Lógica en la Informática / Logic in Computer Science

January 23, 2012. Results published: Wed Jan 25. Review: Thu Jan 26, 10h, Omega 139.

Time: 2h30min. No books, lecture notes or formula sheets allowed.

SOLUTIONS

1) Let us remember the *Heule-3 encoding* for *at-most-one* (*amo*), that is, for expressing in CNF that at most one of $x_1 \dots x_n$ is true. It uses the fact that $amo(x_1 \dots x_n)$ iff $amo(x_1, x_2, x_3, aux)$ AND $amo(\neg aux, x_4 \dots x_n)$. Then the part $amo(\neg aux, x_4 \dots x_n)$, which has $n-2$ variables, can be encoded recursively in the same way, and $amo(x_1, x_2, x_3, aux)$ can be expressed using the quadratic encoding with 6 clauses. In this way, for eliminating two variables we need one auxiliary variable end six clauses, so in total we need $n/2$ variables and $3n$ clauses.

Also remember: an encoding for *amo* is *arc-consistent for unit propagation* if, when one of $x_1 \dots x_n$ becomes true, the SAT solver's unit propagation mechanism will set the other variables to false.

1a Is this Heule-3 encoding for *amo* arc-consistent for unit propagation? Prove it.

Answer: Yes, it is. We can prove it, for example, by induction on n.

Base case: if $n \leq 4$ the quadratic encoding part is the whole constraint. For example, for $n = 4$ we have $\neg x_1 \vee \neg x_2$, $\neg x_1 \vee \neg x_3$, $\neg x_1 \vee \neg x_4$, $\neg x_2 \vee \neg x_3$, $\neg x_2 \vee \neg x_4$, and $\neg x_3 \vee \neg x_4$. For every distinct pair $i, j \subset \{1 \dots 4\}$ we have a clause $\neg x_i \vee \neg x_j$, so if x_i becomes true, all other variables x_j become false by unit propagation.

Induction case: If $n > 4$ the part $amo(x_1, x_2, x_3, aux)$ is expressed using the quadratic encoding with 6 clauses: $\neg x_1 \vee \neg x_2$, $\neg x_1 \vee \neg x_3$, $\neg x_1 \vee \neg aux$, $\neg x_2 \vee \neg x_3$, $\neg x_2 \vee \neg aux$, and $\neg x_3 \vee \neg aux$. Now there are two cases: A) some variable of $\{x_1, x_2, x_3\}$ becomes true, or B) some variable of $x_4 \dots x_n$ becomes true. In case A, as before unit propagation will set the other variables in $\{x_1, x_2, x_3, aux\}$ to false, and hence $\neg aux$ becomes true, and then we can apply the induction hypothesis since $amo(\neg aux, x_4 \dots x_n)$ has two variables less: unit propagation will set all variables in $\{x_4 \dots x_n\}$ to false.

In case B, by induction hypothesis, since $amo(\neg aux, x_4 \dots x_n)$ has two variables less, unit propagation will set all other variables in $\{x_4 \dots x_n, \neg aux\}$ to false and hence aux becomes true, and by the clauses $\neg x_1 \vee \neg aux$, $\neg x_2 \vee \neg aux$, $\neg x_3 \vee \neg aux$, unit propagation will set x_1, x_2, and x_3 to false.

1b We now want to extend the encoding for *at-most-two* (*amt*) constraints. Prove that $amt(x_1 \dots x_n)$ has a model I iff $amt(x_1, x_2, x_3, aux_1, aux_2) \wedge amt(\neg aux_1, \neg aux_2, x_4 \dots x_n)$ has a model I'.

Answer: If $amt(x_1 \dots x_n)$ has a model I, then we can extend it to construct a model I' of $amt(x_1, x_2, x_3, aux_1, aux_2) \wedge amt(\neg aux_1, \neg aux_2, x_4 \dots x_n)$: if no variable of $\{x_1, x_2, x_3\}$ is true in I, then we make aux_1 and aux_2 true in I'; if one variable of $\{x_1, x_2, x_3\}$ is true in I, then we make (for example) aux_1 true and aux_2 false in I'; if two variables of $\{x_1, x_2, x_3\}$ are true in I, then we make aux_1 and aux_2 false in I'. In all three cases I' is a model of $amt(x_1, x_2, x_3, aux_1, aux_2) \wedge amt(\neg aux_1, \neg aux_2, x_4 \dots x_n)$.

Reversely, if $amt(x_1, x_2, x_3, aux_1, aux_2) \wedge amt(\neg aux_1, \neg aux_2, x_4 \dots x_n)$ has a model I', then ("forgetting" the part of the auxiliary variables), I' is also a model of $amt(x_1 \dots x_n)$: if aux_1 and aux_2 are true in I', then no variable of $\{x_1, x_2, x_3\}$ is true in I and at most two of $\{x_4 \dots x_n\}$ are true in I'; if aux_1 is true and aux_2 is false in I' (or vice versa) then at most one variable of $\{x_1, x_2, x_3\}$ is true in I and at most one of $\{x_4 \dots x_n\}$ is true in I'; if aux_1 and aux_2 are false in I' then at most two variables of $\{x_1, x_2, x_3\}$ are true in I and no variable of $\{x_4 \dots x_n\}$ is true in I'.

1c Explain how to encode the part $amt(x_1, x_2, x_3, aux_1, aux_2)$ with no more auxiliary variables.

Answer: Using the following clauses for all subsets of 3 elements out of 5, that is $\binom{5}{3} = 10$ clauses:
$\neg x_1 \vee \neg x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_2 \vee \neg aux_1$, $\neg x_1 \vee \neg x_2 \vee \neg aux_2$, $\neg x_1 \vee \neg x_3 \vee \neg aux_1$, $\neg x_1 \vee \neg x_3 \vee \neg aux_2$, $\neg x_1 \vee \neg aux_1 \vee \neg aux_2$, $\neg x_2 \vee \neg x_3 \vee \neg aux_1$, $\neg x_2 \vee \neg x_3 \vee \neg aux_2$, $\neg x_2 \vee \neg aux_1 \vee \neg aux_2$, and $\neg x_3 \vee \neg xaux_1 \vee \neg aux_2$,

1d How many clauses and auxiliary variables are needed in total for $amt(x_1 \dots x_n)$ in this way?

Answer: The part $amt(\neg aux_1, \neg aux_2, x_4 \ldots x_n)$ has one variable less. So to eliminate one variables, we need 10 clauses and 2 auxiliary variables. So in total we will need $10n$ clause and $2n$ auxiliary variables.

1e Is this *amt* encoding arc-consistent for unit propagation? (That is, if two of $x_1 \ldots x_n$ become true, will unit propagation set the other variables to false?) Prove it.

Answer: No. It is not arc-consistent for unit propagation. For example, if x_1 and x_4 become true, no unit propagation takes place at all. Note that none of the 10 clauses becomes unit.

2) Facebook Catalunya has all the information about its N registered Catalan users and for each user, the list of her friends. Now they want to find a subset of 200 Catalan users that are all friends of each other (every two users in the subset are friends). Explain in detail how they can do this using the Barcelogic SAT Solver.

Answer: Define a SAT encoding with variables x_i, for all i in $1..N$, meaning that "user i is in the subset". Then, for each pair of users (i, i') with $1 \leq i < i' \leq N$ such that i and i' are not friends, we add clause epressing that at least one of them is not in the subset: $\neg x_i \vee \neg x_{i'}$. Finally we need to express that exactly 200 of the N variables x_i are true. This we can do with a cardinality constraint (for example, encoded using sorting networks). The resulting CNF is given to the Barcelogic SAT solver. If it finds a model, from this model we can easily reconstruct the solution for Facebook. If it returns "unsatisfiable", then no solution for Facebook exists.

Another encoding is to have $200N$ variables $x_{i,j}$ with i in $1..N$, and j in 1.,200, and meaning "user i is the j-th member of the subset". Then we need clauses, for each j in 1..200, saying that exactly one of $\{x_{1,j} \ldots x_{N,j}\}$ is true, and we need clauses, for each user i, saying that at most one of $\{x_{i,1} \ldots x_{i,200}\}$ is true. And, as before, for each pair of users (i, i') with $1 \leq i < i' \leq N$ such that i and i' are not friends, we add all clause epressing that at least one of them is not in the subset: for all j, j' in 1..200, all clauses of the form $\neg x_{i,j} \vee \neg x_{i',j'}$.

3) Now we want to solve problem 2) in Prolog. Assume there are 500,000 users and 500,000 clauses like: `friends(3454,[3,7,11,23,37854]).` meaning that (all) the friends of user `3454` are `3,7,11,23` and `37854`.

3a) Define a predicate `list200(L)` that can generate in L under backtracking all lists of 200 different users (200 numbers in $1 \ldots 500{,}000$).

```
subset(0,_,[]):-!.
subset(N,[X|L],[X|S]):- N1 is N-1, subset(N,L,S).
subset(N,[X|L],   S ):-            subset(N,L,S).

listNumbers(0,[]):-!.
listNumbers(N,[N|L]):- N1 is N-1, listNumbers(N1,L).

list200(L):- listNumbers(500000,LNums), subset(200,LNums,L).
```

3b) Define a predicate `friendsforever` that writes a list of 200 friends of each other, if it exists.

```
friendsforever:- list200(L), allfriends(L,L), write(L), nl.

allfriends([],_).
allfriends([X|Rest],L):- friends(X,FrX), isSubset(L,FrX), allfriends(Rest,L).

isSubset([],_).
isSubset([X|Rest],L):- member(X,L), isSubset(Rest,L).
```

3c) This predicate is called `friendsforever` because it may run for a long time (almost forever). Modify your program to make it faster.

Answer: As always, the idea is to not to have a pure "generate and test" in the form of `list200(L)` (generate) and `allfriends(L,L)` (test), but to interleave them (entrelazarlos) instead. One possibility is:

```
friendsforever:- friends(X,FrX),  ff([X],[X|FrX],L),  write(L), nl.

%the 1st argument L  of ff is input, the list built so far.
%The 2nd one      I  is the intersection of all friends of the members of L
%The 3rd one      L1 is the output list.
ff(L,_,L):-  length(L,200),!.
ff(L,I,L1):- member(X,I), \+member(X,L), friends(X,FrX), isSubset(L,Frx),
             intersection(I,[X|Frx],I1), length(I1,K), K>=200, ff([X|L],I1,L1).

intersection([],_,[]).
intersection([X|L],L1,[X|I]):- member(X,L1), !, intersection(L,L1,I).
intersection([_|L],L1,   I ):-                  intersection(L,L1,I).
```

4) Formalize and prove by resolution that sentence D is a logical consequence of the other three:
A: All politicians sometime lie.
B: Friends of football players never lie.
C: Messi is a football player.
D: Messi has no friends that are politicians.

Answer: We prove that $A \wedge B \wedge C \wedge \neg D$ is unsatisfiable.

$A:\quad \forall x\ Pol(x) \rightarrow Lies(x)$
$B:\quad \forall x\ (\exists y\ Friends(x,y) \wedge Player(y)) \rightarrow \neg Lies(x)$
$C:\quad Player(messi)$
$\neg D:\quad \exists x\ Friends(x, messi) \wedge Pol(x)$

In clausal form:
$A:\quad \neg Pol(x) \vee Lies(x)$
$B:\quad \neg Friends(x,y) \vee \neg Player(y)) \vee \neg Lies(x)$
$C:\quad Player(messi)$
$\neg D_1:\quad Friends(c_x, messi)$
$\neg D_2:\quad Pol(c_x)$

Resolution:
$6:\quad Lies(c_x)$ (from $\neg D_2$ and A)
$7:\quad \neg Friends(c_x,y) \vee \neg Player(y))$ (from B and 6)
$8:\quad \neg Friends(c_x, messi)$ (from C and 7)
$9:\quad \square$ (from $\neg D_1$ and 8).