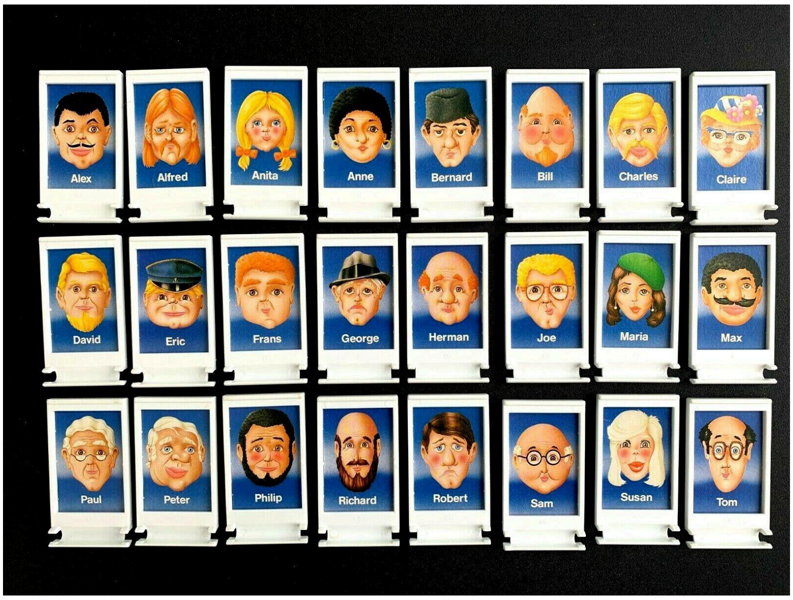
Guess Who?

Project Summary

“Guess Who?” is a two-player board game in which each player chooses a character from a grid of options hidden from the other and they race to guess the identity of the other’s chosen character through a series of questions. Each character has numerous depicted physical attributes, ranging from hair colour to different accessories, shared in common with other characters to focus the game on narrowing the unique combination of traits in a character rather than a single distinct trait.



In our version, a guess must ask a compound question with two traits, with negations allowed. The guesses will be premade and your goal is to make the best guess possible, this will be in a multiple choice format with 1 guess being the best possible option, while the others will be worse options. The goal is to get at least half of the characters down per guess. An example of a question could be, “does your character have glasses and not have brown hair?” The other player must then respond truthfully with either yes or no, to which the guesser eliminates the characters that can be ruled out by said answer. After a series of back-and-forth eliminations, a player will then determine who the opponent’s character is by guessing, or by process of elimination, to win the game.

The image attached is the board we are basing our model off of.

Propositions

is_up(c): checks if the character has been eliminated. is_up(c) is true if the character's card is still able to be guessed.

check_trait(t1, t2): a guess involving two specific traits. check_trait(t1, t2) is true if at least one of t1 or t2 makes is_up(c) false guess_character(c): guesses the passed character and first ensures it is not already eliminated. guess_character(c) is true if is_up(c) is true and if you have guessed your opponent's character correctly.

Constraints

¬(check_trait(c1,t)∧check_trait(c2,t)) - For any two distinct characters c1 and c2, and all traits t, no character can have all the same traits.

(check_trait(c1,t)∧¬check_trait(c2,t)) - For two distinct characters c1 and c2, each question posed by a player for a given trait t must distinguish at least one character c1 from the other c2. Only for characters that are up.

¬(check_trait(c,t)∧check_trait(c,¬t)) - For any character c and trait t, if a character has a trait then they cannot simultaneously have its negation.

((is_up(c1) ∧ ¬is_up(c2)) ∨ (guess_character(c1)) → win?) - For final character c1, and the rest of the characters c2 the goal is to guess the correct character or narrow down the board down to the last character, meaning every other character has been eliminated.

Model Exploration

Here are the traits we have picked out from the board.

- hair .
- hair colour
- facial hair
- gender .
- accessories (including glasses, hat, earrings)
- age
- smiling

Here are the questions involving two traits that we have come up with. This in theory should be able to cover every character on the board.

Male and has glasses
Female and has glasses
Blonde hair and facial hair
Black hair and no hair
Smiling and has accessories
White hair and no facial hair
Male and not smiling
Brown hair and facial hair
Male and has no hair

Orange hair and no facial hair
Female and smiling
Black hair and facial hair

Jape Proof Ideas

Jape Proof 1:

Jape Proof 2:

Jape Proof 3:

Requested Feedback

Are there any general improvements we can make to our project?

Should we include age as a trait? How would that work with middle aged characters?

What are some ideas we can use to make Jape proofs

First-Order Extension

Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.
There is no need to implement this extension!