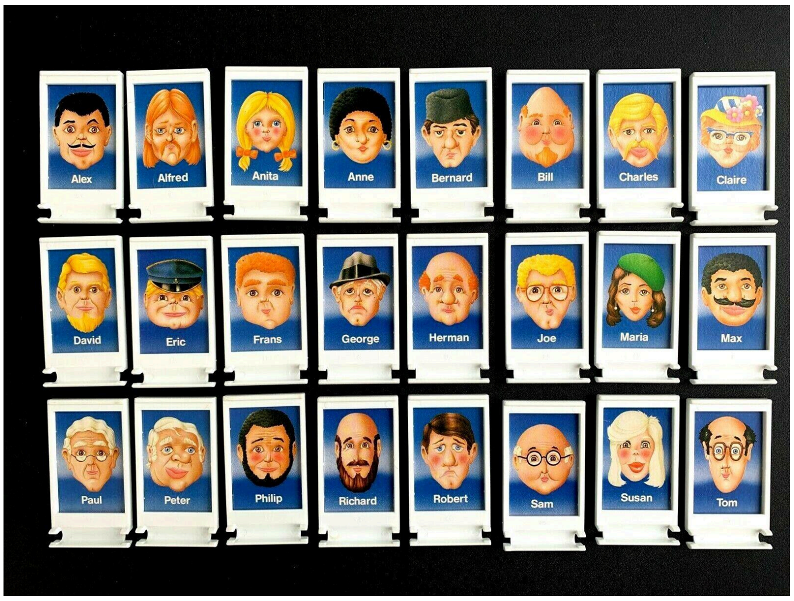
CISC/CMPE 204: Modelling Project

Group 26
Jeffrey Y, Nicholas H, Brian C

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 the goal is to make the best guess possible. An example of a question could be, "does your character have glasses and not have brown hair?" The model uses an inputted character and then uses logic to deduce the best possible set of guesses using the premade guesses.

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

Propositions

question_proposition(): The question_proposition proposition evaluates whether a specific question meets the criteria for selection. If it returns True, that question is deemed suitable and will be included in the final list of asked questions

Constraints

$\sum Q_q \leq k$ - The total number of questions selected Q_q must not exceed the predefined limit k.

$\bigwedge_{c!=user}(Q_{c,1} \vee Q_{c,2} \vee \cdots \vee Q_{c,nc})$ - For every character c that is not the user's chosen character, the system must select at least one question that can distinguish c from the user's character.

$\neg(Q_q \wedge \neg Q_q) \forall q$ - A question q cannot be both selected and not selected simultaneously.

Model Exploration

Model Explanation

In our model of Guess Who we are trying to find out what the guesses are for the least amount of guesses using a set list of questions. The user inputs a character and the model outputs the order of guesses it used. Each question involves guessing two traits at the same time, for example “has hair” & “male”.

Here are the traits that we have curated from the image located at the top of our document:

- has hair
- hair colour (Black hair, Orange hair, Blonde hair, Brown hair, White hair)
- has facial hair
- gender
- accessories, including hat, earrings, bowtie, not including glasses
- is smiling
- has glasses

From this list of traits, we were able to create a dictionary to assign each character a list of 7 traits, you can find this in characters.py.

Here is the list of questions, which is specifically designed to be able to capture every character on the board:

```
    ['male', 'glasses'],
    ['female', 'glasses'],
    ['blonde_hair', 'facial_hair'],
    ['black_hair', 'no_hair'],
    ['smiling', 'accessories'],
    ['white_hair', 'no_facial_hair'],
    ['male', 'not_smiling'],
    ['brown_hair', 'facial_hair'],
    ['male', 'no_hair'],
    ['orange_hair', 'no_facial_hair'],
    ['female', 'smiling'],
    ['black_hair', 'facial_hair'],
    ['orange_hair', 'facial_hair'],
    ['brown_hair', 'accessories'],
    ['brown_hair', 'no_accessories'],
    ['not_smiling', 'accessories'],
    ['female', 'blonde_hair'],
    ['female', 'brown_hair'],
```

When our program runs, it will ask you to choose a character. After you input the character you want, it will spit out the shortest list of questions to ask to identify the character.

Model Difficulties

Originally, our project was done in a completely different way than what we currently have. We realised that the model was not sufficient and was overcomplicated so we scrapped the entire thing and re-made it.

Our first model had three propositions instead of one:
is_up(c): returns true if the character has not been eliminated, or in game terms has its tile still in the upright position.
check_trait(c, t): takes in a character c and a trait t, if character c has trait t, then it will return true
guess_character(c): if character c is the character that the player is trying to find, then it returns true. This was supposed to be equivalent to a “win”.

The problem with formulating our model this way was that our original intent was to have a player playing against an A.I. We were confused if this was the way the project was intended to be done, but after the recommendation of our TA and much deliberation, we decided to redo everything.

Jape Proof Ideas

Jape Proof 1: $\vdash\neg(Q_q\land\neg Q_q)$
No question Q can be simultaneously chosen and not chosen.

Jape Proof 2: $((T_A\lor T_B)\rightarrow(Q_{11}\lor Q_5)),\ T_A\rightarrow Q_{11},\ T_A\vdash Q_{11}\lor Q_5$
If the character’s traits differ in a way that Q_{11} could detect (T_A) or Q_5 could detect (T_B), then at least one of these questions must be asked ($Q_{11} \lor Q_5$). If T_A is true, and we know T_A implies Q_{11}, we can show that we must pick Q_{11} or Q_5.
T_A = Alfred’s traits differ from Anita in a way that Q_{11} can detect
T_B = Alfred’s traits differ from Anita in a way that Q_5 can detect

Jape Proof 3: $(D_A\rightarrow Q_{11}),\ (D_B\rightarrow Q_{12}),\ D_A,\ D_B\vdash Q_{11}\lor Q_{12}$

If two different characters, Alfred and Bernard, both differ from the user's chosen character (Anita), and we know which questions would distinguish each of them individually, then we must at least choose one of those questions.
D_A / D_N = Alfred / Bernard differs from Anita

First-Order Extension

Extending our model to predicate logic we can have predicates represent relationships and properties of objects. This gives us flexibility for more complex scenarios such as partial information or nuanced relationships between traits and characters.

Propositions

Character Traits

Instead of having binary propositions for questions, we can define predicates that will associate traits with characters.
Examples:

- HasTrait(c,t): Character c has trait t
    - HasTrait(Sam, Glasses) is true if Sam has glasses
- HasAllTraits(c, (t1, t2, t3, …)): Character c has all traits in the set
    - HasAllTrait(Alex, (BlackHair, FacialHair, Smiling)) is true if Alex has Black Hair, Facial Hair and is Smiling

Question Representation

Using predicates to represent the structure of a question
Example:

- Question(q, (t1, t2)): Question q checks for traits t1 and t2
    - Question(Q1, (Glasses, BlondeHair) represents the question, "Does your character have glasses and blonde hair?"

Constraints

Character Differentiation

We can add a constraint that ensures for any two distinct character c1 and c2, at least one question can differentiate them:
$\forall c1, c2\ (c1 \neq c2 \rightarrow \exists q\ Eliminates(q, c1, c2))$

- If Alex and Alfred have different hair color, there must be a question about color that can eliminate one of them

<u>Question Limit</u>

Make sure the number of selected questions does not exceed k.
$\sum(\text{Selected}(q)) \leq k$

<u>Trait Validation</u>

Ensure questions are based on valid traits.
$\forall q, c, t\ (\text{Question}(q, \{t1, t2\}) \land \text{HasTrait}(c, t1) \rightarrow \text{Eliminates}(q, c, \_))$