Universidad Nacional Autónoma de México

Facultad de Ciencias

Reporte Proyecto Final

García Rosas Dicter Tadeo - 316085412

Programación Declarativa
Manuel Soto Romero
Juan Pablo Yamamoto Zazueta

0.1 Propuesta

El sudoku es un juego japones creado a finales de la decada de 1970, el cual adquirio popularidad rapidamente en su pais de origen y finalmente gano reconocimiento internacional en 2005, dependiendo de la dificultad algunos rompecabezas, estos pueden tomar entre 10 minutos hasta 1 hora en completarse, la atraccion principal de estos rompecabezas o juegos esta dada por sus reglas simples y que a su vez estas mismas generan una única solución para cada tablero, sin embargo el proceso de razonamiento para completarlo puede volverse complicado rapidamente.

Dada la complejidad y tiempo necesario para completar un rompecabezas podemos hacer de manera natural la siguiente pregunta, ¿hay alguna posibilidad que una computadora acelere el proceso de solución?. Dentro de la computación, el estudio de este juego resulta interesante pues como mencionamos anteriormente en la pregunta, si existen maneras de automatizar la búsqueda de una solución, pues hay dos enfoques desde los cuales podemos atacar este problema independientemente de su dificultad, aunque es importante resaltar que el tiempo de ejecución puede cambiar bastante dependiendo de cual se escoja. Los enfoques mencionados son la fuerza bruta y la representación del juego como un problema SAT.

Para el primer enfoque podriamos hacer uso de el lenguaje prolog, pues se adecúa bien para resolver tareas que involucren combinatoria o búsqueda exhaustiva dado el backtracking que realiza el lenguaje recorriendo asi todo el espacio de búsqueda. Aunque prolog parece la alternativa perfecta, esto no significa que no podamos utilizar otro lenguaje para resolver dicha tarea por lo que una implementación en haskell tambien es viable dada la cantidad de material del mismo que se nos porporcionó durante el curso.

Finalmente para el segundo enfoque, la reducción del problema en un ejemplar que pueda resolver la libreria miniSat sera proporcionada en el lenguaje haskell.

0.2 Introducción

El sudoku es un juego japones creado a finales de la decada de 1970, el cual adquirio popularidad rapidamente en su pais de origen y finalmente gano

reconocimiento internacional en 2005, dependiendo de la dificultad algunos rompecabezas, estos pueden tomar entre 10 minutos hasta 1 hora en completarse, la atraccion principal de estos rompecabezas o juegos esta dada por sus reglas simples y que a su vez estas mismas generan una única solución para cada tablero, sin embargo el proceso de razonamiento para completarlo puede volverse complicado rapidamente.

Dada la complejidad y tiempo necesario para completar un rompecabezas podemos hacer de manera natural la siguiente pregunta, ¿hay alguna posibilidad que una computadora acelere el proceso de solución?. Dentro de la computación, el estudio de este juego resulta interesante pues como mencionamos anteriormente en la pregunta, si existen maneras de automatizar la búsqueda de una solución, pues hay dos enfoques desde los cuales podemos atacar este problema independientemente de su dificultad, aunque es importante resaltar que el tiempo de ejecución puede cambiar bastante dependiendo de cual se escoja. Los enfoques mencionados son la fuerza bruta y la representación del juego como un problema SAT.

0.3 Descripción de un tablero de Sudoku y sus reglas

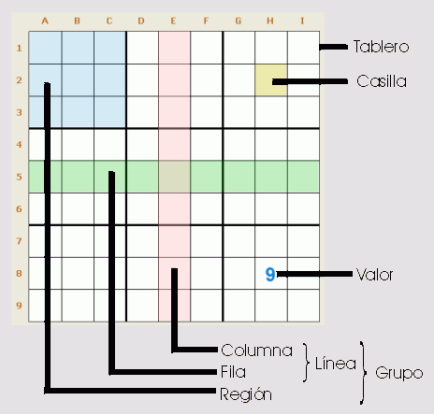
El juego de sudoku esta representado por una cuadricula de 9x9 la cual contiene a su vez 9 subcuadriculas de 3x3 y algunas de las entradas de estas mismas son rellenadas con números en un rango del 1 al 9 mientras que las demás entradas se dejan en blanco.

3	4	5						8
6	1			8	3	5	4	9
7	9			4	5			6
			1	5	7			
				6	4	9		
	7	1	9			4		
		9		2		6		4
	5			1				
2		6				3		

En la imagen anterior se muestra un ejemplar del tablero descrito previamente, pero no basta con solo desribir como se ve un tablero de este rompecabezas para poder solucionarlo, pues se deben seguir ciertas reglas para proporcionar una solución.

Un tablero de sudoku se resuelve asignando numeros entre 1 y 9 a las casillas en blanco de tal forma que cada columna, cada fila y cada sub cuadricula de 3x3 contenga cata uno de los 9 posibles números.
Considerando las reglas anteriores y utilizando un primer enfoque de fuerza bruta podemos observar que prolog se ajusta muy bien a esta tarea, pues por si mismo el lenguaje agota el espacio de búsqueda. Primero representaremos el tablero con una lista plana de 81 elementos aunque su representación más directa es una matriz de 9x9 como se mencionó en la descripción del juego, esta lista de 81 elementos contiene los números que ya son parte de la solución de manera inicial.

Dividiremos el tablero para representarlo en código mediante 3 secciones: filas, columnas y subcuadriculas, las filas se componen de 9 valores a los cuales accederemos calculando el indice de la lista, dado que es una lista plana el indice para cada valor de una fila se calcula mediante (N-1)*9-1 ya que al ser una lista este indice va dando saltos cada 9 posiciones dentro de la lista.



Analogamente se calculan los indices para las filas las cuales también se componen de 9 valores, estos se calculan mediante 0*9+N, de igual forma va

dando saltos cada 9 elementos de la lista. Y finalmente para las cuadriculas internas se calculan mediante la siguiente expresión ((N-1)//3)*27+((N-1) mod 3)*3+1.

Para validar las asignaciones de los 9 números posibles debemos comprobar que no se repitan y que esten dentro de la sección determinada de la lista, por lo que comprobar con la función member cada uno de los 9 números es suficiente, aunque poco eficiente. Después se realiza una iteración sobre la lista y se aplica una función que comprueba que los números estan bien situados y genera a los números que sustituyen a las variables x's con las que identificamos una entrada de al cuadricula como en blanco.

Finalmente se utiliza una función que verifica las condiciones dadas, es decir, que si un número ya fue asignado verificara que este dentro del rango y que no este repetido, si no esta asignado entonces elegirá uno entre 1 y 9 y comprobara las condiciones mencionadas. Si este número no cumple las condiciones se le asigna otro dentro del rango hasta que encuentre uno, en caso de que ningún número funcione hará un bactracking en el maplist y le asignará otro número al elemento anterior y repetirá el proceso hasta que pueda seguir avanzando en la construcción de la solución.

El uso de la fuerza bruta para resolver un tablero de sudoku es un primer enfoque útil dependiendo del tipo de tablero que queramos resolver, pues podemos intuir que entre menos entradas tenga asignado el tablero al inicio del juego, más aumentará la dificultad del mismo siendo 17 el número mínimo de entradas para que sea posible resolver el sudoku, por lo que el uso de la programacion con restricciones sobre el dominio nos permite obtener soluciones a problemas de combinatoria de manera eficiente, en este caso el problema de resolver un tablero de sudoku.

Como mencionabamos anteriormente los problemas pueden ser modelados mediante restricciones y una restricción es una secuencia de variables junto con las combinaciones de valores permitidos para dicha secuencia, es decir los dominios finitos. Podemos definir estas restricciones para el sudoku como los 9 números que podemos asignar a las filas, columnas y subcuadrículas por lo que obtendriamos 27 restricciones.

0.4 Solucion mediante CLP

Inicialmente se utiliza una lista de listas para representar un tablero de sudoku, y se considera resuelto cuando los siguientes enunciados se cumplen:

- hay 9 filas y cada fila tiene una longitud de 9, es decir tiene nueve entradas y el tablero es cuadrado.
- la concatenación de las filas contiene únicamente números del 1 al 9.
- en cada fila los valores contenidos son distintos.
- las columnas son las filas del tablero transpuestas.
- los valores contenidos en cada columna son distintos.
- nombramos a las filas de la primera a la novena A a I, formamos bloques de 3 en 3, A,B,C y D,E,F y G,H,I.

el cpl para el sudoku se utilizapa para la restricción del dominio para los valores que las variables en particular en 'Vs ins 1..9'

0.5 Bibliografía

- Ines Lynce, Joel Ouknine. (s. f.). Sudoku as a SAT Problem. Recuperado el 01 de Junio de 2023 de http://anytime.cs.umass.edu/aimath06/proceedings/P34.pdf
- Fania Raczinski. (2007). Final Report. Recuperado el 03 de Junio de 2023 de https://fania.uk/images/FaniaBSc.pdf
- Andrés Montoya J. La dificultad de jugar sudoku. (2006). Dialnet Recuperado el 02 de Junio de https://dialnet.unirioja.es/servlet/articulo?codigo=6981033
- https://en.wikipedia.org/wiki/Constraint_logic_programming
- Escobar G, Rodrigo. (2011). Programando con Restricciones. Recuperado el 10 de Junio de 2023 de http://www.scielo.org.bo/pdf/ran/v5n1/v5n1a04.pdf
- The Power of Prolog, CLP(FD) and CLP(Z): Prolog Integer Arithmetic. (s.f). Recuperado el 15 de Junio de 2023 de https://www.metalevel.at/prolog/clpz

- Constraint Logic Programming. (s.f). Recuperado el 15 de Junio de 2023 de https://www.swi-prolog.org/pldoc/man?section=clp
- library(clpfd): CLP(FD): Constraint Logic Programming over Finite Domains. (s.f). Recuperado el 15 de Junio de 2023 de https://www.swi-prolog.org/man/clpfd.html