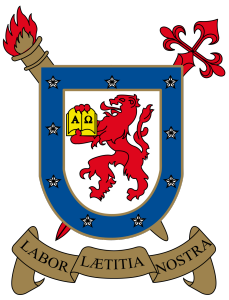

Laboratorio 2
Paradigmas de programación

Programación Lógica - Prolog

Facultad de Ingeniería - Universidad de Santiago de Chile

Estudiante: Mateo Valle Lacourt
Profesor: Gonzalo Martinez Ramirez

Índice

1. Introducción

En el presente informe se verán los aspectos de implementación y desarrollo de un juego de Conecta-4 programado en el lenguaje Prolog bajo el paradigma de programación lógica.

Conecta-4 es un juego de mesa de dos jugadores, los cuales deberán alinear cuatro fichas seguidas de forma vertical, horizontal o diagonal en un tablero posicionado de forma vertical de seis filas y siete columnas. El juego se basa en turnos, en los cuales un jugador coloca una ficha en cualquier columna del tablero y la ficha baja hasta lo mas abajo de este. Los turnos se van alternando hasta que se hayan conectado cuatro fichas seguidas, el cual será el ganador del juego o que no haya espacio en el tablero, lo cual cuenta como empate.

En esta actividad de laboratorio se planteó la creación de un programa que simule este juego. Dicho programa es hecho bajo el paradigma de programación lógica, este paradigma se basa en hechos y reglas lógicas, los cuales serán la base de conocimiento del programa, dicho conocimiento es consultado por un usuario, y el programa, mediante las reglas y hechos que el programador le entregó, debe buscar o inferir alguna posible respuesta.

Primero se verán unas breves descripciones sobre el problema y el paradigma de este laboratorio, después se hará un breve análisis del problema, seguido de una descripción del diseño de solución del problema, para luego detallar los aspectos de la implementación realizada. También se entregarán instrucciones de uso del programa. Finalizando con los resultados obtenidos y conclusiones sobre esta sesión de laboratorio.

2. Descripción breve del problema

El problema presentado en este laboratorio fue realizar un juego de Conecta-4 en el lenguaje de programación Prolog, bajo el paradigma de programación lógica, considerando algunas de las características del juego, tal como:

- Las fichas deben caer en lo más bajo del tablero.
- Verificar si hay cuatro fichas seguidas de forma horizontal, vertical y diagonal.
- Actualizar estadísticas de los jugadores involucrados en el juego.

El desafío del laboratorio fue hacer uso de este nuevo paradigma y lenguaje, los cuales son muy diferentes al resto de los demás.

3. Descripción breve del paradigma

El paradigma lógico se basa en la lógica matemática, donde el conocimiento se representa con hechos y reglas, las cuales pueden ser consultadas por un usuario. Describiendo mejor estos elementos, se puede decir que los hechos y reglas son datos e información que el programador le enseña al programa, y el usuario mediante consultas, puede preguntarle al programa si ciertas proposiciones son verdaderas o no. La respuesta depende únicamente si se le enseñó a inferir bien la información consultada al programa.

Un lenguaje de programación que se basa en este paradigma es Prolog, creado en 1972 por los franceses Alain Colmerauer, Robert Kowalski y Philippe Roussel, con el fin de llevar la programación a un nivel muy alto, parecido al lenguaje humano. Prolog tiene un enfoque declarativo y lógico, por lo que se pueden hacer reglas que funcionen como funciones, usando sus características como las entradas y salidas, incluyendo la recursión como herramienta para resolver problemas más generales y/o complejos.

El paradigma lógico fue unas de las bases de la inteligencia artificial conocida a la fecha, como ChatGPT. Debido a la forma en que funcionan estos dos: el programador le da información al programa, y el usuario consulta esta información.

4. Análisis del problema

La creación de un juego es complejo, por lo que se debe dividir en diferentes sub-problemas, que se complementen para dar una solución adecuada al problema planteado. En este proyecto, el juego se dividió en cuatro principales elementos: Jugadores, Piezas, Tablero y Estados de Juego.

El jugador debe contener información acorde a este, como su identificador, un nombre, el color y cantidad de fichas y estadísticas de victorias, derrotas y empates. Su utilidad es la de brindar información a los demás aspectos de implementación y es uno de los pilares de la solución.

La implementación de un tablero es fundamental para el juego. Es donde se tienen que hacer múltiples tipos de verificaciones, tales como verificar victoria de forma vertical, horizontal y diagonal. También se debe implementar la función de colocar pieza y que quede en la posición mas baja posible, dejando solo espacios libres arriba de esta. Todas estas funciones de juego serán implementadas con recursión, ya que es la única forma en la que el programa puede analizar la totalidad de casos en el tablero.

El estado de juego contiene todos los elementos antes mencionados, teniendo a los jugadores, el tablero actual de cada turno, el turno actual de jugador, y un historial de movimientos realizados.

El conjunto de todos estos elementos conforman una solución al juego de Conecta-4, y cada uno de estos elementos se necesitan entre si para lograr implementar cada función del juego.

5. Diseño de solución

La solución de problema está basado en la implementación de ciertos predicados, los cuales fueron estrictamente necesarios para la solución final. [1]

En las primeras etapas de solución, se implementaron los sub-problemas mencionados anteriormente como TDA's, representados mediante listas.

Lo primero que se construyó fueron los predicados constructores y selectores de los TDA's.

Luego se emplearon las funciones del tablero, para colocar piezas y detectar victorias en el tablero, las cuales se realizaron con recursividad. Esta fue la parte más difícil, ya que para las instrucciones de verificar victoria, se usaron predicados muy simples, y luego se implementaron otros predicados que usan estos simples a gran escala, así recorriendo todo el tablero. La mas complicada fue la de checkeo diagonal, primero se tuvo que implementar un predicado que calcula los primeros cuatro elementos de la diagonal de una matriz, luego aplicar ese predicado a toda una fila, y por ultimo aplicar este ultimo predicado a todas las columnas, llegando a verificar todo el tablero, luego repetir este proceso pero con la diagonal inversa, aplicando un predicado para invertir el tablero.

Una vez terminadas las funciones del tablero, se empezaron a implementar las funciones del TDA juego, como el predicado terminar juego y obtener historial, para la cual se necesitó modificar el TDA creado en un principio, agregando un nuevo elemento(Historial), esto conlleva a la creación de un nuevo constructor.

Finalmente se implementó la función colocar pieza por jugador, la cual fue la mas compleja, ya que requiere muchas verificaciones para que funcione correctamente el 100 % de los casos. Se implementaron predicados que verifican si la jugada puede existir y predicados para crear un nuevo estado de juego, que representa a la jugada hecha, después se verifica si hay ganador, terminando automáticamente el juego, actualizando las estadísticas

de los jugadores, y por ultimo se verifica si termina en empate, haciendo lo mismo descrito anteriormente.

Los TDA's se complementan entre ellos, algunos predicados de un TDA necesitan predicados de otro, cumpliendo con lo dicho en el análisis del problema.

Hay que aclarar que la creación del programa fue bastante sencillo, ya que Prolog tiene aspectos declarativos, se pudo tomar algunos conceptos e ideas de solución de la implementación del laboratorio anterior.

El programa se estructura por cinco archivos .pl, en los cuales se encuentran los predicados de un main, que contiene los requisitos funcionales pedidos [1], y cuatro archivos que contienen los predicados de los TDA anteriormente mencionados.

6. Aspectos de implementación

La totalidad del proyecto fue hecho en el programa SWI-Prolog versión 9.2.8, y la única librería utilizada fue la predeterminada del programa. El proyecto se estructura en cinco archivos principales, los cuales contienen el main y los TDA player, piece, board y game con sus respectivos predicados. Estos archivos son vitales, ya que dependen entre sí, por lo que siempre hay que tenerlos en la misma dirección.

7. Instrucciones de uso

La entrega cuenta con tres archivos de pruebas, los cuales contienen scripts, igualmente es recomendable ver los archivos de TDA y main y leer que hace cada predicado, estos archivos cuentan con información sobre que es lo que hace cada predicado, su dominio, metas primaria y secundarias, y si utiliza recursión o no.

Para correr los scripts, se debe abrir el programa SWI-Prolog, consultar al archivo main, luego se debe abrir cualquier archivo script, copiar su contenido y pegarlo en la consola. Hay ejemplos de jugadas del siguiente tipo:

- Victoria horizontal y diagonal.
- Empate.
- Jugar en un turno no correspondiente.
- Jugar en una columna llena.

Posibles errores:

La implementación no limita a los jugadores a que elijan el mismo color, por lo que, si lo hacen, o el color de los jugadores empieza con la misma letra, habrá problemas para representar el historial, el tablero, y la forma de descubrir si hay ganador en el juego. Pero, según las reglas de Conecta-4, se deberían usar fichas rojas y amarillas, por lo que si se sigue este formato no habrá ningún problema.

8. Resultados y autoevaluación

Se obtuvo un proyecto exitoso y funcional en su totalidad, se realizaron pruebas para todo tipo de caso, como posicionar una ficha en una columna fuera del rango o en una columna llena, que si es el caso, retorna el juego sin cambios, que un jugador intente colocar una pieza cuando no es su turno, consiguiendo el mismo resultado anterior. También se vieron los casos de victoria vertical, horizontal y diagonal, asimismo con el caso de tener un empate.

Se lograron respetar todos los requerimientos funcionales [1], con sus requisitos específicos, como el dominio y la meta primaria, por lo que la autoevaluación considera el puntaje máximo de cada requerimiento.

También agregar que se trabajó constantemente en el proyecto, logrando, casi un commit diario en el repositorio.

9. Conclusiones del trabajo

Para concluir, se puede decir que el proyecto fue difícil en el aspecto de pensar como hacer que el programa entienda lo que queremos transmitir, la abstracción fue muy compleja y necesaria, pero comparado con el paradigma anterior(Funcional), el proceso de implementación fue mucho mas sencillo, la idea de como hacerlo ya estaba hecha.

El tiempo de realización fue mucho más corto, ya que Prolog toma el primer caso verdadero que ve, las implementaciones no tuvieron tanto en cuenta los limites de recursión. Gracias a la constancia, y al tipo de trabajo, el cual es cercano a la vida laboral, se logró un proyecto exitoso y se adquirió un nuevo conocimiento, el cual puede ser muy útil para el futuro laboral y académico, por ejemplo, Prolog es muy parecido a como funciona una base de datos y estos conocimientos se traspasan a otro ramo cursado por el estudiante, lo cual facilita el estudio.

10. Referencias

Gonzalo Martinez – Intro Paradigma Lógico. (2024). https://drive.google.com/drive/folders/1OXEWRCoQUYgSv8Q_Uh3JRveSO8CQFbVW

SWISH – SWI-Prolog for SHaring. (s. f.). https://swish.swi-prolog.org/p/Tutorial%20de%20prolog.swinb

SWI-Prolog – Reference Manual. (s. f.). https://www.swi-prolog.org/pldoc/doc_for?object=manual

11. Anexos

Requerimientos Funcionales	Logro
RF01 - TDAs	1
RF02 - TDA Player - constructor	1
RF03 - TDA Piece - constructor	1
RF04 - TDA Board - constructor	1
RF05 - TDA Board - otros - sePuedeJugar?	1
RF06 - TDA Board - modificador - jugar ficha	1
RF07 - TDA Board - otros - verificar victoria vertical	1
RF08 - TDA Board - otros - verificar victoria horizontal	1
RF09 - TDA Board - otros - verificar victoria diagonal	1
RF10 - TDA Board - otros - entregarGanador	1
RF11 - TDA Game - constructor	1
RF12 - TDA Game - otros - history	1
RF13 - TDA Game - otros - esEmpate?	1
RF14 - TDA Player - otros - actualizarEstadisticas	1
RF15 - TDA Game - selector - getCurrentPlayer	1
RF16 - TDA Game - selector - board-get-state	1
RF17 - TDA Game - modificador - game-set-end	1
RF18 - TDA Game - modificador - realizarMovimiento	1

Figura 1: Requerimientos Funcionales y su alcance de logro