Seminar on Automated Reasoning

S.Jacobs, V.Kuncak, R.Piskac

Problem 1 Resolution, model construction

Let $\Sigma = (\Omega, \Pi)$ be a signature with $\Omega = \{a/_0, f/_1\}$ and $\Pi = \{p/_1\}$. Suppose that the atom ordering $\succ$ is defined in such a way that $p(f^n(a)) \succ p(f^m(a))$ if and only if $n > m \geq 0$. Let N be the following set of clauses:

$$p(f(f(a)))$$

$$\neg p(x) \vee p(f(x))$$

(1)

Sketch how the set $G_\Sigma(N)$ of all ground instances of clauses in N looks like. How is it ordered with respect to the clause ordering $\succ_C$?

(2)

Construct the candidate model $I_{G_\Sigma(N)}$ of the set of all ground instances of clauses in N.

Problem 2 Most General Unifier

Compute, if exists, a most general unifier of the following set of equalities:

$$E = \{f(x, g(x)) = y,\ h(y) = h(v),\ v = f(g(z), w)\}$$

Problem 3 Resolution

Use the resolution calculus to prove the validity of the following formula:

$$\forall x.\exists y.\Big(p(f(f(x)), y) \wedge \forall z.\big(p(f(x), z) \rightarrow p(x, g(x, z))\big)\Big) \rightarrow \forall x.\exists y.p(x, y)$$

Problem 4 Theorem Provers in Practice

The goal of this exercise is that you practice to write down your own specification and formalization of a problem. The focus is not so much on finding the proof but more on writing correct and complete specifications. However, finding the proof will definitely bring you additional points. It is enough if you solve one of the given two problems:

(1) Peano Arithmetic

In the first-order theorem prover of your choice (preferably SPASS) write down the axiomatization of Peano arithmetic. Using this axiomatization prove automatically at least two formulas from the homework on first-order logic, syntax and semantics, given in Problem 2 on Peano arithmetic.
If you do not manage to prove this problem using SPASS, try to use SPASS+T (`http://www.mpi-inf.mpg.de/~uwe/paper/TSPASS-bibl.html`)

(2) Lists

Consider the theory of lists: there are two constructors (functions): `nil` and $\texttt{cons}(L, E)$. `nil` denotes the empty list and $\texttt{cons}(L, E)$ is a list obtained by inserting element E at the beginning of list L. We consider only such inductively defined lists. This means that whenever you are proving properties about lists, you have to prove two theorems: the property hold for `nil` and if the property holds list L, then it also holds for $\texttt{cons}(L, E)$. Consider a similar inductive definition of natural numbers, use constant `z` and function $\texttt{s}(N)$ (`z` stands for 0 and $\texttt{s}(N)$ for the successor function).

- define function $\texttt{card}(L)$ which counts the number of elements in list L
- define predicate $\texttt{isElem}(L, E)$ describing that element E belongs to list L
- describe function $\texttt{del}(L, E)$ which deletes the *first* occurrence of element E from list L. If the element is not in the list, the list should remain unchanged, you do not need raise an exception. Your function needs to be complete and not only partially defined!
- prove the following theorem $\texttt{isElem}(L, E) \rightarrow \texttt{s}(\texttt{card}(\texttt{del}(L, E))) = \texttt{card}(L)$

In both cases, submit your input file and the theorem prover output. Make sure that the input file parses and that we can verify your output. If you use some other theorem prover, you will need to demonstrate that it works on your laptop.