# Lógica en la Informática / Logic in Computer Science

**Wednesday November 11, 2015**

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

**1a)** Let $F, G, H$ be formulas. Is it true that if $F \lor G \models H$ then $F \land \neg H$ is unsatisfiable? Prove it using only the definition of propositional logic.
**Answer:** This is true. $F \lor G \models H$ implies (by def. of logical consequence) that
for all $I$, if $I \models F \lor G$ then $I \models H$, which implies (by def. of $\models$) that
for all $I$, if $eval_I(F \lor G) = 1$ then $eval_I(H) = 1$, which implies (by def of $eval_I(\lor)$ ) that
for all $I$, if $max(eval_I(F), eval_I(G)) = 1$ then $eval_I(H) = 1$, which implies (by def of max) that
for all $I$, if $eval_I(F) = 1$ then $eval_I(H) = 1$, which implies (by arithmetic) that
for all $I$, if $eval_I(F) = 1$ then $1 - eval_I(H) = 0$, which implies (by def $eval_I(\neg)$ ) that
for all $I$, if $eval_I(F) = 1$ then $eval_I(\neg H) = 0$, which implies (by def. of min) that
for all $I$, $min(eval_I(F), eval_I(\neg H)) = 0$, which implies (by def $eval_I(\land)$) that
for all $I$, $eval_I(F \land \neg H) = 0$, which implies (by def of unsatisfiable) that
then $F \land \neg H$ is unsatisfiable.

**1b:** Let $F, G$ be formulas. Is it true that always $F \models G$ or $F \models \neg G$? Prove it using only the definition of propositional logic.
**Answer:** This is false. Counterexample: Assume $F = p$ and $G = q$ for two distinct symbols $p$ and $q$. Then $F \not\models G$, since for the interpretation $I$ where $I(p) = 1$ and $I(q) = 0$, we have $I \models F$ but $I \not\models G$. And also $F \not\models \neg G$, since for the interpretation $I'$ where $I'(p) = 1$ and $I'(q) = 1$, we have $I' \models F$ but $I' \not\models \neg G$.

**2)** Consider the following decision problem, called "minOnes":
Input:
a natural number $k$ and a propositional formula $F$ in CNF over propositional variables $\{x_1 \ldots, x_n\}$
Question:
Is there any model $I$ of $F$ with at most $k$ ones, i.e., where $|\ \{x_i \mid 1 \leq i \leq n \text{ and } I(x_i) = 1\}\ | \leq k$?

**2a)** Do you think that minOnes is NP-complete? Why?
**Answer:** Yes. It is NP-complete. MinOnes is not easier than general SAT, since SAT is the particular case of minOnes where $k = n$. On the other hand, minOnes is also not harder; it is still in NP because one can verify a given solution, an interpretation $I$, in linear time (check whether indeed $I$ is a model of $F$ with at most $k$ ones).

**2b)** How would you use a SAT solver to decide it?
**Answer:** Let $S$ be the set of clauses obtained by encoding the cardinality constraint $x_1 + \ldots + x_n \leq k$. Then, running the solver with input clauses $F \cup S$ will return the desired model with at most $k$ ones iff there exists one.

**2c)** How would you use a SAT solver to solve the optimization version of minOnes, that is, given $F$, to find its model with the smallest possible number of ones?
**Answer:** Run the solver on the input $F$.
A) If it returns unsat, the problem has no solution.
B) If it finds a model with $m$ ones, run again with input $F \cup S$, where $S$ is the set of clauses obtained by encoding the cardinality constraint $x_1 + \ldots + x_n < m$.
Repeat step B (finding each time models with less ones), until the solver returns unsat. The last model found is the optimal one.

Another algorithm is to make calls to the solver with $m = 0$, $m = 1$, $m = 2$,... 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.

**3)** The very large catalan supermarket CATSUP is open every day during 10 hours (from 10am to 8pm). It wants to schedule the working times of its $N$ employees during a 30-day period.

For each hour $h$ in this 300-hour period, CATSUP has made a prediction of the number $N_h$ of employees needed (at least) during hour $h$.

Each employee $i$ in $1..N$ has to work exactly 160 hours in this 30-day period, always at least 9 hours per working day, and no employee gets more than 5 consecutive working days in a row.

Explain in detail how to use a SAT solver for deciding exactly when each employee has to work. Clearly indicate which types of propositional variables you use and their precise meaning, and which properties you impose using which clauses or which constraints. For cardinality or pseudo-Boolean constraints, it is not necessary to give their encodings into clauses. Your solution should be as efficient and simple as possible.

**Answer:**
Variables:

$wh_{i,h}$ means: "worker $i$ works during hour $h$", for $1 \leq i \leq N$ and $1 \leq h \leq 300$
$wd_{i,d}$ means: "worker $i$ works on day $d$", for $1 \leq i \leq N$ and $1 \leq d \leq 30$

Clauses and constraints:

-Relationship between the variables $wh$ and $wd$ for each worker $i$:

For each day $d$ with hours $h1...h10$, we need to express $wd_{i,d} \equiv wh_{i,h1} \vee \ldots \vee wh_{i,h10}$.
Such an OR (as in Tseitin), requires the clause $\neg wd_{i,d} \vee wh_{i,h1} \vee \ldots \vee wh_{i,h10}$ (not needed here) and the ten implications: $wh_{i,h1} \rightarrow wd_{i,d}, \ldots, wh_{i,h10} \rightarrow wd_{i,d}$, which we do need, and which in clausal form are ten clauses: $\neg wh_{i,h1} \vee wd_{i,d}, \ldots, \neg wh_{i,h10} \vee wd_{i,d}$.
For example, to express that if worker 7 works on hour 38 then (s)he works on day 3: $\neg wh_{7,38} \vee wd_{7,3}$.

-Enough people work at each hour:

One cardinality constraint $wh_{1,h} + \ldots + wh_{N,h} \geq N_h$ for each hour $h$ with $1 \leq h \leq 300$.

-Each employee $i$ in $1..N$ has to work exactly 160 hours in this period:

One cardinality constraint $wh_{i,1} + \ldots + wh_{i,N} = 160$ for each worker $i$ with $1 \leq i \leq N$.

-No worker works more than 5 days in a row, that is, no one works on six consecutive days:

For each worker $i$ and day $d$ with $1 \leq d \leq 25$, one clause $\neg wd_{i,d} \vee \neg wd_{i,d+1} \vee \ldots \vee \neg wd_{i,d+5}$.

-If a worker $i$ works on a given day $d$, then (s)h works at least 9h that day:

We forbid all cases where $i$ works on $d$ and, on two different hours of $d$, $i$ does not work:
for each day $d$ and each pair $h, h'$ of two different hours of $d$, one clause $\neg wd_{i,d} \vee wd_{i,h} \vee wd_{i,h'}$.
For example, for worker 7 and day 3, one of the $\binom{10}{2} = 45$ clauses is $\neg wd_{7,3} \vee wh_{7,32} \vee wh_{7,38}$.

Another way of doing this: for each $i$ and $d$, one pseudo-Boolean constraint of the form:
$wh_{i,h1} + \ldots + wh_{i,h10} - 9\ wd_{i,d} \geq 0$
where $h1...h10$ are the hours of day $d$; this expresses that if $wd_{i,d} = 1$ then $wh_{i,h1} + \ldots + wh_{i,h10}$ has to be at least 9 (indeed, if $wd_{i,d} = 0$ the constraint is true independently of the $wh_{i,h}$ variables).