Introducción a la Inteligencia Artificial
Proyecto Final

Arturo Márquez Flores
Maestría en Inteligencia Artificial
Universidad Veracruzana
CIIA – Centro de Investigación en Inteligencia Artificial
Sebastián Camacho No 5, Xalapa, Ver., México 91000
arturomf94@gmail.com
https://github.com/arturomf94/iia-mia

8 de Enero del 2019

1. Introducción

En este reporte se describe el funcionamiento del código de Prolog que resuelve problemas de Sudoku mediante una interfaz gráfica de usuario desarrollada en Python. El proyecto final puede encontrarse en este repositorio. Para poder utilizar se necesita tener instalado SWI-Prolog, Python (al menos 2.7) y las dos principales dependencias: pyswip para la conexión Python-Prolog y Tkinter para poder ejecutar la interfaz gráfica.

2. Código de Prolog

En el Cuadro 1 podemos ver el código en Prolog que se utiliza para resolver los problemas de Sudoku. El código consiste en lo siguiente: en el predicado *sudoku*, la primera línea revisa que *Rows* (un arreglo) tenga una longitud de 9, y que cada fila tenga la misma longitud. La segunda línea concatena *Rows* con variables y verifica que las variables estén en el dominio $\{1, 2, ..., 9\}$. La tercera línea aplica *all_distinct* a todos los elementos de *Rows*; es decir, asegura que todos los elementos sean diferentes. La cuarta línea transpone el arreglo para obtener las columnas, en lugar de las filas y aplica, otra vez, *all_distinct* a las columnas. Las últimas dos líneas definen los nombres de las filas y los *bloques* del juego.

```
:- use_module(library(clpfd)).

:- dynamic
   problem/2.

sudoku(Rows) :-
        length(Rows, 9), maplist(same_length(Rows), Rows),
        append(Rows, Vs), Vs ins 1..9,
        maplist(all_distinct, Rows),
        transpose(Rows, Columns), maplist(all_distinct, Columns),
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
        blocks(As, Bs, Cs), blocks(Ds, Es, Fs), blocks(Gs, Hs, Is).

blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
blocks(Ns1, Ns2, Ns3).

reset :-
    retractall(problem/2).
```

Cuadro 1: Código en Prolog

3. Conexión Python-Prolog

Para establecer una conexión entre Python y Prolog se utilizó la librería llamada pyswip, cuya documentación puede encontrarse aquí. Con el siguiente código del Cuadro 2 se puede establecer una conexión a Prolog.

```
from pyswip import Prolog
prolog = Prolog()
prolog.consult("sudoku.pl")
```

Cuadro 2: Código en Python

Además, pyswip te permite hacer comandos del estilo *assertz* y *consultas* para obtener los resultados de Prolog. En el repositorio del proyecto pueden encontrarse dos scripts donde esto sucede. El primero es **pyswip_test.py**, que es un script de prueba para hacer una conexión con Prolog. El segundo es en el script principal del proyecto, **sudoku_gui.py**, donde estos comandos se integran con Tkinter para formar la interfaz gráfica.

4. Python GUI

La librería utilizada para construir la interfaz gráfica es Tkinter y aquí puede revisarse su documentación. En términos generales, la función **createGUI** es la principal que se encarga de crear la interfaz. Esta función crea las casillas y botones de la ventana que se muestra en la Figura 1. Gran parte del código utilizado aquí fue reciclado de esta implementación.

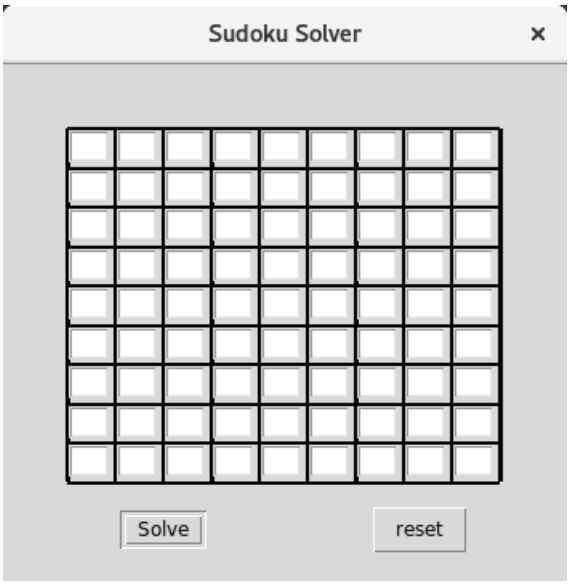

Figura 1: GUI Python

Las casillas pueden ser llenadas por el usuario. Esto crea un *arreglo* que sirve como input para resolver el problema de Sudoku con Prolog. En la siguiente sección se exponen dos ejemplos.

5. Ejemplos

En la Figura 2 podemos observar el primer paso en la interacción entre el usuario y la interfaz. Esta es cuando el usuario ingresa los *inputs* en el recuadro.

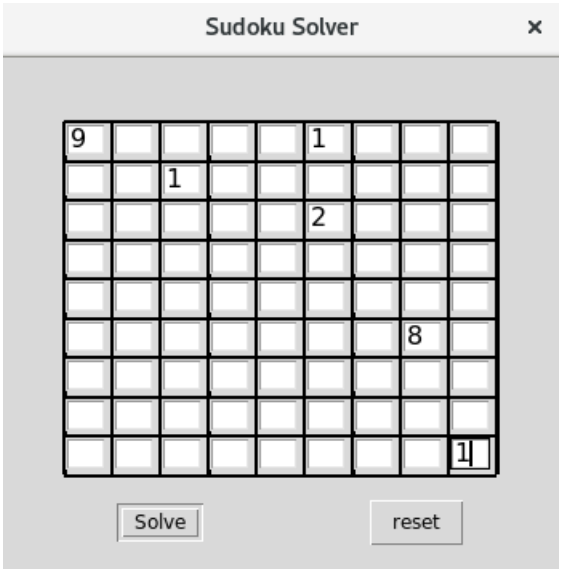

Figura 2: Ejemplo: Usuario ingresa inputs

Posteriormente, cuando se activa el botón *solve* se resuelve automáticamente el problema especificado. Esto se muestra en la Figura 3.

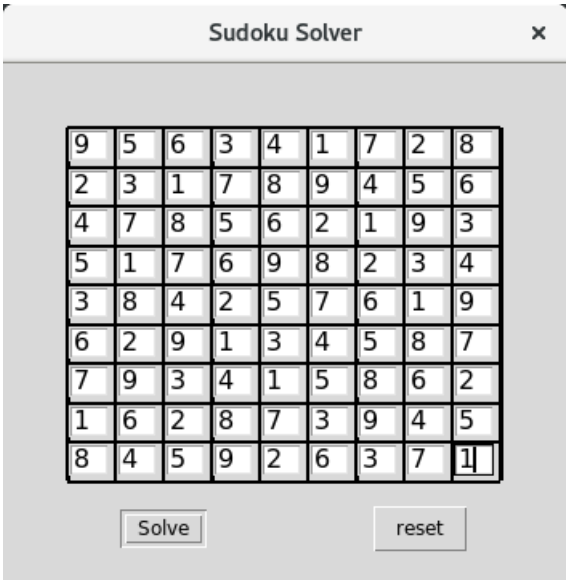

Figura 3: Ejemplo: Programa resuelve problema

Con esto tenemos el problema resuelto. Para resolver un problema con

diferentes inputs sólo es necesario activar el botón *reset* para que, una vez más, las casillas estén vacías.