Práctica de Sistemas Basados en el Conocimiento

Sara Buceta

Pol Casacuberta

Alejandro Espinosa

Department of Computer Science, UPC, Barcelona, Spain.

Primavera 2022

Índice

1. Introducción

El objetivo de nuestra práctica es crear un sistema recomendador de viajes para la agencia de viajes *al fin del mundo y más allá* empleando un sistema basado en el conocimiento (SBC de ahora en adelante). Para ello emplearemos las técnicas y los conceptos estudiados en clase de teoría.

1.1. Descripción del problema

Como ya indicamos en la introducción, el objetivo es crear un SBC que recomiende viajes al usuario, para lo que debemos tener en cuenta las características de dicho usuario.

Las principales características de los usuarios que consideraremos son:

- El número de viajeros y su relación: si viajan solos, en pareja, en familia, en un grupo pequeño o en un grupo grande.
- La edad de los participantes: son factores relevantes si son todos adultos o si hay adolescentes o niños.
- El objetivo del viaje: Descanso, diversión, romántico, trabajo, aventura o cultural.

Emplearemos como restricciones el rango de presupuesto proporcionado por el usuario (de forma más estricta si prefieren perder calidad o duración para mantenerse en el presupuesto), el rango en el que se encuentra la duración deseada del viaje y las preferencias en cuanto a transporte y tipo y calidad de alojamiento.

Tendremos en cuenta también la proximidad entre ciudades, limitando los itinerarios a ciudades del mismo continente si se trata de un viaje de corta duración. Las ciudades se escogerán en función a su adecuación al tipo de viaje y de usuario, y crearemos conjuntos predefinidos de ciudades con características comunes (ciudades románticas, ciudades de gran interés cultural, destinos paradisiacos, ...) que se especificarán en el apartado de esta documentación dedicado a la implementación.

Para obtener toda la información que necesitamos haremos una serie de preguntas al usuario, y a partir de las respuestas generaremos 2 posibles itinerarios en los que incluiremos precio total, ciudades a visitar, días por ciudad y duración total, actividades a realizar en cada ciudad, transportes entre ciudades y alojamientos. En cada itinerario indicaremos qué preferencias del usuario se han tenido en cuenta a la hora de generarlo.

También es necesario tener en cuenta que es posible que nos pidan una recomendación imposible de crear con nuestras opciones disponibles (ya sea por presupuesto, duración, preferencias que sea necesario respetar, ...) por lo que debemos asegurarnos de que si no existe una recomendación que satisfaga las restricciones el programa acaba e informa al usuario.

2. Identificación

2.1. Viabilidad de la construcción del SBC

Un sistema recomendador de este tipo requiere manejar una gran cantidad de datos y posibles combinaciones y requiere emplear criterios cualitativos para escoger soluciones, por lo que es adecuado emplear un SBC, pero ... ¿es viable la construcción de dicho sistema?

Para construir nuestro SBC necesitaremos crear un gran número de instancias de posibles destinos, alojamientos y actividades, así como generar las combinaciones de transporte entre ciudades. Además, es necesario organizar los datos y analizar cómo podemos hacer una buena recomendación a partir de los datos del usuario y las restricciones que este nos indica. Es necesario formalizar los datos, por lo que el problema debe estar bien estructurado, y en nuestro caso podemos identificar los elementos clave y cómo están relacionados, lo que nos permitirá dar este conocimiento al sistema. Finalmente, si se tratase de un caso real sería necesario contar con expertos en el tema, pero en nuestro caso nos guiaremos principalmente por conocimiento general y sentido común, así como por las recomendaciones dadas en el enunciado.

2.2. Fuentes de conocimiento

Nuestras fuentes de conocimiento serán páginas de reserva de hoteles, lo que nos permitirá generar instancias razonables en cuanto a precio, calidad y tipo del alojamiento; blogs de viajes en los que se muestran distintos destinos y actividades a realizar en ellos y compañías de transporte para estimar adecuadamente los precios de los viajes entre ciudades. Toda esta información, combinada con nuestro conocimiento y algo de sentido común también nos permite elaborar las restricciones y preferencias basadas en conocimiento de experto.

Si estuviésemos creando este sistema para una agencia de viajes la información provendría de las bases de datos de dicha compañía y de sus empleados, que nos proporcionarían el conocimiento de experto que necesitamos.

2.3. Objetivos del problema y resultados del sistema

El objetivo del problema es crear dos itinerarios lo más adecuados posible para el usuario, en caso de que sea posible. Esto implica proporcionar las ciudades a visitar, alojamiento para cada ciudad, actividades a realizar y transporte empleado para viajar entre ciudades, así como definir el número de días por ciudad y el número total de días de viaje y calcular el precio total de ambos itinerarios.

Para ello, nuestro sistema debe ser capaz de asociar las características del usuario con las distintas opciones para cada elemento del viaje. En algunos casos estas relaciones serán explícitas (la calidad mínima de alojamiento que el usuario quiere o transportes en los que no quiere viajar), pero en muchos casos se basarán en el conocimiento experto, que nos permitirá crear viajes apropiados para el usuario y no solo viajes que cumplan todas las restricciones indicadas en el enunciado de la práctica.

Las ciudades se seleccionarán por adecuación a la temática del viaje, mejorando la calidad de la recomendación. Se escogerán primero las ciudades, los transportes y los alojamientos, por lo que, en caso de que sea necesario para no pasarse del presupuesto, se sacrificará el número de actividades. Se seleccionan todas las actividades posibles sin superar la duración de la estancia en la ciudad y sin superar el presupuesto.

3. Conceptualización

Una vez establecidos nuestros objetivos, es necesario estudiar el dominio en el que estamos trabajando y plantear cómo será el proceso de resolución.

3.1. Conceptos del dominio

En esta etapa del desarrollo buscamos conocer qué elementos forman el dominio en el que trabajamos, para más adelante construir una ontología. Para ello hicimos *brainstorming* y organizamos en categorías y de forma jerárquica los términos a los que llegamos.

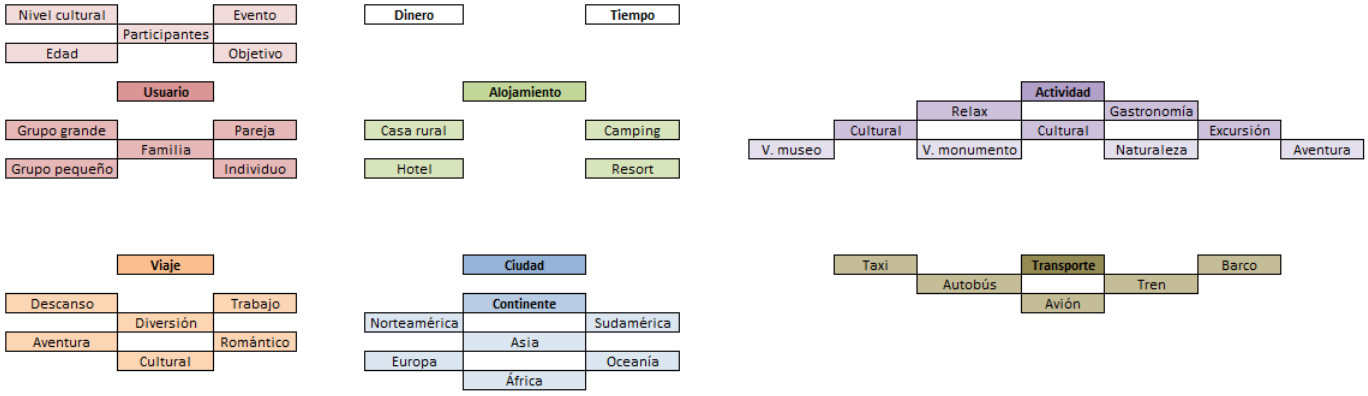

Figura 1: Primera representación gráfica de los conceptos

No todos estos conceptos pasarán a formar parte de nuestra ontología, será necesario seleccionar cuáles son los conceptos principales y qué conceptos serán clases y qué conceptos serán propiedades de otros. Los conceptos que consideramos que serán los más relevantes de cara a la construcción de la ontología son:

- Usuario: El viajero o conjunto de viajeros para el que hay que crear una recomendación de viaje.
- Viaje: Un concepto algo más abstracto del resto. Es un conjunto del resto de elementos.
- Ciudad: Posible destino a visitar.
- Alojamiento: Establecimiento, que puede ser de distintos tipos, en el que los usuarios pueden pasar la noche.
- Actividad: Clasificadas por temática, están relacionadas con la ciudad en la que se realizan.
- Transporte: Permite ir de una ciudad a otra.
- Dinero: Relacionado con los precios y el presupuesto.
- Tiempo: Relacionado con la duración de las actividades y la duración de las estancias y el viaje.

3.2. Problemas y subproblemas

Para elaborar un SBC necesitamos dividir nuestro objetivo en distintos problemas y subproblemas.

El primer problema es la obtención de los datos del usuario, lo que requiere que elaboremos un conjunto de preguntas que nos proporcionen toda la información necesaria pero evitando hacer preguntas innecesarias, ya que buscamos la mayor comodidad del usuario posible.

El segundo problema es la clasificación del usuario a través de la inferencia. Para ello se emplean los datos proporcionados por el usuario. En este problema nos encontramos diversos subproblemas relacionados con cada una de las características del usuario. Estas características son el tipo de usuario (individual, pareja, familia, grupo pequeño o grupo grande), si entre los viajeros hay niños o adolescentes, el tipo de viaje que desean realizar, ...

Una vez inferidas las características del usuario y sus gustos el siguiente problema es la elaboración del itinerario. Contiene diversos subproblemas:

- Asignación de ciudades: Basada en los resultados de la inferencia, se toman ciudades de alguna de las rutas temáticas predefinidas, teniendo en cuenta que tienen que ser del mismo continente si el viaje es corto. Se selecciona también un número de días para la visita dentro del rango impuesto por el usuario.
- Asignación de alojamientos: Para cada ciudad se escoge un alojamiento que respete las restricciones impuestas por el usuario en cuanto a calidad y de un tipo relacionado con la temática del viaje.
- Asignación de transportes: Es necesario seleccionar un medio de transporte para ir de una ciudad del itinerario a otra, respetando las preferencias del usuario.
- Asignación de actividades: Para cada ciudad del itinerario se seleccionará un conjunto de actividades realizables en el tiempo de la estancia en la ciudad. Se tendrá en cuenta el tipo de viaje para escoger las actividades más apropiadas.
- Control de la duración total: La suma de los días que se pasan en cada ciudad debe estar en el rango de la duración del viaje indicada por el usuario.
- Control del precio: Es necesario tener en cuenta los precios de los alojamientos, del transporte y de las actividades para no superar el presupuesto del usuario.

Estos problemas se resuelven para crear dos soluciones.
El último problema es extraer y comunicar los resultados. Para ello es necesario que hayamos guardado adecuadamente los itinerarios calculados y mostrarlos por pantalla en un formato adecuado y fácil de comprender para el usuario.

3.3. Conocimiento experto

El conocimiento experto debía permitirnos diferenciar entre una buena recomendación y una mala, y guiar a nuestro sistema en la dirección correcta. Este conocimiento experto, expresado como reglas explícitas, lo enfocamos principalmente a cómo consideramos que debería ser un buen itinerario, algunos ejemplos son:

- Los itinerarios para viajes de menos de 6 días no pueden pasar por ciudades demasiado alejadas (esto lo simplificamos definiendo que las ciudades deben pertenecer al mismo continente).
- Las ciudades se agrupan por temática y se priorizan las ciudades de temática similar al viaje.
- Un alojamiento es mejor contra más cerca del centro de su ciudad está.

3.4. Proceso de resolución y organización de problemas y subproblemas

Los problemas y subproblemas explicados tenían una estructura bastante clara, por lo que organizamos la resolución de problemas en recolección de datos, inferencia, creación del viaje y devolución de las soluciones. La duración total del viaje y el ajuste del coste al presupuesto son subproblemas que deben tratarse ligados al resto de subproblemas de creación del viaje.

4. Formalización

4.1. Ontología

Para construir una ontología que representara todos los conocimientos necesarios y nos sirviera como base de la futura implementación en CLIPS, decidimos usar una metodología de construcción incremental. Esta metodología se nos recomendó y explicó en clase de teoría, y consiste en una serie de pasos relativamente sencillos que permiten construir una ontología funcional y de alta calidad de forma progresiva y evitando muchos errores.

4.1.1. Construcción

La ya mencionada estrategia de construcción consta de cinco pasos:

1. Identificar todos los elementos del problema.
2. Identificar cuáles de estos elementos son los principales.
3. Establecer relaciones entre los elementos principales del problema.
4. Establecer relaciones entre el resto de elementos y los principales, eliminando posibles elementos innecesarios (o añadiendo si el paso 1 no se ha realizado correctamente).
5. Pensar que atributos tendrán los elementos de cada clase principal.

A continuación mostramos el proceso completo seguido por nosotros, con todos los conceptos y relaciones obtenidas:

1. Identificar todos los elementos del problema

Empleamos como base para construir nuestra ontología los conceptos del dominio ya identificados en el apartado de conceptualización. La imagen con estos elementos se muestra de nuevo a continuación para facilitar el seguimiento del proceso de construcción de la ontología (Figura 2).

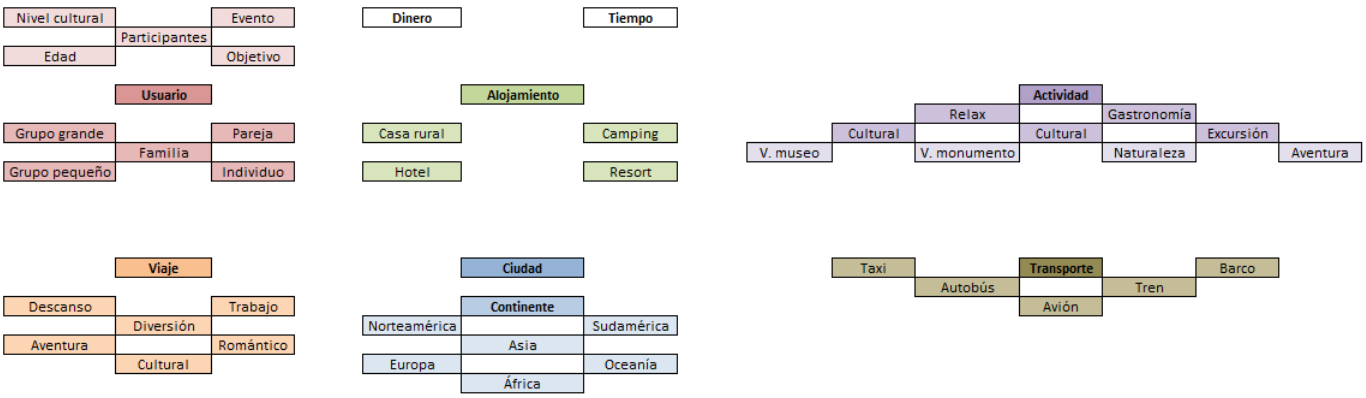

Figura 2: Representación gráfica de los conceptos

2.Identificar elementos principales

Para identificar los principales elementos buscamos los más importantes de entre todos los encontrados, dicho de una forma más clara, entre todos los conceptos buscamos aquellos que son necesarios para darle sentido al problema, por ejemplo el concepto camping no es esencial para el problema, pero el concepto alojamiento sí lo es.
En este paso confirmamos que los elementos principales serían los que esperábamos en la fase de conceptualización junto con algunos elementos nuevos:

1. Usuario: persona que participa en el viaje.
2. Participantes: agrupación de usuarios que participan en el viaje.
3. Viaje: itinerario a realizar por los usuarios.
4. Alojamiento: lugar donde se hospedan los usuarios en cada ciudad.
5. Tiempo: duración de un concepto.
6. Dinero: coste económico de un concepto.
7. Actividad: visita que realizar en una ciudad por los participante.
8. Transporte: método de desplazamiento para viajar entre ciudades.
9. Ciudad: población a visitar por el usuario en el viaje.

Cabe destacar que algunos de estos elementos no serán clases, ya que como veremos más adelante, algunos como elementos como tiempo se convertirán en atributos de otras clases. Todos los conceptos se pueden ver en la figura 3.

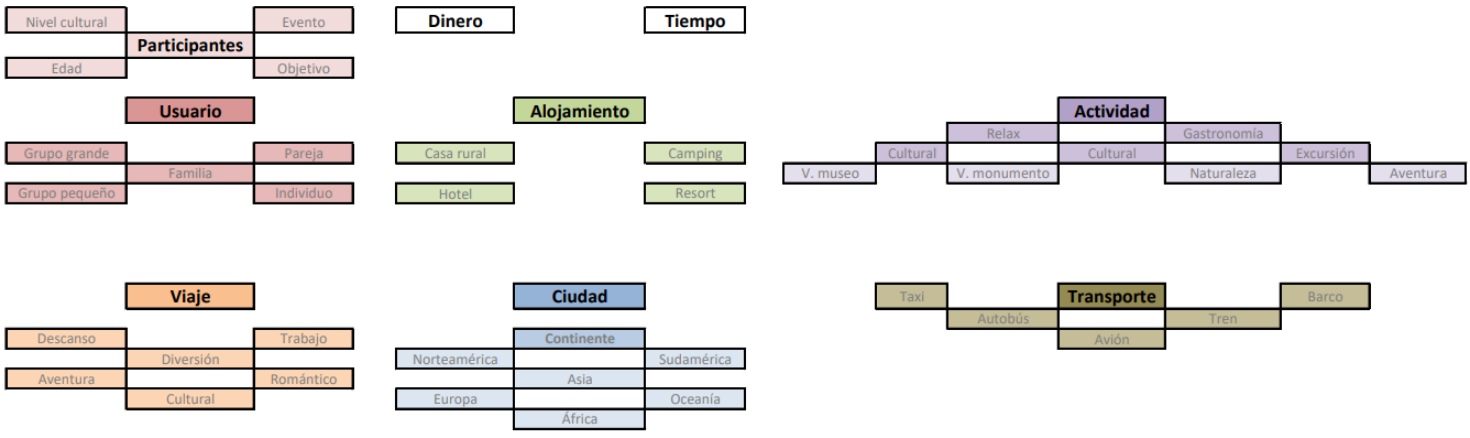

Figura 3: Elementos principales del problema

3. Relaciones entre los principales elementos

Para establecer las relaciones entre los principales elementos (obtenidos en el apartado anterior) decidimos pensar cómo sería un caso real (siguiendo la descripción del enunciado de la práctica), de esta manera obtuvimos las siguientes relaciones:

- Usuario →es un participante

- Participantes →participan en un viaje
- Alojamiento →esta_en una ciudad
- Alojamiento →pertenece a un viaje
- Ciudad →pertenece a un viaje
- Actividad →se_hace_en una ciudad
- Actividad →pertenece a un viaje
- Transporte →parte_de una ciudad
- Transporte →va_a una ciudad
- Transporte →pertenece a un viaje

De esta manera podemos modelar todos los elementos principales del viaje, ya que es tiene sentido que un viaje esté formado por una serie de alojamientos, ciudades, actividades y transporte, todo esto realizado o empleado por unos participantes.
Como se puede observar no hemos obtenido ninguna relación para tiempo y dinero, ya que son conceptos que serán atributos de muchas clases, ya que es mucho más coherente decir que la actividad tiene una duración y un precio, que establecer una relación entre las clases actividad y dinero.

4. Relaciones del resto de elementos, eliminar innecesarios
En este paso vemos qué elementos podemos relacionar y cuáles son innecesarios o se convertirán en atributos.Una vez realizado, obtenemos las siguientes conclusiones:

- Todos los tipos de grupo (familia, pareja...) serán atributos que se usarán y crearán en CLIPS.
- El concepto participante es innecesario, y podemos pasar su relación al usuario, esto hace que un usuario pase a representar todos los usuarios del viaje.
- Los tipos de viaje (romántico, descanso, cultural...) pasan a ser atributos del viaje, especificando el tipo de este.
- Como ya se ha mencionado, tiempo y dinero pasan a ser atributos de las clases viaje, transporte, actividad y alojamiento con los nombres duración y precio, respectivamente.
- Los tipos de alojamiento serán sub-clases de Alojamiento, se elimina el concepto de camping porque nos parece añadir complejidad innecesaria.
- Relax, cultural, excursión, gastronomía y ocio nocturno pasan a ser sub-clases de Actividad, a su vez Naturaleza y aventura pasan a ser sub-clases de excursión y visita museo y monumento sub-clases de cultural.
- Continente será un atributo de la ciudad, todos los posibles continentes desaparecen ya que ya los podremos representar con el atributo continente.
- Avión, tren y barco pasan a ser sub-clases de Transporte, para quitar complejidad al problema decidimos eliminar autobús y taxi.

5. Atributos de las clases principales

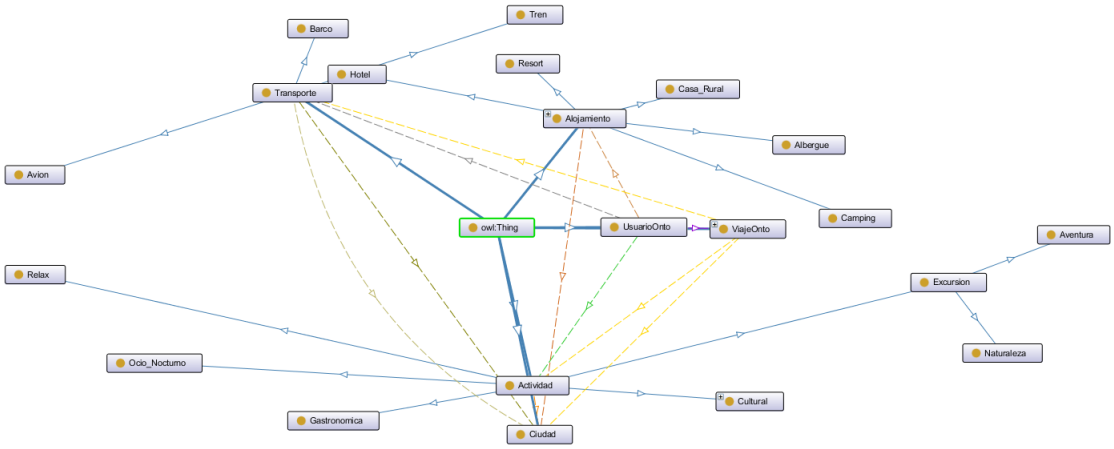

Figura 4: Ontología definitiva elaborada en Protégé

4.1.2. Elementos y descripción

Una vez seguidos todos estos pasos usamos Protégé para hacer nuestra ontología, la ontología obtenida se puede ver en la figura 4

4.2. Resolución de problemas

Una vez tenemos la ontología completada, analizamos cómo la ontología ayuda en la resolución de los problemas de los que hemos hablado anteriormente, para ello exploramos el razonamiento y la metodología de resolución de estos.

4.2.1. Razonamiento de subproblemas

Para poder ver si podemos solucionar los problemas mediante nuestra ontología tenemos primero que ver que elementos son necesarios para cada uno.

Los problemas de los que hablaremos son los identificados anteriormente en el apartado de conceptualización, así que no se entrará en detalle sobre que es cada problema.

1. Obtención de datos del usuario: para este problema requerimos de un sitio donde guardar todos los datos que se le pregunten al usuario, como por ejemplo la edad o el tipo de viaje que se quiere realizar.

2. Clasificación del usuario mediante inferencia: para este problema requerimos de los datos del usuario previamente preguntados.

3. Elaboración del itinerario: para elaborar el itinerario requerimos resolver los siguientes subproblemas:

   a) Asignación ciudades: para asignar ciudades necesitamos acceso a todas las ciudades instanciadas.

   b) Asignación alojamientos: para asignar los alojamientos necesitamos acceso a todos los alojamientos instanciados y las ciudades escogidas.

   c) Asignación actividades: para asignar las actividades necesitamos acceso a todas las actividades instanciadas y las ciudades escogidas.

d) Asignación de transportes: para asignar los transportes necesitamos acceso a todos los transportes instanciados y las ciudades escogidas.

e) Control de la duración: para esto necesitamos acceso a los parámetros de las preguntas (para duración, número de ciudades...), a la duración del viaje y a la duración de toda actividad o estancia.

f) Control del precio: para esto necesitamos acceso al precio de todas las instancias, como por ejemplo transporte o alojamiento.

4. Crear soluciones: para crear las soluciones requerimos de todas las ciudades, alojamientos, actividades y transportes escogidos.

Una vez hemos identificado todas las necesidades vamos a ver si la ontología las satisface.

Para los problemas de obtención de datos e inferencia disponemos de la clase usuario, con la cual podemos almacenar y obtener los valores necesarios.

Para la elaboración del itinerario y sus subproblemas, disponemos de clases para almacenar todas las instancias necesarias y maneras de obtener las ciudades escogidas usando el viaje. Además todas las clases dispondrán de duración y precio, además de los totales, en la clase viaje.

Para el problema de los resultados con la clase viaje podremos solucionarlo fácilmente, ya que tendremos todos los alojamientos, ciudades, actividades y transportes.

4.2.2. Metodología de resolución

La metodología a seguir con la ontología para resolver los problemas es sencilla y se realizará mediante el uso de las relaciones entre clases:

Para el problema de la obtención de datos preguntamos al usuario y almacenamos en los campos del usuario.

Para la inferencia podemos obtener los datos del usuario.

Para el itinerario podemos usar las relaciones entre clases para obtener las instancias necesarias. Podemos, por ejemplo, obtener las ciudades del viaje obteniendo las ciudades que pertenecen a este mediante las relaciones de la ontología. Para comprobar que un transporte va y parte de diferentes ciudades podemos usar las relaciones va_a y parte_de de la clase transporte. Por tanto la metodología a seguir para solucionar el problema del itinerario será sencilla y seguirá el plan mencionado originalmente, ya que la ontología nos permite acceder a las clases y atributos necesarios de manera sencilla.

Para ver los resultados podremos usar los atributos y relaciones de la clase viaje.

5. Implementación

Una vez realizada la ontología decidimos empezar con el código en CLIPS. Decidimos realizar también las instancias con CLIPS directamente, ya que nos pareció más sencillo generar una gran cantidad de instancias mediante scripts que con Protégé, además hacer las instancias con Protégé podría ocasionar algún problema extra a la hora de exportarlas.

5.1. Representación de la ontología

Una vez empezamos a programar en CLIPS nos dimos cuenta de que, por ejemplo, las clases ciudad y usuario de la ontología estaban correctamente planteadas, pero al requerir de una instancia y al ser más complejas de usar decidimos emplear templates para guardar toda la información del usuario y del viaje, de esta manera conseguimos un código más compacto y simple. Las clases usuario y viaje siguen existiendo en la ontología pero no las usamos directamente.
El resto de la ontología la usamos tal y como se explicó en la metodología y razonamiento de subproblemas. La estructura que seguirá nuestro código CLIPS será la siguiente:

- Funciones
- Módulos:
    - Templates necesarios en el módulo o siguientes módulos
    - Reglas del módulo
    - Regla que comprueba que se acaba el módulo y pasa al siguiente módulo

Además tendremos otros dos archivos, llamados sbc.clp y instancias.clp. El primero de ellos contiene lo obtenido con la ontología, es decir, las clases y las relaciones entre ellas. Instancias, como su nombre indica, crea todas las instancias que usamos para el problema.

5.2. Módulos e implementación de estos

Para implementar nuestra práctica hicimos uso de los módulos de CLIPS, empleando un módulo por problema identificado, cada uno de ellos contiene reglas y templates, las funciones las declaramos en el inicio del código.

Ahora entraremos en detalle de las reglas y templates más importantes para que explicar y justificar la implementación, solo explicaremos las reglas las más complejas, obviando reglas como acaban-las-preguntas, que se usan para cambiar de modulo. La lista completa de funciones y módulos con sus templates y reglas:

- Modulo MENU
    - Templates:
        - usuario
        - viaje
    - Reglas:
        - preguntar-edades
        - preguntar-tipo-de-viaje
        - preguntar-tipo-de-viaje
        - preguntar-dias
        - preguntar-ndias-ciudad
        - preguntar-nciudades
        - preguntar-presupuesto
        - preguntar-medios-de-transporte
        - preguntar-calidad-alojamiento
        - preguntar-popularidad-ciudad *

        - preguntar-duracion-o-calidad **
        - acaban-las-preguntas
    - Facts:
        - inicialitzacio
    - Funciones:
        - pregunta-llista
        - pregunta-float
        - pregunta-int
        - pregunta-restri
        - pregunta
        - si-o-no-p

- Modulo Inferencia
    - Templates:
        - fix_dias_por_ciudad
        - ciudad_puntuada
    - Reglas:
        - con-ninos
        - con-adolescentes
        - numero-integrantes
        - obtenertipousuarios
        - longitud_viaje
        - ciudades_romanticas
        - ciudades_descanso
        - ciudades_diversion
        - ciudades_trabajo
        - ciudades_aventura
        - ciudades_cultural
        - initialize_dias_por_ciudad
        - acabainferencia

- Modulo LOGIC
    - Templates:
        - fix_aloj
        - ciudad_escogida
        - fix_trans
        - alojamiento_puntuado
        - nextciudad
        - estructura
    - Reglas:

        - escoger-ciudades-rule-high-fit
        - escoger-ciudades-rule-low-fit
        - escoger-ciudades-rule-low-fit-no-continent
        - comprovar
        - assertsciudades
        - initialize_alojamiento_puntuado
        - escoger_alojamiento
        - escoger_transporte
        - escoger_actividades
        - acaba-la-logica

- Modulo RESULTADOS
    - Reglas:
        - printar_plantilla_rule
        - printar_viaje_rule
        - not_continente
        - acaba-resultados
        - printar_plantilla_rule2
        - preparar_segundo_viaje_aloj_punt
        - preparar_segundo_viaje_next_ciudad
        - preparar_segundo_viaje_fix_trans
        - preparar_segundo_viaje_fix_aloj
        - preparar_segundo_viaje_estr
        - preparar_segundo_viaje

[*] La pregunta popularidad-ciudad al final se ejecuta pero los resultados obtenidos no se usan en la práctica, esto es porque era un expansión y no nos dio tiempo a implementarla. [**] La pregunta duracion-o-calidad al final se ejecuta pero los resultados obtenidos no se usan en la práctica, todo y así hemos decidido dejarla en la entrega. Esta pregunta originalmente iba a ser para un ampliación que teníamos en mente, pero al final no nos dio tiempo a implementar.

5.2.1. Reglas más importantes para solucionar los problemas

Para explicar el funcionamiento del código en CLIPS usaremos la estructura usada hasta el momento, explicando como atacamos cada problema identificado.

Obtención de datos del usuario

De la resolución de este problema se encarga el modulo de MENU, este modulo crea dos templates muy importantes:

- Usuario: el template de usuario guardará toda la información que se le pregunte al usuario y contendrá toda la información sobre el usuario necesaria para organizar un viaje.

- Viaje: el template viaje guardará todos los elementos de la ruta, en nuestro caso estos serán: las ciudades, los días por ciudad, los alojamientos, las actividades por ciudad, los transportes, el coste del viaje y la duración total.

El modulo MENU realiza preguntas al usuario, usando i/o a través de llamadas a las funciones creadas también en este módulo, muchas de estas están inspiradas en el documento FAQS de CLIPS dado como documentación en la asignatura.

En cada reglas se guardan los resultados en el template usuario y al final del modulo se pasa al modulo de INFERENCIA.

Inferir los datos obtenidos para clasificar al usuario

Una vez hemos realizado todas las preguntas necesarias al usuario llegamos al modulo de INFERENCIA, el objetivo de este modulo es resolver nuestro segundo problema, clasificar al usuario infiriendo los datos obtenidos.

En este modulo tenemos 2 templates:

- fix_dias_por_ciudad: este template servirá para unificar todos los posibles días por ciudad, contiene un slot para el número de días.
- ciudad_puntuada: este template servirá para priorizar unas ciudades por encima de otras a la hora de escogerlas en la ruta, contiene un slot con el nombre de la ciudad y uno con un valor que determina la adecuación de una ciudad para el usuario en cuestión.

Las reglas más importantes de este módulo son:

- con-ninos: esta regla comprueba si hay niños en el viaje a partir de inferencia sobre las edades obtenidas en el módulo anterior, la regla con-adolescentes funciona de manera similar.
- numero-integrantes: esta regla calcula el número de integrantes a partir del número de edades introducidas en las preguntas.
- obtenertipousuarios: esta regla infiere el tipo de grupo que viajará (individuo, pareja, familia o grupo), a partir de las edades introducidas y de si tienen niños en el grupo o no.
- longitud_viaje: esta regla comprueba la longitud del viaje que quiere el usuario, si esta es ¿6 se permitirá el viaje entre continentes, en caso contrario no se permitirá.
- ciudades_romanticas: esta regla se ejecuta si el tipo del viaje introducido por el usuario es romántico, después unifica todas las ciudades, si esta es considerada una ciudad romántica se hace un assert de ciudad_puntuada con el nombre y un valor alto, si no se considera ciudad romántica se le asigna un valor bajo. El funcionamiento de las reglas para descanso, diversión, trabajo, aventura y cultural es el mismo que el de esta regla. Cabe destacar que solo se ejecutará una de estas reglas y hará que se prioricen algunas ciudades por delante de otras para que sean adecuadas con el tipo de viaje. Las ciudades de cada tipo han sido escogidas con conocimiento de experto.
- initialize_dias_por_ciudad: esta regla hace un assert de fix_dias_por_ciudad para cada valor entre el mínimo y el máximo de días por ciudad especificados por el usuario, para así poder unificar los días a la hora de escoger el número de días por ciudad.

Crear itinerario Para resolver el problema de crear el itinerario usaremos el módulo de LOGIC, este es el módulo más complejo de todo el código, ya que se encarga de escoger el viaje entero y es el que ha llevado más tiempo implementar.
Este módulo contiene los siguientes templates:

- fix_aloj: este template servirá para ver qué alojamientos se han usado en el viaje.
- ciudad_escogida: este template servirá para ver qué ciudades se han usado en el viaje
- fix_trans: este template servirá para ver qué transportes se han usado en el viaje
- alojamiento puntuado: este template servirá para ver qué "fitness"tienen los alojamientos asociados.
- nextciudad: este template establece una relación temporal entre dos ciudades, indicando que una ciudad es la siguiente a otra en el viaje.
- estructura: template que nos permite asociar ciudades con los días que van a ocupar y cómo de ocupada está la estancia en esa ciudad (por las actividades que se hacen allí).

Como se menciona anteriormente, este problema lo dividimos en los siguientes subproblemas:

Escoger ciudades Para escoger las ciudades usamos las siguientes reglas:

- escoger-ciudades-rule-high-fit: regla por defecto, coge ciudades con un "fitness.ªlto.
- escoger-ciudades-rule-low-fit: regla que se ejecuta si no se llega al mínimo de ciudades que quiere el usuario, coge ciudades con "fitness"bajo.
- escoger-ciudades-rule-low-fit-no-continent: regla que se ejecuta cuando no hay suficientes ciudades en el continente que se está visitando.

Las tres reglas funcionan de manera muy similar, unifican ciudades de los asserts hechos en ciudad_puntuada, comprobando que pertenecen a las instancias del problema, después unifican el número de días que se estará en la ciudad con los asserts hechos anteriormente en fix_dias_por_ciudad. Todo esto pasa por una serie de tests en la parte izquierda de la regla que varían entre reglas (por ejemplo, solo ciudades con un "fitness"mayor que un threshold en la primera regla). Los tests comunes son los relacionados con no pasarse de los máximos (días del viaje, número de ciudades) o comprobar que una ciudad no está ya en la ruta. En la parte derecha de la regla se asignan todos los valores necesarios en el template de viaje.

Una vez se ha ejecutado esta regla, se ejecuta la regla comprobar. Esta regla comprueba que se hayan cumplido todos los mínimos necesarios (número de días del viaje, número de ciudades), el número de días por ciudad no se comprueba, ya que este nunca será menor que el mínimo (debido a que se unifica con asserts que van del mínimo al máximo). Si no llega a los mínimos el programa devuelve error. Como el algoritmo siempre intentará llegar al máximo nos aseguramos de que no falle por haber hecho malas asignaciones.

La última regla relacionada con la selección de ciudades es assertciudades, esta regla establece las relaciones next entre ciudades haciendo asserts de nextciudad para cada par de ciudades escogido.

Escoger alojamientos Para el subproblema de escoger alojamientos usaremos las siguientes reglas:

- initialize_alojamiento_puntuado: esta regla da valor al "fitness"para los alojamientos que unifica dependiendo si estos están cerca del centro o son de la calidad mínima que pide el usuario.

- escoger_alojamiento: esta regla unifica todo alojamiento puntuado para todas las ciudades pertenecientes a la ruta, y escoge los mejores para cada ciudad, además realiza test para asegurar que el alojamiento está en la ciudad para la que se está escogiendo alojamiento y ver que no se están repitiendo alojamientos. También se realiza un test para ver que el alojamiento es de una determinada calidad (calculado en initialize_alojamiento_puntuado). Además, cuando una ciudad de la ruta ya dispone de un alojamiento se hace un assert para no añadir más de un alojamiento por ciudad.

Escoger transporte Para el subproblema de escoger transportes usaremos la siguiente regla:

- escoger_transporte: en esta regla unificaremos en la parte izquierda todos los transportes, para todas las ciudades pertenecientes a la ruta, comprobando que se mantiene la relación next entre las ciudades escogidas. Esto nos obliga a unificar dos ciudades a la vez, lo que añade muchos tests para asegurar que sean diferentes. Además, nos aseguramos que el transporte escogido no pertenezca al tipo que no quiere el usuario. En la parte derecha de la regla añadimos los transportes al viaje y realizamos todos los cambios necesarios.

Escoger actividades Para el subproblema de escoger actividades usaremos la siguiente regla:

- escoger_actividades: de manera muy similar a escoger transporte y alojamientos unificamos todas las actividades para todas las ciudades pertenecientes a la ruta, en este caso para llenar un día con actividades hacemos la siguiente asunción, un día tiene un 100 % de capacidad para actividades, y una actividad ocupa un porcentaje de un día, que puede variar de 0 a valores de 200 o 300 (actividades de múltiples días). Para cada ciudad asignamos actividades dependiendo del número de días que se esté en la ciudad, por ejemplo si es está en París 3 días, tendremos una capacidad de 300 para actividades (3 * 100). Con esto en mente, asignamos actividades a una ciudad hasta que se llega al límite determinado por los días que se pasan en la ciudad. Todo se realiza con tests y unificación en la parte izquierda de la regla usando el template estructura previamente mencionado. En la parte derecha solo actualizamos la ocupación y añadimos las actividades al viaje, cada vez que cambiamos de ciudad también añadimos la ciudad a las actividades para facilitar mostrar los resultados.

Control del precio Para controlar el precio total del viaje en toda regla que añada coste al viaje sumamos el coste al total y comprobamos que no nos pasemos del presupuesto (el test se realiza en la parte izquierda de las reglas). Las reglas en las que se realiza esto son escoger_alojamiento, escoger_transporte y escoger_actividades. Estas se ejecutan de tal manera que si falta presupuesto para el viaje se empezará recortando presupuesto de actividades (cogiendo menos actividades), ya que es coherente que se pueda realizar un viaje con menos actividades pero no sin alojamiento. Si no se puede pagar el alojamiento o el transporte se devuelve error indicando que no se puede realizar un viaje como el pedido con el presupuesto dado.
La programación de esto es igual para todas las reglas donde ocurre.

Control de la duración Como ya hemos explicado, el control de la duración se realiza a la hora de escoger ciudades, es importante destacar que el transporte no añade tiempo al viaje, como no era obligatorio era un añadido que teníamos planteado pero no llegamos a tiempo a ponerlo como extra. La idea era hacer que se tardara un determinado tiempo para transportes entre continentes, si no sería negligible, Esto añadía una casuística muy grande al problema y nos pareció complicado implementarlo.

Imprimir los resultados y generar segundo viaje
Para imprimir los resultados disponemos de dos reglas:

- printar_plantilla_rule: esta regla printa una plantilla para que el output sea más estético

- printar_viaje_rule: esta regla printa el viaje con toda la información, las ciudades visitadas, en su orden de visita, los alojamientos para cada ciudad con sus estrellas, los transportes para ir de ciudad a ciudad, las actividades a realizar en cada ciudad, la duración del viaje y el coste de este.

Con el objetivo de generar un segundo viaje queremos volver a ejecutar el módulo de la lógica pero escogiendo ciudades diferentes. Para este fin deberemos revertir los cambios en la base de hechos de CLIPS y sobre todo modificar el viaje previo y restaurar los valores de sus slots a unos valores por defecto razonables.

- preparar_segundo_viaje_aloj_punt: esta regla quita de la base de datos todos los hechos de alojamiento_puntuado.
- preparar_segundo_viaje_next_ciudad: esta regla quita de la base de datos todos los hechos de next_ciudad.
- preparar_segundo_viaje_fix_trans: esta regla quita de la base de datos todos los hechos de fix_trans.
- preparar_segundo_viaje_fix_aloj: esta regla quita de la base de datos todos los hechos de fix_aloj.
- preparar_segundo_viaje_estr: esta regla quita de la base de datos todos los hechos de estructura.
- preparar_segundo_viaje: esta regla, a parte de quitar de la base de datos hechos que impiden que se pueda escoger el viaje correctamente, restaura a unos valores por defecto al hecho del viaje.
- not_continente: Si se ha añadido alguna ciudad desde la regla donde no hace falta que las ciudades sean del mismo continente, entonces el hecho de continent_rule_not_respected se encontrará en la base de datos y se imprimirá el mensaje por pantalla de que el viaje no ha podido ser del mismo continente.

Creación de las instancias Como último detalle sobre la implementación en CLIPS nos gustaría explicar cómo se han creado las instancias y cuántas hemos creado. Las instancias han sido creadas en CLIPS directamente, para ahorrar problemas haciendo el paso de Protégé a CLIPS, algunos atributos han sido puestos con scripts, pero la mayoría de instancias han sido ejemplos reales y no números (por ejemplo, París en vez de ciudad 001) (nos parecía adecuado y más divertido al ser un creador de rutas). Hemos creado aproximadamente 20 ciudades, para cada ciudad un mínimo de dos alojamientos de diferente categoría y de tres actividades por ciudad. Para el transporte nos parecía adecuado que hubiera un avión para cada par de ciudades y añadir trenes y barcos para algunas combinaciones de ciudades.

5.3. Proceso de implementación con metodología incremental (hasta el primer prototipo funcional)

Para implementar todo el código hemos seguido un proceso incremental, la metodología seguida ha priorizado que no nos quedáramos estancados en el proceso de implementación porque no supiéramos como programar algo en CLIPS (sin duda la curva de aprendizaje del lenguaje y la poca documentación disponible han sido las partes más duras de la práctica).
Pasos seguidos en la implementación:

1. Implementación del módulo que realiza preguntas:
   Nuestro primer objetivo fue implementar unas funciones que usando el input/output de CLIPS (usando ejemplos del FAQS de CLIPS) nos permitieran realizar todas las preguntas necesarias al usuario para realizar el itinerario.
   Este nos pareció un buen punto de partida, ya que disponíamos de muchos ejemplos para ayudarnos y aprender CLIPS de forma práctica. Una vez implementadas todas las reglas (y probadas) decidimos crear lo que serían los orígenes de los templates usuario y viaje para aprender a guardar datos en vez de printar-los.
   Una vez tuvimos implementado esto, ya disponíamos de un módulo de preguntas sólido para usar durante el resto de la práctica, muchas preguntas no se llegaron a usar pero nos sirvieron como base para la práctica.

2. Implementación de un módulo de inferencia primitivo:
   Una vez tuvimos el módulo de preguntas implementado decidimos empezar a programar un módulo de inferencia, desde el principio vimos clara la estructura de preguntar-inferir-lógica-imprimir, así que hicimos un par de funciones que inferían cosas básicas para usar en la lógica del programa.

3. Implementación de una lógica primitiva para la práctica, apuntando a la entrega extra:
   Una vez tuvimos el módulo de preguntas e inferencia implementados decidimos empezar a programar un modelo primitivo del creador de rutas, este decidimos que sería iterativo, pero pondría las bases para la estructura del resto de la práctica. Decidimos que sería iterativo ya que era imposible para nosotros en ese momento usar la unificación de CLIPS para realizar la práctica (nos parecía muy complicada) y queríamos entregar una práctica "funcional"para la entrega extra que nos sirviera como base para el resto de práctica. Esta fue una buena decisión, ya que nuestro nivel de CLIPS aumentó mucho y obtuvimos una base sólida junto a una ruta a seguir para implementar la práctica definitiva.

4. Creación de instancias y output de la solución para la primera entrega:
   En paralelo a la implementación de la lógica de la práctica creamos las instancias necesarias para probar el primer prototipo, además implementamos el módulo de RESULTADOS para ver la solución obtenida (de momento solo un viaje). El módulo de resultados obtenido en esta parte se ha quedado casi igual en la entrega definitiva, ya que al ser output, llegados a este punto ya dominábamos este aspecto de CLIPS.

5.3.1. Primer prototipo funcional

El prototipo obtenido para la entrega extra era capaz de preguntar al usuario todo lo requerido y crear un viaje que cumpliera las restricciones de los máximos, asignando alojamientos, transportes y actividades a cada ciudad. Toda la lógica había estado implementada de manera iterativa para facilitar la curva de aprendizaje y sentar unas bases de cara a la recta final de la práctica.

5.3.2. Proceso de implementación con metodología incremental (Del primer prototipo a la versión final)

Una vez entregado el prototipo inicial para el punto extra teníamos clara la ruta a seguir para lo que faltaba de práctica, esta fue la siguiente:

1. Completar el módulo de lógica usando conocimientos de CLIPS más avanzados:
   Para tener una mejor lógica necesitábamos completar el módulo de INFERENCIA, así que con el conocimiento de CLIPS que habíamos obtenido hasta el momento acabamos el módulo de INFERENCIA.

2. Re-implementar parte de la lógica para usar al unificación de CLIPS:
   Ahora que disponíamos de un conocimiento mucho más avanzado de CLIPS decidimos implementar la lógica de la práctica usando las partes izquierdas de las reglas. Esto lo hicimos ya que sabíamos que de cara al código definitivo un código que explotara la unificación de variables de CLIPS tendría un número de líneas cercano a 800 como mucho, y que, en cambio, uno iterativo podría llegar a tener hasta 10 veces más líneas.
   Al principio fue complicado, ya que de esta parte había muy poca documentación, y al final nuestra fuente de documentación para entender el lenguaje acabó siendo la gramática del compilador de CLIPS, después de un par de días de programar y analizar la gramática ya podíamos programar con soltura en CLIPS, y la parte complicada de la práctica la implementamos rápidamente.

3. Implementar funciones de "fitness"para escoger mejor:
   Una vez teníamos un prototipo funcional que usaba la unificación de CLIPS decidimos mejorar el módulo de

INFERENCIA para que escogiera mejor las ciudades y los alojamientos usando funciones de fitness (clasificar las instancias como mejores o peores). Para ello tuvimos que modificar ligeramente algunas reglas de la lógica (por esto son ligeramente diferentes las reglas que escogen ciudades y alojamientos de las que escogen trasporte y actividades).
También nos hubiera gustado usar este método para transporte y actividades pero decidimos priorizar las partes obligatorias de la práctica.

4. Implementar que se generen dos viajes diferentes:
   Para acabar con la implementación de la práctica implementamos que se obtuvieran en dos viajes totalmente diferentes en la misma ejecución del programa para el mismo input del usuario. Esto fue algo complejo como se ha explicado antes, por eso lo dejamos para cuando tuviéramos un mayor nivel de CLIPS.

5. Pulir código y gestión de errores:
   Como añadido pulimos el código y usamos printouts para notificar al usuario de algunos errores o de porque su viaje no se puede realizar (especificando el motivo).

Nos gustaría destacar que para todo paso de la metodología usada probábamos que el programa funcionara como era esperado para esa etapa, y sobre todo que estuviéramos solucionando los problemas planteados originalmente. Esto lo mencionamos aquí pero se aplica a todos los pasos.

6. Juegos de prueba

Todos los juegos de prueba han sido creados para probar un aspecto específico del programa, durante el desarrollo de este se han usado casos de prueba para comprobar el funcionamiento de algunas cosas, por eso nos parece más adecuada probar el programa con juegos de prueba completos que pongan a prueba todo el código. La estructura que seguiremos para todo juego de prueba será:

1. Presentación del juego de prueba
2. ¿Qué pretendemos probar?
3. Input-output
4. Conclusiones extraídas de la prueba

Es necesario que mencionar que funcionalidades básicas como el funcionamiento de las preguntas, el output de resultados, la inferencia de datos básicos u otros se prueban en todos los juegos de pruebas de manera implícita y no tiene un juego de pruebas explícito para evitar redundancias. Es por eso que para estos casos solo destacaremos comportamientos excepcionales.

6.1. Prueba 1

6.1.1. Presentación del juego de prueba

En este juego de pruebas nos encontramos en un escenario donde un grupo pequeño de jóvenes con mucho dinero quiere hacer un viaje corto y divertido. No les importa coger cualquier tipo de transporte.

6.1.2. ¿Qué pretendemos probar?

En este juego de pruebas pretendemos probar el correcto funcionamiento del sistema para diversos aspectos:

- Viaje dentro del mismo continente (duración de menos de seis días).

- Combinación de transportes, ya que no les importa usar diversos transportes.
- Uso del dinero para realizar un viaje divertido.
- Cumplir los mínimos y máximos requeridos por los usuarios.
- Funcionalidad que ofrece viajes con continentes distintos como opción (incluso si el viaje es corto)

6.1.3. Input-output obtenido

CLIPS input:

```
Escriba las edades de los participantes separados por espacios:
Su edad es: (20 21 23 23)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 3 dias y maximo 5 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 1 dias por ciudad y maximo 2 dias por ciuda
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 3 ciudades y maximo 4 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 4000€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte ()
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 1
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: mixto
```

Clips output:

```
-------------------------------------------------------------------------------------------
-                                      PRIMER VIAJE                                       -
-------------------------------------------------------------------------------------------

Duracion del viaje: 5 dias

Ciudades visitadas: Paris(2), Venecia(2), Granada(1),

Visitas: Visita_Louvre, Paseo_Campos_Eliseos, Paris, Paseo_en_gondola, Murano_Burano_Torcello, Cena_junto_

Alojamientos: Hotel_Darcet, Candil_suite_Realejo, Hotel_Carlton_On_The_Grand_Canal,

Transporte: GRA_VEN_barco, VEN_PAR_avion,
```

```
Coste total del viaje: 1242

-------------------------------------------------------------------------------------------------
-                                        SEGUNDO VIAJE                                          -
-------------------------------------------------------------------------------------------------

Duracion del viaje: 5 dias

Ciudades visitadas: Praga(2), Amsterdam(2), Kioto(1),

Visitas: Visita_al_castillo, Praga, Museo_van_gogh, Murano_Burano_Torcello, Paseo_por_el_barrio_rojo, Amst

Alojamientos: Hotel_the_m_kyoto, Innside_Melia_Amsterdam, Hotel_Royal_Prague,

Transporte: AMS_PRA_tren, KIO_AMS_avion,

Coste total del viaje: 1497

El viaje no puede ser del mismo continente
```

6.1.4. Conclusiones extraídas de la prueba

Después de ejecutar este juego de pruebas obtenemos diversas conclusiones: El primer viaje se mantiene dentro del mismo continente, ofreciendo un viaje por ciudades que se consideran de diversión y mezclando transportes, tal y como esperábamos para el correcto funcionamiento de esta prueba. Todos los mínimos y máximos se cumplen. En el segundo viaje vemos la funcionalidad que ofrece un viaje para cumplir las restricciones, pero mezclando continentes, además notifica de que el viaje no puede ser dentro del mismo continente.
Este es el resultado esperado y consideramos la prueba un éxito.

6.2. Prueba 2

6.2.1. Presentación del juego de prueba

En este caso tenemos un grupo de gente más mayor, que quieren estar más tiempo de viaje y no quieren usar el tren, además quieren alojamientos de más de tres estrellas.

6.2.2. ¿Qué pretendemos probar?

En este juego de pruebas pretendemos probar el correcto funcionamiento del sistema para diversos aspectos:

- Obtener viajes largos que mezclen continentes.
- No usar trenes.
- Solo ver alojamientos de más de tres estrellas.

6.2.3. Input-output obtenido

CLIPS input

```
 Escriba las edades de los participantes separados por espacios:
Su edad es: (34 43 43 45)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 8 dias y maximo 10 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 1 dias por ciudad y maximo 3 dias por ciuda
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 4 ciudades y maximo 5 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 10000€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte (tren)
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 3
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: calidad
```

CLIPS output

```
-------------------------------------------------------------------------------------------
-                                      PRIMER VIAJE                                       -
-------------------------------------------------------------------------------------------

Duracion del viaje: 10 dias

Ciudades visitadas: Miami(3), Venecia(3), Granada(3), Praga(1),

Visitas: 230, Alquiler_yate, Miami, Paseo_en_gondola, Cena_junto_al_canal, Venecia, Visita_alhambra, Murano

Alojamientos: Hampton_miami, Hotel_Royal_Prague, Candil_suite_Realejo, Hotel_Carlton_On_The_Grand_Canal,

Transporte: GRA_VEN_barco, PRA_GRA_avion, VEN_MIA_avion,

Coste total del viaje: 3727

El viaje no puede ser del mismo continente
-------------------------------------------------------------------------------------------
-                                      SEGUNDO VIAJE                                      -
-------------------------------------------------------------------------------------------

No se puede programar un viaje debido a los alojamientos
```

```
Duracion del viaje: 10 dias

Ciudades visitadas: Amsterdam(3), Kioto(3), Tahiti(3), Cancun(1),

Visitas: Museo_van_gogh, Murano_Burano_Torcello, Paseo_por_el_barrio_rojo, Amsterdam, Espectaculo_Maiko_tr

Alojamientos: Dreams_visita_cancun_resort, Piece_Hostel_Sanjo, Innside_Melia_Amsterdam,

Transporte: KIO_AMS_avion, TAH_KIO_avion, CAN_TAH_avion,

Coste total del viaje: 3720

El viaje no puede ser del mismo continente
```

6.2.4. Conclusiones extraídas de la prueba

Después de ejecutar la prueba vemos como en ambos viajes se nos ofrecen rutas que visitan diversos continentes y duran 10 días. En ambos viajes podemos ver como usamos barcos y aviones pero no trenes, como esperábamos. Además, todos los alojamientos obtenidos son de 3 o más estrellas (se puede ver en las instancias de la práctica). Si nos fijamos en el segundo viaje podemos ver que tenemos un error que nos indica que le viaje no se ha podido organizar, ya que falta un alojamiento, esto se debe a que en el destino no hay alojamientos que se ajusten a las necesidades del usuario, así que no se podrá usar esa ruta, ya que aunque parece completa, le falta un alojamiento.

6.3. Prueba 3

6.3.1. Presentación del juego de prueba

En este caso tenemos una familia con niños que no quiere viajar en barco y quieren ir a alojamientos de como mínimo una estrella, además quieren un viaje de diversión.

6.3.2. ¿Qué pretendemos probar?

En este juego de pruebas pretendemos probar el correcto funcionamiento del sistema para diversos aspectos:

- Comprobar que se genera un viaje en familia cumpliendo todas las restricciones especificadas
- Comprobar que se generan solo viajes con alojamientos de al menos una estrella

6.3.3. Input-output obtenido

CLIPS input

```
Escriba las edades de los participantes separados por espacios:
Su edad es: (35 40 10 16)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 3 dias y maximo 10 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
```

```
Introduzca su respuesta: Dias seleccionados, se viajará minimo 1 dias por ciudad y maximo 2 dias por ciuda
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 3 ciudades y maximo 4 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 4000€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte (barco)
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 1
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: mixto
```

CLIPS output

```
    ---------------------------------------------------------------------------------------------
-                                       PRIMER VIAJE                                          -
---------------------------------------------------------------------------------------------

Duracion del viaje: 8 dias

Ciudades visitadas: Paris(2), Venecia(2), Granada(2), Praga(2),

Visitas: Visita_Louvre, Paseo_Campos_Eliseos, Paris, Paseo_en_gondola, Cena_junto_al_canal, Venecia, Visit

Alojamientos: Hotel_Darcet, Hotel_Oya, Candil_suite_Realejo, Hotel_Carlton_On_The_Grand_Canal,

Transporte: GRA_VEN_tren, PRA_GRA_tren, VEN_PAR_avion,

Coste total del viaje: 1674

---------------------------------------------------------------------------------------------
-                                       SEGUNDO VIAJE                                         -
---------------------------------------------------------------------------------------------

No se puede programar un viaje debido a los alojamientos
Duracion del viaje: 8 dias

Ciudades visitadas: Amsterdam(2), Kioto(2), Tahiti(2), Cancun(2),

Visitas: Museo_van_gogh, Murano_Burano_Torcello, Paseo_por_el_barrio_rojo, Amsterdam, Espectaculo_Maiko_tr

Alojamientos: Dreams_visita_cancun_resort, Hotel_the_m_kyoto, Innside_Melia_Amsterdam,

Transporte: KIO_AMS_avion, TAH_KIO_avion, CAN_TAH_avion,

Coste total del viaje: 3510
```

```
El viaje no puede ser del mismo continente
```

6.3.4. Conclusiones extraídas de la prueba

Después de ejecutar el juego de pruebas extraemos las siguientes conclusiones:
Obtenemos dos viajes familiares (al igual que antes, uno de ellos no puede completarse ya que falta un alojamiento), en estos viajes vemos el uso de aviones y barcos, además de alojamientos de todas las calidades (1 o más estrellas implica cualquier calidad de alojamiento).

6.4. Prueba 4

6.4.1. Presentación del juego de prueba

En este caso tenemos un estudiante con muy poco dinero que quiere realizar un viaje largo.

6.4.2. ¿Qué pretendemos probar?

En este juego de pruebas pretendemos probar el correcto funcionamiento del sistema para diversos aspectos:

- Comprobar como para un viaje largo, si no tienes mucho dinero no vas a poder viajar
- Comprobar el correcto funcionamiento de la detección de diversos errores.

Este juego de pruebas puede parecer simple, pero nos ayudará a detectar fallos en el sistema de notificación de errores.

6.4.3. Input-output obtenido

CLIPS input

```
Escriba las edades de los participantes separados por espacios:
Su edad es: (22)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 2 dias y maximo 10 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 1 dias por ciudad y maximo 2 dias por ciuda
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 3 ciudades y maximo 4 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 100€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte (barco)
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 1
```

```
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: calidad
No se puede programar un viaje debido a los alojamientos
No se puede programar un viaje debido a los transportes
```

CLIPS output

```
   ---------------------------------------------------------------------------------------------
-                                        PRIMER VIAJE                                         -
---------------------------------------------------------------------------------------------

Duracion del viaje: 8 dias

Ciudades visitadas: Paris(2), Venecia(2), Granada(2), Praga(2),

Visitas: Paseo_Campos_Eliseos, Paris, Visita_alhambra, Granada, Visita_al_castillo, Praga,

Alojamientos:

Transporte: VEN_PAR_avion,

Coste total del viaje: 97

---------------------------------------------------------------------------------------------
-                                        SEGUNDO VIAJE                                        -
---------------------------------------------------------------------------------------------

No se puede programar un viaje debido a los alojamientos
No se puede programar un viaje debido a los transportes
Duracion del viaje: 8 dias

Ciudades visitadas: Amsterdam(2), Kioto(2), Tahiti(2), Cancun(2),

Visitas: Paseo_por_el_barrio_rojo, Amsterdam, Dia_playa, Tahiti, Crucero_cena_barco_pirata, Cancun,

Alojamientos:

Transporte:

Coste total del viaje: 100

El viaje no puede ser del mismo continente
```

6.4.4. Conclusiones extraídas de la prueba

Como podemos ver en los resultados obtenidos después de haber ejecutado el juego de pruebas, confirmamos que sin dinero no se puede realizar un viaje largo, y además los output de errores funcionan correctamente, explicando

por qué no se pueden generar los viajes (además se ve visualmente que falta, lo que es un buen añadido).

6.5. Prueba 5

6.5.1. Presentación del juego de prueba

En este juego de prueba nos encontramos a una pareja mayor que no quiere ir en avión, quieren hacer un viaje romántico de corta duración.

6.5.2. ¿Qué pretendemos probar?

El objetivo principal de este juego de prueba es comprobar que se pueden general viajes que no empleen aviones como método de transporte, ya que todas las ciudades están comunicadas por avión, pero solo unas pocas están comunicadas por tren o barco.

6.5.3. Input-output

CLIPS input

```
Escriba las edades de los participantes separados por espacios:
Su edad es: (60 58)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 4 dias y maximo 6 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 1 dias por ciudad y maximo 2 dias por ciuda
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 1 ciudades y maximo 2 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 10000€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte (avion)
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 1
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: calidad
```

CLIPS output

```
No se puede programar un viaje debido a los transportes
   ---------------------------------------------------------------------------------------------
-                                       PRIMER VIAJE                                          -
---------------------------------------------------------------------------------------------


Duracion del viaje: 4 dias
```

```
Ciudades visitadas: Nueva_York(2), Venecia(2),

Visitas: Visita_MET, Visita_Central_Park, Nueva_York, Paseo_en_gondola, Murano_Burano_Torcello, Cena_junto

Alojamientos: SpringHill_Suites, Hotel_Carlton_On_The_Grand_Canal,

Transporte:

Coste total del viaje: 1300

El viaje no puede ser del mismo continente
-------------------------------------------------------------------------------------------
-                                        SEGUNDO VIAJE                                    -
-------------------------------------------------------------------------------------------

Duracion del viaje: 4 dias

Ciudades visitadas: Granada(2), Praga(2),

Visitas: Visita_alhambra, Murano_Burano_Torcello, Cena_en_el_centro, Granada, Visita_al_castillo, Praga,

Alojamientos: Hotel_Oya, Candil_suite_Realejo,

Transporte: PRA_GRA_tren,

Coste total del viaje: 722
```

6.5.4. Conclusiones extraídas de la prueba

A partir de los resultados extraemos las siguientes conclusiones, que el primer viaje generado no es posible, ya que no hay transporte (solo hay aviones entre Venecia y Nueva York) y se infringe la regla sobre viajar a distintos continentes en viajes cortos, tal y como indica el mensaje de error; y que el segundo viaje sí es correcto y emplea el tren como único método de transporte,por lo que sí es posible realizar este tipo de viaje, aunque hay algún fallo en el código.

6.6. Prueba 6

6.6.1. Presentación del juego de prueba

En este caso, nuestros usuarios son un profesor de universidad y uno de sus alumnos. Quieren hacer un viaje cultural para realizar un proyecto académico. Este viaje durará 10 días pero se limitará a una única ciudad.

6.6.2. ¿Qué pretendemos probar?

Este viaje no debe tener transporte. Además esperamos que se realicen todas las actividades posibles.

6.6.3. Input-output obtenido

CLIPS input

```
Escriba las edades de los participantes separados por espacios:
Su edad es: (47 20)
¿Que tipo de viaje se quiere realizar? (descanso diversion romantico trabajo aventura cultural) Se realiza
¿Cuál es el minimo de dias que quereis de viaje?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis de viaje?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 10 dias y maximo 10 dias
¿Cuál es el minimo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: ¿Cuál es el maximo de dias que quereis estar en cada ciudad?
Introduzca su respuesta: Dias seleccionados, se viajará minimo 10 dias por ciudad y maximo 10 dias por ciu
¿Cuál es el minimo de ciudades que quereis visitar?
Introduzca su respuesta: ¿Cuál es el maximo de ciudades que quereis visitar?
Introduzca su respuesta: Dias seleccionados, se viajará a minimo 1 ciudades y maximo 1 ciudades
De cuanto presupuesto disponeis (en euros €)
Introduzca su respuesta: Su presupuesto es de 10000€
Que medios de transporte se desean evitar:
Se intentaran evitar los siguientes medios de transporte (avion)
Que calidad de alojamiento se prefiere (de minimo)
Introduzca su respuesta: Se buscaran alojamientos de calidad: 1
¿Prefiere visitar ciudades/lugares poco conocidos? (si no s n) Respuesta TRUE
¿Prefiere duracion, calidad o un mixto? (duracion calidad mixto) Se priorizara la: calidad
```

CLIPS output

```
      ---------------------------------------------------------------------------------------------
-                                       PRIMER VIAJE                                          -
---------------------------------------------------------------------------------------------

Duracion del viaje: 10 dias

Ciudades visitadas: Paris(10),

Visitas: Visita_Louvre, Paseo_Campos_Eliseos, Paris,

Alojamientos: Hotel_Darcet,

Transporte:

Coste total del viaje: 1417

---------------------------------------------------------------------------------------------
-                                       SEGUNDO VIAJE                                         -
---------------------------------------------------------------------------------------------
```

```
Duracion del viaje: 10 dias

Ciudades visitadas: Venecia(10),

Visitas: Paseo_en_gondola, Murano_Burano_Torcello, Cena_junto_al_canal, Venecia,

Alojamientos: Hotel_Carlton_On_The_Grand_Canal,

Transporte:

Coste total del viaje: 2195
```

6.6.4. Conclusiones extraídas de la prueba

Podemos ver en el output que el programa ha tenido el comportamiento esperado, ha seleccionado una única ciudad, ningún transporte y varias actividades de la ciudad visitada.

6.7. Conclusiones juegos de pruebas

Después de haber realizado todos los juegos de pruebas, hemos podido probar el correcto funcionamiento de nuestro programa en diversos aspectos, esto también nos ha hecho ver ámbitos donde podríamos haber implementado mejoras para nuestro SBC. Sobre todo nos hemos dado cuenta de que con casi 5000 líneas de instancias no es suficiente para que el sistema funcione con su máximo potencial, lo que nos permite poner en perspectiva cuántos datos requeriría un sistema realista para su correcto funcionamiento. Aun así todo nuestro sistema se puede probar para casos normales (no viajes de 300 días) y funciona correctamente.

7. Conclusiones de la práctica

Después de haber realizado la práctica de Sistemas Basados en el Conocimiento extraemos diversas conclusiones. La primera de ellas es lo potentes que son este tipo de sistemas, ya que con el correcto planteamiento e implementación se pueden llegar a crear sistemas que nos dan soluciones a problemas de gran complejidad.
La segunda es la importancia de programar de manera declarativa y usando la unificación de variables en CLIPS, si hubiéramos realizado todo el código de manera imperativa habríamos obtenido un producto de mucha menos calidad y la implementación habría sido más lenta. También el tiempo de ejecución se habría visto afectado
La tercera es ver lo importante que es dotar de mucha cantidad de información al sistema para que así obtenga mejores resultados, nuestras instancias podrían parecer excesivas pero son una ínfima parte de los datos que manejan las agencias de viajes reales.
La cuarta es que este ha sido un punto de entrada duro al mundo de los SBC, ya que consideramos que programar en CLIPS por primera vez para crear un proyecto de este tipo es una toma de contacto bastante complicada. El lenguaje tiene un algoritmo de unificación muy potente y es capaz de muchas cosas. Aun así, la poca cantidad de documentación disponible y la extraña y la curva de aprendizaje del lenguaje no lo han puesto nada fácil al principio. Visto en perspectiva, CLIPS es un lenguaje potente pero nuestra falta de conocimiento nos ha impedido explotar sus características al máximo.