UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Informática

PROYECTO DE LABORATORIO 2 – PARADIGMA LOGICO

Paradigmas de Programación

Gerardo Ignacio Pérez Encina 21.343.296-6

Sección:

13204-0-A-1

Profesor:

Edmundo Leiva Lobos

Fecha:

29 de noviembre del 2024

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Conocer diferentes paradigmas de programación es esencial para que los programadores desarrollen soluciones efectivas ante problemáticas considerando todos los puntos de vista posibles y eligiendo los mejores métodos para escribir sus programas.
En el presente laboratorio se nos presenta la creación del famoso juego Conecta 4, mediante un programa en el lenguaje Prolog (PROgrammation en LOGique) bajo el paradigma lógico de programación.
El documento consta de introducción, descripción del problema, análisis del problema, diseño de la solución, consideraciones de implementación, instrucciones de uso, resultados y autoevaluación, y conclusiones.

2. DESCRIPCIÓN DE LA PROBLEMÁTICA

2.1. PROBLEMA

Cuando investigamos más sobre el juego de Conecta 4, nos damos cuentas que nuestro problema es la implementación de este, pero primero hablemos sobre que es Conecta 4, este juego trata sobre un tablero de 6x7 el cual se juega con dos jugadores, cada jugador tiene una cantidad de piezas y en cada turno solo pueden colocar una pieza, la cual cae desde arriba hasta el fondo o hasta donde sea posible, para poder determinar un ganador se debe de hacer una línea vertical, horizontal o diagonal.

2.2. DESCRIPCIÓN DEL PARADGIMA

Para entender los métodos a emplear en la resolución del problema, es útil conocer algunos conceptos claves que facilitan su comprensión. A continuación, se presentan y describen:
1.- **Paradigma**: Es una manera de entender el mundo o una forma establecida de hacer las cosas, especialmente en un contexto especifico. Según Kuhn (1962), se refiere a “realizaciones científicas universalmente aceptadas que, durante un tiempo, ofrecen modelos para plantear y resolver problemas en una comunidad científica”.
2.- **Paradigma de programación lógica**: El paradigma lógico es un modelo de programación que utiliza lógica formal como base para resolver problemas. Los programas consisten en un conjunto de hechos y reglas que describen relaciones entre entidades, y las soluciones se obtienen mediante deducción lógica. En este paradigma nos centramos en declarar que es lo que queremos lograr y no en como lograrlo paso a paso
3.- Hechos y Reglas: En Prolog, los hechos y reglas son fundamentales para representar el conocimiento:
- Hechos: Expresan relaciones que siempre son verdaderas
- Reglas: Definen relaciones derivadas mediante logica

4.- Predicado: Un predicado es una representación lógica que describe propiedades o relaciones entre objetos. Los predicados funcionan como “funciones” en el paradigma funcional, pero en Prolog devuelven verdadero o falso según si las relaciones establecidas son validas

5.- Paradigma lógico en Prolog: El paradigma lógico se basa en el uso de unificación, Para desarrollar la solución, en este primer laboratorio se utiliza el paradigma funcional.

- Unificación: Prolog iguala términos y encuentra valores que satisfacen las relaciones descritas en las reglas
- Búsqueda: Prolog busca las soluciones probando hipótesis y verificando si satisfacen los hechos y reglas
- Backtracking: Si una solución no es válida, Prolog retrocede automáticamente y prueba otras opciones

3. ANÁLISIS DEL PROBLEMA

Para solucionar el problema, se desea crear el juego Conecta 4 en el lenguaje de programación Scheme, bajo el paradigma funcional declarativo. Esto quiere decir que se debe crear un programa que, mediante funciones, se consiga poder jugar una partida de Conecta 4 sin problemas. Por consiguiente, se debe de desarrollar una solución en la cual podamos de realizar el juego Conecta 4 sin ningún problema.

El Conecta 4 está constituido por: Jugadores, tablero y piezas. Y como requisitos, el programa de debe cumplir con las siguientes funcionalidades: Creación, gestión de la partida, gestión de jugadores (Registrar nuevo jugador, modificar información de algún jugador existente, consultar estadísticas de los jugadores), desarrollo del juego (Realizar los movimientos, validar los movimientos según las reglas del juego, detectar victoria, empate o continuación del juego, mostrar el estado actual del tablero).

4. DISEÑO DE LA SOLUCIÓN

Para implementar el juego Conecta 4, surge la necesidad de crear diferentes abstracciones de elementos mediante TDAs. De modo de ir conteniendo tipos de datos abstractos dentro de un gran
TDA que represente el juego
Debido a que el lenguaje Prolog está relacionado con predicados los cuales funcionan a través de verdaderos y falsos, (rellenar más acá). A continuación, se mencionan los tipos de datos abstractos a usar.
TDA Player:
Es el jugador que uno “elige” al crear la partida, este tiene una id, nombre, color, victorias, derrotas, empates y cantidad de fichas, a lo largo que se vaya jugando, sus últimas cuatro características se irán actualizando conforme vaya pasando la partida.
TDA Board:
Es el tablero del juego, la base de este, se podrá crear un tablero, el jugador podrá poner fichas en el tablero y verificar el estado de la partida (Si se ganó, perdió, o hay un empate).
TDA Piece:
Es la pieza que el jugador seleccionara, esta puede ser del color la cual el jugador desee.
TDA Game:
Este es el “TDA madre” el cual tendrá a las demás para poder conformar el juego, este servirá para poder actualizar la partida, es decir, nos mostrara como se ponen las fichas en el tablero, verificar de quien es el turno y etc, también ver el historial y poder visualizar el tablero.

Además de las TDAs, se utilizaron algunas funciones auxiliares que sirven para poder realizar de mejor manera algunas funciones, o también algunos selectores como puede llegar a ser player_color, el cual permite obtener el color del jugador.

Por otro lado, se crearon funciones auxiliares especificas para distintos TDAs que son utilizadas. Por ejemplo, la siguiente función (check_rows) en la cual revisa cada fila del tablero, verificando si es que en cada una hay 4 piezas consecutivas, esta actúa para el TDA Board, en la función de check_horizontal_win

5. CONSIDERACIONES DE IMPLEMENTACIÓN

5.1. ESTRUCTURA DEL PROYECTO

El proyecto se basa en la encapsulación de diferentes TDAs dado que todos están contenidos bajo el TDA principal llamado “game”. Donde cada TDA tienen sus propias funciones que facilitan la operación de estos.

5.2. INTERPRETE USADO

El programa fue totalmente desarrollado en el lenguaje Prolog, escrito en el software Notepad+ y compilado en el programa SWI-Prolog; el cual con esta uno puede realizar un seguimiento con el flujo de ejecución de las consultas con la función “trace”, lo que nos ayuda bastante a la hora de querer saber si el programa se esta comportando de la manera en la cual uno desea.

6. INSTRUCCIONES DE USO

Para poder poner a prueba el programa, uno debe de abrir la aplicación “SWI-Prolog”, luego en el apartado de arriba, apretar donde dice File -> Consult y seleccionar un archivo de los scripts de prueba, luego, para poder hacer la consulta, uno debe de colocar “main.” Sin las comillas.

7. RESULTADOS Y AUTOEVALUACIÓN

Los resultados correspondientes a las RF se detallan mas abajo en el anexo. Sin embargo, se puede afirmar que el programa funciona correctamente hasta el RF 18, siempre y cuando se respeten las instrucciones de uso.

Habiendo dicho eso, el proyecto ha demostrado ser exitoso en la implementación del juego Conecta 4. El código cumple con los objetivos planteados, logrando así una simulación completa del juego. Sin presentar muchos errores significativos.

En cuanto a la autoevaluación, se considera que se ha logrado cumplir con todas las RF esenciales para poder crear el Conecta 4. Los elementos implementados cumplen mis expectativas sobre lo que quería lograr con este trabajo, la expectativa de poder ofrecer una experiencia de juego fluida y sin mayores errores, siempre y cuando se sigan las instrucciones de uso correspondientes.

8. CONCLUSIONES

Este trabajo represento un desafío, especialmente al pasar desde un paradigma funcional, a un paradigma lógico, donde lo único que se puede decir que se “mantiene” es la recursión. Este cambio de enfoque requirió un poco de tiempo para poder adaptarse y pensar mas en “lo que yo quiero que haga” mas que en “los pasos a seguir”. Sin embargo, a medida fue pasando el tiempo, lo que fue complicado en un principio, paso a ser mas y mas fácil, lo que pensaba que podrían llegar a ser varias líneas, eran en realidad unas pocas.

El proyecto se completo en un 100%, esperando que no haya errores o que se hayan introducido bien los pasos a seguir. Disfrute de hacer este trabajo, y me siento satisfecho con lo que he logrado, pero, aun así, me gustaría indagar aún más en este lenguaje de programación debido a que es una de las bases para la inteligencia artificial.

9. REFERENCIAS

Kuhn, T. S. (1962). *La estructura de las revoluciones científicas.*

Spigariol, L. (2005). *Fundamentos teóricos de los Paradigmas de Programación.* Buenos Aires: Facultad Regional Buenos Aires Universidad Tecnológica.

10. Anexo

1.-

Nombre TDA	Dato abstracto	Funciones asociadas
Board	Representación del tablero, donde se colocarán las fichas.	board, can-play? play_piece, check_vertical_win, check_horizontal_win, check_diagonal_win, who_is_winner
Player	Representación del jugador con sus atributos.	Player, update_stats
Piece	Representación de una pieza para jugar sobre el tablero.	Piece
Game	Representación del juego conecta 4, formandose a partir de las TDAs mencionadas anteriormente.	game, game_history, is_draw, get_current_player, game_get_board, end_game, player_play

2.-

Función	Pruebas exitosas	Pruebas falladas	Anotacion
TDA Player	100%	0%	
TDA Board	100%	0%	
TDA Piece	100%	0%	
TDA Game	100%	0%	
can_play?	100%	0%	
play_piece	100%	0%	
check_vertical_win	90%	10%	Los intentos fallidos son debido a que no lograba detectar bien una pieza ya que no era de la forma “red” o “yellow” (Con comillas)
check_horizontal_win	100%	0%	
check_diagonal_win	100%	0%	
who_is_winner	67%	33%	Las veces que fallas es debido a que cuando se hace una victoria diagonal no lo detecta
game_history	100%	0%	
Is_draw?	100%	0%	
update_stats	100%	0%	
get_current_player	100%	0%	

get_current_board	100%	0%	
end_game	100%	0%	
player_play	95%	5%	No logra restar las piezas correctamente