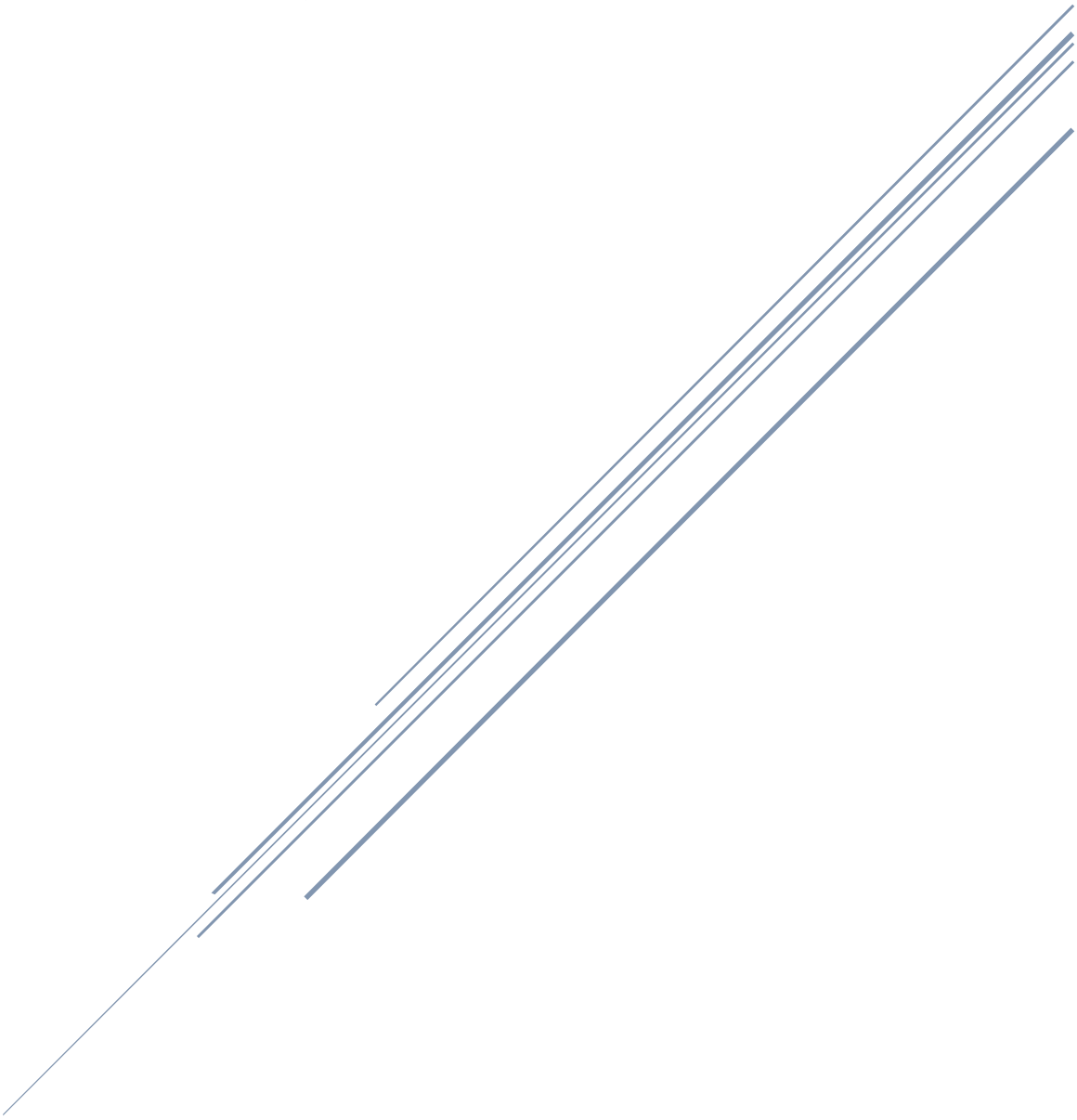
INGENIERIA DEL CONOCIMIENTO

PRÁCTICA 2, 4º curso

Universidad Carlos III Madrid, Ingeniería Informática, 2021



grupo 80 - Colmenarejo - IVAN AGUADO PERULERO – 100405871
grupo 80 - Colmenarejo - JORGE SERRANO PÉREZ – 100405987

Contenido

1.- INTRODUCCIÓN 2

2.- MANUAL TÉCNICO 2

2.1.- OBJETOS 2

2.2.- PREDICADOS 3

2.3.- FUNCIONES 4

2.4.- ACCIONES 4

2.5.- MÉTRICA 5

3.- PRUEBAS REALIZADAS 6

4.- CONCLUSIONES 8

1.- INTRODUCCIÓN

El objetivo de esta práctica es modelar el problema del juego LeHavre para que un planificador automático sea capaz de generar simulaciones de partidas. Este juego consiste en ser el jugador que más dinero acumule, ya sea en metálico o en propiedades, mediante la colocación de trabajadores y la construcción de distintos edificios.

En cuanto al presente documento, en primer lugar, encontramos una introducción. A continuación, un manual técnico en el que se explicará el código implementado explicando las acciones y los predicados. Lo siguiente será otra sección para describir las pruebas que se han realizado para verificar su correcto funcionamiento y un análisis de los resultados obtenidos. Por último, se justificarán las conclusiones técnicas extraídas durante la realización de la práctica, así como una comparación con la práctica anterior, distintos comentarios personales y problemas encontrados.

2.- MANUAL TÉCNICO

En este apartado se realizará una descripción detallada del código implementado, tanto las acciones como los predicados. La solución trata sobre una aproximación al juego Le Havre con distintas relajaciones para facilitar su programación.

2.1.- OBJETOS

Para el uso de predicados y acciones se han definido los objetos *jugador*, *casilla*, *edificio*, *bien*, *oferta*, *oferta_disponible* y *ronda*. No hemos considerado necesario establecer una jerarquía, es decir, definir distintas subclases de cada objeto mencionado anteriormente. En su lugar, hemos empleado distintas constantes para separar mejor y limitar los predicados y las acciones posibles.

Los tipos con sus constantes se definen a continuación:

- ***jugador***: este tipo permite diferenciar los distintos jugadores, aunque en este caso tan solo es necesario uno solo.
- ***casilla***: este tipo identifica las distintas casillas del juego, que se corresponden con el turno de los jugadores. Tiene asociada una oferta en cada caso y hay un total de 7, que hemos declarado como constantes: C1, C2, C3, C4, C5, C6 y C7.
- ***edificio***: este tipo identifica los distintos edificios que se podrán construir y utilizar en el juego. Hemos elegido una pequeña muestra de 6 edificios para simplificar el problema y no aumentar en exceso el número de acciones. Se han creado las constantes correspondientes a los nombres que reciben estos edificios: B1, B2, B3, HUNTINGLODGE, WOOD, CLAYMOUND, ARTSCENTRE y NINGUNO.
- ***bien***: este tipo engloba todos los recursos que pueden utilizar los jugadores. Estos pueden ser: TRIGO, VACA, ARCILLA, MADERA, ACERO, PIEL, CARBÓN, PESCADO y FRANCOS. Destacamos que en este caso podríamos haber creado una subclase que

fuera comida, pero decidimos no hacerlo, como decisión de diseño, porque más tarde nos facilitaría la parte de finalizar la ronda y tener que alimentar.

- ***oferta***: este tipo se corresponde con cada una de las casillas que hay y tiene asociados una serie de bienes que luego se añaden a las ofertas disponibles, que son las que el jugador puede tomar. Sus constantes son: OFERTA1, OFERTA2, OFERTA3, OFERTA4, OFERTA5, OFERTA6 y OFERTA7.
- ***oferta_disponible***: este tipo identifica y asocia cada bien con su correspondiente oferta. Por lo tanto, sus constantes asociadas son: OFERTA-TRIGO, OFERTA-VACA, OFERTA-ARCILLA, OFERTA-MADERA, OFERTA-PESCADO, OFERTA-ACERO, OFERTA-PIEL, OFERTA-CARBÓN y OFERTA-FRANCOS.
- ***ronda***: este tipo identifica las distintas rondas de las que consta la partida. En este caso, al haber solo un jugador, serán un total de 7: R1, R2, R3, R4, R5, R6 y R7.

2.2.- PREDICADOS

En cuanto a los predicados, estos se corresponden con los que no reciben valores numéricos, y se asemejan en cierta medida a los hechos de CLIPS. Los predicados con variables son los siguientes:

- ***(jugadorEn ?j - jugador ?c - casilla)***: indica cuál es la casilla y, por lo tanto, el turno en el que se encuentra el jugador.
- ***(jugadorTieneEdificio ?j - jugador ?e - edificio)***: indica si un jugador ha comprado el edificio y, por lo tanto, no tiene que pagar el coste de entrada para realizar la acción.
- ***(ofertaEn ?o - oferta ?c - casilla)***: indica la relación entre una casilla y su correspondiente oferta.
- ***(ofertaConInteres ?o - oferta)***: indica si una oferta tiene interés, en cuyo caso los jugadores que tengan préstamos deberán pagar 1 franco.
- ***(bienAsociado ?bien - bien ?oferta - oferta)***: indica los bienes que se corresponden a una oferta.
- ***(edificioConstruido ?e - edificio)***: indica si un edificio ha sido construido.
- ***(edificioOcupado ?e - edificio)***: indica
- ***(sinCosteConstruccion ?e - edificio)***: indica que un edificio no tiene coste de construcción, por lo que el jugador no tendrá que pagar nada.
- ***(rondaConCosecha ?r - ronda)***: indica si una ronda tiene cosecha o no, en cuyo caso los jugadores que cumplan unas condiciones reciben algunos bienes adicionales.
- ***(rondaActual ?r - ronda)***: indica la ronda que se está jugando en el momento.
- ***(siguiente-casilla ?c1 ?c2 - casilla)***: indica cuál es la siguiente casilla a la que hay que pasar al terminar un turno.
- ***(siguiente-ronda ?r1 ?r2 - ronda)***: indica cuál es la siguiente ronda a la que hay que pasar al terminar el séptimo turno.
- ***(rondaFinal ?r - ronda)***: indica cuál es la ronda final.

Los que no tienen variables se utilizan principalmente para dirigir el juego en las distintas fases existentes:

- ***(fase1-jugada)***: indica que la fase en la que se añaden los bienes asociados a una casilla (oferta) a su oferta correspondiente se ha completado.
- ***(fase2-jugada)***: indica que la fase en la que el jugador elige su acción principal (tomar los bienes de una oferta o mover su ficha persona para usar un edificio) se ha completado.
- ***(hayQueConstruir)***: se utiliza cuando un jugador ha movido su ficha de persona a uno de los edificios iniciales (B1, B2 o B3), cuya acción es construir un nuevo edificio. Sirve para realizar esa construcción en el momento adecuado.
- ***(faseOpcional-jugada)***: indica que la fase en la que el jugador puede comprar o vender edificios, así como pagar préstamos o no hacer ninguna de las anteriores, se ha completado.
- ***(fin-del-juego)***: se utiliza para finalizar el juego una vez se ha alcanzado la ronda final.

2.3.- FUNCIONES

A continuación, describiremos las funciones, que se utilizan para almacenar, operar y comparar todos aquellos valores numéricos que se deban tener en cuenta. Son las siguientes:

- ***(prestamos)***: indica el número de préstamos que tiene el jugador.
- ***(uds_bien_jugador ?b - bien)***: indica el número de unidades que tiene el jugador de un bien concreto.
- ***(uds_bien_oferta ?b - bien ?o - oferta)***: indica el número de bienes de un tipo que tiene asociado una oferta.
- ***(uds_bien_oferta_disponible ?b - bien ?o - oferta_disponible)***: indica el número de bienes de un tipo que tiene asociado una oferta disponible.
- ***(uds_coste_construccion ?e - edificio ?b - bien)***: indica el número de bienes de un tipo que tiene que pagar el jugador para construir un edificio.
- ***(coste_entrada ?e - edificio)***: indica el coste que tiene que pagar un jugador para poder utilizar un edificio.
- ***(valor ?e - edificio)***: indica el valor que tiene un edificio para realizar el recuento final, y se corresponde también con su precio de compra.
- ***(uds_accion_edificio ?e - edificio ?b - bien)***: indica el número de bienes de un tipo que proporciona un edificio al utilizarlo.
- ***(uds_comida_a_pagar ?r - ronda)***: indica la cantidad de comida que tiene que pagar el jugador al terminar una ronda concreta.
- ***(total-cost)***: función que representa la métrica que hemos elegido para el planificador. Se incrementa en función de las acciones que se vayan eligiendo, es decir, se asigna un coste menor a aquellas acciones que, de acuerdo a nuestra estrategia, son mejores de acuerdo a la puntuación y el objetivo final del juego.

2.4.- ACCIONES

Finalmente, describiremos las distintas acciones posibles a realizar para la correcta ejecución del juego.

- ***fase1-oferta***: se corresponde con la primera fase del juego, en la que se añaden los bienes correspondientes de la casilla (oferta) a la oferta disponible correspondiente.
- ***fase2-tomarBienes***: es una de las acciones principales obligatorias, en la que el jugador elige tomar los bienes de una de las ofertas disponibles.
- ***fase2-moverPersona***: es la otra acción principal obligatoria, en la que el jugador decide mover su ficha de persona para utilizar un edificio, es decir, ejecutar su acción. No se pueden ejecutar esta acción y la anterior simultáneamente.
- ***fase2-construir***: esta acción se ejecuta en el caso de que el jugador se mueva a uno de los edificios iniciales (B1, B2 y B3). Al moverse se activa un predicado para ejecutar inmediatamente esta acción. Tan solo construye el edificio en cuestión pagando el costo indicado.
- ***faseOpcional-comprar***: se corresponde con una acción opcional a realizar durante la fase 2 o fase de acciones principales. Consiste en comprar un edificio para ahorrar los costes de realizar la acción en ese edificio.
- ***faseOpcional-vender***: se corresponde con otra acción opcional a realizar durante la fase 2, y consiste en vender un edificio del que el jugador es propietario para obtener un dinero para pagar al final de la ronda en vez de tener que pedir préstamos.
- ***faseOpcional-pagarPrestamo***: se corresponde con otra acción opcional a realizar durante la fase 2, y consiste en pagar un préstamo, en caso de tenerlo, ya que estos perjudican la puntuación final.
- ***faseOpcional-accionesNoRentables***: esta acción no tiene ningún efecto como tal, se ejecuta en caso de que no sea interesante elegir cualquiera de las otras acciones de la fase opcional.
- ***cambiarCasilla***: es la acción que se ejecuta cuando el jugador ya ha ejecutado todas las acciones que quiere y finaliza su turno. Por tanto, se pasa a la siguiente casilla y se reinician los predicados de control necesarios.
- ***cambiarRonda***: es la acción que se ejecuta cuando se termina el séptimo turno, el de la casilla 7. Si la ronda tenía cosecha se realiza, y a continuación el jugador debe pagar la comida necesaria para alimentar a sus trabajadores o pedir préstamos si no tuviera suficiente. Finalmente, se establece que el jugador vuelve a empezar en la primera casilla y se reinician los predicados de control.
- ***fin-del-juego***: es la acción que se ejecuta cuando se finaliza el juego, es decir, se termina de jugar la séptima ronda.

2.5.- MÉTRICA

Para medir la calidad de los planes generados se ha utilizado la función *total-cost* como métrica. Esta función se debe minimizar, por lo tanto la aumentaremos 1, cuando sea una acción estratégicamente buena, 2 cuando no sea ni buena ni mala y 3 cuando esa acción no beneficia al jugador para conseguir la máxima riqueza a lo largo de la partida. No se añadirá esta funcionalidad en las acciones que se tengan que realizar de forma obligatoria.

A continuación, se expondrán las diferentes estrategias implementadas separadas por acciones.

- ***fase2-tomarBienes***: se ha premiado con un coste de 1 coger francos cuando se han acumulado más de 5 en la oferta. Esto se debe a que los francos son el bien más preciado, con ellos se puede pagar la alimentación de final de ronda, entrar y comprar edificios y sobre todo se tienen en cuenta a la hora de calcular la riqueza total del jugador. Por otro lado, con un coste de 2 se encuentra el tomar bienes de tipo comida. Estos también son importantes, pues al acabar cada ronda son necesarios para alimentar, si no tenemos comida debemos gastar los preciados francos. Por último, con un coste de 3 están el resto de bienes, los cuales en esta versión simplificada del juego son menos importantes.
- ***fase2-moverPersona***: se premia si se mueve a un edificio que tiene en su posesión, pues así no tiene que pagar el coste de entrada. Este movimiento incrementa el coste en 1, frente a 3 de moverse a un edificio que no ha comprado previamente.
- ***fase2-construir***: se ha premiado construir los edificios cuya acción provee francos y comida con un coste de 1, frente a un coste de 3 para construir los que brindan otro tipo de bienes. Esto es debido al mismo motivo expuesto en tomar bienes.
- ***faseOpcional-(comprar/vender/pagarPrestamos/accionesNoRentables)***: dentro de la fase opcional comprar y pagar préstamos tienen un coste de 1. Ambas son acciones bastante buenas estratégicamente. La primera porque a la larga acaba saliendo más barato desplazarte a un edificio comprado que pagar su coste de entrada cada vez que entras en él. En cuanto a los préstamos, si se tienen francos suficientes mejor pagarlos antes de finalizar la partida, pues si esta acaba se te descontará 7 de la riqueza total frente a 5 que cuesta pagarlos. Como acción con el máximo coste dentro de esta fase opcional se encuentra el vender un edificio, pues como se ha mencionado anteriormente ahorras más teniéndolo en tu posesión, y además al venderlo solo obtienes la mitad de lo que pagaste por él. Por último, en la acción para las acciones no rentables el coste es de 2, pues es mejor no hacer nada que vender un edificio en la mayoría de los casos.

3.- PRUEBAS REALIZADAS

Las pruebas se harán de manera secuencial, primero comenzaremos con una ejecución simple del juego y a medida que se vayan superando las pruebas aumentaremos la complejidad, añadiendo nuevas funcionalidades y manteniendo las anteriores. Se han separado de la siguiente manera:

problema-1: menos rondas

Este problema refleja una situación en la que el número de rondas es menor que lo establecido según las reglas para la partida en solitario.

problema-2: sin edificios

La complejidad se reduce quitando los edificios y dejando como única acción la de tomar bienes en la fase 2 del turno.

problema-3: sin intereses ni cosecha

Se prueba el problema sin añadir intereses en las ofertas ni cosecha al final de la ronda, reduciendo la complejidad para observar si influye en el coste y el tiempo de ejecución.

problema-4: completo

Se prueba a ejecutar el problema al completo con todas las funcionalidades implementadas.

problema-5: cambiando ofertas de sitio

Se cambia el orden de las ofertas para ver cómo afecta en la búsqueda de planes.

En la tabla se han incluido los datos del último plan obtenido, es decir, el mejor en cuanto a coste.

Prueba	Con métrica			Sin métrica		
	Acciones	Tiempo	Coste	Acciones	Tiempo	Coste
1	113	3.00	75.00	113	0.34	113.00
2	195	0.02	164.00	195	0.00	195.00
3	197	13.06	132.00	197	0.94	197.00
4	197	15.51	132.00	198	0.53	198.00
5	199	2.40	134.00	198	86.06	198.00

En la primera prueba, podemos observar en los resultados que se nota la reducción de rondas, ya que tanto el número de acciones como el coste es más bajo que en el resto de casos, lo cual es lógico y razonable. Vemos también la importancia de hacer uso de una métrica correcta, ya que disminuye el coste en gran medida.

En la segunda prueba, observamos que los edificios son una parte muy importante del juego, ya que al quitarlos se reduce drásticamente el tiempo de ejecución. Esto se debe a que dos de las acciones opcionales ya no se contemplan, por lo que disminuye el tamaño del árbol de búsqueda.

En la tercera prueba, podemos observar de nuevo la gran importancia del hacer uso de métricas adecuadas, ya que se reduce mucho el coste en comparación con aquella que no utiliza métrica. La única ventaja que podría tener el no utilizarla sería el reducir el tiempo de ejecución.

Por último, en la quinta prueba se puede observar que al cambiar las ofertas de orden se consigue una ejecución más rápida, pero con mayor coste en el caso de haber utilizado la métrica. Sin embargo, al no utilizarla, la búsqueda del mejor plan se vuelve bastante más lenta.

4.- CONCLUSIONES

Tras finalizar la práctica hemos podido comprobar que *pddl* es, en nuestra opinión, un lenguaje más simple e intuitivo que CLIPS. Sin embargo, este limita las funcionalidades que podemos implementar.

El principal problema que hemos encontrado en este proyecto ha sido encontrar cual era nuestro error cuando cometíamos alguno. Esto es debido a que si una de las acciones impedía la correcta ejecución del programa no saltaba un error especificando que la acción x falla, sino un error general. Al no haber prints encontrar el error se complicaba, por lo que tuvimos que ir depurando, añadiendo y quitando acciones, precondiciones y efectos hasta encontrar el problema.

Por otro lado, consideramos que la implementación de la estrategia es bastante más simple que en la práctica anterior, pues el uso de una métrica facilita mucho las cosas.

En definitiva, una práctica que junto con la otra sirven para tener una visión general de las herramientas existentes para resolver este tipo de problemas. Esta visión te permitirá en un futuro saber cuál elegir dependiendo del desafío al que te enfrentes.