CS 4300 - Fall 2022 - Minh Nguyen
Final Project Report

I. **Project:** Sudoku Solver using Propositional Logic

II. PEAS Assessment:

1. **Description & Rules:**
   - Sudoku is a number game consisting of a 9 × 9 grid
   - The player can use only numbers from 1 to 9.
   - Each 3×3 block can only contain numbers from 1 to 9.
   - Each vertical column can only contain numbers from 1 to 9.
   - Each horizontal row can only contain numbers from 1 to 9.
   - Each number in the 3×3 block, vertical column or horizontal row can be used only once.
   - The game is over when the whole Sudoku grid is correctly filled with numbers.
   - In the start state, the board must have at least 17 numbers filled (as clues)

2. **The Percepts (Sensors):**
   The board, which has all rows, columns, the currently filled numbers and their locations

3. **The Actions (Actuators):**
   - Choose a number from 1-9 and fill it in an open space on the board
   - Do a search and to find the solutions

4. **The Environment:**
   a. Observability:
      i. Fully observable
      ii. The board with its rows and columns are known. The filled numbers with their locations and the empty spaces are known. The possible numbers to select are from 1-9 (known)

b. Uncertainty:
    i. Deterministic.
    ii. Filling a number in a square means that that number cannot be used again in its 3x3 block, its vertical column, and its horizontal row

c. Duration:
    i. Episodic.
    ii. The agent, after seeing the initial state of the Sudoku board, can explore the solutions and complete the board in one go

d. Stability:
    i. Static
    ii. The environment will not change, and there are no factors that affect the decision of the agent

e. Granularity:
    i. Discrete
    ii. A number can be filled in a specific space (if it is valid)

f. Participants:
    i. Single agent
    ii. One agent is sufficient solve this problem (hopefully)

g. Knowledge:
    i. Known
    ii. The board with its rows and columns are known. The filled numbers with their locations and the empty spaces are known. The possible numbers to select are from 1-9 (known)

5. Intended Search Strategy:

- Turn the Sudoku game into a constraint satisfaction problem
- Apply propositional logic to solve it
- Another outcome from finishing the game/problem is to learn and understand Prolog and how it can be applied to solve other problems

III. Search Algorithms/Strategies Used:

1. **Prolog Predicate:**
    - A predicate defined in Prolog
    - Receives a lists of rows as input
    - Check if the puzzle is valid (9 rows/columns long)
    - Concatenate all rows into a single list
    - Check if each element (number) in a row is valid (a number from 1 to 9)
    - The following process applies regardless of the strategy (*):
        - Use the chosen strategy to compares every elements is a row
        - Use the chosen strategy to compare every elements in a column
        - Use the chosen strategy to compare every elements in a subsquare

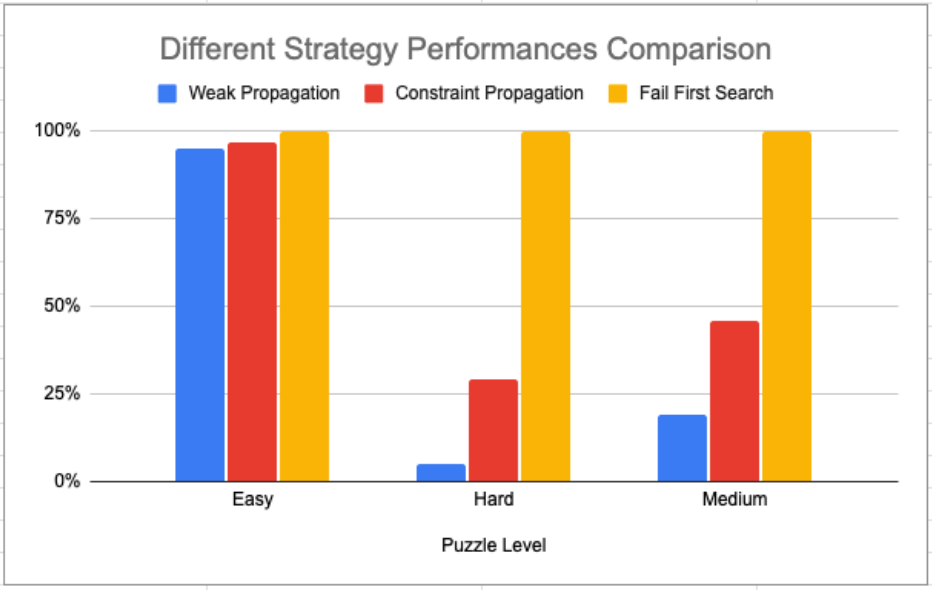
2. **Weak Propagation:**
    - Implements the Prolog predicate strategy
    - (*) = Only use weak propagation to check
    - Use all_different: return True if all variables are different
    - As suggested by its name, this is not a good strategy even though it is good, fast, and able to solve simple Sudoku boards

3. **Constraint Propagation:**
    - Implements the Prolog predicate strategy
    - (*) = Use constraint propagation
    - Use all_distinct: return True if and only if all variables are pairwise distinct
    - This strategy is better than Weak Propagation because it is more efficient and propagates much more strongly

4. **Fail First Search:**
    - Implements the Prolog predicate strategy
    - (*) = Use labeling
        - Enumerate concrete solutions
        - Assigns truth values to the Boolean variables such that all stated constraints are satisfied
        - Label the leftmost variable with smallest domain next, in order to detect infeasibility early.
    - This is a good strategy because it can solve any Sudoku problems
    - However, it is not the most efficient way because it takes a long time to solve a problem

IV. Results:

Puzzle Level	Weak Propagation	Constraint Propagation	Fail First Search
Easy	95%	97%	100%
Hard	5%	29%	100%
Medium	19%	46%	100%



1. **Weak Propagation:**
   - Can solve easy Level problems
   - But as the difficulty increases, the performance decreases
   - This strategy cannot make decision so as it comes across False result (e.g. some the elements in a rows are similar), it cannot backtrack
   - Or in cases where there are multiple possible solutions to a cell in a Sudoku board, this strategy is not capable of choosing which number would go in there
   - Therefore, it may return many possible results but not a correct result.

2. **Constraint Propagation:**

   Since this chart only shows the average performance of each strategy, it seems like Constraint Propagation also has a poor performance. However, this strategy can solve many Medium level problems completely (almost half of the time, the other half usually only miss a few cells on the board)

3. **Fail First Search:**

   Can solve any problems due to the fact that it can "make decisions" by searching every possible combination in order to solve a computational problem.

V. Conclusion:

After implementing the different strategies discussed above, gathering the statistics, and evaluating their performances, I notice that there is a more efficient way to solve a Sudoku problem using this kind of technique. For a given board, we start with constraint propagation, which is fast and is likely to take us close to a solved board, to narrow down the options. Then, for the unsolved cells, since we know the possible numbers that can go into those cells, we apply the search algorithm. By doing it this way, we can solve the problem faster, and still ensure that the solution is correct.

Another interesting thing I find from this project is that by looking at the performance of each strategy, we can find out the level of difficulty of a given Sudoku board. For this project, I ran the program on many different Sudoku boards with different levels of difficulty (easy, medium, hard). Based on the above chart, if a board can be solved solely by weak propagation, it is most likely to be an Easy board, or if it cannot be solved by weak propagation but can be solved using constraint propagation then it is likely to be at least a Medium level board. So this program can both solve the problem and answer how hard the problem is. The main reason why I find this interesting is because of how people generate Sudoku boards at different levels. I believe that they do not simply put random numbers on a board or generate a solved board then take a few numbers away, but rather generate it in a way that it can only be under certain propagation.

For this project, I learned Prolog, a logic programming language that is associated with artificial intelligence and computational linguistics. It is a fascinating language to me because of how a few lines of code (for weak propagation) can solve a Sudoku board. I definitely did not use it to its full potential, and Sudoku is a ‘relatively’ simple AI problem, but from what I have learned about it, I would consider using it for any future AI projects I may be working on.