How to Create a Sudoku Solver Using the DLV Programming Language

What is DLV?

DLV is a deductive database system based on disjunctive logic programming. An easier way the phrase it is opposed to Java, C++, and python where you create a set of instructions for the computer to execute. In DLV all you must do is tell the computer what you do not want, and the computer will generate all possible "worlds" that meet your criteria. We are going to use this language to solve any sudoku puzzle given you provide it the locations of the pre-placed numbers.

Basic Overview of DLV

```
joke.
laugh :- joke.


:- male(X), female(X).
```

- Facts
  - You can create facts that must exist in the "world" that DLV generates.
  - You create a fact in dlv by stating the fact as is in code, such as **"joke."** or if you want the inverse which is no joke **"-joke."**
- Rules
  - Rules are the rules that the DLV program must follow as well.
  - A rule is represented with the name of the subject variable followed by a ":-" then followed by the condition.
  - An example of a rule is **"laugh :- joke.**" this is the equivalent of saying in the program, "if there is joke, then laugh".
- Constrictions
  - Constrictions outline what combinations of facts are not allowed in the "world" that DLV generates
  - A constriction is represented with a ":-" at the start of the line.
  - An example of a constriction would be **":- male(x), female(x)."**
    - This example states that the individual "x" cannot be both a male and female.
- Other Bits and Bobs
  - All statements in DLV end with a "." Opposed to the commonly used ";" to signify the end of a line
  - Comments are lines of code that aren't a part of the program, think of them as notes for yourself while programming.

- When programming in DLV, it does not matter the order in which statements are typed, thus why the figures are not in order.

Materials Needed

- A computer capable of running basic programs.
- A text editor installed on previously mentioned computer.
  - In my example, I will be using Microsoft Visual Studio Code as my text editor, but you can use whatever you are most familiar with.
- A download of DLV on the previously mentioned computer.
  - This can be found at this link: DLV download

Solving a Sudoku in DLV

1. Creating the variables
   - In the sudoku solver, all the facts that we need to complete the sudoku are the individual 81 squares that make up a sudoku board, you can see how they are created in (figure 1).
   - Additionally, we also need facts that give each of the 81 squares a value from 1 to 9 (figure 1).

```
10  %%%%%%%%%
11  % Setup %
12  %%%%%%%%%
13  row(1..9).
14  col(1..9).
15  val(1..9).
```

Figure 1

2. Generating all possible combinations of sudoku boards.
   - we will create a "place" fact that has a row and column with an assigned value.
   - To ensure that each place has only one value, we must use the logic operator "v" to indicate "or".
     - An example of this would be **"place(R, C, 1) v place(R, C, 2) :- row(R), col(C)"**, this line says logically "place(R, C) can have a value of 1 or 2 but not both or neither. (figure 2)

```
% Place numbers
place(R, C, 1)
v place(R, C, 2)
v place(R, C, 3)
v place(R, C, 4)
v place(R, C, 5)
v place(R, C, 6)
v place(R, C, 7)
v place(R, C, 8)
v place(R, C, 9)
    :- row(R), col(C).
```

Figure 2

3. Setting Constrictions
    - One rule to Sudoku is that you cannot have duplicate numbers in either rows or columns, to enforce that rule we set constrictions.
        - We set the constriction:
          :- place(R, C1, VAL), place(R, C2, VAL), C1 != C2. (figure 3)
            - This line says logically "A place in the same row but different column cannot have the same value."
            - We will repeat this step but for the column.

```
% Can't have duplicate numbers in rows or columns
:- place(R, C1, VAL), place(R, C2, VAL), C1 != C2.
:- place(R1, C, VAL), place(R2, C, VAL), R1 != R2.
```

Figure 2

4. Enforce the rule of the small 3 x 3 grids that are in a sudoku.
    - Using math operators, create 9 regions (figure 4).
        - We create a region(REG, N) variable where the first value is what region the is being specified then the second variable is what number is in the region.
    - set a constriction to prevent duplicate values in each region (figure 4).
        - The constriction of **:- not region(REG, VAL), val(VAL), region_set(REG).**
            - This specifies that there cannot be a region that does not have a unique number value in one of the 9 regions.

```
45  % Make sure the sub regions work
46  region(REG, N) :- place(R, C, N), S1X = R-1, S2X = S1X/3, SY = C-1, Y = SY/3 , X = S2X * 10, REG = X + Y.
47  region_set(REG) :- region(REG, _).
48  :- not region(REG, VAL), val(VAL), region_set(REG).
```

Figure 3

5. Fill in the places of the sudoku that you want to solve
    - You do this by creating place facts such **as "place(1,1,4)"**, this indicates that row 1, column, 1 has a value of 4 (figure 5).
        - Continue this for every space that is prefilled when starting the sudoku puzzle.

```
17  % Prepopulate the board
18  place(1,1,4). place(1,2,5). place(1,3,7). place(1,6,8). place(1,8,2). place(2,2,3).
19  place(2,4,5). place(2,5,2). place(2,8,9). place(2,9,4). place(3,1,2). place(4,2,7).
20  place(4,3,1). place(4,6,3). place(5,1,9). place(5,2,6). place(5,5,8). place(5,8,4).
21  place(5,9,1). place(6,4,6). place(6,7,3). place(6,8,5). place(7,9,3). place(8,1,6).
22  place(8,2,2). place(8,5,1). place(8,6,4). place(8,8,7). place(9,2,8). place(9,4,7).
23  place(9,7,4). place(9,8,1). place(9,9,5).
```

Figure 4

6. Run the code and be impressed
    - If you applied all the code correctly and followed the instructions, your code should look like figure 6
        - In order to run the program on your command line (to open a command line on windows, open your windows search bar and type “cmd”)
            - On the command line type “DLV -silent (name of the file). In this instance it was named sudoku.dlv so it should look like this: **$ DLV -silent sudoku.dlv**
    - once you run the code, you will notice that the program outputted all possible ways to solve the sudoku that you supplied it.
    - However, the output will be displayed as place(row, column, value) in a long list of all 81 places.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                _       _           %
%  ___ _   _  __| | ___ | | ___   _  %
% / __| | | |/ _` |/ _ \| |/ / | | | %
% \__ \ |_| | (_| | (_) |   <| |_| | %
% |___/\__,_|\__,_|\___/|_|\_\\__,_| %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Thomas Otten 2022

%%%%%%%%%
% Setup %
%%%%%%%%%
row(1..9).
col(1..9).
val(1..9).

% Prepopulate the board
place(1,1,4). place(1,2,5). place(1,3,7). place(1,6,8). place(1,8,2). place(2,2,3).
place(2,4,5). place(2,5,2). place(2,8,9). place(2,9,4). place(3,1,2). place(4,2,7).
place(4,3,1). place(4,6,3). place(5,1,9). place(5,2,6). place(5,5,8). place(5,8,4).
place(5,9,1). place(6,4,6). place(6,7,3). place(6,8,5). place(7,9,3). place(8,1,6).
place(8,2,2). place(8,5,1). place(8,6,4). place(8,8,7). place(9,2,8). place(9,4,7).
place(9,7,4). place(9,8,1). place(9,9,5).

%%%%%%%%%%%
% Placing %
%%%%%%%%%%%

% Place numbers
place(R, C, 1)
v place(R, C, 2)
v place(R, C, 3)
v place(R, C, 4)
v place(R, C, 5)
v place(R, C, 6)
v place(R, C, 7)
v place(R, C, 8)
v place(R, C, 9)
    :- row(R), col(C).

% Can't have duplicate numbers in rows or columns
:- place(R, C1, VAL), place(R, C2, VAL), C1 != C2.
:- place(R1, C, VAL), place(R2, C, VAL), R1 != R2.

% Make sure the sub regions work
region(REG, N) :- place(R, C, N), S1X = R-1, S2X = S1X/3, SY = C-1, Y = SY/3 , X = S2X * 10, REG = X + Y.
region_set(REG) :- region(REG, _).
:- not region(REG, VAL), val(VAL), region_set(REG).
```

Figure 5

Summary

Despite how confusing it may be at first, DLV is a logic dependent language that requires some hard thinking but nothing too taxing except for maybe the math logic. Once you understand how the language works and its uses, it can be used for a multitude of tasks such has setting up a schedule or solving logic puzzles.