实验作业1

实验任务

使用pure SAT 求解N-Queen 问题, 并对比 PPT 中 SMT 的实现的效率

代码

SMT

参考PPT中的代码，增加时间监测

```
def SMT(n):
    Q = [Int('Q_ % i' % (i + 1)) for i in range(n)]
    val_c = [And(1 <= Q[i], Q[i] <= n) for i in
range(n)]
    col_c = [Distinct(Q)]
    diag_c = [If(i == j, True,
                 And(i + Q[i] != j + Q[j], i + Q[j] !=
j + Q[i]))
              for i in range(n) for j in range(i)]
    solver = Solver()
    solver.add(val_c + col_c + diag_c)

    start = time.time()
    if solver.check() == sat:
        model = solver.model()
        print(model)
    else:
        print(None)
    finish = time.time()
    return finish - start
```

SAT

定义一个n*n的棋盘，用bool变量表示皇后的有无。参照PPT上给出的约束表达式分别增加行列和对角线的约束规则

```
def SAT(n):
    board = [[Bool("x_%s_%s" % (i, j)) for j in
range(n)] for i in range(n)]
    # row constraints & column constraints
    constr = [Or([board[i][j] for j in range(n)]) for
i in range(n)]
```

```
    constr += [Or(Not(board[i][j]), Not(board[i][k])) for i in range(n) for j in range(n) for k in range(j + 1, n)]
    constr += [Or([board[i][j] for i in range(n)]) for j in range(n)]
    constr += [Or(Not(board[i][j]), Not(board[k][j])) for j in range(n) for i in range(n) for k in range(i + 1, n)]
    # diagonal constraints
    for i in range(n):
        for ii in range(i + 1, n):
            for j in range(ii - i, n):
                jj = j - ii + i
                constr += [Or(Not(board[i][j]), Not(board[ii][jj]))]
            for j in range(n - ii + i):
                jj = j + ii - i
                constr += [Or(Not(board[i][j]), Not(board[ii][jj]))]

    solver = Solver()
    solver.add(constr)

    start = time.time()
    for k in range(1):
        if solver.check() == sat:
            model = solver.model()
            solution = []
            for i in range(n):
                for j in range(n):
                    if model[board[i][j]]:
                        solution += [j]
            print(solution)

            # 寻找其他解
            solver.add(Or([Not(board[i][solution[i]]) for i in range(n)]))
        else:
            print(None)
    finish = time.time()
    return finish - start
```

用时比较

考虑约束用时

n的值范围为1-30，用时结果如下

n	SMT	SAT
1	0.009009	0.003003

2	0.004299	0.004077
3	0.010105	0.006078
4	0.009949	0.016001
5	0.012	0.034208
6	0.018084	0.046237
7	0.022871	0.075118
8	0.021694	0.110622
9	0.043	0.149089
10	0.070715	0.226965
11	0.210133	0.300456
12	0.176546	0.366784
13	0.229486	0.478138
14	0.271272	0.849693
15	0.176934	0.880427
16	0.516712	0.858371
17	0.185827	1.048053
18	0.287804	1.263957
19	0.898991	1.659121
20	1.846864	1.746289
21	0.65687	2.007352
22	1.435043	2.558506
23	0.651588	2.792624
24	2.288262	3.542797
25	2.773189	4.188073
26	1.87471	4.468053
27	3.970223	4.738761
28	4.667235	5.873193
29	15.24601	6.642845
30	25.34693	8.251319

不考虑约束用时

n的值范围为1-30，用时结果如下

n	SMT	SAT
1	0.00159	0
2	0.002003	0.000998
3	0.005999	0.00102
4	0.006003	0.002997
5	0.005	0.00397
6	0.010039	0.004018
7	0.010985	0.006924
8	0.010538	0.005998
9	0.028981	0.007221
10	0.053194	0.010119
11	0.078704	0.011997
12	0.118382	0.013036
13	0.095854	0.017053
14	0.059535	0.022006
15	0.324255	0.028986
16	0.48313	0.085308
17	0.512262	0.033999
18	0.355801	0.039325
19	0.339461	0.043005
20	0.53556	0.052698
21	1.930284	0.05232
22	2.995345	0.056
23	3.510686	0.050711
24	1.662449	0.066517
25	2.908194	0.062715
26	2.347658	0.081261
27	3.458813	0.101034
28	11.59356	0.079189
29	15.10143	0.080261

30	8.155843	0.098042

结论

可见SAT的效率都比SMT要高，且在考虑约束时用时会较大幅度的上升

实验作业2

实验任务

使用pure SAT 求解d=a-b, 其中,a,b为正整数

代码

考虑例子13-7=6，均为超过15故`bits=4`，定义carry为进位。借助异或操作实现二进制数的减法。

```
def minus():
    bits = 4
    # d = a - b
    a_num = 13
    b_num = 7
    d_num = 0

    a = [Bool("a_%s" % i) for i in range(bits)]
    b = [Bool("b_%s" % i) for i in range(bits)]
    d = [Bool("d_%s" % i) for i in range(bits)]

    carry = [Bool("c_%s" % i) for i in range(bits + 1)]

    constr = [a[i] == (b[i] == (d[i] == carry[i])) for i in range(bits)]
    constr += [carry[i + 1] == (Or(And(b[i], d[i]), And(b[i], carry[i]), And(d[i],carry[i]))) for i in range(bits)]
    constr += [Not(carry[bits])]
    constr += [Not(carry[0])]
    for i in range(bits):
        if b_num & (1 << i):
            constr += [b[i]]
        else:
            constr += [Not(b[i])]
        if a_num & (1 << i):
            constr += [a[i]]
        else:
            constr += [Not(a[i])]
```

```
27
28      solver = Solver()
29      solver.add(constr)
30      if solver.check() == sat:
31          m = solver.model()
32          for i in range (bits):
33              if m[d[i]]:
34                  d_num += 1 << i
35          print("d_num = %s" % d_num)
```

结果

对于测试的13-7=6可以正确输出结果

```
PS D:\Homework\Formal_Methods> python -u "d:\Homework\Formal_Methods\lab1\plusminus.py"
d_num = 6
```

再测试8-3=5，结果正确

```
PS D:\Homework\Formal_Methods> python -u "d:\Homework\Formal_Methods\lab1\plusminus.py"
d_num = 5
```