CONOCIMIENTO Y RAZONAMIENTO AUTOMATIZADO, 3º CURSO, GII

QUIÉN ES QUIÉN

PECL1 - CURSO 2020/21

Realizado por:

Laura Mambrilla Moreno

Víctor Gamonal Sánchez

Álvaro Golbano Durán

Índice

1. Reparto de tareas

Nuestra forma de trabajar ha consistido en reunirnos diariamente para avanzar poco a poco y de forma conjunta, de tal forma que los tres miembros del grupo hemos ido adquiriendo los mismos conocimientos y hemos contribuido de igual forma en el desarrollo de la práctica. Aún así, cada uno ha destacado más en algo específico:

- **Álvaro Golbano:** ha aportado más conocimiento y soltura sobre la sintaxis del lenguaje.
- **Laura Mambrilla:** ha trabajado más en la idea de incluir ASCII Art y ha contribuido en mayor parte al algoritmo de la máquina en modo difícil.
- **Víctor Gamonal:** refactorización del código y lógica de los predicados

2. Cumplimiento de los requisitos

En el enunciado de la práctica se establecieron 8 objetivos que debíamos poder cumplir para la correcta realización de la misma. A continuación, procedemos a mencionarlos y a explicar si los hemos conseguido o no:

1. ***La solución propuesta debe basarse en el uso de listas para construir la Base de Conocimiento del programa. Éstas tienen que relacionar de algún modo cada uno de los personajes con su conjunto de características propias.***

Conseguido. Hemos establecido una lista de personajes que incluye en su interior, como elementos, listas con la información de cada personaje (nombre y atributos); esto hace que se convierta en una lista de listas como se puede ver en el archivo adjunto *Datos.pl*.

2. **Para construir la Base de Conocimiento se pide aumentar en 3 el conjunto de características de los personajes quedando su tipo a elección del grupo de alumnos.**

Conseguido. Hemos añadido hasta 7 nuevas características con sus respectivos valores:

- **Lunares** (lunares, no_lunares)
- **Barba** (barba, no_barba)
- **Arrugas** (arrugas, no_arrugas)
- **Gorro** (gorro, no_gorro)
- **Pendientes** (pendientes, no_pendientes)
- **Maquillaje** (maquillaje, no_maquillaje)
- **Pelo** con 2 nuevos valores (pelo_rojo, calvo)

3. **Hay que definir un predicado de aridad cero (jugar) que incorpore la preparación del juego y que, además, llame al predicado encargado de gestionar el desarrollo del juego.**

Conseguido. Hemos implementado un predicado jugar() que incorpora, **mediante llamadas a otros predicados**, funcionalidades como asignar personajes a cada jugador, imprimir por pantalla el menú principal del juego y lo más importante que es el desarrollo de la partida (siempre realizado mediante turnos, tanto para el modo Jugador vs Jugador como para el modo Jugador vs Máquina) con las respectivas funciones de cada uno de ellos.

4. **Se contempla la posibilidad de que compita jugador contra ordenador y jugador contra jugador.**

Conseguido. Tenemos un modo de juego Jugador vs Jugador y otro Jugador vs Máquina en el que existen dos niveles de dificultad: fácil y difícil.

5. **Las preguntas que se pueden hacer serán las siguientes: ¿chico?, ¿chica?, ¿gafas?, ¿pelo rubio?, ¿pelo negro?, ¿ojos azules?, ¿ojos marrones?, ¿feliz?, ¿triste?, ¿ropa roja? y ¿ropa verde? a las que habrá que añadir las referentes a las características añadidas por cada grupo de alumnos.**

Conseguido. Como comentamos anteriormente en el punto 2, añadimos hasta 7 nuevas características con 8 nuevas preguntas: barba? ; lunares? ; arrugas? ; gorro? ; pendientes? ; maquillaje? ; pelo_rojo? ; calvo?

6. **Para mejorar la jugabilidad, el programa deberá mostrar por pantalla el subconjunto de candidatos seleccionado en cada turno de juego.**

Conseguido. Hemos seguido la misma estructura que la proporcionada en el documento de ejemplo de partida.

7. **En el caso de la opción de jugador contra ordenador, hay que implementar al menos dos estrategias:**

    a.) La computadora almacena la lista de preguntas a hacer (evitando duplicidades) y escoge una al azar de entre las no realizadas hasta el momento.

Conseguido. Para el modo fácil, la máquina accede a la lista con todas las preguntas que puede formular, cogiendo una de ellas al azar y posteriormente eliminándola para evitar duplicidades.

b.) La computadora aplica un algoritmo que le permite ganar la partida con el mínimo número de preguntas posible.

Conseguido. Lo que hacemos para ello es buscar cuál es la característica que más se repite en todos los personajes del tablero y será esa la que pregunte en cada turno (siendo eliminada posteriormente para evitar duplicidades), lo que aumenta bastante las probabilidades de ganar de la máquina.

8. **Estas estrategias se implementarán en ficheros separados (a consultar según proceda).**

Conseguido. Cada modo de juego (JvJ, JvM fácil y JvM difícil) se encuentra separado en archivos distintos, haciendo todos ellos un consult de los ficheros:

- *Datos.pl* - que incluye todos los datos de los personajes, la lista de personajes y la lista de preguntas que se pueden realizar.
- *Caras.pl* - que incluye todos los ASCII Art que se muestran por pantalla cuando uno de los jugadores gana la partida.

También cabe destacar que hemos incluido **mejoras** como son la adición de personajes llegando hasta un total de 30 y la adición de ASCII Art incluyendo todo tipo de personajes para cuando el jugador gana una partida; por ejemplo, si el personaje que ha adivinado es una mujer que tiene arrugas, saldrá una mujer con arrugas.

3. Errores y/o aspectos no implementados

Todo implementado.

4. Fuentes consultadas

Manual de Prolog: **https://www.swi-prolog.org/pldoc/doc_for?object=manual**

Ampliación de prolog y ejemplos: **https://www.tutorialspoint.com/prolog/index.htm**

Editor online para comprobar funiones: **https://swish.swi-prolog.org**

Para el diseño del ASCII art: **https://www.asciiart.eu/people**

Para encontrar el mayor elemento de una lista: **https://xpasos.wordpress.com/2011/11/03/prolog-busca-numero-mayor-en-lista/**