Hands-On Session 2 (4/9)
AAA528, Spring 2025

Hakjoo Oh

Due: 4/9 23:59

Problem 1 z3를 이용하여 “아인슈타인 퍼즐”을 푸는 프로그램을 작성해보자.

- 가정:
  - 거리에 5개의 집이 일렬로 나란히 있다. 집의 색상은 모두 다르다.
  - 각 집에는 서로 다른 국적의 사람이 살고 있다.
  - 다섯 사람은 각각 어떤 음료를 마시고, 어떤 브랜드의 담배를 피우고, 어느 동물을 기른다.
  - 어느 두 사람도 같은 음료를 마시거나, 같은 담배를 피우거나, 같은 동물을 기르지 않는다.
- 속성:
  - 국적: 영국인, 스웨덴인, 덴마크인, 노르웨이인, 독일인
  - 집 색상: 빨강, 초록, 흰색, 노랑, 파랑
  - 음료: 차, 커피, 우유, 맥주, 물
  - 담배 브랜드: 펠멜, 던힐, 블랜드, 블루매스터, 프린스
  - 동물: 개, 새, 고양이, 말, 물고기
- 단서:
  - 영국인은 빨간 집에 산다.
  - 스웨덴인은 개를 기른다.
  - 덴마크인은 차를 마신다.
  - 초록 집은 흰 집의 바로 왼쪽에 있다.
  - 초록 집에 사는 사람은 커피를 마신다.
  - 펠멜 담배를 피우는 사람은 새를 기른다.
  - 노란 집에 사는 사람은 던힐 담배를 피운다.
  - 가운데 집(3번 집)에 사는 사람은 우유를 마신다.
  - 노르웨이인은 첫 번째 집(1번 집)에 산다.
  - 블랜드 담배를 피우는 사람은 고양이를 기르는 사람의 옆집에 산다.
  - 말을 기르는 사람은 던힐 담배를 피우는 사람의 옆집에 산다.
  - 블루매스터 담배를 피우는 사람은 맥주를 마신다.
  - 독일인은 프린스 담배를 피운다.
  - 노르웨이인은 파란 집 옆에 산다.
  - 블랜드 담배를 피우는 사람은 물을 마시는 사람의 옆집에 산다.
- 질문: 물고기를 기르는 사람의 국적은?

Problem 2 *Using Z3, let us implement a synthesizer that learns boolean functions from input-output examples. Consider boolean functions of the following DNF(Disjunctive Normal Form) form:*

$$f(x_1, x_2, \ldots, x_N) = \bigvee_i^M \bigwedge_j l_{ij}$$

where N is the number of arguments, M is the number of terms (connected by $\wedge$), and l_{ij} denotes literals (x or $\neg x$). For example, the XOR function

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

can be expressed as follows:

$$f(x_1, x_2) = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$$

Write a program that synthesizes the most general boolean function that satisfies given input-output examples.

Examples

1. *The example above (the XOR function) can be specified by the following text file:*

```
2
2
--positive--
01
10
--negative--
11
00
```

The top number indicates N and the second line indicates the number of postive/negative examples. For simplicity, assume the numbers of positive/nagative examples are the same. Beyond `--positive--` *are positive examples that make the function evaluate to 1. Beyond* `--negative--` *are negative examples on which the function evaluates to 0. Given these examples, the goal is to synthesize a boolean function that satisfies them:*

```
f(X) = (!x1 /\ x2) \/ (x1 /\ !x2)
```

2. *The specification for the AND function:*

```
2
3
--positive--
11
11
11
--negative--
01
10
00
```

The expected output of the synthesizer:

```
f(X) = (x1 /\ x2)
```

The systhesis result should be the most general one, where "most general" means that the synthesized function has a minimal number of terms. For example, the following is not the expected result (even though it satisfies the examples):

```
f(X) = (x1 /\ x2) \/ (x1 /\ !x1)
```

3. *Given the specification*

```
3
4
--positive--
000
010
100
110
--negative--
001
011
101
111
```

the synthesizer needs to output the following:

```
f(X) = (!x3)
```

4. *The specification:*

```
4
8
--positive--
0001
0011
0101
1001
1011
1101
1110
1111
--negative--
0000
0010
0100
0110
0111
1000
1010
1100
```

The result:

```
f(X) = (!x3 /\ x4) \/ (x1 /\ x2 /\ x3) \/ (!x2 /\ x4)
```

5. *The specification:*

```
20
16
--positive--
11001001000011111101
10101010001000100001
01101000110000100011
01001100010011000110
01100010100010111000
00001101110000011100
11010001001010010000
00100100111000001000
10001010011001111100
11000111010000000010
00001011101111101010
01100011101100010011
10011011001000100101
00010100101000001000
01111001100011100011
01000000010011011101
--negative--
10101101111110000101
01000101100010100010
10111011010010101001
10101010111111011100
01010110001000000010
01110011110100111100
11110001110110001011
10011100010110000011
11001110001011010011
01101001010110101001
11100001001101100100
00010001010001100100
00110011111110111100
11001001001110011101
11001110001001001001
10110011111011111001
```

The result:

```
f(X) = (!x4 /\ x10 /\ !x12) \/ (!x6 /\ !x10 /\ !x12) \/ (!x1 /\ x9 /\ x11 /\ !x18)
```

SAT Encoding *Let* p_{ij} *and* q_{ij} $(1 \leq i \leq M, 1 \leq j \leq N)$ *be boolean variables defined as follows:*

$$p_{ij} \iff \textit{term } i \textit{ contains } x_j, \qquad q_{ij} \iff \textit{term } i \textit{ contains } \neg x_j$$

Let P *be the number of positive (or negative) examples. Define variables* z_{ik} $(1 \leq i \leq M, 1 \leq k \leq P)$*:*

$$z_{ik} \iff \textit{term } i \textit{ evaluates to true when the } k^{th} \textit{ positive example is given}$$

Let us express the conditions that the variables introduced so far must satisfy. Suppose the first input-output example is $f(1, 1, 0, 0, \ldots, 1) = 1$*. This specification is translated into:*

1. $z_{11} \vee z_{21} \vee \cdots \vee z_{M1}$

2. $(\neg z_{i1} \vee \neg q_{i1}) \wedge (\neg z_{i1} \vee \neg q_{i2}) \wedge (\neg z_{i1} \vee \neg p_{i3}) \wedge (\neg z_{i1} \vee \neg p_{i4}) \wedge \cdots \wedge (\neg z_{i1} \vee \neg q_{iN})$ *for* $1 \leq i \leq M$

Suppose input-output examples include $f(1, 0, 1, 0, \ldots, 1) = 0$*. This negative example can be translated into:*

$$\bigwedge_{i=1}^{M} (q_{i1} \vee p_{i2} \vee q_{i3} \vee p_{i4} \vee \cdots \vee q_{iN})$$