CS 4300 - Final Project

By Minh Nguyen

About This Project

1. Project: Sudoku
2. Tool: Prolog
   a. **PRO**gramming in **LOG**ics
   b. A logical and declarative programming language
3. Project outcomes:
   a. Apply propositional logic to solve a Sudoku board
   b. How it can solve other problems

```
isAmazing(artificial_intelligence).

isGood(life).
```

Knowledge base

```
?- isAmazing(What).
What = artificial_intelligence.

?- isGood(life).
true.

?- isGood(catching_a_cold).
false.
```

Answer questions

Search Algorithms/Strategies Used

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

Search Algorithms/Strategies Used

2. Weak Propagation:
    a. Strategy = weak propagation only
    b. Use predicate **all_different:** return True if all variables are different
    c. Pros: solve simple Sudoku boards fast
    d. Cons: weak, often fail to solve more complex problems

Search Algorithms/Strategies Used

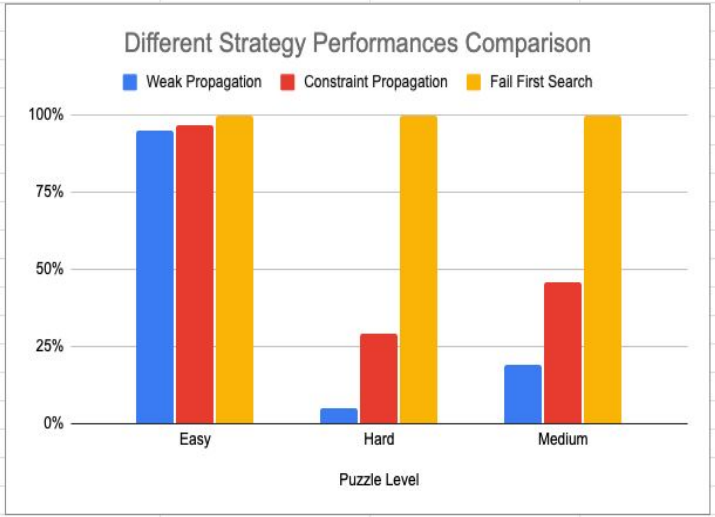
3. Constraint Propagation:
    a. Strategy = constraint propagation
    b. Use predicate **all_distinct:** return True if and only if all variables are pairwise distinct
    c. Pros: more efficient compare to Weak Propagation
    d. Cons: yet, still cannot solve 100% all problems

Search Algorithms/Strategies Used

4. Fail First Search:
    a. Strategy = use labeling
        i. Enumerate concrete solutions
        ii. Assigns truth values to the Boolean variables such that all stated constraints are satisfied
        iii. Label the leftmost variable with smallest domain next, in order to detect infeasibility early
    b. Pros: can solve 100% Sudoku boards
    c. Cons: not efficient as it takes a long time to solve the problem

Results and Comparisons

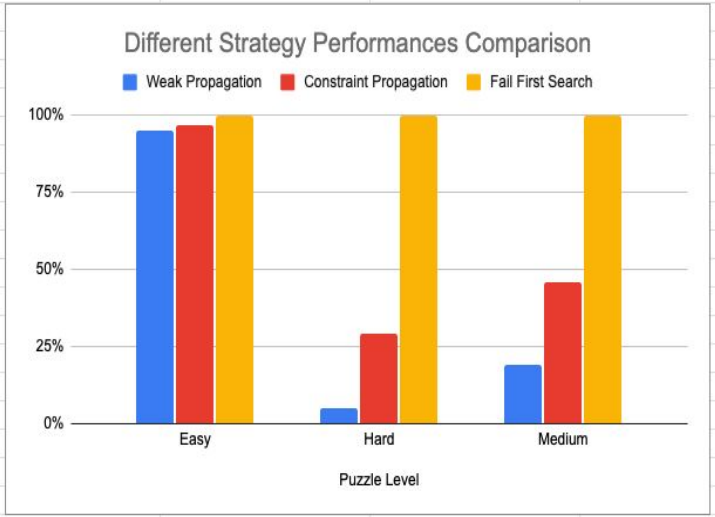
	A	B	C	D
1	Puzzle Level	Weak Propagation	Constraint Propagation	Fail First Search
2	Easy	95%	97%	100%
3	Hard	5%	29%	100%
4	Medium	19%	46%	100%



1. Weak Propagation:
    a. Can solve easy problems
    b. Difficulty increases, performance decreases
    c. Cannot backtrack
    d. If there are many possible solutions to a cell, this strategy cannot make decision

Results and Comparisons

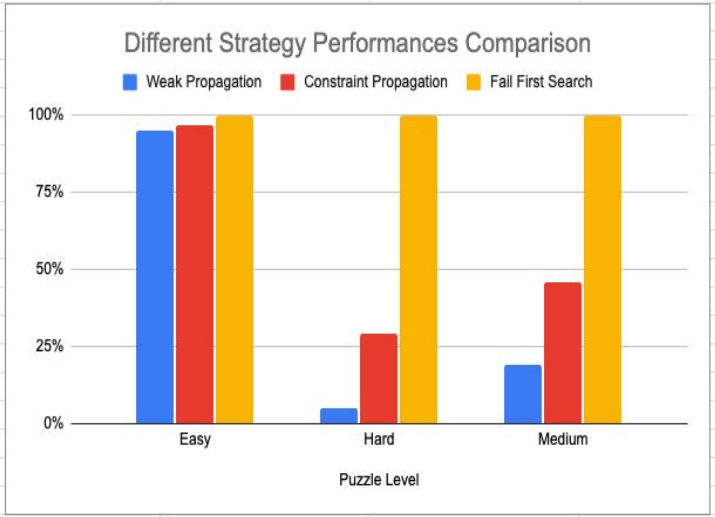
	A	B	C	D
1	Puzzle Level	Weak Propagation	Constraint Propagation	Fail First Search
2	Easy	95%	97%	100%
3	Hard	5%	29%	100%
4	Medium	19%	46%	100%



2. Constraint Propagation:
    a. Can solve many Medium level Sudoku boards completely
    b. It fails due to missing a few cells on the board

Results and Comparisons

	A	B	C	D
1	Puzzle Level	Weak Propagation	Constraint Propagation	Fail First Search
2	Easy	95%	97%	100%
3	Hard	5%	29%	100%
4	Medium	19%	46%	100%



3. Fail First Search:
    a. Solve 100% boards
    b. Can make decisions by searching every possible combination in order to solve a computational problem

Conclusions

1. Better strategy?
    a. Start with Constraint Propagation to narrow down the options
    b. For unsolved cells, used search algorithms
    c. Solve the problems fast and still ensure that the solution is correct

2. Strategies' performance and board difficulties connection:
    a. Solved by Weak Propagation only -> likely to be Easy
    b. Solved by Constraint Propagation -> likely to be Medium
    c. Board generation at different levels

3. Prolog is powerful and helpful

```
presented(slide).

listened(audience).

thanks(audience) :-
    presented(slide),
    listened(audience),
    write('Thanks for listening!').
```

```
[?- thanks(audience).
Thanks for listening!
true.
```