Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática
Paradigmas de Programación

Paradigmas de Programación
Proyecto Semestral de Laboratorio

Laboratorio 2 Paradigma Lógico - Prolog

Alumno: Benjamin Antonio Ortega Quinteros
Profesor: Edmundo Leiva
Carrera: Ingeniería de Ejecución en Computación e Informática
Fecha: 29-11-2024

Índice

Introducción

En el presente informe se enfrenta una problemática sobre la implementación del juego Conecta 4, a través del uso del paradigma lógico en el lenguaje de programación Prolog.
El objetivo es resolver este problema planteado usando la abstracción mediante los conceptos fundamentales del paradigma lógico.
La estructura del informe contiene la introducción, descripción del problema, descripción del paradigma, análisis del problema, diseño de la solución, aspectos de la implementación, instrucciones de uso, resultados y autoevaluación, conclusiones, referencias bibliográficas y finalmente anexos.

Descripción del Problema

Conecta 4 es un juego de estrategia para dos jugadores que se juega en un tablero vertical de 6 filas y 7 columnas. Cada jugador tiene a su disposición 21 fichas, cada jugador tiene asociado un color diferente. Los jugadores se turnan para dejar caer fichas de su color desde la parte superior del tablero. Las fichas caen hasta la posición más baja disponible en la columna seleccionada. El objetivo del juego es ser el primero en formar una línea de cuatro fichas del mismo color, ya sea horizontal, vertical o diagonalmente.

El problema planteado es crear una solución informática del juego Conecta 4 en el paradigma de programación lógico a través del cumplimiento de ciertos requerimientos que deben ser abordados bajo este paradigma y otras condiciones propuestas.

Descripción del Paradigma

El paradigma de programación lógico se basa en la lógica formal y se centra en el “qué” en lugar del “cómo”. Utiliza hechos, reglas y consultas para deducir soluciones de manera declarativa.

Sin embargo, este paradigma presenta varias limitaciones, siendo una de las más destacadas la eficiencia. El uso de técnicas como el backtracking, que consiste en probar diferentes soluciones y retroceder cuando se llega a un punto sin avance, puede ser muy intensivo en términos de recursos. En comparación con otros paradigmas de programación, estas técnicas pueden consumir más recursos computacionales, afectando el rendimiento en problemas complejos y la complejidad de implementación de estos.

Análisis del Problema

Para enfrentar este problema se deben considerar los TDA´s básicos mínimos con los que se pueden implementar los requerimientos funcionales del proyecto.
En este caso son 4 TDA´s necesarios para la implementación del juego Conecta 4, Player, Piece, Board y Game.

Estas abstracciones son necesarias para abordar cada aspecto de la problemática en cuestión, por ejemplo, para determinar una victoria se debe tener una implementación clara del tablero y en este caso sería el TDA board.

Diseño de la Solución

La implementación se basará en el uso de TDAs y sus respectivas funciones para organizar los elementos principales del juego así como para lograr implementar parámetros del juego en cuestión. Dicho esto, se describirán los propósitos específicos de cada TDA y su uso en el programa:

- TDA Player: El propósito de este TDA es la fácil creación de los jugadores del juego así como el fácil acceso a las características de estos. En este TDA se desarrollaran predicados que permitan actualizar las estadísticas de los jugadores.

- TDA Piece: El propósito de este TDA es representar las fichas del tablero, para su posterior uso en el juego.

- TDA Board: El propósito de este TDA es representar el tablero como una lista de listas (matriz) de 6 filas y 7 columnas. Este TDA implementara una representación del tablero, por lo cual se desarrollaran predicados que permitan verificar ciertos estados dentro del juego.

- TDA Game: El propósito de este TDA es la fácil gestión y creación de partidas. Este TDA será el resultado del uso de todos los TDA anteriores y juntos permitirán que la implementación de este juego sea completada.

Cada TDA mencionado anteriormente tendrá sus respectivos predicados de selección, modificación, entre otros, para el correcto uso de la abstracción de los TDA requerida para desarrollar la solución.

Aspectos de la Implementación

La estructura del proyecto consiste en 5 archivos correspondientes a los TDA mínimos mencionados anteriormente, “board”, “piece”, “player”, “game” y finalmente el script de pruebas. Este último ejecuta los requerimientos de manera funcional y muestra el funcionamiento del programa.

No se usó ninguna biblioteca externa para el desarrollo de la solución. El proyecto se desarrolló en el lenguaje de programación Prolog, a través de SWI-PROLOG versión 9.2.8 y se usó el editor de código fuente Visual Studio Code.

Instrucciones de Uso

Los archivos de cada TDA deben estar en la misma carpeta, llamada “codigo_fuente_21542985_OrtegaQuinteros”, mientras que el script de pruebas debe estar afuera para asegurar el correcto funcionamiento del código y evitar cualquier error al momento de ejecutar el script.
Es necesario usar el intérprete de SWI-PROLOG al momento de realizar las consultas para poder usar el programa.

Al ejecutar el intérprete, nos dirigiremos a la sección “File” arriba a la izquierda y le daremos a la opción de consult, posteriormente nos dirigiremos a la carpeta en donde se encuentran los archivos.pl y seleccionamos el archivo “script_base_21542985_OrtegaQuinteros.pl” y le damos a abrir, luego, en la entrada de línea ingresamos “main.” y se ejecutará el programa, finalmente, cuando termine el script se debe presionar “Enter” para finalizar la ejecución.

Para efectos prácticos se agrego una línea de código a los scripts, dicha línea de código es:
game_get_full_board(G11, FullCurrentBoard)
Este predicado otorga el tablero completo, incluyendo el historial, y dada la implementación realizada muchos predicados reciben como parámetro el tablero completo y no solo el board. También, se modificó la línea de **is_draw** para que muestre en pantalla el resultado y no se pare la ejecución.

Resultados y Autoevaluación

En cuanto a los resultados de los requerimientos del proyecto, se logró cumplir las expectativas esperadas de los requerimientos funcionales y no funcionales.
A continuación se expresa el alcance de cada uno de estos requerimientos:

Requisitos Funcionales

- **TDA player:** Logrado Completamente
    - **update-stats:** Logrado Completamente
- **TDA piece:** Logrado Completamente
- **TDA game:** Logrado Completamente
    - **game-history:** Logrado Completamente
    - **is-draw:** Logrado Completamente
    - **get-current-player:** Logrado Completamente
    - **game-get-board:** Logrado Completamente
    - **end-game:** Logrado Completamente
    - **player-play:** Logrado Completamente
- **TDA board:** Logrado Casi Completamente
    - **can-play?:** Logrado Completamente
    - **play-piece:** Logrado Completamente
    - **check-vertical-win:** Logrado Completamente
    - **check-horizontal-win:** Logrado Completamente
    - **check-diagonal-win:** Logrado Completamente
    - **who-is-winner:** Logrado Casi Completamente

Cabe recalcar que en los casos en donde el alcance es de “Logrado Casi Completamente”, la principal diferencia es que el predicado funciona pero solo para una entrada en específica, en este caso, para el predicado board-who-is-winner, en donde el player 1 debe tener como color “red” para el correcto funcionamiento de este.

Conclusión

Podemos concluir que la implementación realizada que se logró en el desarrollo del juego Conecta-4 en el paradigma de programación funcional es bastante completa, ya que se implementan correctamente la gran mayoría de los requisitos funcionales requeridos para el desarrollo del juego de manera correcta.

En comparación al paradigma funcional, el declarar hechos en vez de funciones y la forma de escribir el código es más sencilla y fácil de entender.

La mayor dificultad del paradigma lógico es la manera de aplicar la abstracción para poder implementar predicados así también como entender la recursión en el lenguaje de programación Prolog.

Referencias

fedecarrion137. (19 de agosto de 2015). Paradigma Logico. TuParadigma. https://tuparadigma.wordpress.com/2015/08/19/paradigma-logico/

Ferestrepo, C. A. (s.f.). *Programación lógica: Teoría*. Paradigmas de programación. Recuperado de https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/proglogica.html