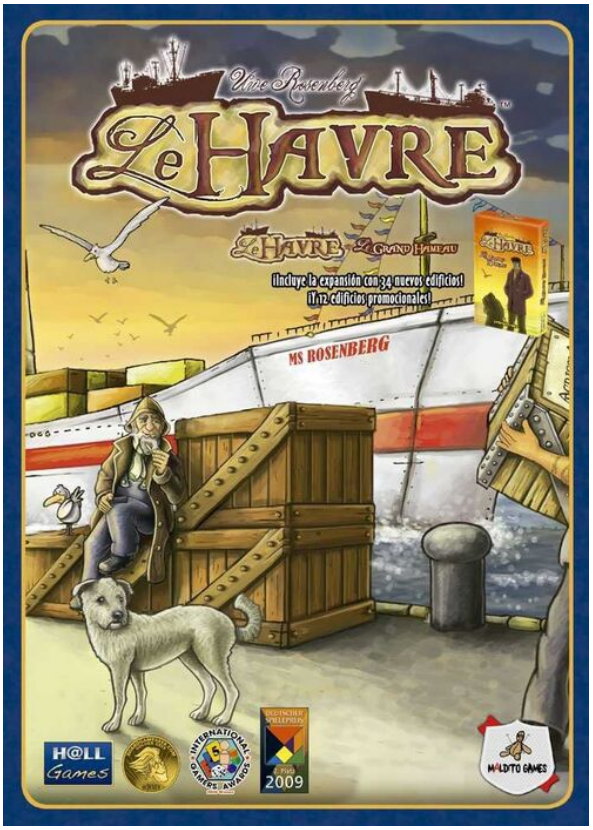
uc3m | Universidad Carlos III de Madrid

Sistema Basado en el Conocimiento para el juego Le Havre con *CLIPS*



Noviembre 2021

Ingeniería del Conocimiento

Práctica I

Diego Fernández Sebastián
100387203

Ricardo Grande Cros
100386336

Doble Grado Ingeniería Informática y Administración de Empresas

Tabla de Contenidos

1. Introducción

En este documento se presenta la realización de la práctica 1 de la asignatura Ingeniería del Conocimiento, dedicada a la implementación del juego Le Havre en CLIPS a través de sistemas basados en el conocimiento. En concreto se ha implementado una "partida corta" del juego para 2 jugadores de la versión en *Java* creada por Grzegorz Kobiela[1].

En primera instancia se hace alusión a la metodología CommonKADS, la cual es el estándar empleado para la implementación de Sistemas Basados en el Conocimiento. En este primer apartado se justifica el enfoque dado al problema a través de esta metodología.

A continuación se expone un manual técnico que pretende explicar todo el conocimiento existente en el dominio del problema. Este manual principalmente está enfocado a la explicación del código en CLIPS implementado. Aunque también se describe el conocimiento de inferencia y de tareas que dan lugar a la estrategia implementada para resolver el problema.

Seguidamente se presenta un manual de usuario con la información sobre cómo ejecutar el programa y cómo modificar la situación inicial de la partida para producir distintos escenarios. Tras los manuales, se justificará el funcionamiento del sistema a través de una batería de pruebas realizadas. Por último, se exponen unas conclusiones técnicas sobre el desarrollo de la práctica y unos comentarios finales acerca de la misma.

[1] enlace disponible en:https://boardgamegeek.com/user/Ponton

2. Aplicación Metodología CommonKADS

El objetivo principal del presente documento consiste en construir un Sistema Basado en el Conocimiento multijugador para el juego de mesa Le Havre aplicando la metodología *CommonKADS*.

Recordar que las fases de construcción de un SBC difieren de los modelos de Ingeniería de Software tradicionales. El proceso de adquisición del conocimiento se lleva a cabo gradualmente durante todas las etapas del proceso de construcción.

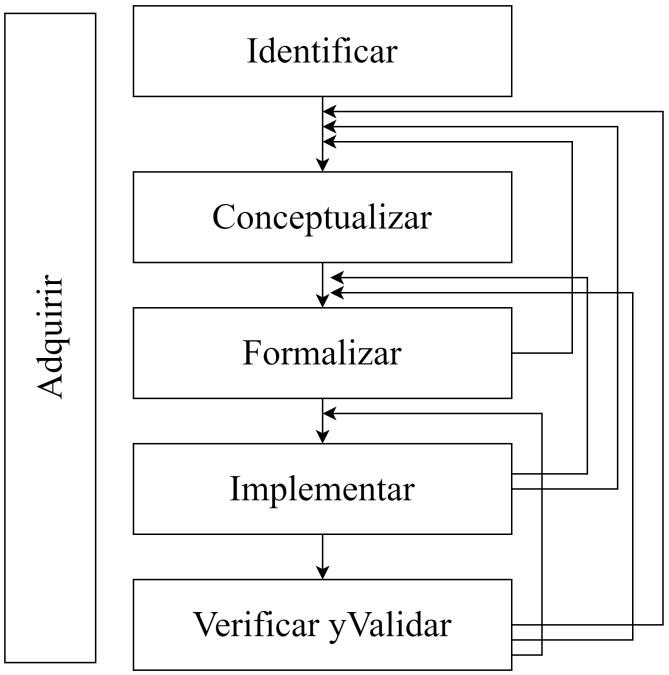


Figura I - Fases Construcción Sistema Experto

En anteriores entregas se han presentado la identificación del problema, una primera aproximación a la conceptualización y una técnica de adquisición de conocimiento. En esta entrega se pretende abordar la conceptualización del conocimiento, la formalización del mismo y su posterior implementación con sus correspondientes pruebas para validar y verificar el sistema experto producido aplicando la metodología *CommonKADS*.

CommonKADS es una metodología de construcción de sistemas basados en el conocimiento y actualmente es considerada el estándar para la Ingeniería del Conocimiento y de los Sistemas Basados en el Conocimiento.

CommonKADS subdivide la construcción de un sistema experto en una serie de modelos de comportamiento que reflejan diferentes puntos de vista del conocimiento inmerso en un problema. Los modelos de *CommonKADS* son: modelo de la organización, de tareas, de agentes, de conocimiento, de comunicaciones y del diseño.

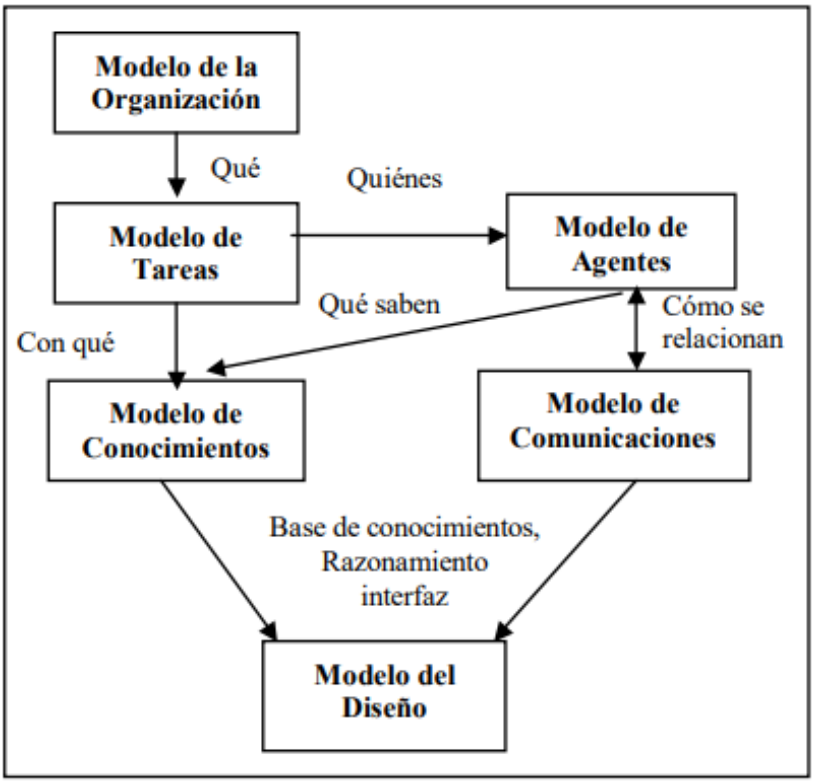


Figura II - Modelos de CommonKADS

Aunque este documento está centrado principalmente en el modelo de Conocimiento habrá que tener en cuenta aquellos actores que se prevén que interactúen en el sistema experto, es decir, el modelo de agentes. En concreto, al tratarse de un sistema multijugador cerrado para dos jugadores se ha previsto que habrá dos agentes. Estos serán los jugadores desarrollados para el sistema experto y serán los encargados de ejecutar las tareas. Sin embargo, no habrá ningún agente externo.

Dado que el objetivo principal es obtener un sistema experto capaz de obtener resultados decentes para el juego Le Havre, en el modelo de tareas se han obtenido aquellos procesos encaminados a alcanzar este objetivo. Las tareas obtenidas son:

- Obtener edificios
- Obtener barcos
- Pagar deudas
- Transformar recursos
- Mantener derecho de cosecha
- Coger recurso de la oferta.
- Comerciar en la Compañía Naviera
- Pagar Demanda Comida
- Cambiar decisión del recurso alimenticio empleado para pagar tarifas de entrada

El modelo de conocimiento pretende explicar en detalle todo tipo de conocimiento empleado en la realización de una tarea. En *CommonKADS* el conocimiento queda segregado en tres niveles.

- *Conocimiento del dominio*: aquel conocimiento explícito del dominio de la aplicación. En este caso, será representado por la ontología propuesta.
- *Conocimiento de inferencia*: aquel conocimiento abstracto que permite vincular una solución con el conocimiento del dominio. Son pasos de razonamiento básicos. En este caso, serán

representados con deseos, es decir, dada una situación, el jugador generará una serie de deseos para llevar a cabo una acción.

- *Conocimiento de la tarea*: aquel conocimiento que refleja cómo alcanzar las metas de la solución del problema. Tiene como objetivo especificar los pasos de inferencia básicos. En este caso, será la estrategia del jugador propuesta.

Para la representación del conocimiento de tareas, *CommonKADS* proporciona una serie de plantillas abstractas reutilizables divididas en tareas analíticas y sintéticas. La propuesta encaja principalmente en la tarea *analítica de valoración*, ya que el jugador tendrá unos criterios específicos con los que valorará sus alternativas y decidirá qué acción tomar durante su turno. Aunque también puede incluirse en la categoría de *planificación*. Para valorar alguna de sus alternativas, primero realizará subactividades parcialmente ordenadas para determinar si le interesa o no.

3. Manual técnico

El manual técnico pretende sintetizar la formalización e implementación del conocimiento llevado a cabo. En esta sección se presentan los distintos tipos de conocimiento existentes siguiendo el modelo de Conocimiento de *CommonKADS*.

3.1. Conocimiento del Dominio

El conocimiento del dominio queda representado a través de la ontología. En este apartado se describe la ontología empleada como representación del dominio del problema. Todos los conceptos usados fueron extraídos durante las sesiones de adquisición del conocimiento.

3.1.1. Clases de la Ontología

Recurso

Los recursos del juego son: *FRANCO, MADERA, PESCADO, ARCILLA, HIERRO, GRANO, GANADO, CARBÓN, PIEL, PESCADO AHUMADO, CARBÓN VEGETAL, LADRILLOS, ACERO, PAN, CARNE COQUE* y *CUERO*.
Además, tendrán un atributo para representar el precio unitario al que puede venderse el recurso en la Compañía Naviera.

Recurso Alimenticio

Existen algunos recursos cuya finalidad principal es satisfacer la demanda de comida al finalizar cada ronda. Nótese que no todos los recursos con este fin generan las mismas unidades de comida por unidad. Por ende, esta subclase de *RECURSO* permite representar la cantidad de comida que genera un determinado recurso a través de su correspondiente atributo.

Bonus

Algunas cartas del juego otorgan a quien las posee una bonificación al usar ciertos edificios. Esta bonificación puede ser de dos tipos: *pescador* o *martillo*. La bonificación, o *bonus* a partir de ahora, incrementa las unidades de recursos obtenidas al usar los edificios que se ven afectados por esta característica.

Los edificios que otorgan bonus por su posesión son:

Bonus *Pescador*
- Piscifactoría
- Ahumador
- Compañía naviera

Bonus *Martillo*
- Mina de carbón
- Montículo de arcilla
- Herrería
- Constructora 1, 2 y 3

Además, los edificios que otorgan más cantidad de recursos de salida cuando son usados son:
- Piscifactoría y montículo de arcilla: otorgan una unidad adicional por cada bonus en propiedad (*Pescador* o *Martillo* respectivamente).
- Mina de carbón: sólo otorga una unidad adicional si se posee un *Martillo*.

Mazo

Realiza la misma función que los mazos en el juego físico. Permite representar una agrupación de cartas bajo el mismo *id*. Cuenta con dos atributos:
- ID mazo: permite identificar el mazo.
    1. Mazo de edificios 1
    2. Mazo de edificios 2
    3. Mazo de edificios 3
    4. Mazo de barcos de madera
    5. Mazo de barcos de hierro
    6. Mazo de barcos de acero
    7. Mazo de barcos de lujo
- Número de cartas en mazo: contador de cartas en el mazo. Se actualizará en las reglas según entren o salgan cartas del mazo.

Ronda

Representa las rondas del juego y sus atributos más relevantes. Estos son:
- Nombre ronda
- Coste comida: la demanda de comida a cubrir al finalizar la ronda.
- Hay cosecha: atributo booleano que indica si al final de la ronda se deben entregar recursos a los jugadores. Los recursos son: 1 de grano si el jugador tiene al menos 1 de grano y 1 de ganado si el jugador tiene al menos 2 de ganado.

Participante

Aunque físicamente no exista un jugador humano centrado exclusivamente en dirigir el correcto flujo del juego a través del *AYUNTAMIENTO*; son los propios jugadores humanos los encargados de simular las acciones del mismo, será necesario poder representar las relaciones que este tiene con las cartas y los recursos.

Por tanto, a la hora de modelar este comportamiento se ha visto necesario tener una clase que represente a todos los participantes activos del juego. Esta clase tendrá únicamente como atributo un nombre identificativo y descriptivo para cada participante.

Jugador

Subclase de *PARTICIPANTE*, representa a los jugadores humanos participantes dentro del juego. Están definidos por:

- Nombre
- Deudas: contador de préstamos que el jugador va acumulando.
- Número de barcos que el jugador posee.
- Capacidad de envío: según los barcos que posea, esta capacidad aumenta.
    - +2 por cada barco de madera.
    - +3 por cada barco de hierro.
    - +4 por cada barco de acero.
- Demanda de comida cubierta: poseer barcos reduce la cantidad de comida a pagar al finalizar una ronda.
    - -4 por cada barco de madera
    - -5 por cada barco de hierro
    - -7 por cada barco de acero
- Riqueza: indica la riqueza en términos de barcos y cartas que el jugador dispone en todo momento.

Carta

Las cartas contienen los siguientes atributos:

- Nombre
- Tipo: un enumerado con los valores {*BASICO, INDUSTRIAL, COMERCIAL, BARCO, NINGUNO*}
- Valor que otorga por poseerla. Además, en la mayoría de los casos coincide con su coste de compra.

Carta Edificio Generador

Para los edificios generadores se hizo necesario implementar una clase más específica, que hereda los atributos de *CARTA* pero además implementa:

- Número de recursos de salida: indica cuántos recursos distintos de salida ofrece la carta.

Carta Banco

Del mismo modo que en la *CARTA EDIFICIO GENERADOR*, para representar al *Banco* se hizo necesaria una clase que implementa herencia de *CARTA*, y que además incluye:

- El tipo se fija en *COMERCIAL*.
- Coste: el Banco no otorga el mismo valor que su coste. De hecho, este último es considerablemente mayor. Por este motivo se ha incluido este atributo y se ha necesitado implementar una clase específica para el caso.

Barco

Los barcos del juego se han modelado a través de una herencia de la clase *CARTA*. Implementan además los siguientes atributos:

- Tipo fijado en *BARCO*
- Coste: mismo motivo que en *Banco*.

- Unidades de comida que genera: cuántas unidades reduce del pago de demanda de comida del final de la ronda.
- Capacidad de envío: cantidad de unidades de recurso que puede enviar cuando se usa el edificio *Compañía Naviera*.

Nótese que el barco lujoso, al no tener los atributos de coste, unidades de comida que genera y capacidad de envío será sencillamente una instancia de la clase *CARTA*.

Loseta

Representa las casillas del tablero por las que se mueven los jugadores. Contiene los siguientes atributos:

- Posición: un valor entero del 0 al 6.
- Visibilidad: indica si está visible o no. Realmente este atributo no tiene mayor utilidad que la de aumentar la fidelidad de la implementación en CLIPS con el juego físico.
- Intereses: atributo booleano que indica si la casilla obliga a pagar intereses cuando un jugador con deudas pasa por ella.

3.1.2. Hechos estructurados

Oferta Recurso

Este hecho estructurado representa la cantidad del recurso ofertado. Advierta que no todos los recursos son ofertados. Sus atributos son:

- recurso: enumerado con los recursos ofertados. *{FRANCO, MADERA, PESCADO, ARCILLA, HIERRO, GRANO, GANADO}*
- cantidad

Barco Disponible

Representa el hecho de que el barco esté disponible para ser construido o comprado. Los barcos empiezan apilados en sus mazos correspondientes pero no podrán ser adquiridos hasta que la ronda lo desbloquee.

Contador Compañía Naviera

Representa el contador empleado para poder iterar de mayor prioridad a menor prioridad los recursos del jugador cuando este ha entrado a la compañía naviera.

3.1.3. Relaciones

Participante Tiene Recurso

Se emplea para enlazar a los participantes con todos los recursos que poseen. Sus atributos son:

- Nombre del jugador.
- Tipo: enumerado que indica el tipo de recurso. Puede tomar los valores. {*COMIDA, DINERO, OTRO*} Atributo empleado para determinar qué recurso puede ser empleado para pagar los costes de comida.
- Recurso: contiene el nombre del recurso.

- Cantidad: valor numérico.

Participante Tiene Carta

Representa la propiedad de un participante sobre una carta. Sus atributos son:

- Nombre del jugador
- Nombre de la carta

Jugador Tiene Bonus

Esta relación permite conocer de qué bonus dispone un jugador.

- Nombre del jugador
- Tipo de bonus
- Cantidad de bonus del tipo acumulados.

Jugador Está en Edificio

Cuando un jugador entra en un edificio, se queda en él hasta que decide entrar en otro. Esta situación supone una restricción importante debido a que, salvo en la última ronda, sólo puede haber un jugador en un edificio. Por este motivo empleamos esta relación, para poder almacenar la localización del jugador en los edificios.

- Nombre edificio: Inicialmente los jugadores no están en ningún edificio, por lo que este valor estará instanciado como una cadena vacía.
- Nombre jugador

Jugador Ha Usado Edificio

Los jugadores pueden permanecer dentro de un edificio, pero éste sólo puede ser usado una vez. Por este motivo, encontramos conveniente emplear una relación que represente si un jugador ha empleado ya un edificio. Sus atributos son:

- Nombre edificio
- Nombre jugador

Aunque esta relación contenga los mismos atributos que la relación *Jugador Está en Edificio*, necesitamos ambas para poder representar e identificar las dos situaciones.

Jugador Está En Loseta

Representa la posición de un jugador en el tablero.

- Nombre del jugador
- Posición de la loseta

Loseta Tiene Recurso

Almacena qué recursos tiene una loseta para poder añadirlos a la oferta de recursos cuando un jugador pase por la loseta. Sus atributos son:

- Posición de la loseta
- Tipo de recurso
- Cantidad del recurso (en nuestro caso siempre es 1 pero se ha modelado para permitir alteraciones en el juego que permitan añadir más de una unidad del mismo recurso.

Carta Tiene Bonus

Permite indicar si una carta contiene un bonus sin necesidad de incluir atributos a las clases ni añadir complejidad a la herencia de *Carta*. Ciertamente son pocas las cartas que contienen bonus, y por eso se decidió emplear una relación implementada como clase.

- Nombre Carta
- Bonus

Carta Pertenece A Mazo

Permite conocer en qué mazo se encuentran las cartas así como su posición dentro del mismo. Sus atributos son:

- ID del mazo.
- Nombre de la carta.
- Posición de la carta en el mazo.

Coste Entrada Carta

Representa el coste de entrar a un edificio. Sus atributos son:

- Nombre de la carta
- Clase de recurso: *COMIDA* o *DINERO*. Es útil representar este atributo para poder dar libertad a las reglas estratégicas en cómo estructurar el pago de comida (pueden incluirse varios tipos de recurso "comida").
- Cantidad

Edificio Input

Modela qué debe introducirse en un edificio generador de recursos para que produzca su output. Los atributos de esta relación son:

- Nombre de la carta
- Recurso
- Cantidad máxima que se puede introducir. Para edificios que no tengan cantidad máxima se ha optado por asignar un número positivo muy elevado, simulando que no hay límite superior.

Edificio Output

Representa los recursos que un edificio generador puede producir. Sus atributos son:

- Nombre de la carta
- Recurso
- Cantidad mínima generada por unidad. Ciertos edificios producen sus salidas siguiendo una proporción en función de la cantidad de entradas. Un ejemplo sería el edificio *Ahumador*, que genera 0.5 francos por cada unidad de pescado introducida.

Carta Output Bonus

Representa los recursos extra que pueden obtenerse en un edificio gracias a la acumulación de bonuses. Sus atributos son:

- nombre carta
- bonus: enumerado representando los tipos de bonus. {*MARTILLO, PESCADOR*}

- Cantidad máxima permitida: No todas las cartas otorgan las mismas unidades de recurso por bonus. Existen cartas dónde sólamente te genera un recurso extra independientemente de la cantidad de bonuses del tipo que se posean (mina de carbón). Por el contrario, otras generarán tantos recursos adicionales como de bonuses disponga el jugador. Este comportamiento será representado haciendo el mínimo entre el número de bonuses y la cantidad máxima permitida por esta relación.

Coste Construcción Carta

Representa el coste de construir un edificio al emplear edificios constructores. Incluye un atributo por cada tipo de recurso empleado para construir cualquier carta {*MADERA, ARCILLA, LADRILLO, HIERRO, ACERO*}, de manera que una instancia contenga toda la información de construcción de un edificio, facilitando así su uso en reglas.

Ronda Introduce Barco

Indica qué rondas añaden barcos al tablero. Sus atributos son:

- Nombre de la ronda
- Nombre de la carta que introducen (en este caso siempre son barcos)

Ronda Asigna Edificio

Al final de ciertas rondas, el juego adjudica un edificio de la parte superior de un mazo al ayuntamiento. Esta relación permite modelar precisamente ese comportamiento, empleando los siguientes atributos:

- Nombre de la ronda
- ID del mazo del cual se tomará la carta para asignársela al ayuntamiento.

3.1.4. Reglas Gestión del Juego

Comienzo Partida

Esta regla será siempre la primera en ejecutarse. En ella se puede especificar si se quieren generar o no trazas para debugear y el fichero donde se guardará el registro de la partida. Además, también será la encargada de determinar el fichero de salida, la fecha de ejecución y el sistema operativo empleado.

3.1.5. Reglas del Juego

A continuación se listan las reglas propias del juego. En este caso por completitud, se han implementado reglas del juego que aunque no sean empleadas por la estrategia diseñada, si son funcionales por si otra persona quisiera determinar una estrategia distinta con el código proporcionado.

Por tanto, aunque en este apartado se expliquen exclusivamente aquellas reglas que si son usadas en la estrategia, el código entregado contiene todas las reglas implementadas. Aquellas no usadas están al final del fichero comentadas para evitar posibles confusiones.

Destapar Loseta

En el juego físico de *Le Havre*, las losetas de recursos están inicialmente tapadas. Durante la primera ronda, cada vez que un jugador avanza a una loseta, esta se destapa, mostrando los recursos que aporta a la oferta.

Mediante esta regla, hemos buscado simular el mismo comportamiento, que aunque no aporte nada "útil" en el desarrollo del juego, sí nos pareció interesante añadir por completitud y por simular fielmente una partida del juego.

La regla simplemente comprueba que la loseta en la que se encuentra un jugador está tapada, y si esto es cierto, la destapa.

Pagar Intereses Francos

Esta regla reduce los francos del jugador si éste tiene préstamos pendientes de pagar y si se encuentra en la loseta que obliga al pago de intereses. Se le reducirán los francos en una unidad, sin importar cuántas deudas tenga.

La regla emplea el semáforo de "fin actividad principal" para que esta regla se ejecute únicamente cuando el jugador ya ha tomado recursos de la oferta o ha entrado a un edificio, de manera que pueda beneficiarse de los beneficios obtenidos por su acción para emplearlos en el pago de intereses.

Además, una vez ha pagado sus intereses se genera un hecho de "intereses pagados" para evitar que la regla entre en bucle (el principio de refacción no lo evita puesto que modificamos los recursos del jugador y por tanto los hechos de las precondiciones cambian).

Pagar Intereses Endeudándose

Es posible que el jugador no tenga francos suficientes para pagar los intereses. Por este motivo hemos creado una regla que aumenta la deuda del jugador en la cantidad necesaria para poder pagar los intereses. Al obtener deudas, los intereses quedan automáticamente pagados.

Actualizar Mazo

Cada vez que se obtiene una carta de un mazo, ya sea por su compra, construcción o porque se le otorga al ayuntamiento al final de una ronda, el mazo debe ser actualizado. Esto se debe a que cada carta del mazo tiene una posición en el mismo (almacenada en la relación *CARTA_PERTENECE_A_MAZO*).

Esta regla emplea un hecho conformado por el *id* del mazo, el número de actualizaciones restantes (se inicializa con el número de cartas en el mazo – 1) y la posición por la que se está actualizando. De esta manera, se ejecuta un bucle en el cual, por cada carta del mazo, a su posición se le resta 1, lo mismo ocurre con el valor de cartas restantes por actualizar en el hecho semáforo *actualizar_mazo*. El hecho semáforo también se modifica aumentando en 1 la posición de la carta por la que se está iterando.

Fin Actualizar Mazo

Para detectar el final del proceso de actualización de mazos descrito en la regla anterior, se ha programado una nueva regla que comprueba si el hecho semáforo *actualizar_mazo* tiene el valor 0 en el campo que indica el número de actualizaciones restantes.

Si esto es así, se elimina el hecho *actualizar_mazo*.

Comprar Edificio Al Ayunto. No Bonus

Las reglas del juego permiten que los jugadores compren edificios al ayuntamiento con francos. Este comportamiento se implementa en esta regla, la cual tiene dos variantes, una en la que el edificio comprado no aporta ningún bonus al jugador y otra en la que sí.

Para el caso en el que el edificio no tiene bonus, simplemente se comprueba que sea el turno del jugador, que haya terminado su acción principal (se puede comprar siempre, independientemente de si el jugador ha tomado recursos de la oferta o ha entrado a un edificio) y que el edificio a comprar sea del ayuntamiento.

Además, se comprueba la existencia de un deseo de comprar el edificio, el cual es generado en las reglas estratégicas.

Cuando las precondiciones se cumplen, se asigna el edificio al jugador, se elimina del ayuntamiento y se reducen los recursos del jugador y su riqueza.

Comprar Edificio Al Ayunto. Si Bonus

Siguiendo la exposición de la regla anterior, en el caso en el que la carta a comprar sí contenga bonus, se debe, además de todo lo explicado anteriormente, añadir al jugador una unidad en el tipo de bonus que corresponda.

Comprar Barco

Los barcos tienen la peculiaridad de que su valor no es igual a su coste de compra. Es por esto que tienen un atributo para cada caso, por lo que no se puede generalizar su compra con las reglas de comprar cartas al mazo (las cartas de edificios sólo tienen el atributo valor).

Su funcionamiento es igual que el de cualquier compra de cartas, salvo porque actualizan los atributos del jugador referentes a los barcos (capacidad de envío, demanda de comida satisfecha y número de barcos).

Comprar Edificio Al Mazo No/Si Bonus

Del mismo modo que se permite comprar cartas al ayuntamiento, el juego también contempla las compras de cartas directamente al mazo. Al igual que con las compras al ayuntamiento, se ha creado una regla para cuando el edificio otorga bonus al jugador y para cuando esto no se cumple.

Además, la diferencia con respecto a la compra de cartas al ayuntamiento es que, tras obtener una carta del mazo, éste debe ser actualizado. Para ello se genera el hecho semáforo *actualizar_mazo*, y se rellena con el identificador del mazo, el número de cartas pendientes por actualizar y en el valor

correspondiente al iterador de cartas en el mazo se introduce el valor 2, para que comience actualizando por la carta que se encuentra en la segunda posición del mazo.

Vender Carta No/Si Bonus

Para vender un edificio, se comprueba que éste pertenezca al jugador y que el mismo tenga un deseo de venderlo (hecho semáforo) que se genera en las reglas estratégicas (en nuestro caso, consideramos que vender edificios nunca es beneficioso, por lo que la parte estratégica no se ha incluido.

Al vender una carta, el jugador obtiene la mitad de su valor y la carta del edificio es entregada al ayuntamiento. Se reduce la riqueza del jugador y se aumenta el valor de su recurso *francos*. Además, si la carta suponía un bonus para el jugador, éste es reducido.

Vender Barco

A diferencia de la venta de edificios, los barcos no se entregan al ayuntamiento tras su venta, sino que se añaden a su mazo correspondiente. Si se trata de un barco de madera, se añadirá al final del mazo de barcos de madera.

La otra peculiaridad de esta regla frente al resto de ventas de cartas es que actualiza los atributos del jugador referentes a los barcos (reduce la capacidad de envío, reduce el número de barcos y reduce la demanda de comida cubierta).

Pagar Deuda

Esta regla permite a los jugadores pagar sus deudas, siempre que tengan alguna pendiente y que tenga el deseo de pagarla (a través del hecho semáforo *deseo_pagar_deuda*). La regla produce el pago de tantas deudas como se indique en el deseo.

Añadir Carta Ayunto. Final Ronda

Cuando finaliza una ronda, se asigna una carta de uno de los mazos al ayuntamiento. El mazo del que se extrae la carta está indicado en la relación instanciada *RONDA_ASIGNA_MAZO*. La regla añade la carta al ayuntamiento y genera el semáforo para que se actualicen los mazos (explicado en la regla Actualizar Mazo).

Pagar Demanda Comida

Al finalizar las rondas, se obliga a los jugadores a realizar un pago de comida. Nos referimos a dicho pago con *demanda de comida*. La demanda de comida puede ser cubierta con cualquier tipo de alimento, queda a disposición del jugador y de su estrategia el decidir con qué recursos se paga.

La regla emplea un hecho semáforo denominado *cambiar_ronda* cuyo valor es *True* cuando el proceso de cambio de ronda se ha iniciado (hay varias reglas que requieren de este semáforo). Además, necesita que el jugador genere un deseo de pagar demanda, rellenando un valor por cada recurso de comida existente (puede poner a 0 los que no desea emplear).

La regla reduce la demanda pendiente de pagar con los recursos que aporta el jugador, ponderando cada uno por su multiplicador de comida (por ejemplo, la carne aporta 3 unidades de comida). Si el jugador excede la demanda de comida con sus recursos aportados, perderá el excedente.

Pagar Comida Endeudándose Módulo (Distinto) 0

En ocasiones, los jugadores no pueden aportar suficientes recursos de comida para pagar la demanda del final de la ronda. En tal situación, la demanda es pagada endeudando al jugador. Para calcular la cantidad de préstamos necesarios para pagar la demanda pendiente, se divide la cantidad restante por pagar entre 4, que es el valor de una deuda.

Sin embargo, si el resultado de esta operación fuese decimal, se necesita realizar un redondeo a la alza. Esta operación no está implementada en *CLIPS*, por ello hemos tenido que realizarla manualmente.

El problema encontrado es que, para divisiones cuyo resultado es entero, el proceso de redondeo no funcionaba (añadíamos una unidad de más). Por esto hemos necesitado dos reglas, que en función de si la división tiene resto o no, aplican una división con redondeo o simplemente dividen.

Añadir Ganado/Grano Por Cosecha

En las reglas del juego se incluyen las cosechas, que implican una entrega de recursos al jugador si cumple ciertas condiciones. En el caso de la cosecha por ganado, al jugador se le entrega una unidad extra de ganado o grano si en sus recursos tiene dos o más (ganado) o una unidad o más (grano), una vez haya pagado la demanda de comida de la ronda. Esta regla está dedicada únicamente a que dicho comportamiento se cumpla.

Pasar Turno

Para pasar turno, será necesario que ya se hayan añadido los recursos a la oferta, el jugador haya terminado su actividad principal, que no haya ningún proceso de actualización de mazos pendiente y que se haya generado un hecho semáforo (*proceso_limpieza_finalizado*) que indica que se han limpiado los hechos generados por la estrategia del jugador que no interesan mantener.

Tras cumplirse dichas precondiciones, se modifica la posición del otro jugador de la partida, moviéndole 2 losetas e iniciando su turno.

Pasar Turno al Final de la Ronda

El paso de turno al final de la ronda realiza las mismas comprobaciones que el paso de turno normal pero añade la precondición de que un jugador deba estar en la última loseta del tablero y el otro en la penúltima, y que además el jugador que está en la última loseta haya terminado su acción principal.

Esta regla genera los hechos necesarios para producir el cambio de ronda, como puede ser cambiar el semáforo *cambiar_ronda* a TRUE. Además, inicia el contador de comida demandada para que se inicie

Pasar Turno Ronda Extra Final

La diferencia de esta regla y la de paso de turno normal, es que como se debe a la ronda extra final, no es necesario actualizar las posiciones de los jugadores. También hay que restringir que esta regla se active a la ligera. Tiene implementado un contador de turnos que se inicializa con 2. Cuando los turnos restantes son 0 no se permitirá pasar de turno ya que la partida habrá llegado a su fin.

Añadir Recursos Oferta

Esta regla se encarga de añadir los recursos correspondientes a una loseta a la oferta cuando un jugador inicia su turno.

Tomar Recurso Oferta

Si el jugador desea tomar recursos de la oferta (lo indica a través del semáforo *deseo_coger_recursos*) se le añade el recurso de la oferta que desea. Esto constituye la actividad principal de un turno, por lo que en las postcondiciones se genera el hecho *fin_actividad_principal*, además de modificar la cantidad del recurso tomado en la oferta a 0.

Entrar Edificio Gratis Rondas

Parte de los edificios del juego no tienen coste de entrada, o si son propiedad del jugador, no debe pagar su entrada. Esto supone que el jugador puede entrar a ellos siempre que lo desee (*deseo_entrar_edificio*), que sea su turno, que no haya realizado la actividad principal de su turno todavía y que el edificio esté disponible (es propiedad de alguien y no hay nadie dentro).

Si las precondiciones se cumplen, se modifica la posición del jugador.

Entrar Edificio Con Coste Entrada Dinero Rondas

Para los edificios que sí tienen coste de entrada, pero el coste es únicamente pagable con francos, se requiere que el jugador haya generado el deseo de entrar al edificio, indicando el pago de FRANCOS.

Además, en los edificios con coste de entrada hay que obtener quién es el propietario de dicho edificio para hacerle la entrega de los recursos que el jugador pagó para entrar.

Entrar Edificio Con Coste Entrada Alimento Rondas

Para los edificios cuyo coste de entrada es COMIDA, se requiere que el jugador haya generado el deseo de entrar al edificio, indicando *COMIDA* y el recurso alimenticio a usar. Dicho recurso será entregado al propietario del edificio.

Entrar Edificio Gratis Ronda Extra Final

Esta regla sigue la misma lógica que la de Entrar Edificio Gratis Rondas, salvo por la diferencia de que en la ronda extra final no existe la restricción de que el edificio esté vacío para poder entrar.

Entrar Edificio Con Coste Entrada Alimento/Monetario Ronda Extra Final

En la ronda extra final no es necesario que el edificio esté vacío para poder entrar. El resto de la regla sigue la misma lógica que la de entrar a un edificio con coste durante rondas normales.

Utilizar Edificio Constructor No/Si Bonus

Cuando el jugador entra en una "Constructora", se le permite construir un edificio siempre y cuando tenga suficientes recursos y la carta del edificio esté en la primera posición de uno de los mazos. Además, el jugador deberá haber generado en su estrategia el deseo *deseo_construcción*, indicando qué edificio desea construir.

Una vez se han realizado las comprobaciones se eliminará la carta del mazo, se entregará al jugador y se actualizará el mazo. Además, se actualiza la riqueza del jugador y se finaliza su actividad principal. También se genera el hecho "edificio_usado" para que el jugador no lo vuelva a usar si no ha entrado a otro edificio antes.

La diferencia entre la regla que incluye bonus y la que no es que, si el edificio a construir tiene bonus (*CARTA_TIENE_BONUS*), debe actualizarse dicha información en los datos del jugador.

Edificios Generadores

Los edificios generadores son aquellos que permiten a un jugador generar recursos, ya sea transformando un *input* o simplemente obteniendo los recursos por entrar al edificio. Los edificios generadores presentan una gran heterogeneidad, lo que nos imposibilitó el planteamiento de una regla genérica que englobase a todos los edificios. Esto se debe seguramente al diseño de la ontología que realizamos, es posible que con un planteamiento alternativo la generalización sí fuese posible.

Por este motivo, nos vimos obligados a desarrollar una regla para cada caso de edificio generador. La casuística encontrada se resume en la siguiente tabla:

Input	Output	Bonus	Edificios
0	1	1	Piscifactoría, Montículo de arcilla, Mina de Carbón
1	1	0	Horno de carbón vegetal, Fábrica de hierro
1	1	0	Siderurgia
1	2	0	Matadero, peletería y coquería
1	2	0	Ahumador, fábrica de ladrillos y peletería

Para los edificios sin input, directamente se le entrega el recurso generado al jugador, comprobando que esté dentro del edificio y que no ha usado el edificio en el turno anterior (*JUGADOR_HA_USADO_EDIFICIO)*. Afortunadamente, todos los edificios sin input tienen bonus, por lo que pudimos incluir en una única regla su comportamiento. En función del edificio, se entregan al jugador las unidades adicionales del recurso correspondientes teniendo en cuenta cuántos bonus tiene.

Una vez el jugador ha empleado el edificio, se finaliza su actividad principal, se modifica el edificio usado por él y sus recursos y se eliminan los deseos empleados.

Para el resto de casos se realiza la misma lógica, pero teniendo en cuenta el input necesario. Este input provoca que el deseo empleado por el jugador en las reglas estratégicas deba ser algo más complejo, debido a que debe indicar qué recurso introducir en el edificio y cuánta cantidad. Toda esta información debe indicarse en un hecho *deseo_generar_con_recurso*.

Los recursos se transforman en base a un ratio indicado por el edificio en la relación *EDIFICIO_OUTPUT*. En ocasiones, el ratio genera cantidades decimales del recurso. Siguiendo las reglas del juego, dichas cantidades son redondeadas al entero inferior más próximo.

Comerciar en Compañía Naviera

La Compañía Naviera cuenta con un comportamiento único. Para ello, hemos implementado una regla específica para esta carta. Además, esta carta cobra una gran relevancia en nuestra estrategia puesto que es usada con frecuencia en la ronda extra final.

La compañía naviera ofrece a los jugadores la posibilidad de vender recursos. En concreto, tantos recursos como capacidad tengan con sus barcos. Esta capacidad está representada en el atributo *capacidad envío* del jugador.

Para poder usar la Compañía Naviera, el jugador debe generar un hecho *deseo_usar_compañía_naviera*, a través del cual indica qué recursos quiere vender. La suma de las cantidades de los recursos no debe exceder la capacidad de envío.

Si todo se cumple, se eliminarán los recursos del jugador y se le entregará la cantidad de francos correspondiente al valor de los recursos vendidos. Dicho valor está indicado en la propia regla y se obtiene de las reglas del juego. Tras usar el edificio, se finaliza la actividad principal del jugador y se elimina su deseo de usar la compañía naviera.

Usar Edificio Muelle

El edificio Muelle permite a los jugadores fabricar barcos. Esta carta no deja de ser un edificio constructor, pero no se ha podido incluir en las reglas de *USAR_EDIFICIO_CONSTRUCTOR* puesto que la generación de barcos requiere de la modificación de ciertos atributos en el jugador que con otras cartas no sería necesaria.

Para su funcionamiento, la regla necesita de un hecho *deseo_construccion* que indica qué barco ha de ser construido. Este hecho se genera en la estrategia del jugador. Tras esto, se comprueba que el jugador tiene suficientes recursos para construir el barco. Además, se comprueba la disponibilidad del barco (ya ha sido introducido en la partida) y su posición en el mazo (debe ser la primera carta).

Si se cumplen las precondiciones, se entrega el barco al jugador, se reducen los recursos empleados, se le añade capacidad de envío, número de barcos y demanda de comida cubierta, se elimina el deseo empleado y se finaliza su actividad principal.

Pasar Ronda

Esta regla ejecuta el cambio de rondas. Para que se de lugar su ejecución, es necesario que se haya iniciado ya el cambio de ronda (iniciado al pasar turno al final de la ronda). Además, se comprueba que los jugadores ya hayan pagado la demanda de comida.

Si se cumplen las precondiciones, se cambia de ronda y se introduce el barco correspondiente a la nueva ronda

Comenzar Ronda Extra Final

La ronda extra final tiene un comportamiento muy especial. Por este motivo, no puede ser iniciada con la regla de pasar ronda.

La regla emplea un semáforo para evitar ejecutarse en bucle, Tras ello, genera los hechos de *permitir_realizar_accion* para que los jugadores puedan ejecutar sus últimas voluntades en la partida.

Calcular Riqueza Jugador

Si la ronda actual es la ronda extra final, los jugadores han finalizado su actividad principal y ya tienen bloqueada cualquier otra acción, se debe calcular la riqueza de cada jugador para poder mostrar las puntuaciones finales. La riqueza final se calcula con el atributo *riqueza* del jugador, restándole los préstamos pendientes de pagar y sumándole la cantidad de francos en su posesión.

Mostrar Resultados Partida

Esta regla es la última en ejecutarse en la partida, y es la encargada de mostrar las puntuaciones finales por pantalla además de mostrar ciertas estadísticas para su posterior análisis.

3.2. Conocimiento de Inferencia

En este apartado se representan los pasos de razonamiento básicos implementados. Estos serán representados mediante deseos. Se pueden diferenciar dos bloques: aquel empleado para deshacer los deseos generados por el jugador durante su proceso de decisión y aquel empleado para generar deseos.

3.2.1. Deseos

Se han implementado los siguientes deseos para describir los pasos de razonamiento básicos seguidos por el experto.

- *Deseo conseguir recurso*: si se necesita un recurso que no existe en la oferta, cómo por ejemplo ladrillos o acero, se lanzará el deseo de conseguir el recurso para intentar obtenerlo.
- *Deseo entrar al edificio*: debido a que los edificios pueden tener tarifa de entrada, se tienen deseos de entrar al edificio simples donde sólo se indica el edificio objetivo o deseos complejos donde se indicará además el recurso que se quiere emplear para pagar.
- *Deseo generar con recurso*: existen recursos que pueden ser generados a través de otros. Se lanzará para intentar obtener aquel recurso. Por ejemplo, si el jugador quiere ladrillos, se lanzará el deseo generar con recurso que contenga arcilla.
- *Deseo coger recurso*: si el recurso se necesita ó hay una cantidad elevada de él en la oferta se lanzará.

- *Deseo construcción*: si el jugador dispone de suficientes materiales para construir un determinado edificio se lanzará.
- *Recurso construcción*: representa aquellos recursos necesarios para la construcción de un edificio.
- *Contador recurso construcción*: contador auxiliar que lleva la cuenta de los deseos *recurso construcción*.
- *Objetivo prioridad generado*: información que permite al jugador centrarse en aquel objetivo prioritario generado.
- *Decisión pago comida entrada edificios*: deseo que contempla el recurso empleado por el jugador para pagar las tarifas de entrada de los edificios.
- *Deseo pagar deudas*: si el jugador tuviese deudas, puede ver necesario deshacerse de ellas.
- *Deseo pago individual*: deseo auxiliar que representa la disposición del jugador de emplear cierta cantidad de recurso para pagar la demanda de comida.
- *Deseo pagar demanda*: deseo que representa la combinación de recursos que emplea el jugador para pagar la demanda de la ronda.
- *Deseo comprar un edificio:* deseo que representa la intención de comprar un edificio.
- *Comerciar en compañía*: deseo auxiliar que representa la disposición del jugador de emplear cierta cantidad de recurso para comerciar con él en la compañía naviera.
- *Deseo usar compañía naviera*: deseo que representa la intención de usar la compañía naviera con una combinación de recursos dada.

3.2.2. Limpieza de deseos

Tras la toma de decisión del jugador y antes de pasar de turno, será necesario concluir limpiando de la base de hechos aquellos deseos que hayan sido instanciados durante el proceso y que no hayan sido empleados.

La idea subyacente implementada es la siguiente, cuando un jugador termine su correspondiente acción, se lanzará un proceso de limpieza. Mientras esté activo se ejecutarán reglas de limpieza de deseos. Cuando todo haya sido completamente eliminado, se finalizará este proceso dando paso al cambio de turno.

Dado que en las reglas del juego se emplean estos deseos como precondiciones, si finalmente se termina ejecutando la regla, se eliminará el deseo utilizado. Sin embargo, puede darse el caso que el jugador evalúe varias propuestas antes de tomar una decisión provocando que sea necesario ejecutar el proceso de limpieza a través de la regla *COMENZAR LIMPIEZA ESTRATEGIA DEL TURNO JUGADOR.*

Si este no fuese necesario, a través de la regla *NO NECESARIA LIMPIEZA ESTRATEGIA DEL TURNO JUGADOR* se permitirá continuar con el flujo del juego pertinente.

Si por el contrario, se lanza el proceso de limpieza, cuando este termine se permitirá continuar con el flujo correspondiente a través de la regla *FINALIZAR LIMPIEZA ESTRATEGIA DEL TURNO JUGADOR*.

Las reglas empleadas para eliminar todo deseo existente son las siguientes:

- *LIMPIAR OBJETIVOS CONSEGUIR RECURSOS*
- *LIMPIAR DESEOS ENTRAR EDIFICIOS SIMPLES*
- *LIMPIAR DESEOS ENTRAR EDIFICIOS COMPLEJOS*
- *LIMPIAR DESEOS GENERAR CON RECURSO*
- *LIMPIAR DESEOS COGER RECURSO*
- *LIMPIAR DESEOS CONSTRUCCIÓN SUELTOS*

- *LIMPIAR DESEO CONSTRUIR EDIFICIO NO PUEDE PAGAR ENTRADA CONSTRUCTORA*
- *LIMPIAR OBJETIVOS PRIORIDADES*
- *LIMPIAR DESEO RECURSO CONSTRUCCION VACÍOS*
- *LIMPIAR DESEO RECURSO CONSTRUCCION RESTANTES FIN TURNO*
- *LIMPIAR CONTADOR RECURSO CONSTRUCCION FIN TURNO*

3.2.3. Generación de deseos

Para la obtención de todos los deseos descritos anteriormente han sido necesarias numerosas reglas. A continuación se sintetizan las siguientes reglas:

- *GENERAR DESEO OBTENER BARCO*
- *GENERAR DESEO DEUDAS*
- *CAMBIAR DECISIÓN PAGO COMIDA ENTRAR EDIFICIOS*
- *TRANSFORMAR ALIMENTOS*
- *COMPROBAR EXISTENCIA RECURSOS CONSTRUCCIÓN EN OFERTA*
- *GENERAR DESEO CONSEGUIR RECURSOS*
- *OBTENER DESEO ENTRAR EDIFICIO PARA CONSEGUIR RECURSO*

Bajo este nombre se pretende agrupar las cuatro reglas que han sido necesarias para generar los deseos de entrar a los edificios. Dado que se ha intentado ser lo más fiel posible a la realidad, la máxima simplificación que se ha obtenido dada la ontología propuesta ha sido representar cada combinación de input-coste de entrada con una regla distinta.

Input	Coste Entrada	Edificios
NO	NO	Piscifactoría
NO	SI	Montículo de arcilla, herrería, mina de carbón.
SI	NO	Carbón Vegetal, Peletería
SI	SI	Panadería, Matadero, Ahumador

A continuación se listan los nombres de las reglas:
OBTENER DESEO ENTRAR EDIFICIO PARA CONSEGUIR RECURSO SIN INPUT SIN COSTE ENTRADA
OBTENER DESEO ENTRAR EDIFICIO PARA CONSEGUIR RECURSO CON INPUT SIN COSTE ENTRADA
OBTENER DESEO ENTRAR EDIFICIO PARA CONSEGUIR RECURSO CON INPUT CON COSTE ENTRADA
OBTENER DESEO ENTRAR EDIFICIO PARA CONSEGUIR RECURSO SIN INPUT CON COSTE ENTRADA

- *OBTENER DESEOS CONSTRUCCIÓN PRIORIDADES*

Bajo este nombre se agrupan las seis reglas que han sido necesarias para generar los deseos de prioridades empleados en la estrategia. Como se comentará en la siguiente sección, los expertos normalmente tienen ciertas prioridades en los edificios que construyen para obtener ventajas competitivas.

OBTENER DESEOS CONSTRUCCIÓN PRIORIDAD 1
OBTENER RECURSOS PARA CONSTRUIR PRIORIDAD 1
OBTENER DESEOS CONSTRUCCIÓN PRIORIDAD 2
OBTENER RECURSOS PARA CONSTRUIR PRIORIDAD 2
OBTENER DESEOS CONSTRUCCIÓN PRIORIDAD 3
OBTENER RECURSOS PARA CONSTRUIR PRIORIDAD 3

- *OBTENER DESEO ENTRAR EDIFICIO CONSTRUCTOR*

Debido a que los edificios pueden tener tarifa de entrada, se tienen deseos de entrar al edificio simples donde sólo se indica el edificio objetivo o deseos complejos donde se indicará además el recurso que se quiere emplear para pagar. Además, al tratarse de edificios constructores, estos tienen distintas propiedades y comportamiento a los generadores.

$$deseo\ entrar\ edificio\ (nombre\ jugador_i,\ nombre\ edificio)$$
$$deseo\ entrar\ edificio\ (nombre\ jugador_i,\ nombre\ edificio,\ tipo,\ recurso)$$
$$tipo \in \{COMIDA,\ DINERO\}$$
$$recurso \in \{PESCADO,\ PESCADO\ AHUMADO,\ CARNE,\ PAN,\ FRANCO\}$$

OBTENER DESEO ENTRAR EDIFICIO CONSTRUCTOR SIN COSTE ENTRADA
OBTENER DESEO ENTRAR EDIFICIO CONSTRUCTOR CON COSTE ENTRADA

- *DESEOS DEMANDA DE COMIDA*

Bajo este nombre se agrupan las reglas necesarias para determinar el deseo final de demanda de comida. Como se explicará en el siguiente apartado, la demanda de comida requiere incluir un determinado número de reglas para poder implementar el conocimiento extraído del experto.

Advierta que nuevamente aparece el problema del módulo 0. Dado que *CLIPS* no dispone de una función *ceil* se ha necesitado implementar una fórmula matemática para replicar esta funcionalidad. Sin embargo, cuando $x\ mod\ 4\ =\ 0$ la expresión no funciona y es necesario aplicar en las postcondiciones otra expresión.

Además, *FRANCOS* tiene una regla individual debido a que se busca que primero se intancien los pagos con el resto de recursos. Los francos serán el último recurso que usarán los jugadores a la hora de pagar su demanda de comida como más tarde se explicará.

GENERAR PAGOS INDIVIDUALES
GENERAR PAGOS INDIVIDUALES PROBLEMA MODULO 0
GENERAR PAGOS INDIVIDUALES CUANDO CONTADOR YA ES 0
GENERAR PAGO INDIVIDUAL FRANCOS
RELLENO DESEO PAGAR DEMANDA

- *MANTENER DERECHO COSECHA GANADO*
- *MANTENER DERECHO COSECHA GRANO*
- *OBTENER DESEO COMPRAR CARTA NO PUEDE SER CONSTRUIDA*

Excepción implementada para aquellas cartas que no pueden ser construidas con materiales. Para ello, se podrá generar el deseo de comprar la carta si se dispone de suficientes *FRANCOS*.

- *GENERAR DESEO COGER OFERTA PRIORITARIO*
- *GENERAR DESEO COGER OFERTA NO PRIORITARIO*
- *DESEOS COMPAÑÍA NAVIERA*

Bajo este nombre quedan agrupadas las reglas necesarias para el correcto funcionamiento de la tarea comerciar con compañía Naviera. Como se explicará más adelante, han sido necesarias todas y cada una de las reglas para obtener el funcionamiento óptimo de este edificio.

OBTENER DESEO ENTRAR COMPAÑÍA NAVIERA NO EN PROPIEDAD
OBTENER DESEO ENTRAR COMPAÑÍA NAVIERA EN PROPIEDAD
CAMBIAR PRIORIDAD RELLENAR DESEO COMPAÑÍA NAVIERA
RELLENAR DESEO COMPAÑÍA NAVIERA SIN COSTE
RELLENAR DESEO COMPAÑÍA NAVIERA CON COSTE

BLOQUEAR RECURSO EMPLEADO PARA PAGAR ENTRADA COMPAÑIA NAVIERA
RELLENAR DESEO COMPAÑÍA NAVIERA 2

- *GENERAR DESEO COGER OFERTA RONDA EXTRA FINAL*

3.3. Conocimiento de Tarea

3.3.1. Acciones posibles del jugador en su turno

Antes de empezar en profundidad con el conocimiento de tarea, será fundamental entender todas las alternativas de las que dispone un jugador durante su turno en una partida cualquiera del juego. El turno de un jugador puede desglosarse acorde a la ronda actual dónde se encuentre:

- Rondas de la 1 a la 8:
  - ***Actividad principal***: únicamente está permitido realizar una por turno. Las actividades principales que pueden llevarse a cabo a su vez son:
    - Entrar a un edificio para construir, transformar o comerciar.
    - Tomar un recurso de la oferta
  - ***Actividades secundarias***: no se encuentran restringidas, pueden llevarse a cabo tantas como se deseen. Las actividades secundarias disponibles son las siguientes:
    - Comprar / vender cartas, ya sean barcos o edificios.
    - Pagar deudas
- Ronda extra final:
  - ***Actividad principal***: únicamente está permitido realizar una por turno. Las actividades principales que pueden llevarse a cabo a su vez son:
    - Entrar a un edificio para transformar o comerciar.
    - Tomar un recurso de la oferta
  - ***Actividades secundarias***: no se encuentran restringidas, pueden llevarse a cabo tantas como se deseen. Las actividades secundarias disponibles son las siguientes:
    - Vender cartas, ya sean barcos o edificios.
    - Pagar deudas

3.3.2. Tareas

Como se mencionó anteriormente, la propuesta del documento encaja principalmente en la tarea analítica de ***valoración***. La mayoría de tareas funcionan evaluando una serie de criterios para determinar si es llevada a cabo o no. Es el caso de pagar deudas, mantener derecho de cosecha, coger recursos de la oferta y obtener barcos.

Sin embargo, la propuesta también tiene tareas sintéticas de ***planificación***. Es el caso de transformar recursos, obtener edificios, comerciar en la Compañía Naviera y pagar demanda de comida.

Pagar Deudas

En el caso de pagar deudas, cuando el jugador tenga deudas pendientes y disponga de cinco francos, deseará pagar la deuda. A través de sucesivas partidas, se ha observado que contra antes se pagan las deudas más maniobra se tiene en la partida.

Mantener derecho de Cosecha

Mantener el derecho de cosecha es fundamental para obtener una buena puntuación en la partida. A modo de recordatorio, la cosecha funciona de la siguiente manera: tras finalizar una ronda, si el jugador dispone de 2 o más unidades de *GANADO* ó 1 o más unidades de *GRANO* se le otorgará una unidad de cada recurso respectivamente.

Almacenar recursos de *GANADO* y/o *GRANO* es determinante ya que permiten cubrir varias demandas de comida con relativa facilidad al ser transformados en comida.

Coger Recursos de la Oferta

Coger recursos de la oferta también dependerá del número de unidades que haya disponibles en la oferta. Por tanto, se ha establecido una prioridad, cuando haya más de 4 unidades se deberá coger el recurso de forma prioritaria. Esta tarea también puede ser desempeñada de forma no prioritaria siempre que el jugador no disponga de una mejor alternativa.

Obtener Barcos

Para obtener barcos se evaluará si el muelle puede ser usado y si se dispone de suficientes recursos, en caso afirmativo se deberá obtener un barco. La construcción de barcos es fundamental debido a las reducciones de costes de comida que generan.

Cambiar Decisión del recurso alimenticio empleado para pagar tarifas de entrada

Esta tarea permite a los jugadores determinar qué recurso alimenticio emplearán para pagar las tarifas de entrada de los edificios. Aunque es cierto que los expertos empleaban a veces varios recursos para satisfacer las tarifas, se ha optado por simplificar este aspecto para que empleen exclusivamente aquel recurso alimenticio del que dispongan una mayor cantidad.

Por tanto, esta tarea se llevará a cabo cuando el jugador tenga un recurso que le proporcione una mayor cantidad de alimento del que tiene actualmente para este propósito.

Transformar Recursos

La tarea de transformar recursos permite a los jugadores transformar recursos simples en elaborados en los edificios pertinentes. Aunque es cierto que se le da cierta prioridad a la hora de transformar recursos alimenticios a través de una regla exclusiva, esta tarea sirve para transformar cualquier tipo de recurso simple en elaborado.

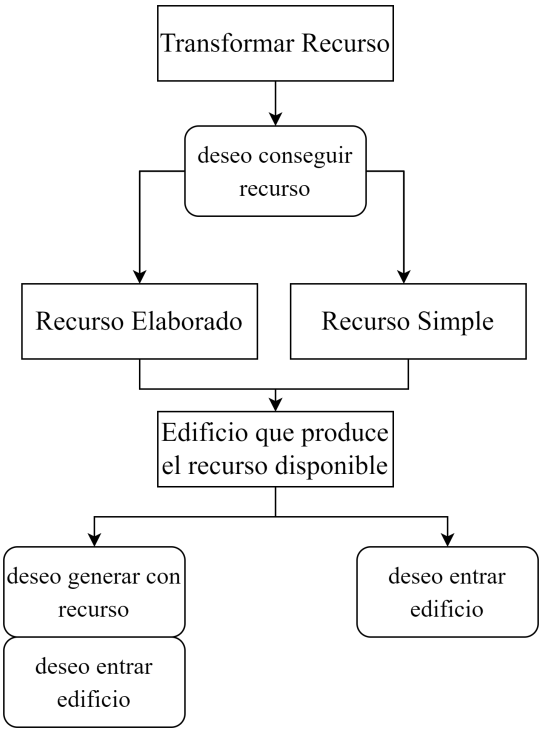


Figura III - Tarea Transformación recursos

El jugador tiene 5 unidades de *GANADO*, por tanto, decide transformar el *GANADO* en *CARNE*. Esta decisión es representada a través del *deseo de conseguir recurso* en el cual se le indica que el jugador quiere conseguir 5 unidades de *CARNE*.

$$deseo\ conseguir\ recurso\ (jugador_i,\ CARNE,\ 5)$$

Si el jugador puede acceder al edificio donde se obtiene la carne, es decir, el MATADERO, generará el deseo de entrar al MATADERO y el deseo de generar en el edificio 5 de carne empleando 5 de ganado.

$$deseo\ entrar\ edificio\ (jugador_i,\ MATADERO,\ DINERO,\ FRANCO)$$

$$deseo\ generar\ con\ recurso(jugador_i,\ MATADERO,\ GANADO,\ 5)$$

El proceso de abstracción seguido trata de emular al experto. Aunque pueda parecer contraintuitivo a la hora de programarlo, es como los expertos toman las decisiones en este juego. Tienen *GANADO* y quieren transformarlo en *CARNE*, para ello, primero determinan que el *GANADO* produce *CARNE* y posteriormente buscan aquel edificio donde se puede transformar *GANADO* en *CARNE*, en este caso el *MATADERO*. Al final, el experto decide cuántas unidades de *GANADO* quiere transformar. Para simplificar, las unidades que se quieren transformar serán predeterminadas en el deseo de conseguir recurso.

Obtener Edificios

La tarea de obtener edificios permite a los jugadores obtener edificios construyéndo. Es una tarea simplificada ya que la obtención de edificios puede realizarse también comprando edificios. Sin embargo, los expertos que compraban edificios siempre obtenían peores estadísticas que aquellos que no compraban y construían. Por tanto, los jugadores expertos implementados se dedicarán exclusivamente a construir edificios.

Además, los expertos no construían cualquier edificio, estos les daban prioridades a los edificios para obtener ventajas competitivas con respecto a sus rivales. Es el caso de la *FÁBRICA DE LADRILLOS*, este edificio es un embudo natural ya que todo edificio intermedio requiere de ladrillos para su construcción. Quien tenga esta carta se asegurará tener un flujo de comida constante debido a la tarifa de entrada que tiene el edificio.

Se han determinado tres niveles de prioridades. Esto permite a los jugadores no malgastar sus recursos de construcción y centrarse en aquellos edificios que son prioritarios para ellos.

Para determinar si se puede obtener o no un determinado edificio, será imprescindible llevar a cabo una planificación a corto plazo, es decir, el objetivo de obtener un edificio puede que no se alcance siempre en un mismo turno. Sin embargo un edificio podrá ser obtenido a través de varios turnos siempre y cuando se den las condiciones.

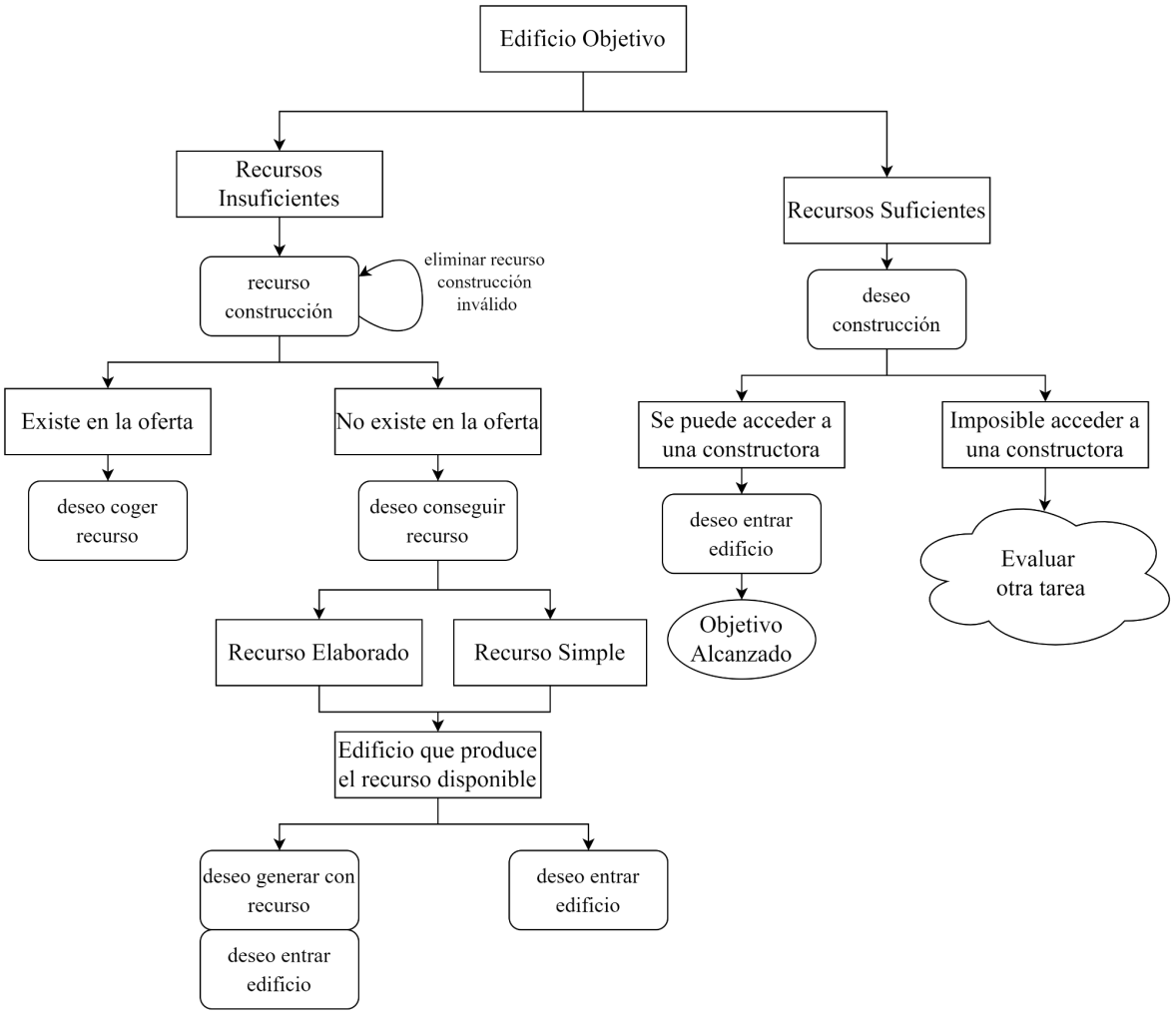


Figura IV - Tarea Obtener Edificios

El diagrama mostrado en la figura IV pretende mostrar todo el proceso de abstracción obtenido por los expertos a la hora de construir un edificio. Para entender mejor esta tarea y la planificación a corto plazo de la que se habla se analiza el siguiente ejemplo:

Dado un edificio objetivo con una determinada prioridad. Se comprueba si se tienen recursos necesarios para construirlo. En el caso de que no se disponga de estos recursos, se comprobará si estos pueden ser obtenidos de la oferta.

Continuando con el ejemplo, el recurso que se necesita no existe en la oferta. Por tanto, se genera un deseo de conseguir el recurso. Para conseguir este recurso habrá que comprobar si el edificio que lo genera está disponible. Si lo estuviese, se deseará entrar al edificio.

Esto provoca que la acción del jugador sea entrar al edificio que genera aquel recurso para obtenerlo. Como se ha visto, se trata de una planificación a corto plazo. Advierta que al pasar el turno no se guarda ningún deseo, todo es limpiado de la base de hechos. Por tanto, no existe una planificación a largo plazo.

Debido al sistema de prioridades establecido, si el edificio estuviera disponible cuando vuelva a tener el turno, el jugador intentará obtener el edificio realizando de nuevo otra planificación.

Comerciar en la Compañía Naviera

La tarea de comerciar en la Compañía naviera también queda dentro de las tareas de planificación. Los jugadores expertos suelen emplear este edificio en la *RONDA EXTRA FINAL* para obtener una puntuación extra intercambiando recursos que no aportan riqueza por francos.
Los recursos que pueden ser comerciados en este edificio se pueden clasificar según el valor unitario en francos que proporcionan.

Valor Unitario en Francos	Recursos
1	*PESCADO, MADERA, GRANO, FRANCO*
2	*HIERRO, PIEL, PESCADO AHUMADO, CARBÓN VEGETAL, LADRILLO, CARNE*
3	*GANADO, CARBÓN, PAN*
4	*CUERO*
5	*COQUE*
8	*ACERO*

El proceso implementado es el siguiente: El jugador quiere acceder a la compañía naviera como última acción de la partida. Si le pertenece el edificio, con generar un deseo de entrar en un edificio simple será suficiente. En caso contrario, habrá que generar un deseo de entrar al edificio compuesto donde entra en juego el recurso empleado por el jugador para pagar los costes de entrar al edificio.

Además, dado que comerciar francos es una pérdida de capacidad de envíos, se ha optado por instanciar el deseo auxiliar de comerciar en compañía con 0 de *FRANCOS*.

Dada la prioridad descrita anteriormente, los recursos del jugador son recorridos de mayor a menor en términos de valor para gastar capacidades de envío en recursos que generen mayor riqueza. Si el jugador tiene que pagar tarifa de entrada para acceder al edificio, será necesario bloquear aquella cantidad necesaria para poder realizar el pago. En caso contrario, este proceso no serviría de nada.

Por último, tras haber rellenado todos los deseos auxiliares de comerciar en compañía, se genera el deseo de usar compañía naviera. Este contiene el nombre del jugador y una combinación de recursos del jugador posible.
En la siguiente figura se presenta este proceso de una forma gráfica.

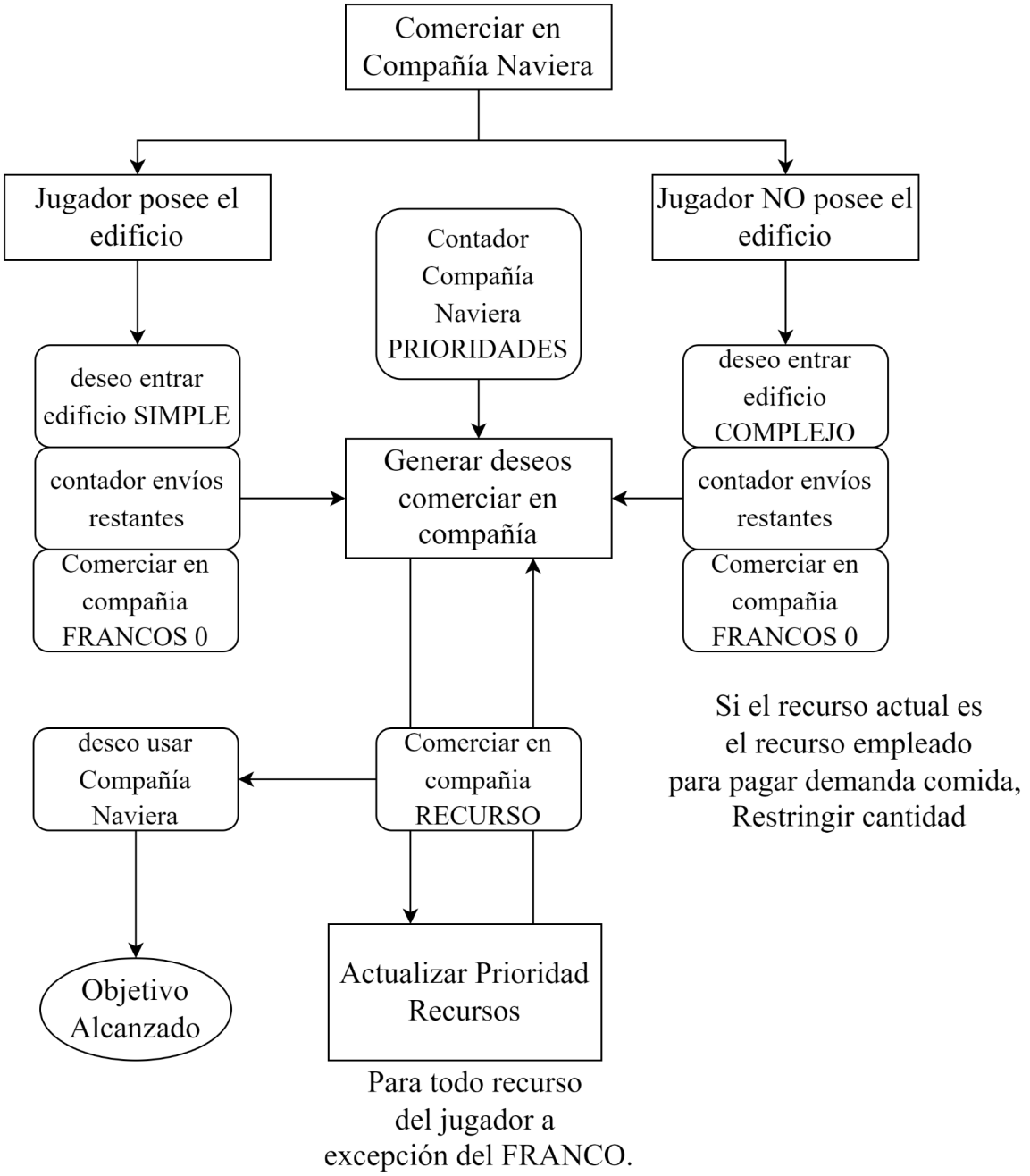


Figura V - Tarea Compañía Naviera

Pagar Demanda Comida

En las ocho primeras rondas de la partida, es necesario hacer frente al pago de una demanda de comida que va incrementándose a medida que van pasando las rondas. Esto provoca una limitación a la hora de avanzar en la partida.

De los expertos se ha obtenido la conducta de evitar pagar con *FRANCOS* a toda costa. Prefieren siempre afrontar estos pagos con recursos de comida simples y elaborados antes de emplear los *FRANCOS*. Otra alternativa que se ha visto en repetidas ocasiones es la de adquirir barcos para cubrir cierta parte de esta demanda.

El proceso implementado es el siguiente: Dada una cantidad de demanda a cubrir se va iterando entre todos los recursos alimenticios del jugador para ir rellenando la demanda de comida. Para simplificar este proceso la única diferencia se hace con los *FRANCOS*, el resto de recursos son recorridos indistintamente sin orden de prioridad hasta cubrir la totalidad de la demanda.

En el caso de que el jugador no tuviese recursos alimenticios suficientes, se emplearán los francos del jugador. Si aún así sigue sin poder cubrir la demanda de comida, se le asignarán tantas deudas como sean necesarias.

El funcionamiento de las deudas es el siguiente, adquirir una deuda te otorga 4 *FRANCOS*. Pagar una deuda te cuesta 5 *FRANCOS*. Tener una deuda al finalizar la partida te penaliza con menos 7 de riqueza.

3.3.3. Estrategia Implementada

La estrategía implementada para los jugadores inteligentes se divide en dos partes. Por un lado, se tiene la estrategia empleada para cualquier ronda de la primera a la octava. Esta estrategia tiene las siguientes prioridades.

1. Siempre que el jugador tenga alguna deuda y tenga *FRANCOS* suficientes, se deshará de ella una vez haya terminado su actividad principal.
2. Siempre que el jugador disponga de una cantidad mayor que 4 de recursos que pueden ser transformados en recursos alimenticios llevará a cabo la acción de transformar estos recursos siempre que el edificio se encuentre libre.
3. Construir edificios de prioridad I
4. Construir edificios de prioridad II
5. Construir edificios de prioridad III
6. Mantener el derecho de cosecha.
7. Comprar con *FRANCOS* aquella carta que no pueda ser construida con materiales. (*MONTÍCULO DE ARCILLA*)
8. Coger recursos de la oferta de forma prioritaria. Es decir, tomar un recurso de la oferta cuando su cantidad sea mayor a 3.
9. Coger recursos de la oferta de forma NO prioritaria. Es decir, tomar un recurso de la oferta cuando su cantidad sea mayor a 0.

Mencionar que en paralelo todo jugador podrá cambiar el recurso alimenticio que emplea para pagar tarifas de entrada de los edificios. Por otro lado, la estrategia implementada para la ronda *EXTRA FINAL* sigue las siguientes prioridades:

1. Comerciar en la Compañía Naviera.
2. Tomar recurso de la oferta cuando este es mayor que 0.

Para finalizar, comentar que la estrategia tiene un componente estático. Estas son las prioridades introducidas a priori para los edificios. El jugador en sí mismo no determina qué edificio es mejor de acuerdo a la situación de la partida sino que este le viene prefijado de antemano.

4. Manual de usuario

El Sistema Basado en el Conocimiento implementado está compuesto por la siguiente estructura de directorios y ficheros:

pruebas
- *plantilla_pruebas.clp*
- *prueba1.clp*
- *prueba2.clp*
- *prueba3.clp*
- *prueba4.clp*
- *prueba5.clp*

salidas
- *salida-prueba1.clp*
- *salida-prueba2.clp*
- *salida-prueba3.clp*
- *salida-prueba4.clp*
- *salida-prueba5.clp*

ontology.clp
rules.clp

Los pasos a seguir para poder ejecutar el sistema y obtener la salida son los siguientes:

1. Tener instalada la versión de *CLIPS 6.4*
2. Cargar la ontología con el comando `(load ontology.clp)`
3. Cargar el escenario deseado con el comando `(load pruebas/pruebaX.clp)`
    a. Nótese que *X* es un número entero que representa un escenario concreto.
4. Cargar las reglas con el comando `(load rules.clp)`
5. Tras haber cargado los tres ficheros, introducir por consola, ya sea en el *CLIPS IDE* o desde el terminal el comando `(reset)`
6. Posteriormente, introducir el comando `(run)`
7. El fichero de salida se encontrará en el directorio *salidas*

Para no sobreescribir las salidas de una prueba con otra, será necesario abrir el fichero `rules.clp` y en la regla *COMIENZO_PARTIDA* especificar en el comando (dribble-on nombre_fichero.txt) el nombre del fichero que se quiera. Además, en esta regla se podrán añadir / eliminar las trazas para *debuggear* en caso de error.

Por último, si se desea introducir un escenario para curiosear la solución propuesta, se ha dejado en concreto un fichero llamado *plantilla_pruebas.clp* donde se podrán permutar las prioridades de los objetivos introducidas estáticamente.

$$(objetivo\ carta\ jugador\ jugador_i\ edificio_j\ prioridad)\ \forall\ prioridad \in \{1,\ 2,\ 3\}$$

5. Pruebas realizadas

En este apartado se pretende validar el Sistema Basado en el Conocimiento implementado a través de 5 escenarios.

5.1. Escenarios de pruebas

La primera prueba consta de un escenario copiado del juego proporcionado de *java*. Por tanto, la disposición de losetas, edificios y barcos es la misma. Esta prueba permite comprobar si el sistema experto obtiene o no una solución. Además, las prioridades de los edificios iniciales son idénticas para ambos jugadores.

La segunda prueba pretende mostrar que el sistema experto no ha sido programado específicamente para pasar la primera prueba. Para ello, se han permutado los recursos que proporcionan las losetas a la oferta y la loseta de intereses. Las prioridades de los edificios iniciales siguen siendo idénticas para ambos jugadores.

La tercera prueba ha consistido en permutar el orden de las cartas en el mazo manteniendo la estrategia estática de prioridades de edificios. De hecho, se ha forzado una situación algo extrema. Edificios esenciales como la *PISCIFACTORÍA* (posición 3 mazo 2), la *PANADERÍA* (posición 5 mazo 2) o *MATADERO* (posición 3 mazo 3) han sido colocados al final de la pila de mazos. Esto permite observar cómo se han comportado los jugadores con límites de recursos alimenticios.

La cuarta prueba ha consistido en permutar las prioridades de los edificios en el caso extremo de la prueba 3 para comprobar si hay margen de mejora al cambiar las prioridades y hacer que se centren en otros objetivos.

La quinta y última prueba ha sido proponer sobre el escenario de la primera prueba prioridades diferentes a los jugadores para que no vayan ambos a por los mismos edificios y tengan, como en la vida real, comportamientos distintos dentro de la partida.

5.2. Análisis de Resultados

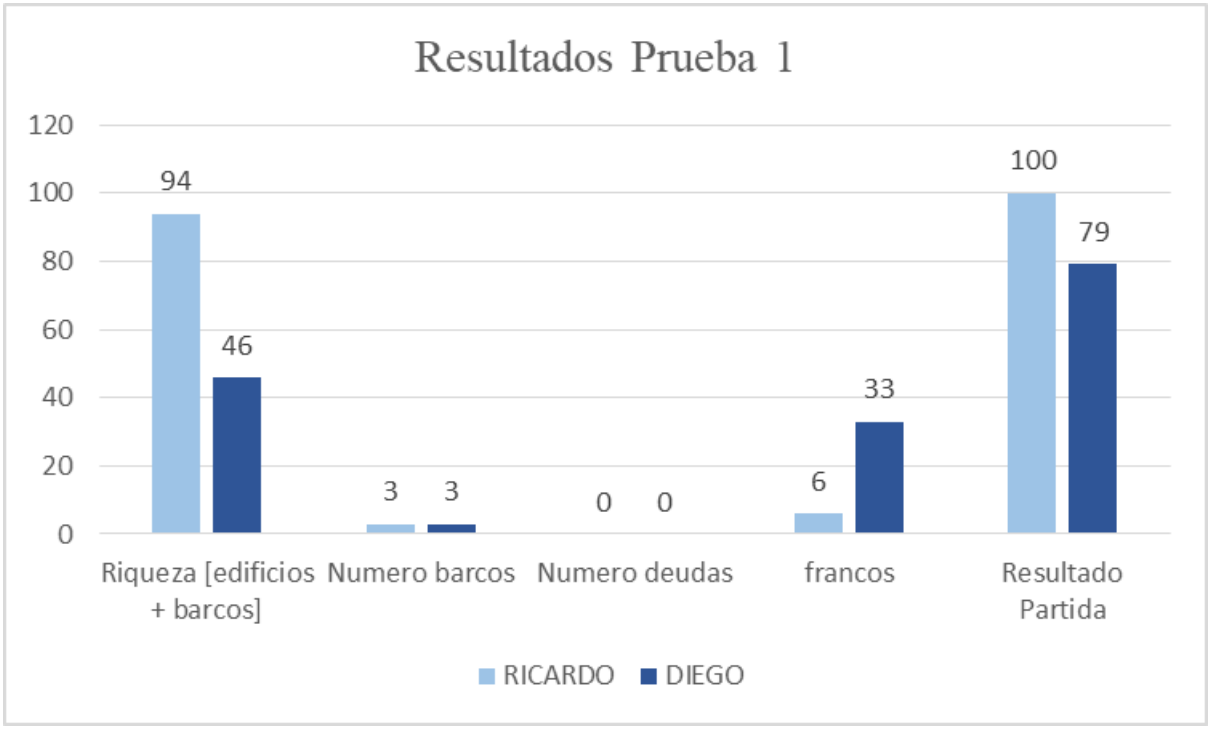


Figura VI - Resultados Prueba 1

Como se muestra en la figura VI, los resultados obtenidos son relativamente buenos. Ningún jugador termina la partida con deudas, ambos son capaces de construir barcos y edificios. Sin embargo, nótese que la cantidad de francos del jugador *DIEGO* es absurdamente alta. Esto es debido a que no se ha implementado en la estrategia que los jugadores compren edificios o barcos con francos si estos pueden ser construidos.

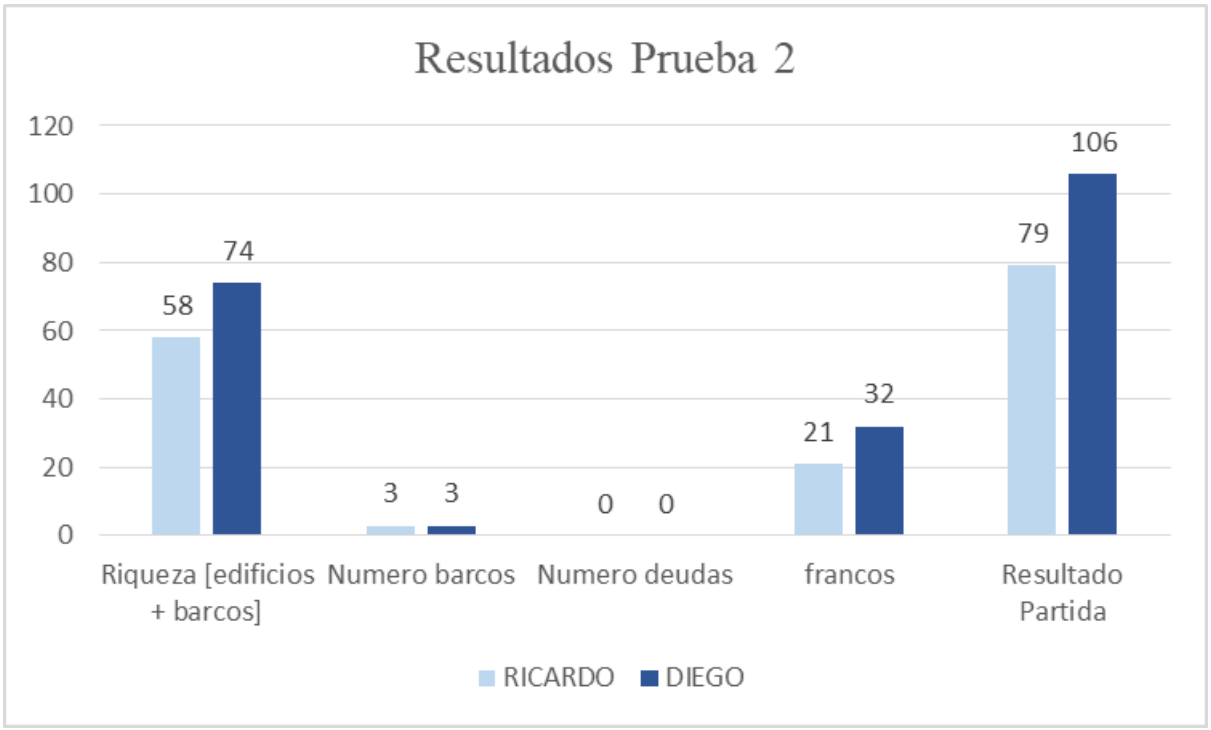


Figura VII - Resultados Prueba 2

Como se observa, los jugadores son capaces de desenvolverse en escenarios nuevos. La permutación de las losetas de oferta sí implica obtener un resultado distinto pero con valores parecidos. Ambos jugadores han sido capaces de obtener bastante riqueza a través de edificios y barcos pero siguen teniendo el problema del efectivo.

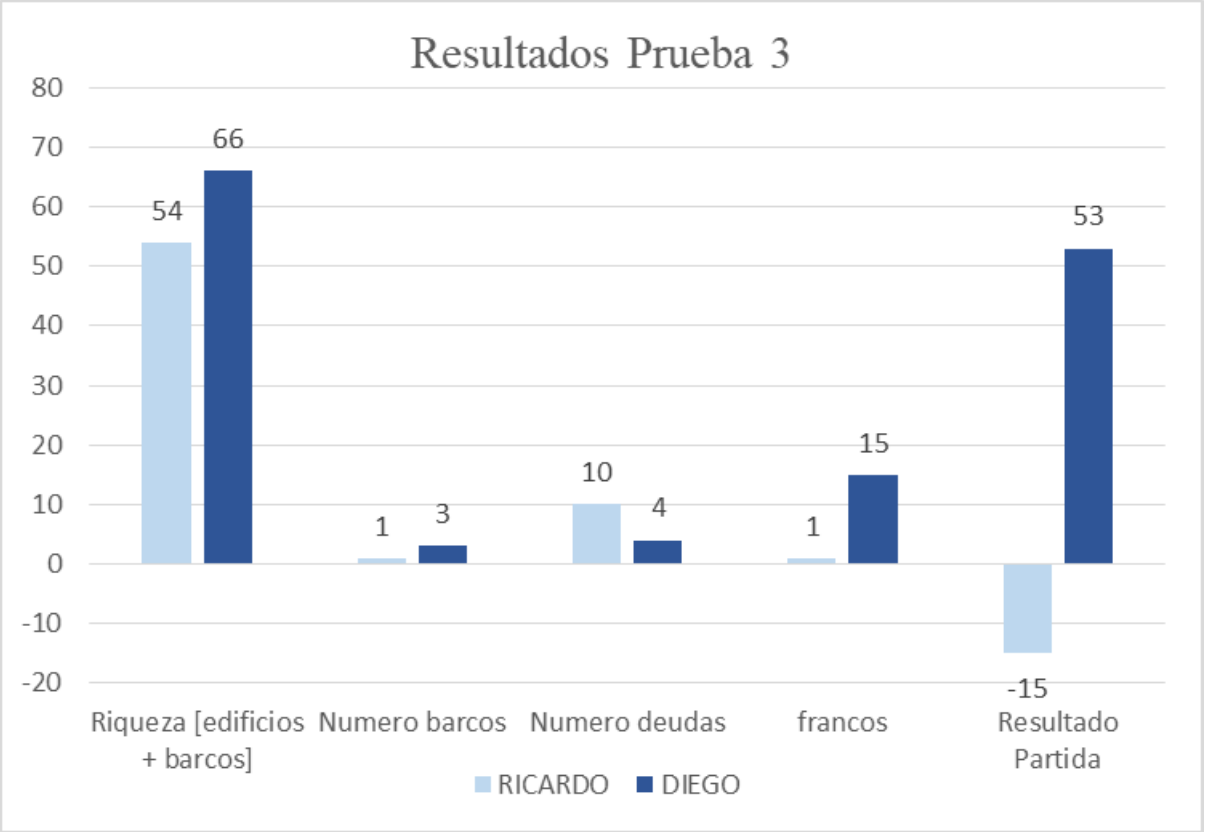


Figura VIII - Resultados Prueba 3

En este caso, al enfrentar a los jugadores expertos a semejante situación adversa se puede observar cómo efectivamente, la escasez de recursos alimenticios obliga a los jugadores a endeudarse. Sin embargo, la estrategia principal sigue funcionando. Son capaces de construir edificios y barcos.

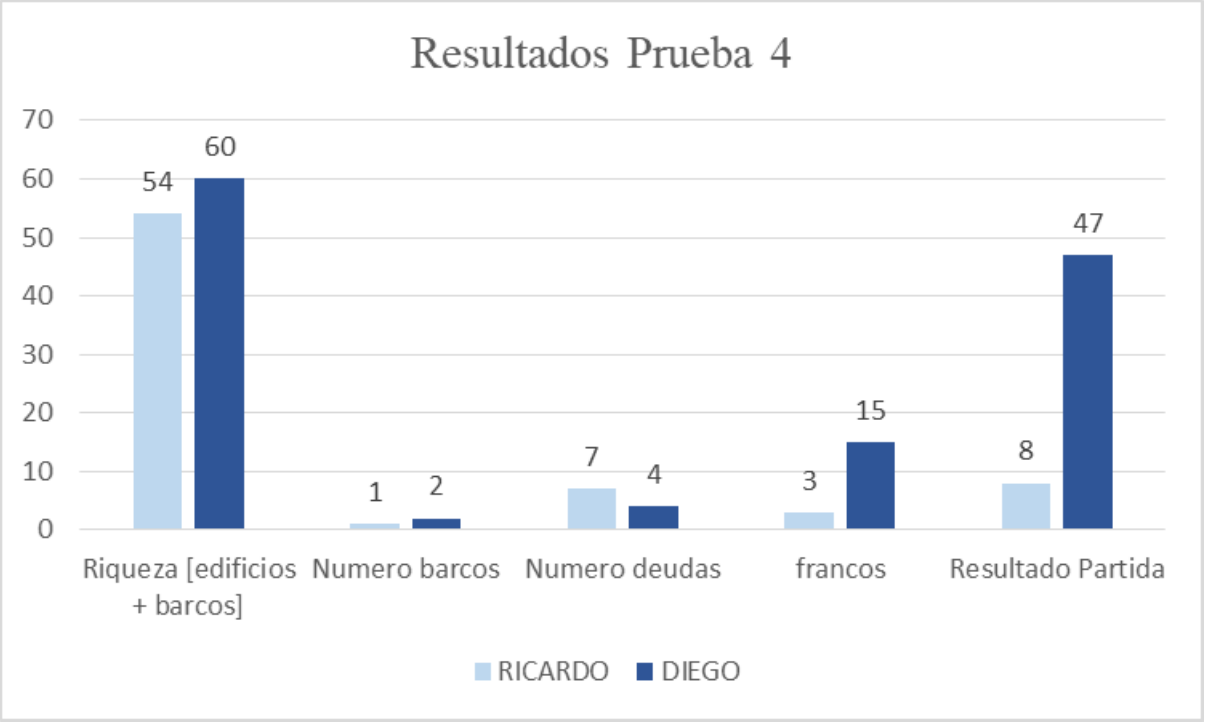


Figura IX - Resultados Prueba 4

Tras haber modificado las prioridades de los edificios, se puede determinar que las posiciones de las cartas en un mazo son determinantes para el resultado de la partida. Siguen necesitando endeudarse en grandes cantidades para poder pasar de ronda.

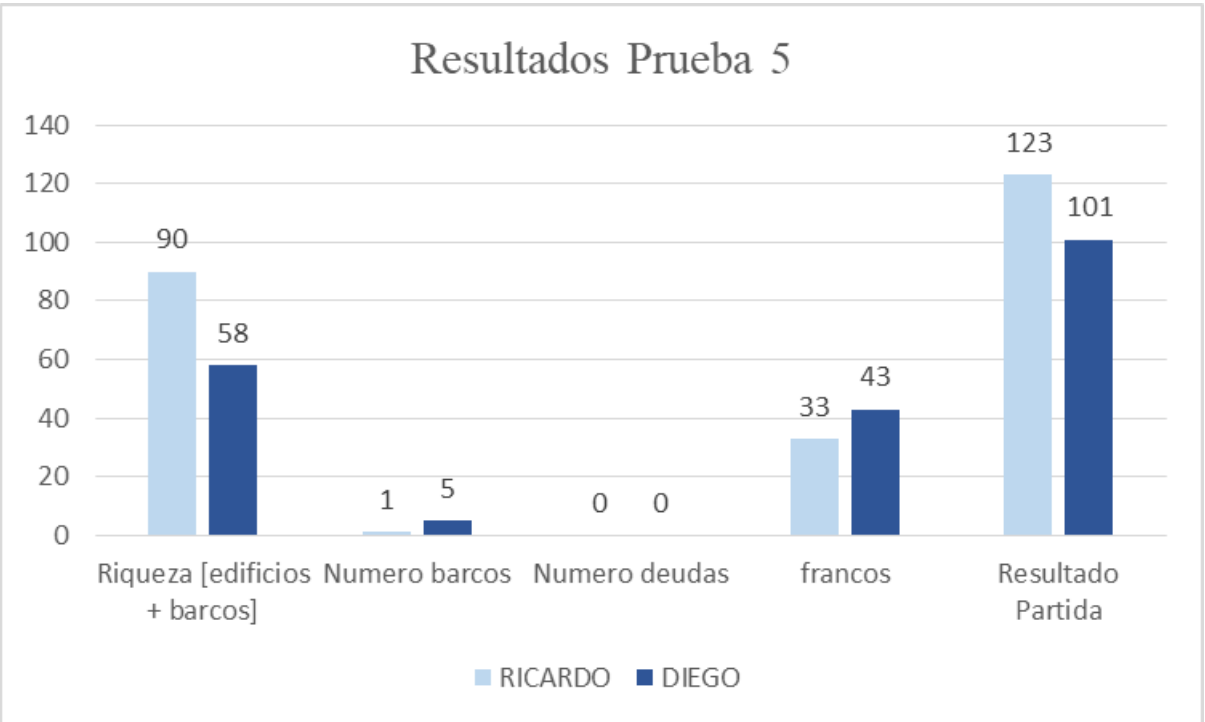


Figura X - Resultados Prueba 5

Por último, en esta prueba se permite obtener el máximo rendimiento de los jugadores expertos. Ambos jugadores tienen distintas prioridades en los edificios lo que permite que cada

jugador vaya a desempeñar su estrategia sin grandes repercusiones en la estrategia del otro jugador. Sin embargo, se sigue observando que los jugadores acumulan ingentes cantidades de francos.

Como se puede observar, la estrategia no implementa la compra de edificios con *FRANCOS* salvo casos excepcionales. Esto hace que los jugadores vayan almacenando cantidades sustanciales de *FRANCOS* sin ningún sentido. El enfoque que pensamos en su momento era el de dejar los francos como liquidez en los jugadores, esto simplifica mucho la estrategia ya que al ser aversos al riesgo no van a arriesgarse a comprar un edificio que posteriormente pueden necesitar vender para hacer frente a ciertos pagos.

6. Conclusiones

Consideramos que la solución propuesta como implementación del juego *Le Havre* a través de sistemas basados en reglas es adecuada. Los resultados de los jugadores podrían ser mejores, dado que en las sesiones de adquisición de conocimiento los jugadores superaban los 100 puntos de riqueza con relativa facilidad.

No obstante, las estrategias implementadas y su funcionamiento en *CLIPS* otorgan resultados satisfactorios aunque mejorables. Los jugadores desarrollan la partida e incluso a través de las instanciaciones y ejecuciones de reglas podemos ver una "simulación" del razonamiento de un experto.

La generalización de las reglas ha sido una de las mayores dificultades encontradas. En nuestra opinión, seguramente sea posible agrupar de manera más eficiente las reglas de ejecución de edificios, así como las de obtención de deseos de las reglas estratégicas. Debido a las dificultades encontradas, al tiempo invertido y a la restricción del mismo, no nos ha sido posible mejorar nuestra solución en este sentido.

El funcionamiento de las reglas estratégicas del juego es relativamente complejo, es cierto que en ocasiones toma conocimiento estático y las decisiones no son generadas puramente por el transcurso de la partida, pero en la mayoría de los casos sí es así, y por tanto quedamos satisfechos con la completitud aportada.

Las simplificaciones realizadas sobre el juego son muy reducidas. La más relevante quizás sea la relajación de los costes energéticos. La inclusión de este tipo de restricción en el uso de cartas no debería ser demasiado complejo, podrían incluso integrarse como "inputs" necesarios para el uso de un edificio. Sin embargo, sí puede suponer un reto a la hora de replantear la estrategia.

7. Comentarios finales

Para finalizar el documento, nos gustaría exponer nuestra opinión sobre la herramienta *CLIPS* y el desarrollo de la práctica.

Hemos podido entender la complejidad que supone tratar de transferir el conocimiento de un experto a un sistema basado en reglas. En concreto, la parte dedicada al razonamiento del experto y sus estrategias ha supuesto un gran reto debido a la casi nula experiencia que teníamos en

planteamientos de este tipo. Estamos seguros de que los sistemas basados en reglas son una pieza fundamental en el dominio de la Inteligencia Artificial.

Somos conscientes de que seguramente nuestro planteamiento ontológico tenga margen de mejora, y que posiblemente el cómo hemos modelado el juego sea la causa de nuestros mayores quebraderos de cabeza mientras programábamos la solución. Ante esto, sólo se nos ocurre una manera de mejorarlo: practicar y enfrentarnos a más problemas de este tipo, y por ello concluimos afirmando que la realización de esta práctica nos ha sido de verdadera utilidad.