Lógica en la Informática / Logic in Computer Science

Tuesday November 22nd, 2016

Time: 1h45min. No books, lecture notes or formula sheets allowed.

1) Let S be a set of propositional clauses over a set of n predicate symbols, and let $Res(S)$ be its closure under resolution. For each one of the following cases, indicate whether $Res(S)$ is infinite or finite, and, if finite, of which size. Very briefly explain why.

1a) If clauses in S have at most two literals.

Answer: [Remember: *Resolution* is a deduction rule where from two clauses of the form $p \vee C$ and $\neg p \vee D$ (the *premises*), the new clause $C \vee D$ (the *conclusion*) is obtained. Here p is a predicate symbol, and C and D are (possibly empty) clauses. The *closure under resolution* $Res(S)$ contains all clauses that can be obtained from S by zero or more resolution steps; formally, it is the union, for i in $0..\infty$, of all S_i where $S_0 = S$ and $S_{i+1} = S_i \cup Res_1(S_i)$, where $Res_1(S_i)$ is the set of clauses that can be obtained by one step of resolution with premises in S_i.]

Note that if clauses in S have at most two literals, clauses in $Res(S)$ too. Each clause is a set (i.e., no repetitions) of literals. If there are n predicate symbols, there are $2n$ literals. There are $\binom{2n}{2}$ clauses (i.e., subsets) of two literals, plus $2n$ clauses of 1 literal, plus the empty clause, which altogether is $O(n^2)$. Therefore, this is the maximal size of $Res(S)$ in this case.

1b) S is a set of Horn clauses.

Answer: Note that if clauses in S are Horn (i.e., they have at most one positive literal), clauses in $Res(S)$ too. How many Horn clauses over n predicate symbols exist? There are n negative literals, so there are 2^n clauses with only negative literals. For each one these, adding to it one positive literal, gives us the remaining Horn clauses. Total: $2^n + n \cdot 2^n$, so this is the maximal number of Horn clauses that exist and hence the maximal size of $Res(S)$ in this case.

1c) Every clause in S has either two literals or is a Horn clause.

Answer: Starting from Horn clauses and two-literal clauses in S does not help in bounding the size of $Res(S)$. In fact, from certain S of this form, by resolution one can obtain *any* clause: for any clause with a negative literal $\neg p$, such as $\neg p \vee C$, by one resolution step with a two-literal clause $p \vee q$ we get the clause $q \vee C$, and another step with $\neg q \vee p$ we get the clause $p \vee C$, i.e., we can change the sign of any literal of any clause.

Still, for a given set of n predicate symbols, $Res(S)$ is always finite, because each clause is a subset of the set of $2n$ literals, so there are at most 2^{2n} clauses in $Res(S)$.

1d) S is an arbitrary set of propositional clauses.

Answer: Each clause is a subset of the set of $2n$ literals, so there are at most 2^{2n} clauses in $Res(S)$. In fact S can already have 2^{2n} clauses.

2) Let C_1 and C_2 be propositional clauses, and let D be the conclusion by resolution of C_1 and C_2.

2a) Is D a logical consequence of $C_1 \vee C_2$? Prove it formally, using only the definitions of propositional logic.

Answer: If D is the conclusion by resolution of C_1 and C_2, then C_1 and C_2 are of the form $p \vee C_1'$ and $\neg p \vee C_2'$, for some predicate symbol p and clauses C_1' and C_2', and D is $C_1' \vee C_2'$.

In general, it is not true that $p \vee C_1' \vee \neg p \vee C_2' \models C_1' \vee C_2'$. Note that $p \vee C_1' \vee \neg p \vee C_2'$ is a tautology. So $C_1 \vee C_2 \neg \models D$ whenever D is not a tautology. Proof by counter example: take $C_1' = C_2' = q$ for some symbol q and any interpretation I with $I(q) = 0$. Then $I \models p \vee q \vee \neg p \vee q$ but $I \not\models q$, which completes our proof.

2b) Is D a logical consequence of $C_1 \wedge C_2$? Prove it formally, using only the definitions of propositional logic.

Answer: It true that $(p \vee C_1') \wedge (\neg p \vee C_2') \models C_1' \vee C_2'$. By definition of logical consequence, we have to prove that for all I, if $I \models (p \vee C_1') \wedge (\neg p \vee C_2')$ then $I \models C_1' \vee C_2'$.

We prove it by case analysis. Take an arbitary I. Assume $I \models (p \vee C_1') \wedge (\neg p \vee C_2')$.

Case A): $I(p) = 1$.

$I \models (p \vee C_1') \wedge (\neg p \vee C_2')$ implies,	by definition of satisfaction, that
$eval_I((p \vee C_1') \wedge (\neg p \vee C_2')) = 1$ which implies,	by definition of evaluation of $\wedge$, that
$min(eval_I(p \vee C_1'), eval_I(\neg p \vee C_2')) = 1$ which implies,	by definition of min, that
$eval_I(\neg p \vee C_2') = 1$ which implies,	by definition of evaluation of $\vee$, that
$max(eval_I(\neg p), eval_I(C_2')) = 1$ which implies,	by definition of evaluation of $\neg$, that
$max(1 - eval_I(p), eval_I(C_2')) = 1$ which implies,	by definition of $eval_I(p)$, that
$max(1 - I(p), eval_I(C_2')) = 1$ which implies,	since $I(p) = 1$, that
$max(0, eval_I(C_2')) = 1$ which implies	
$eval_I(C_2') = 1$ which implies	
$max(eval_I(C_1'), eval_I(C_2')) = 1$ which implies,	by definition of evaluation of $\vee$, that
$eval_I(C_1' \vee C_2') = 1$ which implies,	by definition of satisfiction, that
$I \models C_1' \vee C_2'$.	

Case B): $I(p) = 0$.

The proof is analogous to Case A, with the difference that now from $min(eval_I(p \vee C_1'), eval_I(\neg p \vee C_2')) = 1$ we obtain $eval_I(p \vee C_1') = 1$ and hence (since $I(p) = 0$) $eval_I(C_1') = 1$ which implies $eval_I(C_1' \vee C_2') = 1$ and hence $I \models C_1' \vee C_2'$.

2c) Let S be a set of propositional clauses and let $Res(S)$ be its closure under resolution. Is it true that S is satisfiable if, and only if, the empty clause is not in $Res(S)$? Very briefly explain why.

Answer: Yes.

$\Longrightarrow$: S satisfiable implies that S has a model I, and, since $S \models Res(S)$ (see below), this implies $I \models Res(S)$, which implies, since the empty clause is unsatisfiable, that the empty clause is not in $Res(S)$. We have $S \models Res(S)$ since $Res(S)$ is obtained from S by finitely many times adding to the set a new clause that (as we have seen in 2b) is a logical consequence of two clauses we already have.

$\Longleftarrow$: See the solution of lecture notes 3, excercise 20: if the empty clause is not in $Res(S)$ then we can build a model I for $Res(S)$ (and hence for its subset S), which proves that S is satisfiable. This is done by induction on the number N of symbols in $Res(S)$. The base case $N = 0$ is obvious. For the induction step, case $N > 0$, pick a symbol p that appears in $Res(S)$. Consider the set S' of all clauses of $Res(S)$ without p. Then by the induction hypothesis S' has a model I' (because S' has one symbol less, it is closed under resolution and it has no empty clause). Now we can extend I' to a model I for $Res(S)$ by setting $I(p)$ adequately. If $I' \models C$ for all $p \vee C$ in $Res(S)$ we set $I(p) = 0$ and we are done because then I becomes a model of all clauses with p in $Res(S)$. Otherwise, we have $I' \not\models C$ for some $p \vee C$ in $Res(S)$. Then we must set $I(p) = 1$ to satisfy this clause, and now we need to prove that then I is still a model of all clauses of the form $\neg p \vee D$ in $Res(S)$; this is true because if $\neg p \vee D$ and $p \vee C$ are in $Res(S)$, then also $C \vee D$ is in $Res(S)$ and hence in S' (because $C \vee D$ has no p). But then $I' \models C \vee D$ and $I' \not\models C$ implies $I' \models D$ and hence $I \models D$ and $I \models \neg p \vee D$.

3) Consider the well-known NP-complete *vertex cover* problem: Given a natural number k and a graph with n vertices and m edges $\{(u_1, v_1), \ldots, (u_m, v_m)\}$ with $u_i, v_i \in \{1 \ldots n\}$, it asks whether the graph has a k-cover, that is, a subset of size k of the vertices such that for each edge (u_i, v_i) at least one of u_i and v_i is in the cover.

3a) How would you use a SAT solver to decide it?

Answer:
Variables:

x_j, meaning "vertex j is in the k-cover" for $j \in \{1 \ldots n\}$ (n variables).

Clauses/constraints:

-there is one two-literal clause $x_u \vee x_v$ for each edge (u, v)

-one cardinality constraint $x_1 + \ldots + x_n \leq k$, expressing that the cover has at most k vertices.

The cardinality constraints can be encoded into clauses with, for example, a sorting network. We can use any SAT solver: the set cover problem has a solution iff this set of clauses has a model.

3b) How would you use a SAT solver for the optimization version of vertex cover, that is, given only the graph, to find its smallest possible k-cover?

Answer: Note that for $k = n$ we are sure there is a solution. Run the solver on the set of clauses with initial $k = n - 1$. If it finds a model with M ones (M elements in the cover), then run again, replacing the cardinality constraint by $x_1 + \ldots + x_n < M$. Repeat this, (each time finding models with smaller cover), until the solver returns unsat. The last model found is the optimal one.

Another algorithm is to make calls to the solver with $k = 0$, $k = 1$, $k = 2$,... and then the first model found is optimal. Yet another algorithm is to do a binary search. But the first algorithm given here works very well, because A) it is usually easier to find a model than to prove unsatisfiability and B) the M frequently decreases in large jumps.

3c) In one exam last year, we considered the following decision problem, called *minOnes*: given a natural number k and a set S of clauses over variables $\{x_1, \ldots, x_n\}$, it asks whether S has any model I with at most k ones, that is, with $I(x_1) + \ldots + I(x_n) \leq k$. Its optimization version is, given only S, to find its model with the minimal number of ones. If the set of clauses S only has clauses with at most two literals, does the optimization version of minOnes become polynomial? Explain briefly why.

Answer: NO! (unless P=NP). It is the same as the NP-complete vertex cover problem (see 3a)!! If it were polynomial, vertex cover would also become polynomial: take S consisting of only the (2-literal!) clauses for the edges, find the model of S with the minimal number of ones, and check whether this minimal number is larger than k or not.