



Universidad  
Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2020/2021

**Ingeniería del Conocimiento**

# **Práctica 1: Sistemas de Producción**

## **Autores:**

Alba Reinders Sánchez      100383444  
Alejandro Valverde Mahou      100383383

Grupo 83      Leganés

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Manual técnico</b>	<b>4</b>
2.1. Ontología . . . . .	5
2.2. Reglas . . . . .	6
2.2.1. Reglas genéricas . . . . .	6
2.2.2. Reglas Twister . . . . .	6
2.2.3. Reglas Tres en Raya . . . . .	7
<b>3. Manual de usuario</b>	<b>8</b>
<b>4. Pruebas realizadas</b>	<b>9</b>
<b>5. Conclusiones</b>	<b>10</b>
<b>6. Comentarios personales</b>	<b>11</b>

## 1. Introducción

En esta primera práctica se aborda la implementación de un **sistema de producción (SP)** en CLIPS que lleve a cabo la ejecución de una sesión de un especialista con un paciente en la que interactúan un humano y un robot NAO, donde este último adopta el papel de especialista.

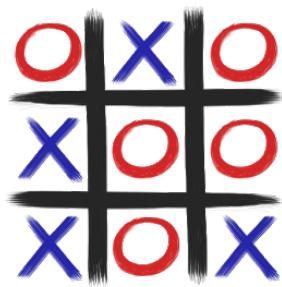


Robot NAO

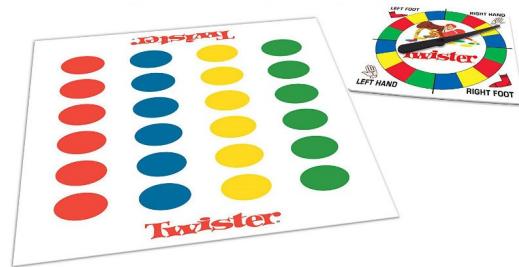
En concreto, se crea un *SP* con toda la información respecto a los personajes y el entorno en el que se lleva a cabo la interacción, así como posibles desviaciones durante la sesión en las que el robot debe reaccionar y modificar sus acciones en consecuencia.

La interacción entre el paciente y el robot se realiza a través del desarrollo de 2 juegos: el **Twister** y el **Tres en raya**. Además, el sistema está adecuado para tratar con pacientes que puedan presentar 2 personalidades distintas, aparte del comportamiento base, **despistado** y **energético**.

Se han elegido estos juegos por ser ejemplos de actividades sencillas, que requieren de un número reducido de reglas, para demostrar las aplicaciones de los *SP*. Por el mismo motivo, se han elegido únicamente 2 posibles comportamientos básicos. Si se deseara aplicar sobre un problema real no serían suficientes, y sería necesario tener en cuenta un mayor número de personalidades.



Juego del Tres en Raya



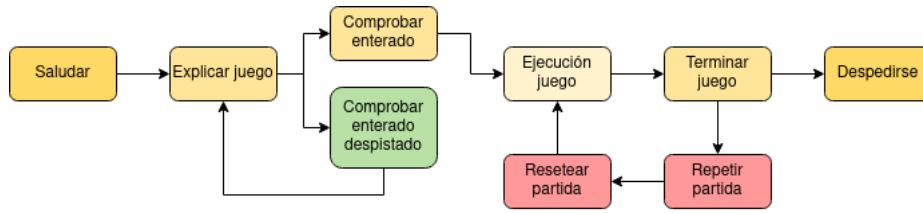
Juego del Twister

El documento consiste en el manual técnico con la descripción de la implementación, el manual de usuario con la explicación de cómo usar el programa, las pruebas realizadas y el análisis de los resultados, y para finalizar una serie de conclusiones y comentarios personales.

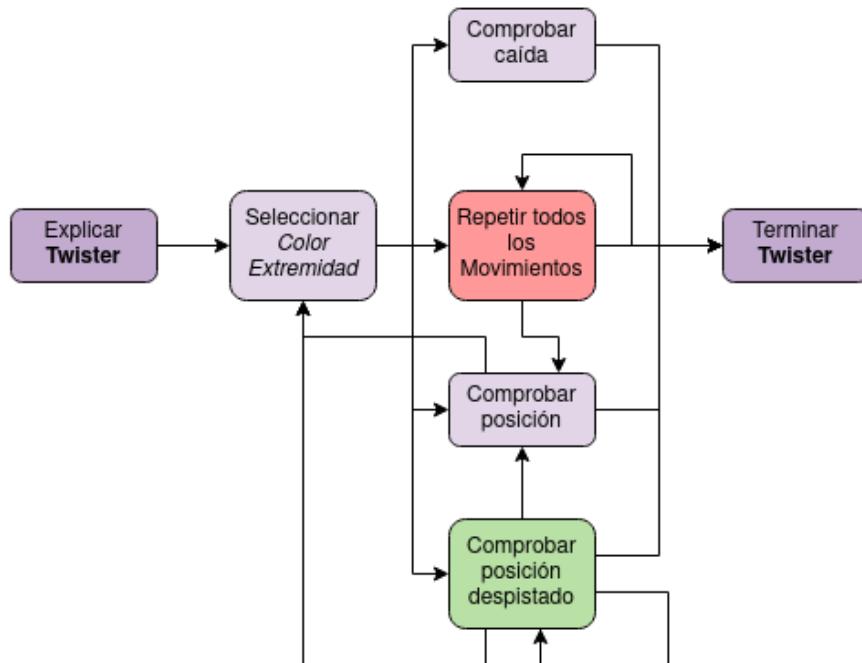
## 2. Manual técnico

El SP está compuesto por la **ontología** y las **reglas**. A continuación, se explican ambas justificando todas las decisiones que se han tomado durante la implementación.

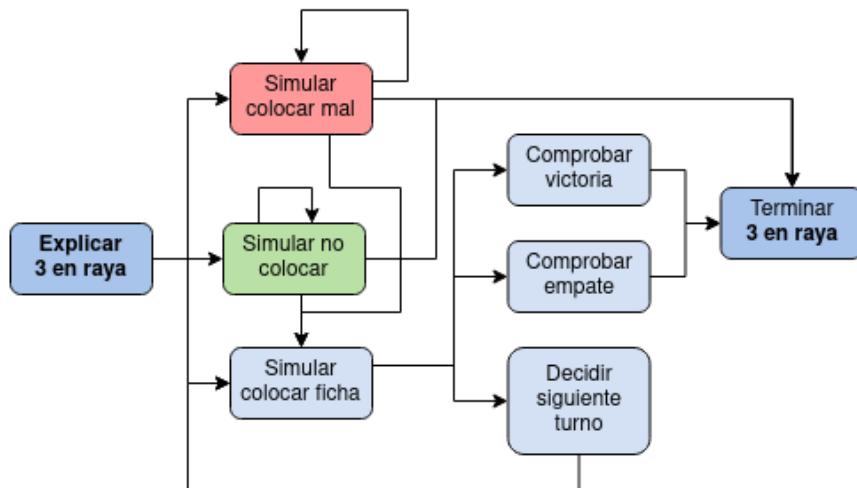
Los flujos de acciones que se han planteado para resolver los problemas son los siguientes, donde los estados de color rojo solo se pueden realizar si el *paciente* es *enérgico* y los de color verde si el *paciente* es *despistado*.



Flujo genérico de una sesión



Flujo del Twister



Flujo del Tres en Raya

## 2.1. Ontología

La ontología que se plantea para la solución está compuesta de las siguientes clases.

- **JUEGO** → Representa el juego que se va a llevar a cabo a lo largo de la sesión. Contiene, además del nombre (*id*) del juego, la explicación del mismo, junto con la indicación de quién debe empezar (*paciente* o *robot*). Esto se ha decidido porque para cada juego comienza uno de los dos:

En el caso del ***Twister***, el *robot* será siempre el que empiece la partida, dado que será el *paciente* el que siempre 'juegue', mientras que el *robot* elige las acciones.

Por otro lado, se ha decidido que para el caso del ***Tres en Raya*** siempre empiece el paciente. Esto se ha hecho porque, en este juego, el jugador que comienza tiene ventaja y, dado que el juego forma parte de una terapia, se quiere potenciar que el *paciente* gane con mayor probabilidad.

- **CONTROL** → Esta clase tendrá una única instancia, y se usa para localizar las distintas operaciones que controlan el sistema en una sola instancia. En el caso excepcional de que se repita la partida (comportamiento que solo puede darse con *pacientes* de personalidad *enérgica*) esta instancia se borra y se reinicia.

Esta clase permite controlar de quién es el turno, el número de ciclos, junto al número máximo de ciclos, el número de fallos, el juego que se lleva a cabo en la sesión, y el número de veces que se repite la partida.

- **PACIENTE** → Representa al paciente que está en la sesión. Se ha decidido usar una clase en lugar de un *fact* para representarlo porque, además de la personalidad del paciente, también almacena el nombre del mismo. Esto se hace para poder enviar mensajes personalizados, y que el paciente se encuentre en un ambiente más cómodo e informal.

- **COLOR** → Representa cada uno de los cuatro colores posibles para el juego del ***Twister***. Puede ser *rojo*, *verde*, *amarillo* o *azul*.

- **EXTREMIDAD** → Representa cada una de las cuatro extremidades posibles para el juego del ***Twister***. Puede ser *pie derecho*, *pie izquierdo*, *mano derecha* o *mano izquierda*.

- **ELECCION** → Representa cada una de las elecciones o comandos que hace el *robot* y que tiene que realizar el *paciente* en el juego del ***Twister***. Además del color y la extremidad elegidas, para poder realizar comprobaciones adicionales, se incluye el número de veces que se ha tenido que repetir una misma elección, y el orden por el que se han dicho.

- **CASILLA** → Representa cada una de las 9 casillas del juego del **Tres en Raya**. Además de sus coordenadas, se tiene su valor, que puede ser *vacío* si no se ha colocado nada todavía, o *paciente* o *robot*, en función de quién coloque la ficha.

## 2.2. Reglas

### 2.2.1. Reglas genéricas

Estas reglas son las que sirven para ambos juegos y donde da igual la personalidad del paciente:

- **Saludar**: el *robot* saluda de manera personalizada al *paciente* por su nombre.
- **Explicar**: el *robot* explica al *paciente* el juego que se haya elegido.
- **Comprobar-enterado**: el *robot* comprueba que el *paciente* se haya enterado de la explicación del juego.
- **Despedirse**: el *robot* se despide de manera personalizada del *paciente* por su nombre.

Si el *paciente* tiene de personalidad *despistado*, existe la regla **Comprobar-enterado-despistado** para repetir la explicación del juego y la regla **Demasiadas-equivocaciones-despistado** para terminar la sesión porque se ha equivocado más de 5 veces y por tanto se considera que el *paciente* puede no estar muy atento para continuar con el juego.

Se añade la regla **Repetir-partida** para volver a jugar otra partida. Esto sucede exclusivamente si el *paciente* es *enérgico*, ya que se considera que puede que tenga más energías para seguir jugando.

Por lo tanto, una vez que se acaba una partida se activa esta regla y, dependiendo del juego, las siguientes: **Resetear-partida-tres-en-rayo** y **Resetear-partida-twister**. Estas sirven para reiniciar las variables del juego.

Además, se crea la regla **Auxiliar-fin-reseteo**, que lo que hace es finalizar el proceso de reinicio de variables.

Por último, se tiene la regla **Demasiadas-equivocaciones-energico**, la cual tiene como fin terminar la sesión si el *paciente* intenta colocar un ficha cuando no es su turno más de 3 veces. Este comportamiento es exclusivo del *enérgico*, ya que se considera que ha hecho demasiadas trampas para una sola partida.

### 2.2.2. Reglas Twister

En cuanto a las reglas exclusivas para el **Twister**, para un comportamiento normal, se crean las siguientes:

- **Robot-selecciona**: el *robot* selecciona de forma aleatoria un color y una extremidad para que el *paciente* realice la acción.
- **Comprobar-posicion**: el *robot* comprueba que el *paciente* ha realizado correctamente la acción que le ha dicho y aumenta el contador de ciclos.
- **Comprobar-caida**: el *robot* comprueba que el *paciente* se ha caído y que por tanto se acaba el juego.
- **Terminar-juego**: por otro lado, si el contador de ciclos alcanza los ciclos máximos, se termina el juego.

Según la personalidad del paciente se pueden activar además las siguientes reglas:

Si es *despistado*:

- **Comprobar-posicion-despistado**: si el *paciente* es *despistado* se puede activar esta regla con el objetivo de que el *robot* le repita la acción al *paciente* porque no la esté haciendo.
- **Saltar-ciclo**: cuando el *robot* ha tenido que repetir la misma acción más de 2 veces porque el *paciente* no la realiza se procede a pasar al siguiente movimiento.

Si es *enérgico*:

- **Comprobar-posicion-energico:** si el *paciente* es *enérgico* se puede activar esta regla para simular que el *paciente* se ha movido de su posición porque no se esté quieto y por tanto el *robot* le avisa y le repite todos los movimientos.
- **Repetir-orden:** regla para repetir todos los movimientos en orden gracias al orden que acompaña a cada elección.
- **Auxiliar-repetir-orden:** sirve para dar por finalizado la repetición de los movimientos.

### 2.2.3. Reglas Tres en Raya

Las reglas creadas para un comportamiento normal del **Tres en Raya** son:

- **Simular-colocar:** sirve para representar que uno de los jugadores ha colocado una ficha en una casilla vacía.
- **Comprobar-victoria:** engloba a **Comprobar-victoria-h**, **Comprobar-victoria-v**, **Comprobar-victoria-d1** y **Comprobar-victoria-d2**. Comprueban si, tras colocar una ficha, alguno de los dos jugadores gana la partida. Tiene que dividirse en varias reglas porque existen estas 4 posibles condiciones de victoria (horizontal, vertical, y las dos diagonales). Estas reglas tienen que tener mayor prioridad que la regla de empate, para que el sistema no pueda elegir empate cuando se cumple victoria.
- **Comprobar-empate:** se comprueba si se han rellenado las 9 casillas, y no ha habido victoria de ningún jugador.
- **Decidir-siguiente-robot:** como el *paciente* siempre comienza la partida, será el turno del *robot* siempre que haya colocadas un número impar de fichas.
- **Decidir-siguiente-paciente:** por el mismo motivo que la regla anterior, será el turno del *paciente* siempre que haya colocadas un número par de fichas.

Para cuando el *paciente* es *enérgico* se tiene la regla **Simular-colocar-mal** que simula cuando el *paciente* intenta colocar una ficha sin ser su turno porque tiene mucha energía y por tanto no puede esperar a su turno. La regla lo que hace es sumar uno al contador de fallos y decirle que no puede colocar porque no es su turno.

Si el *paciente* es *despistado* se tiene la regla **Simular-no-colocar** que simula cuando el *paciente* no coloca una ficha cuando es su turno porque no está muy atento y se le olvida que es su turno. La regla lo que hace es sumar uno al contador de fallos y recordarle que es su turno.

### **3. Manual de usuario**

#### 4. Pruebas realizadas

## 5. Conclusiones

## 6. Comentarios personales