Exercises for Computability and Complexity, Spring 2016, Sheet 6

Please return your solutions in the Thursday lecture on April 21.

For this exercise sheet please prepare your solutions with a text processor and submit a printout!

Problem 1. Design a λ-expression **sortincreasing**, which applied to a list whose entries are Church numerals returns a list of the same length with the same entries, but sorted in ascending order. You may assume that you already have combinators $<, >, \leq, \geq$, which when applied to two Church numerals reduce to **true** or **false** in the obvious fasion. Also you may use all the combinators for Boolean logic, list processing and arithmetics introduced in the lecture notes. Example of what your λ-expression should do: **sortincreasing** (1::3::2::1::**nil**) →* (1::1::2::3::**nil**). I suggest to lean on bubblesort in your construction.

Notice. The next exercise is a L5 programming exercise, using the online L5 interpreter at http://www.csse.monash.edu.au/~lloyd/tildeFP/Lambda/Ch/ . This interpreter as a peculiarity that one should know about. If within a `let rec` environment some `funname = lambda x ...` is introduced, and later this `funname` is used, e.g. as in `funname a::b`, then `funname` binds stronger than later bindings, that is, `funname a::b` would be interpreted as `(funname a)::b` and not as `funname (a::b)`. Be generous with brackets!

Problem 2. Write an L5 program that implements your solution from the first homework problem. Supply a printout your progam and of a run of your program on (1::3::2::1::**nil**).

Problem 3. Show that there exists a decidable language over {0, 1} which is not in NP. Hint: one quick way to show this is by diagonalization.

Solution (one possibility – there are many ways to tackle this problem). Let (M_i, k_i) ($i = 1, 2, ...$) be an effective enumeration of all pairs of nondeterministic TMs M with tape alphabet {0, 1} and positive integers k. Let L_i be the language decided by M_i within time $|x|^{k_i}$. Notice that the set of all these languages is NP, and that every such L_i is decidable. Let $w_1, w_2, ...$ be the lexicographic enumeration of {0,1}*. Then the diagonal language $\{w_i \in \{0,1\}^* \mid w_i \notin L_i\}$ is decidable and different from all of the L_i, hence not in NP.

Optional advanced exercise: Write an L5 program that carries out differentiation of polynomials of the kind $(a_1x^{b_1} + ... + a_nx^{b_n})' \rightarrow a_1\, b_1\, x^{b_1-1} + ... + a_n\, b_nx^{b_n-1}$, where the a_i, b_i are integers (the a_i may be negative). A core part of this exercise is to first design a coding of such terms in a format suitable for L5 – I would suggest a list-based coding. Deliverables: a description of your coding in words, plus a printout of your L5 program and of the result on inputting a (coded version of) $2x^3 - 1 + x^3 - 3x$. The output should represent polynomial terms in ascending order w.r.t. to the exponent, and there should be no multiple instances of terms with the same exponent. Thus, on input $2x^3 - 1 + x^3 - 3x$, the output should be a coded version of $-3 + 9x^2$. *Remark:* the first symbolic math programs for automated algebraic transformations (such as differentiation) were all written in LISP. In general, functional programming languages are perfectly suited to tasks that have a symbolic-transformational, "calculus" flavour – not only mathematical algebra systems, but also, for instance, linguistic grammar processing, theorem proving, or knowledge representation in AI.