计算机程序理论与模型

第 1 次作业

12345678 张三

1 PySAT 求解 Refutation example 1 中的问题

1.1 Refutation example 1 中的问题介绍

我们的知识库 (Knowledge Base, KB) 包含以下信息：

- FirstGrade
- FirstGrade $\rightarrow$ Child
- Child $\wedge$ Male $\rightarrow$ Boy
- Kindergarten $\rightarrow$ Child
- Child $\wedge$ Female $\rightarrow$ Girl
- Female

我们需要使用 PySAT 从 KB 推断出 Girl，即证明 KB $\models$ Girl。

Atom	**Boolean Variable**
FirstGrade	x_1
Child	x_2
Male	x_3
Boy	x_4
Kindergarten	x_5
Female	x_6
Girl	x_7

表 1: 原子命题与布尔变量的对应关系

1.2 使用 PySAT 求解上述问题

接下来我们通过将 PySAT 来求解上述问题。

1.2.1 将问题转换为 SAT 问题

为了求解上述问题，我们首先需要将原问题转换为 SAT 问题，换句话说，我们需要将“证明 KB $\models$ Girl”转换为证明某个布尔公式不可满足，为了简化问题，我们先将各个原子命题 (atom) 转换为布尔变量 x_i，KB 中的 7 个原子命题对应 7 个布尔变量，其中的对应关系如表 1 所示。

为了证明 KB $\models$ Girl，即证明 KB $\wedge \neg$ Girl 是不可满足的 (unsatisfiable)，我们可以将 KB $\wedge \neg$ Girl 转换为公式 (formula)，该公式中的原子命题都通过表 1 转换为了对应的布尔变量，此时问题就转换为证明以下公式 (1) 是不可满足的。

$$x_1 \wedge (x_1 \to x_2) \wedge (x_2 \wedge x_3 \to x_4) \wedge (x_5 \to x_2) \wedge (x_2 \wedge x_6 \to x_7) \wedge x_6 \wedge \neg x_7 \tag{1}$$

接下来，我们需要将公式 (1) 转换为 CNF 范式。

首先，我们将其中的条件语句转换为析取语句，即将 $p \to q$ 转换为 $\neg p \vee q$，对于公式 (1) 应用该转换规则之后即可得到公式 (2)。

$$x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg(x_2 \wedge x_3) \vee x_4) \wedge (\neg x_5 \vee x_2) \wedge (\neg(x_2 \wedge x_6) \vee x_7) \wedge x_6 \wedge \neg x_7 \tag{2}$$

接着再应用德摩根定律 (De Morgan's Laws)，将其中 $\neg(x_2 \wedge x_3) \vee x_4$ 转换为 $(\neg x_2 \vee \neg x_3) \vee x_4$，以及将 $\neg(x_2 \wedge x_6) \vee x_7$ 转换为 $(\neg x_2 \vee \neg x_6) \vee x_7$，接着将其中多余的括号去掉，即可得到公式 (3)。公式 (3) 是一系列析取子句的合取，故公式 (3) 是 CNF 范式。

$$x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_5 \vee x_2) \wedge (\neg x_2 \vee \neg x_6 \vee x_7) \wedge x_6 \wedge \neg x_7 \tag{3}$$

此时原问题就转换为如何证明公式 (3) 不可满足，即原问题转换为了求解 SAT 问题。

1.2.2 用 PySAT 求解上述 SAT 问题

接下来我们将用 PySAT 来求解上述 SAT 问题，碍于篇幅，完整代码请看附录 A 或者附件，在这一小节我们只展示核心代码部分。

首先我们需要将之前得到的 CNF 范式输入到 PySAT 中，也就是将公式 (3) 输入到 PySAT 中。由于 PySAT 中使用自然数 i 来表示布尔变量 x_i，故公式 (3) 可以按照下面的方式输入到 PySAT 中：

```
# 输入 CNF 范式，其中 "1" 表示 "x1"，"2" 表示 "x2"
cnf = CNF(from_clauses=[
    [1], [-1, 2], [-2, -3, 4], [-5, 2], [-2, -6, 7], [6], [-7]
])
```

接下来利用 PySAT 中的 Solver 来判断该 CNF 范式是否可满足：

```
# 使用 Solver 判断该 CNF 范式是否可满足
with Solver(bootstrap_with=cnf) as solver:
    # 输入该 CNF 范式是否可满足
    print('formula is', f'{"s" if solver.solve() else "uns"}atisfieable')
```

在命令行中运行上述 python 代码，即可得到图 1 中的结果，可以看到该 CNF 范式是不可满足的，即可得出 KB $\models$ Girl，证毕。

```
(pysat_env) arnohu@ubuntu-server:~/pysat_homework$ python pysat_refutation_example_1.py
formula is unsatisfieable
```

图 1: PySAT 求解公式 (3) 是否可满足

2 使用 PySAT 求解 N 皇后问题

2.1 N 皇后问题描述

给定 N 个皇后和一个 $N \times N$ 的棋盘，要求找到一种在该棋盘上放置 N 个皇后的方法，使得每个皇后不能互相攻击彼此，即对于每个皇后 i 来说，它的同一行，同一列，同一主次对角线都不会出现另一个皇后 j。

2.2 将 N 皇后问题转换为 SAT 问题

我们需要用 CNF 范式将 N 皇后问题表示出来。为了方便后续的讲解，我们用布尔变量 $q_{i,j}$ 表示棋盘的第 i 行中的第 j 放置了一个皇后。

由于棋盘大小为 $N \times N$，并且我们有 N 个皇后，再加上同一行不能出现两个皇后，所以每一行必须有一个皇后存在。若我们想要表示第 i 行必须有一个皇后存在，可以用 $\bigvee_{j=0}^{N-1} p_{i,j}$ 来表示。那么“每一行必须有一个皇后存在”则可以用 $\bigwedge_{i=0}^{N-1} \bigvee_{j=0}^{N-1} p_{i,j}$ 来表示。同理“每一列必须有一个皇后存在”可以用 $\bigwedge_{j=0}^{N-1} \bigvee_{i=0}^{N-1} p_{i,j}$ 来表示。

接下来我们将皇后的各个互斥规则转换为公式。

对于第 i 行来说，不同列不能同时出现皇后，即 $\bigwedge_{j=0}^{N-2} \bigwedge_{k=j+1}^{N-1} (\neg(p_{i,j} \wedge p_{i,k}))$，通过德摩根定律即可得到 $\bigwedge_{j=0}^{N-2} \bigwedge_{k=j+1}^{N-1} (\neg p_{i,j} \vee \neg p_{i,k})$，故“每一行都不能同时有两个皇后”即为 $\bigwedge_{i=0}^{N-1} \bigwedge_{j=0}^{N-2} \bigwedge_{k=j+1}^{N-1} (\neg p_{i,j} \vee \neg p_{i,k})$。

同理“每一列都不能同时有两个皇后”即为 $\bigwedge_{i=0}^{N-1} \bigwedge_{j=0}^{N-2} \bigwedge_{k=j+1}^{N-1} (\neg p_{j,i} \vee \neg p_{k,i})$。

若第 i 行第 j 与第 k 行第 l 列在同一主对角线，那么一定有 $i - k = j - l$，故“同一主对角线不能同时有两个皇后”即为 $\bigwedge_{\substack{i,j,k,l=0 \\ i-k=j-l \\ i \neq k}}^{N-1} (\neg q_{i,j} \vee \neg q_{k,l})$。

同理“同一次对角线不能同时有两个皇后”即为 $\bigwedge_{\substack{i,j,k,l=0 \\ i+j=k+l \\ i \neq k}}^{N-1} (\neg q_{i,j} \vee \neg q_{k,l})$。

综上，将 N 皇后转换为 CNF 范式即为以下 6 个公式的合取：

$$\bigwedge_{i=0}^{N-1}\left(\bigvee_{j=0}^{N-1} q_{i,j}\right) \tag{4}$$

$$\bigwedge_{i=0}^{N-1}\left(\bigvee_{j=0}^{N-1} q_{j,i}\right) \tag{5}$$

$$\bigwedge_{i=0}^{N-1}\bigwedge_{j=0}^{N-2}\bigwedge_{k=j+1}^{N-1}\left(\neg q_{i,j}\vee\neg q_{i,k}\right) \tag{6}$$

$$\bigwedge_{j=0}^{N-1}\bigwedge_{i=0}^{N-2}\bigwedge_{k=i+1}^{N-1}\left(\neg q_{i,j}\vee\neg q_{k,j}\right) \tag{7}$$

$$\bigwedge_{\substack{i,j,k,l=0\\ i-k=j-l\\ i\neq k}}^{N-1}\left(\neg q_{i,j}\vee\neg q_{k,l}\right) \tag{8}$$

$$\bigwedge_{\substack{i,j,k,l=0\\ i+j=k+l\\ i\neq k}}^{N-1}\left(\neg q_{i,j}\vee\neg q_{k,l}\right) \tag{9}$$

此时解决 N 皇后问题就转换为求解 SAT 问题。

2.3 使用 PySAT 求解上述 SAT 问题

碍于篇幅，这里只展示部分核心代码，完整代码请看附录 B 或者附件。在 PySAT 中自然数 i 代表着布尔变量 x_i，而我们在 2.2 中的布尔变量都是 $p_{i,j}$ 的格式，故我们需要一个辅助函数来将 $p_{i,j}$ 转换成 x_i，换言之就是输入两个自然数 i 和 j，然后得到新的自然数作为布尔变量的下标。该辅助函数如下：

```
def varnum(i, j):
    # n 为 N 皇后问题的参数 N
    return n * i + j + 1
```

接下来我们要初始化一个空的 CNF 范式，然后依次将公式 (4) (9) 填充进去，初始化 CNF 范式如下所示：

```
formula = CNF()
```

接下来我们要往 formula 中填充公式 (4) 与公式 (6)，这两个公式能够确保同一行不出现两个皇后，且至少保证棋盘上有 N 个皇后，填充的过程如下：

```
# 确保每行只有一个皇后，且保证棋盘中有 N 个皇后
for i in range(n):
    # 对应公式 (4)
```

```
    formula.append([varnum(i, j) for j in range(n)])
    for j in range(n-1):
        for k in range(j+1, n):
            # 对应公式 (6)
            formula.append([-varnum(i, j), -varnum(i, k)])
```

同理，我们可以用类似的方式来填充公式 (5) 和公式 (7)，这两个公式能够确保同一列不出现两个皇后，且保证棋盘上有 N 个皇后，填充的过程如下：

```
# 确保每列只有一个皇后，且保证棋盘中有 N 个皇后
for j in range(n):
    # 对应公式 (5)
    formula.append([varnum(i, j) for i in range(n)])
    for i in range(n-1):
        for k in range(i+1, n):
            # 对应公式 (7)
            formula.append([-varnum(i, j), -varnum(k, j)])
```

而公式 (8) 和公式 (9) 分别对应着主对角线最多只能有一个皇后和次对角线最多只能有一个皇后，我们可以直接公式写出以下代码：

```
# 确保没有两个皇后在同一主对角线，同时确保没有两个皇后在同一次对角线
for i in range(n):
    for j in range(n):
        for k in range(n):
            for l in range(n):
                if i != k and i - k == j - l:
                    # 对应公式 (8)
                    formula.append([-varnum(i, j), -varnum(k, l)])
                if i != k and i + j == k + l:
                    # 对应公式 (9)
                    formula.append([-varnum(i, j), -varnum(k, l)])
```

至此，我们已经将 N 皇后问题对应的 CNF 范式输入进了 formula 变量中，接下来使用 SAT 求解器进行求解即可，若求解出某个变量为正数，则表示那个变量对应的位置可以放一个皇后，为了对结果进行可视化，可以放皇后的地方用”Q” 来表示，而不可以放皇后的地方则用”.” 表示，如下所示：

```
with Solver(bootstrap_with=formula) as solver:
    if solver.solve():
        model = solver.get_model()
        board = [['.' for _ in range(n)] for _ in range(n)]
        for val in model:
```

```
            if val > 0: # 检查正的变量值，正的变量表示当前位置可以放一个皇后
                i = (val - 1) // n # 从 val 中解析出行号
                j = (val - 1) % n # 从 val 解析出列好
                board[i][j] = 'Q'
        return board
    else:
        return None
```

当我们输入 N = 8 的时候，结果如图 2 所示，当我们输入 N = 10 的时候，结果如图 3 所示，可以看出我们的求解的结果是正确的。

```
(pysat_env) arnohu@ubuntu-server:~/pysat_homework$ python pysat_nquees.py
N = 8
. . . Q . . . .
. Q . . . . . .
. . . . . . Q .
. . Q . . . . .
. . . . . Q . .
. . . . . . . Q
. . . . Q . . .
Q . . . . . . .
```

图 2: PySAT 求解 8 皇后问题

```
(pysat_env) arnohu@ubuntu-server:~/pysat_homework$ python pysat_nquees.py
N = 10
. . . . Q . . . . .
. . . . . . . Q . .
. Q . . . . . . . .
. . . Q . . . . . .
. . . . . . . . . Q
. . . . . . Q . . .
. . . . . . . . Q .
. . . . . Q . . . .
. . Q . . . . . . .
Q . . . . . . . . .
```

图 3: PySAT 求解 10 皇后问题