Our project models the well-known word guessing game Hangman. The goal of this game is to guess the word that is being thought of by the administrator as the players. In our case, the administrator will be the human using the software, and our program will be the player. We hope to build a system that models that game using belief revision in prolog that allows the player to attempt guessing the word.

In this first paragraph, I'd like to define a bit of vocabulary to make it easier to understand what it is we are referring to. Our Knowledge Base is the list of words the program thinks the possible answer will be this knowledge base will shrink as the game goes on. Next is our “dictionary” which is the list of known words that the administrator gets to pick from. This is required to allow the program to know the word it is guessing.

We are going to represent the beliefs using a dictionary and a knowledge base that will be altered by guesses. This knowledge base will first consist of all the words in the dictionary. This dictionary will be part of the microworld we construct. Along with that, the microworld is also going to consist of a set of rules dictating the game, which will mostly be per the normal rules of hangman (cite 1). There will be additional rules implemented.

The basic scheme of the program will work as follows: first, there will be a word given to the program. Then the program will check if that word is within the “dictionary” if it is it will confirm. Second, the program will start playing the game. It will start by asking how many letters the word is. Third, it will start guessing letters by selecting randomly from a list of the most common letters. After a few guesses of this list, it will start guessing from a larger list of less likely letters and so on until it either guesses all the letters or runs out of guesses. Fourth, the program will guess one letter at a time and the player will tell the computer yes or no depending on whether the letter is in the word or not. Fifth, the program will ask you in what position the letter is in, in the word. Followed by the program asking the user if there are any other spots the letter is in, in the word, if yes then it will again ask you where. Rinse and repeat until you reply no when asked if there are any more spots.

The Program will revise its beliefs under two conditions and for each in several different ways. As said before it will trim down its potential guess first when given the length of the word. Secondly, it will revise its beliefs by eliminating words that don't contain correctly guessed letters. This is very simple yet will allow us to represent belief revision.

The techniques used to do this will be as described before a knowledge base and state that will be updated. Similarly, to what we did for the block world. Where we will have the program recursively go down and modify a state until the resulting word is found and then cry that word back out to the top and let the user know the word. This will allow us to replay the game if desired as well as allow us to break out of the recursive loop when the game is done without having to deal with breaking the loop and going back out of all the reductions.

There are a few improvements of variation we could do to our program but due to the limited time we have, we will begin going with a very simple approach. In this paragraph, we will discuss the different things that could be done. First, there can be an implementation of a confidence meter of sorts that depending on the word and the letters in the word the computer guesses that word. This thought would only be useful if there is more than one letter left to guess. Another idea that would be just brilliant would

be if the administrator would pick words outside of the program's dictionary and when the program ultimately fails to guess it asks the administrator the word and adds that word to its dictionary.

In conclusion there, the modeling of hangman using a revised belief system in prolog is very interesting with several different aspects. There are several more things that could be implemented. Hopefully, these ideas can be realized in future projects.