Razonamiento Automatizado 2020-2
Proyecto 1

Selene Linares Arévalo.

Fecha de entrega: 5 abril de 2020

1. Objetivo

Dar solución a un problema reduciéndolo a un problema de satisfacibilidad en Lógica proposicional.

2. Desarrollo

- Elegir un problema para ser resuelto utilizando satisfacibilidad. Puede seleccionarse alguno de la lista en la siguiente sección o bien, proponer algún otro que llame su atención en particular.
- Crear un reporte en donde se presenten:
  1. El enunciado del problema.
  2. Explicación detallada de cómo se aborda el problema para ser resuelto con satisfacibilidad.
- Implementar un programa que genere un archivo con la fórmula o fórmulas que formalizan el problema. El código debe estar debidamente documentado. Se debe incluir el código en la entrega del proyecto.
- Agregar una sección al reporte donde se explique a grandes rasgos cómo el programa genera la fórmula del punto anterior.
- Verificar la satisfacibilidad de la fórmula utilizando *Z3* o *Prover9*. Incluir una sección en el reporte, con los resultados y observaciones obtenidos.
- Realizar una presentación del proyecto desarrollado. Las exposiciones se realizarán en la semana del 13 al 17 de abril y tendrán duración máxima de 20 min.
- La entrega del proyecto se enviará al ayudante con copia a la profesora.

3. Propuestas

3.1. Sliding-block puzzle

Considerar el problema del rompecabezas de piezas deslizantes [5]:

The **8-puzzle**, an instance of which is shown in Figure 3.4, consists of a 3×3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach a specified goal state, such as the one shown on the right of the figure. The standard formulation is as follows:

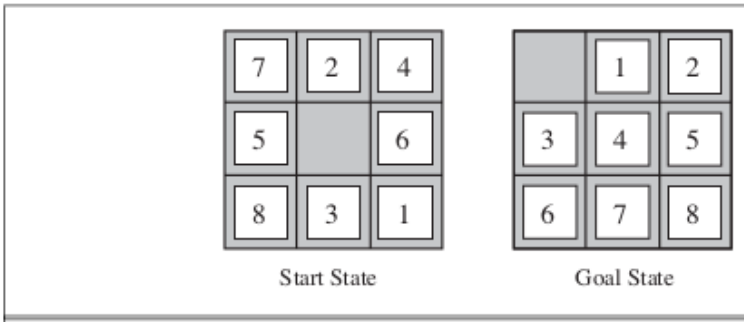


Figure 3.4 A typical instance of the 8-puzzle.

- **States**: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state**: Any state can be designated as the initial state. Note that any given goal can be reached from exactly half of the possible initial states (Exercise 3.4).
- **Actions**: The simplest formulation defines the actions as movements of the blank space *Left*, *Right*, *Up*, or *Down*. Different subsets of these are possible depending on where the blank is.
- **Transition model**: Given a state and action, this returns the resulting state; for example, if we apply *Left* to the start state in Figure 3.4, the resulting state has the 5 and the blank switched.
- **Goal test**: This checks whether the state matches the goal configuration shown in Figure 3.4. (Other goal configurations are possible.)
- **Path cost**: Each step costs 1, so the path cost is the number of steps in the path.

What abstractions have we included here? The actions are abstracted to their beginning and final states, ignoring the intermediate locations where the block is sliding. We have abstracted away actions such as shaking the board when pieces get stuck and ruled out extracting the pieces with a knife and putting them back again. We are left with a description of the rules of the puzzle, avoiding all the details of physical manipulations.

Lo anterior se puede resolver como una instancia del problema de planeación, de tal forma que se verifique si hay un plan (solución del rompecabezas) en 10, 14, o 10 pasos, por ejemplo.
Con el objetivo de reducir el número de cláusulas, se puede considerar un rompecabezas más sencillo como el siguiente:

Nota: El programa que genera la fórmula que formaliza este problema, debe recibir como entrada la configuración del puzzle y con ello construir la fórmula.

3.2. Buscaminas

Ejercicio tomado de [5].

> **7.22** Minesweeper, the well-known computer game, is closely related to the wumpus world. A minesweeper world is a rectangular grid of N squares with M invisible mines scattered among them. Any square may be probed by the agent; instant death follows if a mine is probed. Minesweeper indicates the presence of mines by revealing, in each probed square, the *number* of mines that are directly or diagonally adjacent. The goal is to probe every unmined square.
>
> **a.** Let $X_{i,j}$ be true iff square $[i, j]$ contains a mine. Write down the assertion that exactly two mines are adjacent to [1,1] as a sentence involving some logical combination of $X_{i,j}$ propositions.
>
> **b.** Generalize your assertion from (a) by explaining how to construct a CNF sentence asserting that k of n neighbors contain mines.
>
> **c.** Explain precisely how an agent can use DPLL to prove that a given square does (or does not) contain a mine, ignoring the global constraint that there are exactly M mines in all.
>
> **d.** Suppose that the global constraint is constructed from your method from part (b). How does the number of clauses depend on M and N? Suggest a way to modify DPLL so that the global constraint does not need to be represented explicitly.
>
> **e.** Are any conclusions derived by the method in part (c) invalidated when the global constraint is taken into account?

Con el objetivo de reducir el número de cláusulas, se puede considerar un tablero con número reducido de columnas y renglones.
Nota: El programa que genera la fórmula que formaliza este problema, debe recibir como entrada la configuración del tablero y con ello construir la fórmula.

3.3. Subgráfica bipartita

Enunciado tomado de [3].

[GT25] BIPARTITE SUBGRAPH

INSTANCE: Graph $G = (V,E)$, positive integer $K \leqslant |E|$.
QUESTION: Is there a subset $E' \subseteq E$ with $|E'| \geqslant K$ such that $G' = (V,E')$ is bipartite?

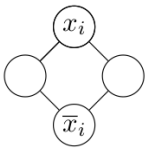
Reference: [Garey, Johnson, and Stockmeyer, 1976]. Transformation from MAXIMUM 2-SATISFIABILITY.
Comment: Remains NP-complete for graphs with no vertex degree exceeding 3 and no triangles and/or if we require that the subgraph be connected [Yannakakis, 1978b]. Solvable in polynomial time if G is planar [Hadlock, 1975], [Orlova and Dorfman, 1972], or if $K = |E|$.

Nota: El programa que genera la fórmula que formaliza este problema, debe recibir como entrada una gráfica G y un entero positivo $k \leq |E|$ y a partir de ellos construir la fórmula.

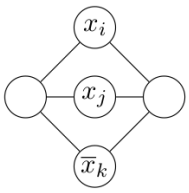
3.4. Camino con vértices prohibidos

Enunciado tomado de [2].

Exercise 21. The problem PATH WITH FORBIDDEN PAIRS:
Input: A graph $G = (V, E)$, vertices $s, t \in V$, and a set of pairs $(u_1, v_1), \ldots, (u_e, v_e) \in V \times V$.
Question: Does there exist a path from s to t that visits at most one vertex of each pair (u_i, v_i)?
Hint: see the gadgets below.



variable



clause $x_i \wedge x_j \wedge \overline{x}_k$

Nota: El programa que genera la fórmula que formaliza este problema, debe recibir como entrada una gráfica G, dos vértices s, t y un conjunto de pares; a partir de ellos se construye la fórmula.

3.5. Triángulo monocromático

Enunciado tomado de [3].

[GT6] MONOCHROMATIC TRIANGLE

INSTANCE: Graph $G = (V,E)$.
QUESTION: Is there a partition of E into two disjoint sets E_1, E_2 such that neither $G_1 = (V,E_1)$ nor $G_2 = (V,E_2)$ contains a triangle?

Reference: [Burr, 1976]. Transformation from 3SAT.
Comment: Variants in which "triangle" is replaced by any larger fixed complete graph are also NP-complete [Burr, 1976]. Variants in which "triangle" is replaced by "k-star" (a single degree k vertex adjacent to k degree one vertices) is solvable in polynomial time [Burr, Erdös, and Lovasz, 1976].

Nota: El programa que genera la fórmula que formaliza este problema, debe recibir como entrada una gráfica G y a partir de ella construye la fórmula.

3.6. Otro problema de planeación.

Enunciado tomado de [4].

Ejercicio 9.27 ([104]) *Considere un puzzle consistente en un vector de 7 posiciones, con la siguiente configuración inicial:*

N	*N*	*N*	*B*	*B*	*B*	*V*

Hay tres fichas negras (N), tres blancas (B) y una casilla vacía (V). El puzzle permite los siguientes movimientos:

1. *Una ficha puede moverse a una casilla vacía adyacente, con coste unidad*
2. *Una ficha puede saltar sobre otras dos (como máximo) hasta una celda vacía, con coste igual al número de fichas sobre las que ha saltado.*

El objetivo del puzzle es tener todas las fichas blancas a la izquierda de todas las negras (sin importar la posición de la celda vacía). a) Defina una función heurística, h, para este problema; b) Obtenga un plan que resuelva este problema.

Nota: se puede omitir el costo de mover fichas.

3.7. Knight's tour.

Enunciado tomado de [1].

> **Almost a millennium before Euler's fateful summer in East Prussia, a Kashmiri poet named Rudrata had asked this question: Can one visit all the squares of the chessboard, without repeating any square, in one long walk that ends at the starting square and at each step makes a legal knight move?**

Nota: Si la respuesta es sí, se debe mostrar el recorrido.

Referencias

[1] Chapter 8: Np-complete problems. `https://people.eecs.berkeley.edu/~vazirani/algorithms/chap8.pdf`. Accessed: 2020-03-25.

[2] Show that the following problems are np-complete. `https://www.hse.ru/mirror/pubs/share/217960059`. Accessed: 2020-03-25.

[3] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., USA, 1990.

[4] Iranzo, P., and Alpuente, M. *Programación Lógica. Teoría y Práctica.* Pearson Educación, Madrid, Spain, 2007.

[5] Stuart Russell, P. N. *Artificial Intelligence: A Modern Approach. 3rd Edition.* Pearson, 2009.