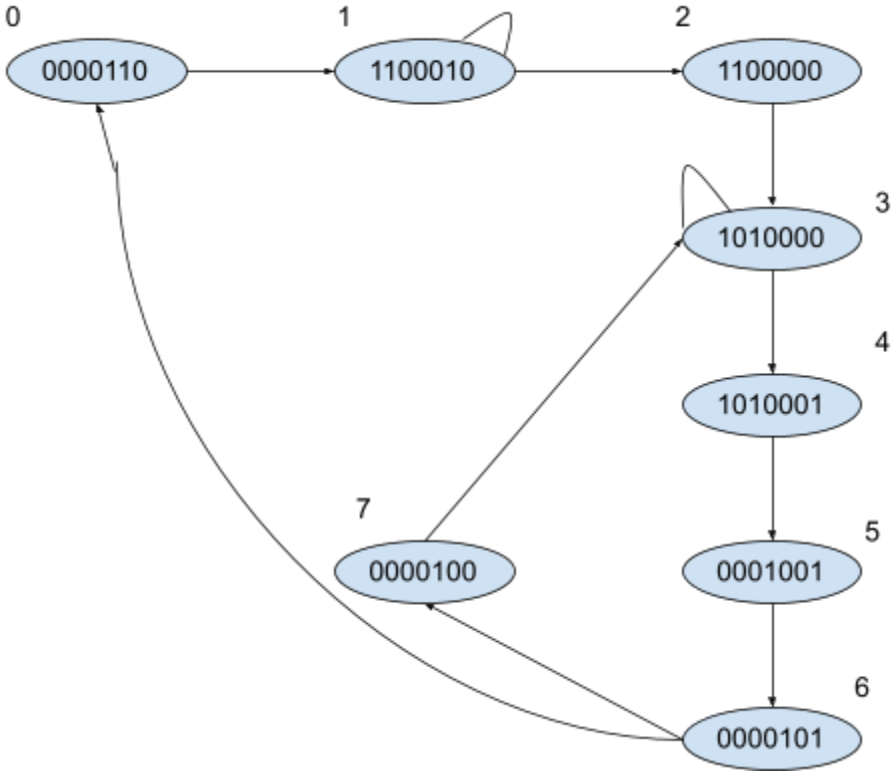
Cameron J Skidmore
1/25
Math Logic

Homework 1

The object of this assignment was to create a system in SBSAT to represent a traffic control system. This system is supposed to contain 6-8 bits of information. I decided to create one based on a traffic light. This system focuses on the behavior of a light in one direction of an intersection. This is a light that contains a red, yellow, and green light for moving straight, as well as a left turn green arrow. This system also includes a monitor for cross traffic waiting that tells the light to switch to allow this traffic to cross the intersection. The bits of information are as follows:
Note: in this system, direction 1 indicates the direction traffic is flowing with the light and direction 2 indicates traffic coming in the opposite direction

- V1: Light green for direction 1
- V2: Left turn light green
- V3: Light green direction 2
- V4: Light yellow direction 1
- V5: Light red direction 1
- V6: Traffic in direction 1 waiting to turn left
- V7: Cross traffic wait light detected

The following state diagram represents this system



In this state diagram, the states are all labeled with state 0 being the starting state. In this state, the light is red with at least one car waiting to turn left. The starting state of this can be defined as the following:

$$V_1^{0'} \wedge V_2^{0'} \wedge V_3^{0'} \wedge V_4^{0'} \wedge V_5^{0} \wedge V_6^{0} \wedge V_7^{0'}$$

The state transitions can be defined with the following statements:

0 -> 1: $(V_1^1 \equiv V_1^{0'}) \wedge (V_2^1 \equiv V_2^{0'}) \wedge (V_3^1 \equiv V_3^0) \wedge (V_4^1 \equiv V_4^0) \wedge (V_5^1 \equiv V_5^{0'}) \wedge (V_6^1 \equiv V_6^0) \wedge (V_7^1 \equiv V_7^0)$

1 -> 2: $(V_1^2 \equiv V_1^1) \wedge (V_2^2 \equiv V_2^1) \wedge (V_3^2 \equiv V_3^1) \wedge (V_4^2 \equiv V_4^1) \wedge (V_5^2 \equiv V_5^1) \wedge (V_6^2 \equiv V_6^{1'}) \wedge (V_7^2 \equiv V_7^1)$

2 -> 3: $(V_1^3 \equiv V_1^2) \wedge (V_2^3 \equiv V_2^{2'}) \wedge (V_3^3 \equiv V_3^{2'}) \wedge (V_1^3 \equiv V_1^2) \wedge (V_4^3 \equiv V_4^2) \wedge (V_5^3 \equiv V_5^2) \wedge (V_6^3 \equiv V_6^2) \wedge (V_7^3 \equiv V_7^2)$

3 -> 4: $(V_1^4 \equiv V_1^3) \wedge (V_2^4 \equiv V_2^3) \wedge (V_3^4 \equiv V_3^3) \wedge (V_4^4 \equiv V_4^3) \wedge (V_5^4 \equiv V_5^3) \wedge (V_6^4 \equiv V_6^3) \wedge (V_7^4 \equiv V_7^{3'})$

4 -> 5: $(V_1^5 \equiv V_1^{4'}) \wedge (V_2^5 \equiv V_2^4) \wedge (V_3^5 \equiv V_3^4) \wedge (V_4^5 \equiv V_4^{4'}) \wedge (V_5^5 \equiv V_5^4) \wedge (V_6^5 \equiv V_6^4) \wedge (V_7^5 \equiv V_7^4)$

5 -> 6: $(V_1^6 \equiv V_1^5) \wedge (V_2^6 \equiv V_2^5) \wedge (V_3^6 \equiv V_3^5) \wedge (V_4^6 \equiv V_4^{5'}) \wedge (V_5^6 \equiv V_5^{5'}) \wedge (V_6^6 \equiv V_6^5) \wedge (V_7^6 \equiv V_7^5)$

6 -> 0: $(V_1^0 \equiv V_1^6) \wedge (V_2^0 \equiv V_2^6) \wedge (V_3^0 \equiv V_3^6) \wedge (V_4^0 \equiv V_4^6) \wedge (V_5^0 \equiv V_5^6) \wedge (V_6^0 \equiv V_6^{6'}) \wedge (V_7^0 \equiv V_7^{6'})$

6 -> 7: $(V_1^7 \equiv V_1^6) \wedge (V_2^7 \equiv V_2^6) \wedge (V_3^7 \equiv V_3^6) \wedge (V_4^7 \equiv V_4^6) \wedge (V_5^7 \equiv V_5^6) \wedge (V_6^7 \equiv V_6^6) \wedge (V_7^7 \equiv V_7^{6'})$

7 -> 3: $(V_1^3 \equiv V_1^{7'}) \wedge (V_2^3 \equiv V_2^7) \wedge (V_3^3 \equiv V_3^{7'}) \wedge (V_4^3 \equiv V_4^7) \wedge (V_5^3 \equiv V_5^{7,}) \wedge (V_6^3 \equiv V_6^7) \wedge (V_7^3 \equiv V_7^7)$

With these state transitions all figured out, we can put this whole system in sbsat to check for it satisfiability:

```
p cnf 56 137

c --Express starting state
-1 0
-2 0
-3 0
-4 0
5 0
6 0
-7 0


c -- This is setting as and instead of or for starting state clauses

c -- Equivalence (no change between states)
7 -3 0
-7 3 0

c -- XOR (bit change)
-8 -4 0
8 4 0

c -- Transition from 0 -> 1
-8 -1 0
8 1 0

-9 -2 0
9 2 0

10 -3 0
-10 3 0

11 -4 0
-11 4 0
```

Note: the entire file for this system is attached

Now we can run this through sbsat:

```
cam@pop-os:~/data/MathLogic$ sbsat homework1.cnf
Reading File homework1.cnf  ....
Reading CNF 137/137
Simplify removed 0 clauses
Found 63 XOR functions
Simplify removed 126 clauses
Found 0 MAJV= functions
Simplify removed 0 clauses
Found 0 ITE= functions
Simplify removed 0 clauses
Found 0 AND=/OR= functions
Simplify removed 0 clauses
Built 11 clause BDDs
Simplify removed 11 clauses
Preprocessing .... Done
Satisfiable
Total Time: 0.007
cam@pop-os:~/data/MathLogic$
```

Looks good! The system is satisfiable!