tags: `Report`

Real-Life Applications of Computational Algorithms - HW2

309551130 謝柏威

Implementation

- This program is written in `python3`.
- Package `pyeda` is required.
- Reference: https://pyeda.readthedocs.io/en/latest/sudoku.html

1. parse input

- First step, read all the numbers as the sudoku input, and check whether sudoku dimension is valid.

```
grid = [c for c in fin.read() if c.isdigit()]
N = int(math.sqrt(len(grid)))
assert len(grid) == N ** 2
n = int(math.sqrt(N))
assert N == n ** 2
```

2. construct boolean expression and corresponding decision diagram

- `exprvars` is a function for returning arrays of arbitrary dimension, and can be used later to apply constrain.
- `expr2bdd` convert arbitrary expressions to binary decision diagram.

```
DIGITS = "".join([str(i + 1) for i in range(N)])
X = exprvars('x', (1, N + 1), (1, N + 1), (1, N + 1))
solver = And(rule_constrain(), input_constrain(grid))
solver = expr2bdd(solver)

print(solver.satisfy_count(), file = fout)
```

3. sudoku rule constrain

- This is the part where we apply sudoku rule constrain.
- `OneHot` function returns a formula in conjunctive normal form. The formula is eqivalent to "only one of the variable in the clause is true", which is useful during our sudoku rule construction.

```
# value constrain
V = And(*[
        And(*[
            OneHot(*[ X[r, c, v]
                for v in range(1, N + 1) ])
            for c in range(1, N + 1) ])
        for r in range(1, N + 1) ])

# row constrain
R = And(*[
        And(*[
            OneHot(*[ X[r, c, v]
                for c in range(1, N + 1) ])
            for v in range(1, N + 1) ])
        for r in range(1, N + 1) ])

# column constrain
C = And(*[
        And(*[
            OneHot(*[ X[r, c, v]
                for r in range(1, N + 1) ])
            for v in range(1, N + 1) ])
        for c in range(1, N + 1) ])

# box constrain
B = And(*[
        And(*[
            OneHot(*[ X[n*br+r, n*bc+c, v]
                for r in range(1, n + 1) for c in range(1, n + 1) ])
            for v in range(1, N + 1) ])
        for br in range(n) for bc in range(n) ])

return And(V, R, C, B)
```

4. sudoku input constrain

- This is the part we apply sudoku input constrain. We iterate through whole sudoku grid and apply sudoku constrain.

```
## apply sudoku input constrain
def input_constrain(grid):
    return And(*[ X[i // N + 1, i % N + 1, int(c)]
        for i, c in enumerate(grid) if c in DIGITS ])
```

Results

Constructing a decision graph takes a lot of time. (6487s in my own experience)

```
$ python3 src/main.py data/sudoku_4x4_9.txt answer.txt && cat answer.txt
9

(242ms)

$ python3 src/main.py data/sudoku_9x9_125.txt answer.txt && cat answer.txt
125

(6487s)
```

Discussion

- I noticed that constructing a decision diagram takes a lot of time. If I didn't convert the bool expression to decision diagram, It literally takes no time to solve the problem.
- The `pyeda` document is quite nice and comprehensive. After reading through the document on the website, I'm able to finish this project easily.