problemSet5

Problem Set 5

Ben Nguyen

7.15 (15 pts)

Use resolution to prove the sentence ¬A∧¬B from the clauses in Exercise 7.25

Exercise 7.25 Convert the following set of sentences to clausal form.

1. S1: A ⇔ (B∨E).

(¬A∨(B∨E))∧(¬(B∨E)∨A)

(¬A∨B∨E)∧((¬B∧¬E)∨A)

(¬A∨B∨E)∧(¬B∨A)∧(¬E∨A)

2. S2: E⇒D.

¬E∨D

3. S3: C∧F⇒¬B.

¬(C∧F)∨¬B

¬C∨¬F∨¬B

4. S4: E⇒B.

¬E∨B

5. S5: B⇒F.

¬B∨F

6. S6: B⇒C.

¬B∨C

Give a trace of the execution of DPLL on the conjunction of these clauses.

Proof by Contradiction: ¬(¬A∧¬B) = A∨B

A∨B	¬B∨A
A∨A	Simplify
A	¬A∨B∨E
B∨E	¬E∨B
B	¬B∨C
C	¬C∨¬F∨¬B
¬F∨¬B	¬B∨F
¬B∨¬B	Simplify
¬B	B
$\emptyset$	

Contradiction - B must be true because it is correct given the steps, then ¬B must also be true given the steps. Since both cannot be true, this proves the contradiction

7.16 (5+5+5 = 15 pts)

This exercise looks into the relationship between clauses and implication sentences.

1. Show that the clause $(\neg P_1 \vee \cdots \vee \neg P_m \vee Q)$ is logically equivalent to the implication sentence $(P_1 \wedge \cdots \wedge P_m) \Rightarrow Q$.

$$(\neg P_1 \vee \cdots \vee \neg P_m \vee Q)$$

$$(\neg(P_1) \vee \neg P_2 \vee \cdots \vee \neg P_m \vee Q)$$

$$(\neg(P_1 \wedge P_2) \vee \neg P_3 \vee \cdots \vee \neg P_m \vee Q)$$

…

$$(\neg(P_1 \wedge P_2 \wedge \cdots \wedge P_m) \vee Q)$$

$$(P_1 \land \cdots \land P_m) \Rightarrow Q$$

2. Show that every clause (regardless of the number of positive literals) can be written in the form $(P_1 \land \cdots \land P_m) \Rightarrow (Q_1 \lor \cdots \lor Q_n)$, where the P’s and Q’s are proposition symbols. A knowledge base consisting of such sentences is in implicative normal form or Kowalski form.

$$(P_1 \land \cdots \land P_m) \Rightarrow (Q_1 \lor \cdots \lor Q_n),$$

$$\neg(P_1 \land \cdots \land P_m) \lor (Q_1 \lor \cdots \lor Q_n),$$

$$(\neg P_1 \lor \cdots \lor \neg P_m) \lor (Q_1 \lor \cdots \lor Q_n),$$

$$(\neg P_1 \lor \cdots \lor \neg P_m) \lor (Q_1 \lor \cdots \lor Q_n),$$

$$\neg P_1 \lor \cdots \lor \neg P_m \lor Q_1 \lor \cdots \lor Q_n,$$

3. Write down the full resolution rule for sentences in implicative normal form.

$$\frac{\neg P_1 \lor \cdots \lor l_i \lor \cdots \lor \neg P_m \lor Q_1 \lor \cdots \lor Q_n \qquad \neg A_1 \lor \cdots \lor \neg A_m \lor B_1 \lor \cdots \lor r_j \lor \cdots \lor B_n}{\neg P_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor \neg P_m \lor \neg A_1 \lor \cdots \lor \neg A_m \lor Q_1 \lor \cdots \lor Q_n \lor B_1 \lor \cdots \lor r_{j-1} \lor r_{j+1} \lor \cdots \lor B_n}$$

7.21 (3+4+3+3 = 13 pts)

A propositional 2-CNF expression is a conjunction of clauses, each containing exactly 2 literals, e.g.,

(A∨B)∧(¬A∨C)∧(¬B∨D)∧(¬C∨G)∧(¬D∨G).

1. Prove using resolution that the above sentence entails G.

(A∨B)

(¬A∨C)

(¬B∨D)

(¬C∨G)

(¬D∨G)

Proof by Contradiction: Assume ¬G

(¬G)	(¬C∨G)

(¬C)	(¬A∨C)
(¬A)	(A∨B)
(B)	(¬B∨G)
(G)	(¬G)
∅	

2. Two clauses are semantically distinct if they are not logically equivalent. How many semantically distinct 2-CNF clauses can be constructed from n proposition symbols?

There are $\binom{2}{2n} - (n-1)$ semantically distinct clauses. There are $2n$ symbols, because each symbol can be inverted *eg. ¬A or A*, and each of the $2n$ symbols can be chosen for each of the 2 symbols. However, since the problem says semantically distinct we cannot count clauses like ¬A∨A, because they just evaluate to true, so that takes $(n-1)$ from our total.

3. Using your answer to (b), prove that propositional resolution always terminates in time polynomial in n given a 2-CNF sentence containing no more than n distinct symbols.

$$\binom{2}{2n} - (n-1) = 2n^2 - 2n + 1$$

Two 2-CNF clauses will always create at maximum another 2-CNF clause when they resolve, therefore there at maximum the algorithm will go through all $2n^2 - 2n + 1$ clauses, which is polynomial time.

4. Explain why your argument in c does not apply to 3-CNF.

Two 3-CNF clauses does not always create at maximum another 3-CNF clause, so the same logic does not apply

7.22 (4+4+4 = 12 pts)

Prove each of the following assertions:

1. Every pair of propositional clauses either has no resolvents, or all their resolvents are logically equivalent.

Case 1:

If a pair of propositional clauses do not resolve, there must be zero resolvents.

Case 2:

If a pair of propositional clauses resolves to one resolvent, then all resolvents must be logically equivalent because there is only one resolvent.

Case 3:

If a pair of propositional clauses resolves to more than one resolvent, then there must be more than one shared symbol in the pair of propositional clauses where they are the inverse of each other in each clause. For example, ¬P will be in clause 1, and P will be in clause 2. Therefore, if you resolve each of those resolvents with every single shared symbol, they will all resolve to the exact same thing in the end, making them all logically equivalent.

This covers all cases of every different number of resolvents, therefore every pair of propositional clauses either has no resolvents, or all their resolvents are logically equivalent.

2. There is no clause that, when resolved with itself, yields (after factoring) the clause (¬P∨¬Q).

The only clauses that can be resolved with themself must have at least one literal and the corresponding opposite literal in the same clause.

Clause 1: ¬A∨A∨…

Clause 2: ¬A∨A∨…

Resolvent: ¬A∨A∨…

Therefore, resolving with themself will lead to a clause with the form: ¬A∨A∨… which is always true

3. If a propositional clause C can be resolved with a copy of itself, it must be logically equivalent to True.

The only clauses that can be resolved with themself must have at least one literal and the corresponding opposite literal in the same clause.

Clause: ¬A∨A∨…

However, having a literal and the corresponding opposite literal in the same clause will always force it to be true

Any clause that contains ¬A∨A is always true

7.23 (5+5+5 = 15 pts)

Consider the following sentence:

[(Food⇒Party)∨(Drinks⇒Party)]⇒[(Food∧Drinks)⇒Party].

1. Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.

Food	Party	Drinks	[(Food⇒Party)∨(Drinks⇒Party)]⇒[(Food∧Drinks)⇒Party]
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	T

Since all possible valid inputs satisfy the implication, the implication is true.

2. Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).

[(Food⇒Party)∨(Drinks⇒Party)]⇒[(Food∧Drinks)⇒Party]

[(F⇒P)∨(D⇒P)]⇒[(F∧D)⇒P]

[(¬F∨P)∨(¬D∨P)]⇒[¬(F∧D)∨P]

[¬F∨P∨¬D∨P]⇒[¬(F∧D)∨P]

$[\neg F \lor \neg D \lor P] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg(F \land D) \lor P] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg F \lor \neg D \lor P] \Rightarrow [\neg F \lor \neg D \lor P]$

Therefore, this sentence is satisfiable because both sides compile down to the same CNF

3. Prove your answer to (a) using resolution.

$[(Food \Rightarrow Party) \lor (Drinks \Rightarrow Party)] \Rightarrow [(Food \land Drinks) \Rightarrow Party]$

$[(F \Rightarrow P) \lor (D \Rightarrow P)] \Rightarrow [(F \land D) \Rightarrow P]$

$[(\neg F \lor P) \lor (\neg D \lor P)] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg F \lor P \lor \neg D \lor P] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg F \lor \neg D \lor P] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg(F \land D) \lor P] \Rightarrow [\neg(F \land D) \lor P]$

$[\neg F \lor \neg D \lor P] \Rightarrow [\neg F \lor \neg D \lor P]$

$\neg(\neg F \lor \neg D \lor P) \lor \neg F \lor \neg D \lor P$

$(F \land D \land \neg P) \lor \neg F \lor \neg D \lor P$

$(F \lor \neg F \lor \neg D \lor P) \land (D \lor \neg F \lor \neg D \lor P) \land (\neg P \lor \neg F \lor \neg D \lor P)$

$(True) \land (True) \land (True)$

True

By resolution, this statement is always true

7.26 (10 pts)

Convert the following set of sentences to clausal form. Give a trace of the execution of DPLL on the conjunction of these clauses.

1. $A \Leftrightarrow (B \lor E)$.

$(\neg A \lor (B \lor E)) \land (\neg(B \lor E) \lor A)$

(¬A∨B∨E)∧((¬B∧¬E)∨A)

(¬A∨B∨E)∧(¬B∨A)∧(¬E∨A)

2. E⇒D.

¬E∨D

3. C∧F⇒¬B.

¬(C∧F)∨¬B

¬C∨¬F∨¬B

4. E⇒B.

¬E∨B

5. B⇒F.

¬B∨F

6. B⇒C.

¬B∨C

```
function DPLL_Satisfiable(s) returns true or false
    inputs: s, a sentence in propositional logic
    clauses <- the set of clases in the CNF representation of s
    symbols <- a list of the proposition symbols in s
    return DPLL(clauses, symbols, {})

function DPLL(clauses, symbols, model) returns true or false
    if every clause in clauses is true in model then return true
    if some clause in clauses is false in model then return false
    P,value <- Find-Pure-Symbol(symbols, clauses, model)
    if P is non-null then return DPLL(clauses, symbols - P, model \union {P=va
    P,value <- Find-Unit-Clause(clauses, model)
    if P is non-null then return DPLL(clauses, symbols - P, model \union {P=va
    P <- First(symbols); rest <- Rest(symbols)
    return DPLL(clauses, rest, model \union {P=true}) or
        DPLL(clauses, rest, model \union {P=false}) or
```

Iteration	Symbols	Model	Pure?	Unit?	return?

Iteration	Symbols	Model	Pure?	Unit?	return?
0	{A,B,C,D,E,F}	{}	D = True	N/A	
1	{A,B,C,E,F}	{D = True}	N/A	N/A	
2	{B,C,E,F}	{D = True, A = True}	N/A	N/A	
3	{C,E,F}	{D = True, A = True, B = True}	N/A	$\neg B \lor F \implies F =$ True	
4	{C,E}	{D = True, A = True, B = True, F = True}	N/A	$\neg C \lor \neg F \lor \neg B \implies C =$ False	
5	{E}	{D = True, A = True, B = True, F = True, C = False}	N/A	N/A	$\neg B \lor C$ returns false
6	{C,E,F}	{D = True, A = True, B = False}	N/A	$\neg A \lor B \lor E \implies E =$ True	
7	{C,F}	{D = True, A = True, B = False, E = True}	N/A	N/A	$\neg E \lor B$ returns false
8	{B,C,E,F}	{D = True, A = False}	E = False	N/A	
9	{B,C,F}	{D = True, A = False, E = False}	B = False	N/A	
10	{C,F}	{D = True, A = False, E = False, B = False}	N/A	N/A	return true