Ingeniería del Conocimiento

Práctica 1:
SBC con CLIPS

Doble Grado de Ingeniería en Informática
y Administración de Empresas

Curso 2020/2021

Ramses Maurice Contreras Sulbarán [100386209]

Carlos Castrejón Sánchez [100386198]

Índice

1. Introducción

En este pdf se encuentra la información referente a la práctica realizada por Ramsés Contreras Sulbarán y Carlos Castrejón Sánchez sobre la primera práctica grupal de Ingeniería del Conocimiento.
Esta práctica consiste en programar el funcionamiento de un robot terapéutico para que juegue a dos juegos a elegir con niños con distintas personalidades. El robot debe de saber adaptarse a cómo reaccionan los niños con las distintas personalidades y actuar en consecuencia. El robot debe de saber las reglas de los juegos para así poder jugar con los niños.
Los juegos elegidos para llevar a cabo esta práctica fueron el Tres en Raya y el Juego de Memoria.
Las personalidades que hemos elegido para los niños son Neutra, Impaciente y Distraído.
Los temas que se van a tratar a continuación son:

- Un manual técnico donde vamos a comentar la implementación que decidimos hacer para completar esta práctica y cuál ha sido nuestro proceso de razonamiento para nuestras distintas elecciones de diseño.
- Un manual de usuario para explicar cómo funciona el programa y cómo cualquier usuario puede ejecutarlo.
- Las distintas pruebas que hemos ido realizando para verificar que nuestro programa funcionase correctamente y qué conclusiones hemos extraído de estas pruebas.
- Distintos comentarios personales que tenemos sobre la práctica como que nos ha parecido, cuánto tiempo nos ha tomado, si cambiaríamos algo, etc.

Nuestro programa para el trabajo lo hemos hecho completamente interactivo, por lo cual no es necesario que haya una base de hechos iniciales para comenzar su ejecución, ya que el programa recoge esta información mediante inputs del usuario.
Aún así para las pruebas de ejecución hemos puesto como hechos iniciales tanto la personalidad del niño como la elección del niño, por lo que al ejecutar el programa, se saltará la primera regla (la regla *pickGamePersonality*).

2. Manual técnico

2.1. Ontología

Nuestra ontología se compone de dos clases:

2.1.2. Clase CONTROL

Tiene distintos slots que nos almacenan toda la información relativa a la sesión de terapia y juego, a excepción de la disposición y valor de las casillas. En esta clase tenemos la información referente al juego elegido por el niño (Eleccion), su personalidad (Personalidad), de quien es el turno (Turno), la ronda en la que nos encontramos (Ronda), el tiempo que le toma al niño jugar su turno (Cronometro) y un atributo especial que sirve para controlar si ya se le ha dado una segunda oportunidad de elegir una carta en el juego de la memoria al niño, si se ha equivocado eligiendo dos cartas distintas, con distintos animales.

Turno a turno, y ronda tras ronda, los atributos de esta clase se van actualizando según las decisiones tomadas por los jugadores, hasta finalizar la partida, sirviendo como parámetros de las restricciones que se aplican en las reglas de nuestro Sistema Basado en el Conocimiento.

2.1.2. Clase CASILLA

Uno de los motivos por los que decidimos elegir tanto el juego Tres en Raya como el Juego de Memoria se debe a que podíamos generalizar bastante su funcionalidad gracias a esta misma clase. Los atributos de esta clase tienen distinto significado según el juego:

- Activada: Indica, en el caso del Tres en Raya, si algún jugador ha marcado la casilla. Por otro lado, en el Juego de Memoria, indica si esa carta ya fue descubierta, tras haber sido encontrada su pareja por algún jugador.
- Valor: Guarda el valor de la casilla en Valor. En el juego de Tres en Raya, dicho valor indica qué jugador marcó esa casilla según su símbolo (X para el robot, O para el niño), mientras que, en el Juego de Memoria, que figura (animal) tiene esa carta.
- x / y: Coordenadas x e y, en las que se encuentra la casilla/carta en el tablero de juego.

2.2. Reglas

Una vez explicada la ontología, planteamos las reglas que sirven para controlar el flujo de ejecución de nuestro programa, en base a las entradas del usuario. Decidimos que el usuario metiese todos los datos iniciales y decisiones de cada turno, mediante input al

programa para que así este mismo fuese más interactivo en lugar de hacerlo mediante una base de hechos inicial.

2.2.1. Reglas Generales de Configuración Inicial de la Sesión

- ➢ pickGameNPersonality: Esta regla se ejecuta cuando no existen aún entradas de elección de usuario, y tiene como consecuente la solicitud al usuario de dichas entradas, que se almacenan en la base de hechos.
- ➢ readGameNPersonality: En base a las entradas obtenidas con la regla anterior se crea una instancia de la clase CONTROL, con el input indicado, comprobando primero que es correcto, de lo contrario se ejecuta la regla `incorrectChoices` para solucionarlo. Asimismo, se añaden a la base de hechos una serie de hechos de control:
  - warningBeforeDone, que sirve para indicar al SBC que ya se ha hecho o no el aviso al niño antes del turno del robot en la ronda correspondiente (en las reglas warningBeforeTurn_Impacient y warningBeforeTurn_Impacient) que se inicializa en False.
  - kidPlayed: Inidica si el niño ha jugado ya en la ronda correspondiente.
  - robotPlayed: Indica si el robot ha jugado ya en la ronda correspondiente.
- ➢ incorrectChoices: Se ejecuta si el input del usuario en pickGameNPersonality es incorrecto, es decir, el juego no es 3R ni JM (códigos correspondientes a 3 en Raya y Juego de Memoria) y la personalidad de el usuario no es Neutro, Impaciente ni Distraido. Tiene como consecuente, indicar el error al usuario y pedir una nueva entrada correcta.

2.2.2. Reglas Generales de Control de Flujo durante el Juego

- ➢ changeRound: Esta regla se ejecuta cuando tanto el niño como el robot han participado ya ambos en el juego y justo ha acabado el turno del niño. Esto lo sabemos ya que en la base de hechos podemos encontrar que tanto kidPlayed como robotPlayed son True. Cuando esta regla se ejecuta, en la base de hechos volvemos a establecer kidPlayed y robotPlayed a False, así como warningBeforeDone a False. Por último imprimimos por pantalla el cambio de ronda.
- ➢ changeTurn: Se ejecuta una vez se tiene conocimiento en la BH de que el robot ya ha jugado en su turno, y simplemente modifica la instancia CONTROL para cambiar el turno al niño, indicando al mismo que el robot ya ha terminado su turno. También añadimos a la base de hechos lo siguiente:
  - timeWarningDone: Este hecho sirve para indicar si el robot ha regañado previamente al niño debido a que ha tardado mucho tiempo o ha tardado poco tiempo en elegir su movimiento en función de la personalidad del niño.

2.2.3. Reglas Generales de Interacción Robot-Paciente

- overTimeDistractedKid: Esta regla se ejecuta cuando un niño distraído ha tardado más de 15 segundos en elegir su acción. Antes de ejecutarse comprobamos que aún no se ha realizado ningún warning previo comprobando que el valor del hecho timeWarningDone sea False. Una vez hecho esto, el robot le advierte al niño que se concentre y cambie el valor del hecho timeWarningDone a True.
- tooFastImpacientKid: Se ejecuta cuando se trata a un niño impaciente, que tarda 3 segundos o menos en tomar una acción, comprobando que aún no se ha realizado el aviso correspondiente en esta ronda. Tiene como consecuencia, un aviso del robot al niño de que ha elegido muy rápido y le recomienda tomarse su tiempo para pensar. Asimismo, se indica en la BH que se realizó el aviso.
- warningBeforeTurn_Impacient: Se ejecuta cuando se trata a un niño impaciente, y aún no se ha realizado el aviso que se lanza antes de que juegue el robot (warningBeforeDone = False). Tiene como consecuencia, el aviso y la indicación en la BH de que ya se realizó el aviso en la ronda correspondiente.
- warningBeforeTurn_Distracted: Se ejecuta cuando se trata a un niño distraído, y aún no se ha realizado el aviso que se lanza antes de que juegue el robot (warningBeforeDone = False). Tiene como consecuencia, el aviso y la indicación en la BH de que ya se realizó el aviso en la ronda correspondiente.

2.2.4. Reglas de Configuración y Victoria - 3 en Raya

- initBoard3R: Esta regla se ejecuta cuando nos encontramos en la ronda 0, antes de que ningún jugador actúe aún y simplemente establecemos las 9 casillas para jugar al Tres en Raya, con todas sus coordenadas.
- CondicionVictoria_3R: Comprueba tras marcar cada jugador en su turno, si ha logrado marcar tres casillas en raya, indica en la BH el jugador y elimina la instancia de CONTROL para que no se continúe con la ejecución del juego.

2.2.5. Reglas de Estrategia del Robot - 3 en Raya

- JuegaRobot_3R_Ronda1: Esta regla se ejecuta cuando nos encontramos en la primera ronda del juego. La estrategia básica que pensamos para el robot es que en su primer movimiento siempre decida marcar la casilla (1, 1), ya que al elegir una esquina luego tendrá cierto margen de maniobra. Comprobamos mediante un test que la personalidad sea Neutra o que el robot le haya ayudado al niño en función de su personalidad mediante un comentario de cara a esta ronda. Si esta comprobación falla, el robot no actúa aún. También se comprueba que el robot aún no haya jugado. Una vez que todo lo anterior se cumple, la casilla queda marcada y el hecho robotPlayed pasa a ser True. imprimimos por pantalla que el robot ha marcado la casilla (1, 1).
- JuegaRobot_3R_Ronda2: Esta regla se ejecuta en la 2da ronda de juego. Continuando con la estrategia del robot, este decide marcar la casilla (1,3) siempre y

cuando esta y la casilla del medio (2,2) estén disponibles. Una vez marcada la casilla, se indica modificando la instancia correspondiente y se actualiza la BH para indicar que el robot ya ha jugado en esta ronda.

- ➢ JuegaRobot_3R_RRondas: Esta regla se ejecuta cuando la ronda es mayor que 2 o cuando las casillas (2, 2) o la (1, 3) están ya marcadas. El resto de comprobaciones son iguales que para las anteriores reglas. Si todo lo anterior se cumple, el robot elige una casilla que marcar y el hecho robotPlayed pasa a ser True. También imprimimos por pantalla la casilla marcada por el robot. La elección de la casilla puede ser al azar, si se indica (set-strategy random), u ordenada, si no se indica.

2.2.6. Reglas de Input del Usuario - 3 en Raya

- ➢ inputKid_3R: Solicita x e y al usuario, y se hace lectura del sensor del robot que mide el tiempo que tarda en decidir.
- ➢ juegaKid_3R: Esta regla se ejecuta cuando es el turno del niño ((Turno Kid)). Gracias a la regla anterior, extraemos de la base de hechos los inputs previamente realizados por el niño. En esta regla realizamos un test para comprobar que o el niño es distraído y ha realizado la acción en menos de 15 segundos o que el niño es impaciente y ha realizado la acción en más de 3 segundos o que el warning por motivos de tiempo ya se ha realizado o bien que la personalidad del niño es neutra. Si alguna de estas cuatro situaciones anteriores se cumplen, entonces la regla puede ejecutarse. Cuando la regla se ejecuta, se marca la casilla elegida por el niño y el hecho kidPlayed pasa a ser True, además de imprimir por pantalla la casilla marcada por el niño.

2.2.7. Reglas para Interacción Robot/Paciente - 3 en Raya

- ➢ corregirCasillaOutOfBounds_3R: Corrige si la casilla que ha ingresado el usuario se encuentra fuera del rango (1 >= x,y <= 3). Como consecuencia, se eliminan de la BH los hechos que contienen la entrada del usuario y, como se volverá a solicitar una entrada, se indica que aún no se ha realizado el aviso timeWarning.
- ➢ corregirCasillaOcupada_3R: Corrige si la casilla que ha ingresado el usuario ha sido marcada previamente. Como consecuencia, se eliminan de la BH los hechos que contienen la entrada del usuario y, como se volverá a solicitar una entrada, se indica que aún no se ha realizado el aviso timeWarning.
- ➢ winner3RRobot_DistractedKid: Cierre de la sesión para un niño distraído que ha perdido en el 3 en raya.
- ➢ winner3RRobot_ImpacientKid: Cierre de la sesión para un niño impaciente que ha perdido en el 3 en raya.
- ➢ winner3RKid: Cierre de sesión para un niño que ha ganado el 3 en raya.

2.2.8. Reglas de Configuración y Victoria - Juego de Memoria

- initBoardJM: Esta regla se ejecuta cuando nos encontramos en la ronda 0, antes de que ningún jugador actúe aún y simplemente establecemos las 24 cartas (casillas) para jugar al Juego de Memoria, con todas sus coordenadas y valores de las cartas (animales).
- CondicionVictoria_JM: Comprueba tras voltear dos cartas cada jugador en su turno, si han logrado encontrar las 12 parejas, y finaliza la sesión.

2.2.9. Reglas de Estrategia del Robot - Juego de Memoria

- JuegaRobot_JM_Rondas: El robot escoge dos cartas distintas correctas (tienen el mismo animal) una vez se ha hecho el aviso de warningBefore. La elección de las cartas puede ser al azar, si se indica (set-strategy random), u ordenada, si no se indica. Se modifican las cartas correspondientes, indicando que ya fueron descubiertas sus parejas (Activada = True), y se indica en la BH que el robot ya ha jugado su turno en esta ronda.

2.2.6. Reglas de Input del Usuario - Juego de Memoria

- inputKid_JM: Solicitan dos cartas al usuario (x e y), y se hace lectura del sensor del robot que mide el tiempo que tarda en decidir.
- juegaKid_JM: Se comprubea que la entrada del usuario sea correcta, es decir, las cartas son distintas, dentro del rango y no han sido volteadas aún, y además que tienen el mismo animal. También, se comprueba que ya se realizó el aviso por exceso o falta de tiempo, según la personalidad del niño. Como consecuencia, se modifican las instancias y hechos correspondientes, al igual que en el caso de JuegaRobot, y se elimina el hecho timeWarningDone, pues se reseteará al cambiar de ronda.

2.2.7. Reglas para Interacción Robot/Paciente - Juego de Memoria

- corregirCasillaOutOfBounds_JM: Esta regla se ejecuta cuando el usuario ya ha introducido un input (se comprueba si los hechos xKid y yKid existen). También realizamos un test para comprobar que los inputs del usuario sean mayores que 24 o menores que 1, dado que si este es el caso entonces esta regla debe ejecutarse. Si lo anterior se cumple, el robot avisa al niño de que ha realizado una acción incorrecta.
- corregirCasillaOcupada_JM: Esta regla se ejecuta cuando el usuario ya ha introducido input, al igual que en la anterior regla. También realizamos un test que comprueba que la primera carta que ha elegido el usuario ya haya sido levantada o que la segunda carta elegida por el usuario ya haya sido levantada. Si esto es así, el robot advierte al usuario de esto.
- corregirCasillaIgual_JM_Distraido: Esta regla se ejecuta cuando la personalidad del niño es distraído. También es necesario que el niño ya haya introducido previamente

un input. Mediante un test, comprobamos si el niño ha intentado levantar la misma carta dos veces debido a que no recuerda las reglas del juego. Si esto es así el robot le recuerda las reglas del juego para que no vuelva a cometer el mismo error.

- ➢ corregirCasillaIgual_JM_Impaciente: Esta regla se ejecuta cuando la personalidad del niño es impaciente. Al igual que en la regla anterior, es necesario que el usuario ya haya introducido previamente un input. Además realizamos un test que compruebe si el niño ha intentado levantar dos veces la misma carta ya que ha ido con prisas al ser un niño impaciente. Si esto es así, el robot le aconseja que en su próximo turno tome la decisión con más calma y se fije mejor.
- ➢ secondChance_JM_Distraido: Se le da una segunda oportunidad a un niño distraído que ha elegido dos cartas distintas que no se han activado aún, pero que no tienen el mismo animal. Por tanto, elimina los hechos de entrada de las dos cartas del usuario, y se indica que se ya dió la segunda oportunidad modificando el atributo correspondiente en la instancia de la clase CONTROL.
- ➢ cardsNotEqual_JM_Distraido: Una vez se ha dado la segunda oportunidad a un niño distraído, y este vuelve a equivocarse (con una entrada legal), no se vuelve a dar otra oportunidad, tan sólo se indica que el niño ya ha jugado (para pasar a la siguiente ronda), y se eliminan los hechos que almacenan la entrada el usuario para que se soliciten de nuevo en la siguiente ronda.
- ➢ cardsNotEqual_JM_Impaciente: Esta regla se ejecuta cuando el usuario ya ha introducido un input y el niño impaciente no consigue encontrar una pareja. Si esto ocurre, el robot le reconforta y le aconseja que en la próxima ronda intente dedicarle más tiempo a su decisión para tener más éxito en la próxima ronda.

3. Manual de usuario

Este programa consiste en un robot que interactúa con el usuario mediante la información de los inputs extraídos del usuario. Las interacciones del robot consisten en jugar a dos juegos que el usuario puede elegir (Tres en Raya o Juego de Memoria) y adaptarse a cómo actúe el usuario. Para este problema se interpreta al usuario como si fuese un niño.

Para empezar, una vez ejecutes el programa, le pedirá al usuario/terapeuta/padre que introduzca el juego al que quiere jugar además de la personalidad del niño.

A continuación el robot procederá a establecer todos los preparativos necesarios para llevar a cabo el juego y procederá a actuar él primero. Una vez el robot haya actuado, le pedirá al usuario los inputs para saber que dos cartas quiere levantar (en el Juego de Memoria) o que casilla quiere marcar (en el Tres en Raya), a la vez que se procede a la lectura del sensor exterior del robot que mide el tiempo (se almacena el valor como un input del usuario).

Las rondas se irán sucediendo una detrás de otra hasta que el juego se complete, y se realizarán una serie de avisos/recomendaciones al paciente según su personalidad, el tiempo que tome en realizar una acción y si resulta victorioso o no (este último sólo para el juego de 3 en raya).

En el caso del Tres en Raya hemos decidido que el juego sea competitivo entre el niño y robot, es decir, que o gana uno de los dos o el juego acaba en empate. Por otro lado, en el Juego de Memoria el objetivo del juego es que tanto niño como robot cooperen para conseguir levantar las 12 parejas correctamente en el menor número de rondas posible. Si ambos aciertan todo el rato, el número de rondas mínima es 6.

Una vez el juego se haya completado, se declara al ganador y el número de rondas que ha tomado acabar al juego (en el caso del Tres en Raya) o se declara el número de rondas que se han tomado para finalizar el juego y se da la enhorabuena al niño por participar en el caso del Juego de Memoria.

4. Pruebas realizadas

Respecto a las pruebas, como ya mencionamos en la introducción, el programa lo habíamos planteado para que fuese completamente interactivo, sin necesidad de que hubiese una base de hechos iniciales, por lo que a la hora de realizar los archivos de prueba simplemente añadimos en los hechos iniciales la elección del juego por parte del usuario y la personalidad del niño elegida por el usuario. Esto provoca que la primera regla no se llegue a ejecutar.

Para el resto de pruebas de inputs, hemos puesto mediante comentarios que debe de escribir el profesor para conseguir los resultados que queremos conseguir con cada prueba.

PRUEBA 1

Para la primera prueba elegimos que la elección del juego del usuario sea el Tres en Raya y que la personalidad del niño sea Distraído.
En esta prueba el niño introduce en las rondas:

1. x = 1, y = 3, tiempo = 20
2. x = 2, y = 2, tiempo = 5

En este caso, todo el programa se ejecuta sin complicaciones. Lo único que merece la pena mencionar es que como el niño es distraído el robot le recuerda que después de que sea su turno, es el turno del niño.
El robot gana en este caso.

PRUEBA 2

Para la segunda prueba elegimos que la elección del usuario sea el Tres en Raya y que la personalidad del niño sea Impaciente.
En esta prueba el niño introduce en las rondas:

1. x = 4, y = 1, tiempo = 1
2. x = 1, y = 3, tiempo = 5
3. x = 3, y = 1, tiempo = 20
4. x = 2, y = 2, tiempo = 5

En este caso, el niño gana al robot. En el primer turno, el primer movimiento, el niño lo realiza de manera precipitada (en un 1 segundo), y por tanto el robot le reprocha por ello. Le aconseja que sea más paciente a la hora de tomar sus decisiones. Además el niño elige una casilla que no es válida, ya que no existe una casilla con esas coordenadas, por lo que el robot le advierte sobre esto y le dice que vuelva a introducir un nuevo input.
Como el niño es impaciente, el robot le recuerda que el niño debe esperar a que el robot realice su movimiento antes de que el niño actúe.
También podemos observar que como el niño no es distraído, da igual lo mucho que tarde en elegir, que el robot no le llamará la atención. De hecho, es positivo que un niño impaciente tarde mucho en elegir una opción.

PRUEBA 3

Para la tercera prueba, elegimos que la elección del usuario sea el Juego de Memoria y que la personalidad del niño sea Distraído.
En esta prueba el niño introduce en las rondas:

1. carta1 = 12, carta2 = 13, tiempo = 20
2. carta1 = 1, carta2 = 24, tiempo = 1
3. carta1 = 3, carta2 = 22, tiempo = 1
4. carta1 = 5, carta2 = 20, tiempo = 1
5. carta1 = 7, carta2 = 18, tiempo = 1
6. carta1 = 9, carta2 = 16, tiempo = 1
7. carta1 = 11, carta2 = 14, tiempo = 1

En este caso, podemos observar que el juego se completa sin problemas, excepto que al principio el niño eligió una carta a la que el robot ya le había dado la vuelta y por tanto el robot le llama la atención. El robot le pide que vuelva a seleccionar dos cartas de nuevo.

PRUEBA 4

Para la cuarta prueba, elegimos que la elección del usuario sea el Juego de Memoria y que la personalidad del niño sea Impaciente.
En esta prueba el niño introduce en las rondas:

1. carta1 = 1, carta2 = 24, tiempo = 1
2. carta1 = 1, carta2 = 24, tiempo = 5
3. carta1 = 4, carta2 = 5, tiempo = 5
4. carta1 = 5, carta2 = 4, tiempo = 5

5. carta1 = 4, carta2 = 21, tiempo = 5
6. carta1 = 6, carta2 = 19, tiempo = 5
7. carta1 = 8, carta2 = 17, tiempo = 5
8. carta1 = 10, carta2 = 15, tiempo = 5

En este caso, podemos observar que en el paso 1 el robot le regaña al niño por tardar muy poco tiempo, ya que es impaciente y le recomienda al niño que piense más detenidamente sus acciones.
En el turno 2, el robot le regaña al niño porque este intenta levantar cartas que ya han sido previamente levantadas por el robot..
El resto del programa se ejecuta de forma correcta.

PRUEBA 5

Para la quinta prueba, elegimos que la elección del usuario sea el Juego de Memoria y que la personalidad del niño sea Distraído.
En esta prueba el niño introduce en las rondas:

1. carta1 = 1, carta2 = 2, tiempo = 1
2. carta1 = 1, carta2 = 24, tiempo = 1
3. carta1 = 3, carta2 = 22, tiempo = 1
4. carta1 = 5, carta2 = 20, tiempo = 1
5. carta1 = 7, carta2 = 18, tiempo = 1
6. carta1 = 9, carta2 = 16, tiempo = 1
7. carta1 = 11, carta2 = 14, tiempo = 1

En este caso, en el primer turno, el niño elige 2 cartas que son elegibles, pero lamentablemente no forman una pareja. Debido a que el niño es distraído, decide darle una segunda oportunidad para que haga memoria e intente elegir otro par de cartas. Siempre que el niño levanta dos cartas que puede levantarse pero falla al formar la pareja, el robot le da una segunda oportunidad para que recuerde las cartas que lleva levantadas hasta el momento. El resto del programa se ejecuta con normalidad. El robot no le reprocha por el tiempo ya que le toma poco tiempo tomar las decisiones, pero como la personalidad del niño es distraída, el robot decide no regañarle.

5. Conclusiones

Las conclusiones que sacamos de esta práctica es que CLIPS es un lenguaje más potente de lo que pensamos en un inicio. Al principio del curso, no estábamos muy contentos con cómo funcionaba CLIPS, ya que era un lenguaje de programación nuevo que teníamos que aprender y algo distinto a los lenguajes a los que estábamos acostumbrados.
Tras acabar esta práctica nos hemos dado cuenta de que CLIPS es un lenguaje muy fácil de depurar y bastante intuitivo a la hora de programar con él. Es cierto que por lo general

preferimos otros lenguajes de programación debido a que es más fácil conseguir programar ciertos programas y tambíen en menos líneas de código.
Como he comentado anteriormente, por lo general, nuestra perspectiva sobre este lenguaje de programación ha cambiado drásticamente y ahora creemos dominarlo a un nivel decente como para poder trabajar con él en proyectos más complejos y también dominarlo de cara al examen final.

6. Comentarios personales

Por lo general la prácticas nos ha parecido muy didáctica y el nivel es el adecuado para estas alturas del curso. Creemos que gracias a esta práctica ya controlamos el lenguaje de programación CLIPS a un nivel elevado de cara al examen final.

En Colmenarejo consideramos que se le ha dedicado el tiempo necesario para que los alumnos comprendiésemos la práctica y nuestro profesor le dedicó el tiempo necesario para resolver nuestras dudas y ayudarnos con los problemas que pudiésemos encontrar. Sin embargo, luego de plantear el problema inicialmente con la identificación del mismo y de los conceptos que lo componen, al comenzar a escribir código nos encontramos con diversos problemas con nuestro planteamiento original. Por ello, decidimos volver a reestructurar nuestro fichero ontologia.clp. Decidimos basarnos en la última práctica de la clase magistral para llevar a cabo esta práctica.

En resumen, estamos contentos con cómo se ha desarrollado toda la práctica y nuestra experiencia ha sido positiva y didáctica.