helped us develop our algorithm. Using this algorithm, we will then use logic to prove that our algorithm terminates.

B. Road-Map

This paper is organized as follows. Section II elaborates on the tools we will use and other background on our topic. Section III will explain more in depth about the Monsters and Munchkin game and other similar problems. Section IV will be about writing out our traditional algorithm. Section V will be on generalizing our traditional algorithm and broadening the scope of our problem. Section VI focuses on our findings and other related experimental results which includes our pseudo-code and process. Section VII is our conclusion with summarizing reflections and further thoughts.

II. Background

A. Model and Notation

In our notation for the algorithm, we will be using ACL2s. Our model includes a few defdatas that we have created. First, we will represent monsters as a natural number which is now many monsters there are. Next, we will represent munchkins as a natural number which is how many munchkins there are. The maximum capacity of the boat is also a natural number. Finally, the side that the boat is on is either ‘left or ‘right.

After creating these defdatas, we considered how to represent all variables on one side of the river at a given time. We then have the defdata count which will be a list of monsters and munchkins which is a list of

natural natural. All information relevant to the boat can also be represented with a defdata which includes the capacity and side it's on.

A function we created to convey a step to be taken in our algorithm is "move". This prints out in list form how many monsters and munchkins to move to which side of the river. We will be using all these tools as well as ACL2s contracts to represent our algorithm.

B. The Game Structure

We will be modeling a single player game. To represent our game, we will have 5 starting variables - the number of monsters on the left side of the river, the number of munchkins on the left side, the number of monsters on the right side, the number of munchkins on the right side, and the maximum boat capacity. Each state of our game will keep track of all of these variables where given a starting state, we will print out a list of instructions of each following game state that leads to the correct solution.

C. Satisfiability

Propositional satisfiability problem is known as SAT and is the problem of developing if a set of sentences in propositional logic is satisfiable. This is relevant to our problem since our game can be reduced to that of propositional satisfiability [3].

To begin, a formula is valid if it is true for all values of its terms. Satisfiability means that there is an existence of a combination of terms to make the expression true. A proposition is satisfiable if there is at least one true situation. In our problem, satisfiability refers to there are certain conditions that allow our algorithm to give a set of instructions on how to solve the monsters and munchkins problem.

To prove satisfiability, if our algorithm can output a list of steps, it means that there is a solution to the problem. Based on specific conditions that we have deduced, our algorithm proves that it is unsatisfiable.

III. Monster and Munchkins game

The Monster and Munchkins game is another variation of the classic river crossing logic puzzle, "Missionaries and Cannibals", where the goal of the game is to move all of the monsters and munchkins on one side of the river to the other. The earliest river-crossing puzzle is found in the manuscript Propositiones ad Acuendos Juvenes, or "problems to sharpen the young". This manuscript contains 3 river-crossing problems which are the fox, goose, and bag of beans puzzle, the jealous husbands problem, and the bridge and torch problem. The jealous husbands problem is the problem that missionaries and cannibals—or monsters and munchkins, the name we prefer—is modeled after. The monsters and munchkins must travel on a boat of a given capacity, and there cannot be more monsters than munchkins existing on any river bank or traveling in the boat at any given time.

There are many other similar games to the monsters and munchkins variation where players are given different variables to manipulate such as a wolf, lamb, and cabbage and the jealous husbands problem. Each of these have their own little twist that makes them unique. For example, a wolf eats a lamb that eats a cabbage and so, a wolf cannot be left with a lamb and a lamb cannot be left with a cabbage. In the jealous husband's problem, a wife cannot be left with men without her husband being present on her bank. Beyond these, there are even more variations of the game that add their own level of complexity to solve.

Related solutions to similar river crossing logic puzzles include the *Graphical Solution of Difficult Crossing Problems* completed by students from Harvey Mudd and Pomona College, where the algorithm for the variation "Missionaries and Cannibals" was done through graphical representation of plotted points and drawn diagrams, and the reasoning behind the invariant of a boat capacity greater than or equal to four was proved through those models [4].

The boundaries to our work include… **[to-do]**

Our metrics to success will be determined by termination. This means that if there is a satisfiable solution, the steps to that solution should be returned. If there is no solution, the algorithm should terminate and relay that.

IV. Methodology of Traditional Algorithm

[to-do once algorithm is finished]

V. Generalization of Traditional Algorithm

To generalize our traditional algorithm that only works for monsters and munchkins starting on the left side of the river and a boat capacity of 4.

A boat capacity of 4 is chosen for the simple reason that lesser capacities do not fully solve the problem. A capacity of 1 does not work, as no net change could possibly occur (1 monster or munchkin is required to drive the boat). For a capacity of 2, 4 or more pairs of monsters and munchkins does not work (though it does work for up to 3 pairs) [5]. For capacity of 3, up to 5 pairs can cross [5]. A boat capacity of 4 is chosen because for all number pairs of monsters and munchkins, a complete crossing is possible.

Other cases we have considered include when there are no restrictions on the amount of monsters and munchkins in the starting position. There are a couple of situations that arise from this generalization. First, when there are more monsters than munchkins, this situation is rejected because if there are more monsters than munchkins, we can automatically terminate the game. This is because it is impossible to follow the game restriction that at no point in time can there be more monsters than munchkins on either side of the river bank. The next situation with an arbitrary number of monsters and munchkins is where there are more munchkins than monsters. This case is trivial because we can treat this problem as if there are an equal number of monsters and munchkins and follow the algorithm as so. After following those steps, all remaining munchkins on the left river bank can then be brought to the right river bank, making the problem solvable.

Additionally, if we increase the boat capacity to be greater than 4, we are sure that there is a possible solution. This is because if it is possible for a boat capacity of 4, even if we increase the boat capacity, we can treat the problem as if the boat capacity is 4 and follow the algorithm we have created. This would produce a working solution that satisfies all conditions.

VI. Experimental Results and Metrics

The purpose of this project was to figure out the generalized algorithm to prove the satisfiability of the classic river crossing logic puzzle "Monsters and Munchkins" and prove its termination. By encoding our algorithm in ACL2s and

proving its termination in a proof checker, we have successfully completed our project.

VII. Conclusion

[to-do once finished]

VIII. References

[1] B. B. Marklund, P. Backlund and H. Engstrom, "The Practicalities of Educational Games: Challenges of Taking Games into Formal Educational Settings," 2014 6th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), Valletta, Malta, 2014, pp. 1-8, doi: 10.1109/VS-Games.2014.7012170.

[2] Hodges, Wilfrid and Jouko Väänänen, "Logic and Games", *The Stanford Encyclopedia of Philosophy* (Fall 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/fall 2019/entries/logic-games/>.

[3] Introduction to Logic: Propositional Satisfiability. (n.d.). Retrieved April 13, 2021, from http://intrologic.stanford.edu /extras/satisfiability.html

[5] Franci, Raffaella (2002). "Jealous Husbands Crossing the River: A Problem from Alcuin to Tartaglia". In Dold-Samplonius, Yvonne; Dauben, Joseph W.; Folkerts, Menso; van Dalen, Benno (eds.). From China to Paris: 2000 Years Transmission of Mathematical Ideas. Stuttgart: Franz Steiner Verla. pp. 289–306. ISBN 3-515-08223-9.

Progress Report

1. Scope of Setup and Work Thus Far

Our project revolves on finding a universal algorithm that is satisfiable for the classic river crossing logic puzzle "Monsters and Munchkins" and demonstrating its termination in ACL2s. Our group will follow the following timeline (see Figure 1) to complete our project.

Figure 1 Timeline

Completed?	To-Do	Finish By
No	Algorithm in ACL2s	Friday 4/9
Yes	Algo in English/ Pseudo-code	Friday 4/9
Yes	Make Github Repository	Monday 4/12
No	Proof done	Tues 4/13
No	Project Outline Everything written	Tues 4/13
No	DEADLINE of DRAFT	Wed 4/14

Our group has written out the pseudo-code for an algorithm of narrower scope (the traditional variation of the problem) to transliterate into ACL2s, but we have not yet completed the final algorithm in ACL2s due to contract issues. We have shown that our algorithm does work in Racket, so it's a matter of fixing the nuances in ACL2s. We also plan on finishing the proof (using the Proof Checker) to show termination of our algorithm after the ACL2s portion is completed. Furthermore, we have made a general outline with the basic information in each area so that we will be ready to fill out the rest of the methodology once our encoded representation runs in ACL2s.

2. Current Concerns and Issues

The current implementation of our algorithm in ACL2s does not terminate, and we are still in the process of figuring out why this is, or what contracts we may need to add or remove to accomplish termination. In our repository, we have a Racket file that contains identical functions to the ones in our ACL2s file (though without contracts), and they work as we expect them to, assuming that we give them "legal" inputs. Because of this, we are unsure if the issue with our algorithm has something to do with unaccounted edge cases or contracts, or if a deeper ACL2s-specific problem is accountable. Simply put, we do not know why our function does not terminate - stemming from a lack of understanding of termination in ACL2s. Going forward, we'd like to address this problem as early as possible with TAs and/or the professor, as it would be difficult to complete the rest of our paper without a working algorithm.

3. To-Do

We have yet to finish writing our coded implementation in ACL2s due to contract issues and failure to terminate, and we will only then be able to prove termination in the proof checker once that is completed. We plan on going to office hours to complete our proof in ACL2s, and then we will be able to finish the final project outline. Additional progress on the project can really only be done after the initial ACL2s pass. When/if it passes, one step we would need to take is extrapolating the things we have learned from the Traditional algorithm (which is of more limited scope) and making more judgements about the general problem, which includes arbitrary starting states. This might include more statements about which states are unsolvable, or might result in an infinite

loop. A more general variation of the problem would have many more edge cases that would make it more difficult or impossible to solve (consider a case where 5 munchkins start on the left and 4 monsters start on the right). Another step would be to explain why our ACL2s code actually works, either through a formal proof or an in-writing outline. Another possibility for additional work is to come up with alternative algorithms that may also work, although it is not in the scope of our project to prove if our algorithm is optimal (requiring the fewest steps).

4. First Draft: See above.
5. Code Status: In Progress

Have not been able to successfully run the code in ACL2 (will go to office hours).

6. Link to Repository

https://github.com/mihalehang/monsters-munchkins