ENSTA Paris
2022-2023

Decision Procedures for Artificial Intelligence – INF656L

Alexandre Chapoutot

SAT Part - Practical work 1

Goal(s)

- Modeling decision problem with propositional logic;
- First algorithm to prove satisfiability of propositional logic formula.

Exercise 1

Let's consider the interpretation I such that $\{p \mapsto F, q \mapsto T, r \mapsto T\}$. Does I satisfy the following propositional formulas?

1. $(p \implies \neg q) \vee \neg(r \wedge q)$
2. $(\neg p \vee \neg q) \implies (p \vee \neg r)$
3. $\neg(\neg p \implies \neg q) \wedge r$
4. $\neg(\neg p \implies (q \wedge \neg r))$

Solution:
I satisfies 1, 3 and 4 and does not satisfy 2

Exercise 2 – Truth table method

Question 1

Determine with a truth table method whether the formula

$$(p \implies q) \vee (p \implies \neg q)$$

is valid.

Solution:

P	Q	$((P$	$\rightarrow$	$Q)$	$\vee$	$(P$	$\rightarrow$	$(\neg$	$Q)))$
1	1	1	1	1	**1**	1	0	0	1
1	0	1	0	0	**1**	1	1	1	0
0	1	0	1	1	**1**	0	1	0	1
0	0	0	1	0	**1**	0	1	1	0

Question 2

Use the truth tables method to determine whether the formula $\phi \equiv (p \wedge \neg q) \implies (p \wedge q)$ is a logical consequence of the formula $\psi \equiv \neg p$.

Solution:

P	$(\neg$	$P)$
1	**0**	1
0	**1**	0

P	Q	$((P$	$\wedge$	$(\neg$	$Q))$	$\rightarrow$	$(P$	$\wedge$	$Q))$
1	1	1	0	0	1	**1**	1	1	1
1	0	1	1	1	0	**0**	1	0	0
0	1	0	0	0	1	**1**	0	0	1
0	0	0	0	1	0	**1**	0	0	0

All the interpretation making ψ satisfiable make ϕ satisfiable, *i.e.*, $\psi \models \phi$.

Question 3

Socrate says:

> If I am guilty, I must be punished; I'm not guilty. Thus I must not be punished.

Is the argument logically correct?

Solution:

- P stands for "I am guilty"
- Q stands for "I must be punished"

So we have

$$\phi \equiv (P \implies Q) \wedge \neg P$$

and

$$\psi \equiv \neg Q \quad \text{I must not be punished}$$

We have to prove that $\phi \models \psi$ so

Q	$(\neg$	$Q)$
1	**0**	1
0	**1**	0

and

P	Q	$((p$	$\rightarrow$	$Q)$	$\wedge$	$(\neg$	$P))$
1	1	1	1	1	**0**	0	1
1	0	1	0	0	**0**	0	1
0	1	0	1	1	**1**	1	0
0	0	0	1	0	**1**	1	0

Consider $P =$ false and $Q =$ true so $\phi \not\models \psi$

Exercise 3 – Logical formalization

A stands for "Aldo is Italian" and B stands for "Bob is English".

Question 1

Formalize the following sentences:

1. Aldo isn't Italian
2. Aldo is Italian while Bob is English
3. If Aldo is Italian then Bob is not English
4. Aldo is Italian or if Aldo isn't Italian then Bob is English
5. Either Aldo is Italian and Bob is English, or neither Aldo is Italian nor Bob is English

Solution:

1. $\neg A$
2. $A \wedge B$
3. $A \implies \neg B$
4. $A \vee (\neg A \implies B)$ same than $A \vee B$
5. $(A \wedge B) \vee (\neg A \wedge \neg B)$ same than $A \Leftrightarrow B$

Exercise 4 – Enigma

Aladdin finds two trunks T_A and T_B in a cave. He knows that each of them either contains a treasure or a fatal trap. On trunk T_A is written:

"At least one of these two trunks contains a treasure."

On trunk T_B is written:

"In T_A there's a fatal trap."

Aladdin knows that either both the inscriptions are true, or they are both false.

We want to answer the questions:

- Can Aladdin choose a trunk being sure that he will find a treasure?
- If this is the case, which trunk should he open?

Question 1

Give the propositional formulas associated to inscriptions on trunks

Solution:
Let's consider propositions

- A stands for "Trunk T_A contains the treasure"
- B stands for "Trunk T_B contains the treasure"

Model of the inscriptions

- $A \vee B$ stands for "At least one of these two trunks contains a treasure".
- $\neg A$ stands for "T_A contains a trap".

Question 2

Model the problem with logical formulas

Solution:
Model of the problem

- $(A \vee B) \leftrightarrow \neg A$ stands for "either both the inscriptions are true, or they are both false".

Question 3

Give the truth table of problem's model and answer the questions

Solution:

a	b	$((a$	$\vee$	$b)$	$\leftrightarrow$	$(\neg$	$a))$
1	1	1	1	1	**0**	0	1
1	0	1	1	0	**0**	0	1
0	1	0	1	1	**1**	1	0
0	0	0	0	0	**0**	1	0

Remark: solutions generated from `https://cs.uwaterloo.ca/~cbright/logic/truthtable.php`
The only valuation of variable which satisfies the formula is $A =$ false and $B =$ true so Aladdin can open trunk T_B, being sure that it contains a treasure

Exercise 5 – Diner

Question 1
Model with propositional logic the following problem:

> I would like to invite some of the following people to a party: Alice, Ben, Chris and David. If I invite Alice, I should also invite Benoît. I cannot invite Benoît and Christophe to the same party. I want to invite at least three of them (this condition must also be expressed as a logical formula).

Solution:

- A stands for "I invite Alice"
- B stands for "I invite Ben"
- C stands for "I invite Chris"
- D stands for "I invite David".

The logical model is then

1. $A \implies B$ (if I invite Alice then I invite Ben)
2. $\neg(B \wedge C)$ (I cannot invite Ben and Chris in the same time)
3. At leat 3 persons are invited

$$(A \wedge B \wedge C) \vee (A \wedge B \wedge D) \vee (A \wedge C \wedge D) \vee (B \wedge C \wedge D)$$

Question 2
Put these formulas in conjunctive normal form

Solution:

1. $B \vee \neg A$
2. $\neg B \wedge \neg C$
3. $(A \vee B) \wedge (A \vee C) \wedge (A \vee D) \wedge (B \vee C) \wedge (B \vee D) \wedge (C \vee D) \wedge (A \vee B \vee C) \wedge (A \vee B \vee D) \wedge (A \vee C \vee D) \wedge (B \vee C \vee D) \wedge (A \vee B \vee C \vee D)$

Exercise 6 – Simple decision procedure
An encoding of the PL formula in CNF is given by the DIMACS CNF format. A sample file is:

```
c  simple_v3_c2.cnf
c
p cnf 3 2
1 -3 0
2 3 -1 0
```

All lines beginning with a "c" character are comments. The contents of the file begin with the words 'p cnf', followed by the number of n variables and the number of c clauses of the problem. In a DIMACS CNF file, a variable is represented as follows by an integer between 1 and n. The negation: is represented by by the $-$ sign. A clause is represented as a list of literal, separated by spaces, and ending with a 0. A problem is represented as a succession of clauses. By example, the DIMACS CNF file given as an example encodes the formula

$$(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1)$$

We will not deal with the reading of a CNF DIMACS file, but we are inspired by the encoding of PL formula in CNF.

The objective of this exercise is to implement a simple SAT algorithm based on a *model checking* method. The idea is to generate all the possible interpretations and to test one by one these interpretations to find those which satisfy the logical formula.

Question 1

Give a simple data structure that represents a PL formula in CNF according to the DIMACS CNF format.

> **Solution:**
> A list of lists of integers

Question 2

Write an evaluation function of a PL formula in CNF given an interpretation of the variables, *i.e.*, a Boolean value for each variable.

Question 3

Write a function that generates all binary words of length n given in parameter.

Question 4

Using the previous two functions, define a decision procedure.

Question 5

Transform PL formula of Exercise 3 into DIMACS format and apply this decision procedure to find a solution.

Question 6

Try solving this problem

```
c  an example from Quinn's text, 16 variables and 18 clauses.
c Resolution: SATISFIABLE
c
p cnf 16 18
  1    2  0
 -2   -4  0
  3    4  0
 -4   -5  0
  5   -6  0
  6   -7  0
  6    7  0
  7  -16  0
  8   -9  0
 -8  -14  0
  9   10  0
  9  -10  0
-10  -11  0
 10   12  0
 11   12  0
 13   14  0
 14  -15  0
 15   16  0
```

Solution:

```python
def bitfield(n):
    return [True if digit=='1' else False for digit in bin(n)[2:]] # [2:] to chop off the "0b" part

def createModel (n):
    allValues = []
    for i in range(0, pow(2, n)):
        w = bitfield(i)
        rest = n - len(w)
        wr = ([False] * rest) + w
        allValues.append(wr)
    return allValues

def evalCNF (cnf, model):
    formulaValue = True
    for clause in cnf:
        clauseValue = False
        for lit in clause:
            if (lit > 0):
                clauseValue = clauseValue or model[lit-1]
            else:
                clauseValue = clauseValue or (not model[(-lit)-1])
        formulaValue = formulaValue and clauseValue
    return formulaValue

def decisionProcedure (nbVar, cnf):
    allValues = createModel(nbVar)
    flag = False
    for w in allValues:
        if (evalCNF(cnf, w)):
            print ("SAT with ", w)
            return
    if (not flag):
        print ("UNSAT")

cnf = [
    [1, -3], [2, 3, -1]
]
print ("Problem 1 is")
decisionProcedure (3, cnf)

cnf2 = [
    [1, 2], [-2, -4], [3, 4], [-4,-5],
    [5, -6], [6 -7], [6, 7], [7 -16],
    [8, -9], [-8, -14], [9, 10], [9, -10],
    [-10, -11], [10, 12], [11, 12], [13, 14],
    [14, -15], [15, 16]
]
print ("\nProblem 2 is")
decisionProcedure (16, cnf2)
```