CS 2742 (Logic in Computer Science) – Fall 2008
Lecture 7

Antonina Kolokolova

September 21, 2008

3 Resolution.

Recall that a formula is in the CNF (conjunctive normal form) if it is a $\wedge$ of $\vee$s of literals (variables or their negation.)

In this lecture we will talk about proving (or, actually, finding contradictions) statements that are in this special form. As a warm-up exercise, let us see how to simplify CNF formulas further.

Theorem 1. *Any CNF formula is equivalent to a CNF formula with no more than 3 literals in each clause.*

Proof. Suppose some clause has more than 3 literals. Take the last two literals of the clause, call them l_i and l_j. Now, introduce new variable y which will have the same value as $(l_i \vee l_j)$ and replace $(l_i \vee l_j)$ in the original clause containing $(l_i \vee l_j)$ with y. To force y to have the same value as $(l_i \vee l_j$, add clauses $(y \leftrightarrow (l_i \vee l_j))$, converted to CNF.

How do we convert $(y \leftrightarrow (l_i \vee l_j))$ to CNF? Use definition of $\leftrightarrow$ and logical identities.

$$\begin{aligned}
(y \leftrightarrow (l_i \vee l_j)) &\iff (y \rightarrow (l_i \vee l_j)) \wedge ((l_i \vee l_j) \rightarrow y) \\
&\iff (\neg y \vee l_i \vee l_j) \wedge (\neg(l_i \vee l_j) \vee y) \\
&\iff (\neg y \vee l_i \vee l_j) \wedge ((\neg l_i \wedge \neg l_j) \vee y) \\
&\iff (\neg y \vee l_i \vee l_j) \wedge (\neg l_i \vee y) \wedge (\neg l_j \vee y)
\end{aligned}$$

□

Example 1. $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4)$. The second clause is replaced with $(\neg x_1 \vee \neg x_2 \vee y)$ and added to the formula are $(\neg y \vee x_3 \vee \neg x_4) \wedge (\neg x_3 \vee y) \wedge (x_4 \vee y)$.

Notice that what we do here is bringing a formula in a different (simpler?) form by adding variables. But when we trying to determine if a formula is a tautology/contradiction, then we want to remove variables rather than adding them: each added variable doubles the size of the truth table. So we want to do the opposite: try to simplify the formula by getting rid of some variables.

Definition 1 (Resolution rule)**.** *: Given two clauses of the form $C \vee x$) and $(D \vee \neg x)$, where C and D are (possibly empty) disjunction of variables, can derive a (possibly empty) clause $(C \vee D)$.*

That is,

$$(C \vee x) \wedge (D \vee \neg x) \rightarrow (C \vee D)$$

where $C = (l_1 \vee \cdots \vee l_k)$ and $D = (l'_1 \ldots l'_{k'})$ for some literals. If there is a repeated literal, only write it once.

Note that you may end up deriving an "empty" clause, that is, a $\vee$ of zero literals. But the only way this could happen is when two clauses being resolved are (x) and $(\neg x)$. But $x \wedge \neg x$ is always false, a contradiction. So deriving an empty clause proves that the original formula was a contradiction (which is what we usually want to show).

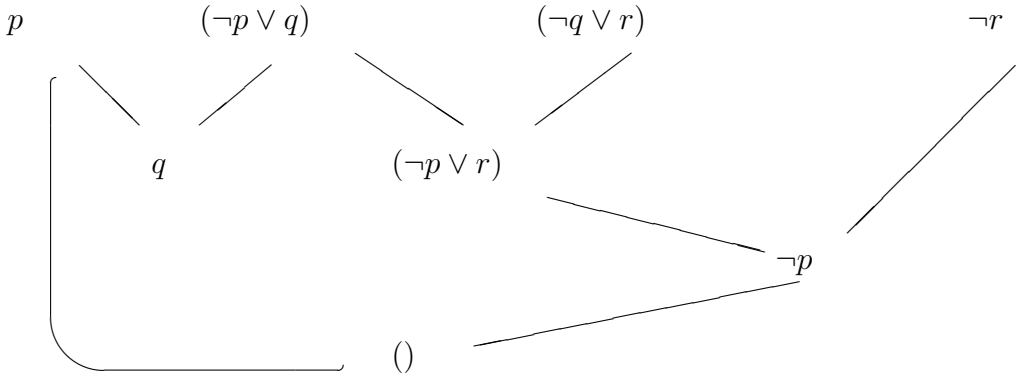
Example 2. Consider the following statements:
p: it is sunny
q: the weather is good
r: I spend the day outside

Now consider the following argument:

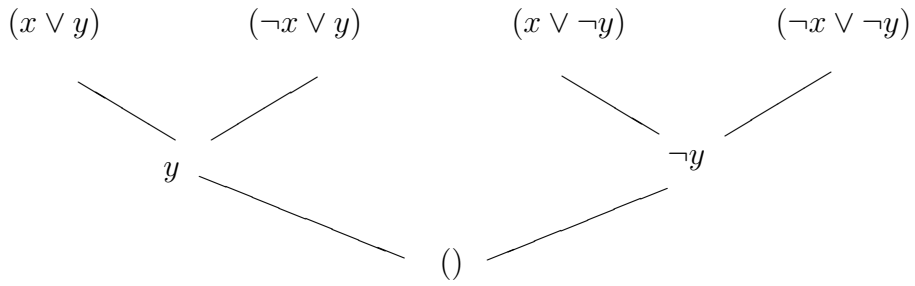
p
$p \rightarrow q$
$q \rightarrow r$
$\therefore r$

We want to prove that this is a valid argument, that is, p, $p \rightarrow q$ and $q \rightarrow r$ together imply r (this in general is called a *transitivity* law). Resolution proof method allows us to find contradictions: so we represent this problem as a contradiction $p \wedge (p \rightarrow q) \wedge (q \rightarrow r) \wedge \neg r$. Note that here again we are using the fact that the negation of an implication (in this case, premises imply the conclusion) is a conjunction of premises and negation of the conclusion.

Resolution works with CNF formula, so the first step is to convert the formula above into CNF. In this case, it is very easy: just apply the definition of implication to the second and third clause to obtain $p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$.



Example 3. Let's look at another example. Now there are only two variables, and the four clauses contain all possible combinations (so the formula is a contradiction) $(x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$. You can easily see that any truth assignment to x and y falsifies one clause.



Resolution proof system is very powerful in that given any formula in CNF form it can check whether it is a contradiction. However, resolution is not very efficient. Although it is much more efficient in practice than using truth tables (for example, the $(x_1 \wedge (x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \cdots \wedge (x_{n_1} \to x_n) \wedge \neg x_n$ can be done by a resolution proof system in n steps, whereas the truth table would have 2^n lines), there are examples of problems on which resolution proof system has to explore all possibilities, and so is similar to the truth table method in efficiency. The most prominent such example is the PigeonHole Principle, which says that it is impossible to put $n + 1$ element (pigeon) into n places (holes) without two elements getting into the same place. You needed to use this principle, this kind of counting reasoning to solve the last lecture puzzle.

The resolution is still the most popular method of programming automated proof systems because if its clarity and simplicity. However, when working with or programming such a system keep in mind that resolution "can't count".

Puzzle 7. Somebody asked a logician: "is it true that if you love Beth, then you also love Jane?". He said: "If this is so, then I love Beth". Does he love Jane as well?