Problem solving 9: Reduce operations

Gilles Audemard - Ibn Sina School

Problem solving in C – `http://ibnsina.alfweb.net/`

July 2015

Files to download: `09/...`

Exercise 1 (Look for literals occurences – `solving_09_01.c`)
It is quite common to ask for literal occurences of a SAT formula: one wants to get all clauses (their number in the formula) where a literal appears. For example, with the formula (`0, 1, 2` represent clauses index):

```
0 : 10 20
1 : -10 30 20
2 : 30 -20
```

Then,

- Occurrences of `10 : 0`
- Occurrences of `-10 : 1`
- Occurrences of `20 : 0, 1`
- Occurrences of `-20 : 2`
- Occurrences of `30 : 1, 2`
- Occurrences of `-30 :`

1. In order to do that, implement function (**and only this one**): `vecInt* getOccurrences(Formula f, Literal l)`
2. test it: `make tests0901`

Exercise 2 (Try on a large formula – `solving_09_02_main.c`)
Take the instance `korf-15.cnf` of the friday project.

1. Create a `main` function that loads this formula using the previous solving sessions and use the function `getOccurrences` in order to print the **number** of occurrences of all positive literals.
2. Look the cpu time needed to achieve this processus, by launching your main like that `/usr/bin/time solving_09_02_main`

Exercise 3
This seems too much. We are going to store all literal occurrences inside the formula and use more memory, but less CPU time.

Look at the end of the file `solving_09_01.h`.

1. First of all, change `FORTESTS` and put the value 1 (this is for activate tests!)
2. Modify the structure `Formula`. Add to it `Occurrences* literalOccurrences;`. Each literal has a vector of occurrences.
3. In `solving_07_02.c`:
   (a) Modify the function `createFormula` in order to allocate all memory space (look like the allocation of clauses)

(b) Modify the function `freeFormula`

(c) Modify the function `addLiteralInClause`: you know that this literal is in that clause. Put it in occurrences.

(d) You have made some changes in `solving_07_02.c`. Run related tests again in order to check if you have a regression in your code

(e) Modify `getOccurences`. You can return directly the vecInt!

(f) You have made some changes in `solving_09_01.c`. Run related tests again in order to check if you have a regression in your code

(g) try again `/usr/bin/time solving_09_02_main`

Exercise 4 (Modify status of a clause : `int statusClause(Clause c,Interpretation I)`)
It is common to ask the satisfiability of a formula on closed interpretations (with a hamming distance near or equal to 1). We can avoid to traverse the clause again in case of satisfiability by moving the first literal of the clause in the first position. Like that, if it does not change its value, we directly known the satisfiability of this clause.

1. Modify the code of `statusClause` to do that
2. Run again the related tests to avoid code regression