CSC 503 Homework Assignment 3

Due September 15, 2014

September 8, 2014

1. [20 points] Construct a formula ϕ in **DNF** based on the following truth table:

p	q	ϕ
T	T	F
T	F	T
F	T	T
F	F	T

Answer

From the above truth tables we can derive the DNF with following steps

(a) Rows with truth values 'T' are $2, 3, 4$.

(b) The formula ϕ can be represented in DNF as $(Row2 \vee Row3 \vee Row4)$ which means that except for these there can be no other values as T.

(c) Each Row can be represented by the corresponding values of the literals. So the formula becomes ϕ : $(p \wedge \neg q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$

Thus the DNF for the given truth table is $(p \wedge \neg q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$

2. [20 points] Construct a formula ϕ in **CNF** based on the following truth table:

p	q	r	ϕ
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	T

Answer

(a) Rows with truth values F are $3, 4, 6, 7$

(b) The formula ϕ can be represented in CNF as $(\neg Row3 \wedge \neg Row4 \wedge \neg Row6 \wedge \neg Row7)$ which means that except for the false value of the rows $3, 4, 6, 7$ others are T.

(c) Each Row can be represented by the corresponding values of the literals. So the formula becomes ϕ : $(\neg(p \wedge \neg q \wedge r) \wedge \neg(p \wedge \neg q \wedge \neg r) \wedge \neg(\neg p \wedge q \wedge \neg r) \wedge \neg(\neg p \wedge \neg q \wedge r))$

(d) Reducing the above formula using De'Morgans Law we get ϕ : $((\neg p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (p \vee \neg q \vee r) \wedge (p \vee q \vee \neg r))$

Thus the CNF for the given truth table is $((\neg p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (p \vee \neg q \vee r) \wedge (p \vee q \vee \neg r))$

3. [40 points] Apply algorithm HORN from page 66 of the textbook to the following Horn formula.

$$\begin{array}{ll}(p \wedge q \wedge w \to \bot) & \wedge \\ (t \to \bot) & \wedge \\ (r \to p) & \wedge \\ (\top \to r) & \wedge \\ (\top \to q) & \wedge \\ (r \wedge u \to w) & \wedge \\ (u \to s) & \wedge \\ (\top \to u) & \end{array}$$

Your answer should list propositional letters in the order in which they are marked as well as giving the overall answer.

Answer

(a) Mark all occurances of $\top$.

(b) Mark r, q, u from $(\top \to r), (\top \to q), (\top \to u)$

$$\begin{array}{ll}(p \wedge \mathbf{q} \wedge w \to \bot) & \wedge \\ (t \to \bot) & \wedge \\ (\mathbf{r} \to p) & \wedge \\ (\top \to \mathbf{r}) & \wedge \\ (\top \to \mathbf{q}) & \wedge \\ (\mathbf{r} \wedge \mathbf{u} \to w) & \wedge \\ (\mathbf{u} \to s) & \wedge \\ (\top \to \mathbf{u}) & \end{array}$$

(c) Mark p, w, s from $(r \to p), (r \wedge u \to w), (u \to s)$

$$\begin{array}{ll}(\mathbf{p} \wedge \mathbf{q} \wedge \mathbf{w} \to \bot) & \wedge \\ (t \to \bot) & \wedge \\ (\mathbf{r} \to \mathbf{p}) & \wedge \\ (\top \to \mathbf{r}) & \wedge \\ (\top \to \mathbf{q}) & \wedge \\ (\mathbf{r} \wedge \mathbf{u} \to \mathbf{w}) & \wedge \\ (\mathbf{u} \to \mathbf{s}) & \wedge \\ (\top \to \mathbf{u}) & \end{array}$$

(d) Mark $\bot$ from $(\mathbf{p} \wedge \mathbf{q} \wedge \mathbf{w} \to \bot)$

The order order of propositional letters is $r, q, u, p, w, s, \bot$. Since we have marked $\bot$, the horn formula is **'Unsatisfiable'**.

4. [20 points] Explain in what sense the SAT solving technique, as presented in lecture and the book, can be used to check whether formulas are tautologies.

Answer

SAT solving technique is an extension of the HORN Marking algorithm where we can compute the contraints on the subformulas of the general formula ϕ that makes ϕ true.

There are multiple SAT solving techniques. One that solves in linear time of the number to nodes in a directed acyclic graph(DAG) of the formula phi and the other one which is an enhancement to this which solves in cubic time.

Linear Solver We use the marking algorithm on the parse tree of the formula. Before running the algorithm we transform the formula into the semantically equivalent form having same propositional atoms. The algorithm to check whether formulas are as follows:

(a) Take the formula ϕ we want to check for tautology.

(b) Convert the formula ϕ using the following transformations

i. $T(p) = p$

ii. $T(\neg\phi) = \neg T(\phi)$

iii. $T(\phi_1 \wedge \phi_2) = T(\phi_1) \wedge T(\phi_2)$

iv. $T(\phi_1 \vee \phi_2) = \neg(\neg T(\phi_1) \wedge \neg T(\phi_2))$

v. $T(\phi_1 \rightarrow \phi_2) = \neg(T(\phi_1) \wedge \neg T(\phi_2))$

(c) Now we can only prove if the formula is tautology if we can prove that its negation is never satisfiable. So consider the $\neg\phi$ as the new formula and we would like to see if that is satisfiable.

(d) Construct the parse tree for the formula $\neg\phi$

(e) Consider the root node of the tree to be true and then apply the following rules to find out the corresponding contraints of the child nodes.

i. Negation Laws: If the current node is $\neg$ and has the value p then child node should have the constraint $\neg p$ i.e. if the current node has constraint T then the child node will have $\neg T = F$ and vice versa.

ii. Conjunction Laws: In this case if the current node is T then each child node will have a constraint T. But if the current node is F then atleast one of the child must have F. So anyone of the child can be false.

(f) Continue doing the expansion till we reach the leaf nodes or reach a contradiction. If we reach the contradiction then we say that the formula is not satisfiable. Which means $\neg\phi$ is not satisfiable and thus ϕ must be a tautology. However if the leaf node is reached then we go bottom-up tp verify whether all the contraints in conjunction hold good and the root node is still in T state. If this is the case then we say that $\neg\phi$ is satisfiable and hence ϕ is not a tautology.

Note: In the above algorithm we might reach a state where we have not reached even the contradiction nor the leaf node. In this case we will have some free (not constrained) nodes. In such cases assume temperory values for these nodes. Say first time we assume T for the node and we reach a contradiction them we assume the permanent value of that node as $\neg T = F$ and re- compute the constraint. If we reach a contradiction again then we can conclude that $\neg\phi$ is not satisfiable. Otherwise it is satisfiable and ϕ is not a tautology. This extension of the first algorithm takes cubic time to compute and hence is called a **cubic solver**.