FM-HW 3

1 N-Queen问题的SAT和SMT求解

实验记录

SAT(n=8):

```
PS D:\Course\formal_methods\HW\FM-HW3> python .\n-queen_sat.py
n:8
[Q_3_1 = False,
 Q_1_2 = False,
 Q_3_2 = False,
 Q_1_3 = False,
 Q_6_5 = False,
 Q_5_5 = False,
 Q_8_1 = True,
 Q_6_7 = False,
 Q_3_4 = False,
 Q_8_3 = False,
 Q_1_8 = False,
 Q_5_4 = False,
 Q_1_4 = True,
 Q_7_4 = False,
 Q_4_2 = False,
 Q_6_1 = False,
 Q_3_8 = False,
 Q_7_1 = False,
 Q_4_6 = False,
 Q_3_6 = False,
 Q_5_7 = False,
 Q_2_1 = False,
 Q_3_5 = False,
 Q_6_8 = True,
 Q_1_5 = False,
 Q_4_1 = False,
 Q_2_4 = False,
 Q_2_6 = False,
 Q_8_2 = False,
 Q_8_6 = False,
 Q_5_1 = False,
 Q_4_5 = False,
 Q_3_3 = False,
 Q_4_7 = False,
 Q_5_3 = False,
 Q_1_6 = False,
 Q_4_4 = False,
 Q_7_3 = False,
 Q_8_5 = False,
```

```
 Q_1_7 = False,
 Q_2_8 = False,
 Q_7_5 = True,
 Q_8_8 = False,
 Q_3_7 = True,
 Q_4_8 = False,
 Q_5_6 = True,
 Q_6_2 = False,
 Q_1_1 = False,
 Q_2_2 = True,
 Q_7_6 = False,
 Q_7_2 = False,
 Q_7_8 = False,
 Q_8_7 = False,
 Q_8_4 = False,
 Q_6_6 = False,
 Q_2_5 = False,
 Q_5_8 = False,
 Q_2_7 = False,
 Q_6_3 = False,
 Q_5_2 = False,
 Q_4_3 = True,
 Q_6_4 = False,
 Q_2_3 = False,
 Q_7_7 = False]
solve time:  0.046875
PS D:\Course\formal_methods\HW\FM-HW3>
```

SMT(n=8):

```
PS D:\Course\formal_methods\HW\FM-HW3> python .\n-queen_smt.py
n:8
[Q_5 = 1,
 Q_8 = 7,
 Q_3 = 8,
 Q_2 = 2,
 Q_6 = 3,
 Q_4 = 6,
 Q_7 = 5,
 Q_1 = 4]
solve time:  0.03125
```

SMT/SAT性能比较

n	SMT time(s)	SAT time(s)
5	0.015625	0.015625
10	0.03125	0.109375
15	0.171875	0.484375
20	0.40625	1.40625

理论上：SAT solver的性能相对SMT solver更好，所需时间应该更短；

实际上：性能的发挥与程序员的工程水平有很大关系，在程序员水平欠佳的情况下，SAT solver的性能可能会比SMT solver更差。

2 用pure-SAT解决二进制减法问题

编码思路

$$
\begin{gathered}
d_i \leftrightarrow (a_i \leftrightarrow (b_i \leftrightarrow c_i)) \\
c_{i-1} \leftrightarrow ((a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i)) \\
\neg c_n \\
\neg c_0
\end{gathered}
$$

以上命题约束了$d = a + b$在n个二进制位且无进位的情况下的运算法则。通过逻辑命题给出b和d的初始条件，就可以通过SAT solver给出a的可能取值。即求解

$$
\begin{gathered}
a = d - b \\
\text{b and d are desribed in logic proposition}
\end{gathered}
$$

在实际的编码过程中，c_0需要被单独进行处理（或者改变命题中c的下标），以防止越界的情况发生。编码中各约束条件的意义如下：

约束条件	意义
`b_c`	根据二进制数b给出其逻辑命题描述
`d_c`	根据二进制数d给出其逻辑命题描述
`sum_c`	$d_i \leftrightarrow (a_i \leftrightarrow (b_i \leftrightarrow c_i))$
`carry_c`	$c_{i-1} \leftrightarrow ((a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i))$，当$i \geq 2$
`leftmost_carry`	$c_{i-1} \leftrightarrow ((a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i))$，当$i = 1$
`leftmost_carry_c`	$\neg c_0$
`rightmost_carry_c`	$\neg c_n$

使用文档

$$
a = d - b
$$

1. 输入操作数在二进制下的位数n；
2. 使用二进制（0/1串）输入b的值，保证输入为n位；
3. 使用二进制（0/1串）输入d的值，保证输入为n位；
4. SAT solver给出结果；

实验结果

```
PS D:\Course\formal_methods\HW\FM-HW3> python .\BinarySub_sat.py
n (in bits):3
BinarySub: a = d - b
b (in n bits):001
d (in n bits):100
[B_1 = False,
 B_2 = False,
 B_3 = True,
 A_1 = False,
 A_2 = True,
 C_1 = True,
 A_3 = True,
 C_2 = True,
 D_1 = True,
 D_3 = False,
 D_2 = False,
 C_0 = False,
 C_3 = False]
```

```
PS D:\Course\formal_methods\HW\FM-HW3> python .\BinarySub_sat.py
n (in bits):5
BinarySub: a = d - b
b (in n bits):00111
d (in n bits):10100
[B_1 = False,
 B_3 = True,
 C_3 = True,
 A_1 = False,
 B_5 = True,
 A_2 = True,
 C_4 = True,
 A_3 = True,
 D_4 = False,
 D_3 = True,
 C_2 = True,
 D_2 = False,
 C_0 = False,
 D_5 = False,
 B_2 = False,
 A_4 = False,
 C_1 = True,
 A_5 = True,
 D_1 = True,
 C_5 = False,
 B_4 = True]
PS D:\Course\formal_methods\HW\FM-HW3>
```