Written Examination, December 11th, 2002 Course no. 02140

The duration of the examination is 4 hours.

Course Name: Languages and parsing

Allowed aids: All written material

The problem set consists of 6 problems which are weighted as follows:
Problem 1: 15%, problem 2: 15%, problem 3: 20%, problem 4: 15%,
problem 5: 25%, problem 6: 10%

Marking: 13-scale.

Problem 1 (15%)

Let $L \stackrel{\text{def}}{=} \{a^n(bc)^n \in \{a,b,c\}^* \mid n \geq 0\}$.

1. Prove that L is not regular.
2. Is L context-free? Justify your answer.

Problem 2 (15%)

1. Give a construction which, given arbitrary DFA $A = (Q, \Sigma, \delta, q_0, F)$, yields a DFA accepting the language
$$L^A_{\text{even}} \stackrel{\text{def}}{=} \{w \in L(A) \mid |w| \text{ is even}\} \ .$$
2. Provide a construction which, given arbitrary DFA $A = (Q, \Sigma, \delta, q_0, F)$ and $n > 0$, yields a DFA accepting the language
$$L^A_{\text{nondiv}} \stackrel{\text{def}}{=} \{w \in L(A) \mid |w| \text{ is not divisible by } n\} \ .$$

Problem 3 (20%)

Show the following claims, where all the languages are over alphabet $\Sigma = \{a, b, c\}$:

1. the set L_1 of all words $w \in \Sigma^*$ with the property "w ends with *abba*" is regular;
2. the set L_2 of all words $w \in \Sigma^*$ with the property "if each occurrence of the symbol c in w is replaced by the string *aba* then the resulting word is a palindrome" is context-free;
3. the set $L_3 = L_1 \cap L_2$ of all words which have the above properties is context-free.

Problem 4 (15%)

Answer "true" or "false" to the following claims, and give a brief explanation to each answer. All strings considered are over some alphabet Σ containing at least two symbols.

1. The language $\{w \in \Sigma^* \mid |w| \text{ is non-prime}\}$ is regular.
2. If L is accepted by an NFA with n states then $\mathrm{pref}(L)$ is accepted by some DFA with 2^n states, where $\mathrm{pref}(L) \stackrel{\text{def}}{=} \{x \mid xy \in L \text{ for some } y \in \Sigma^*\}$.
3. The intersection of a context-free language with the complement of a context-free language is regular.
4. If L is the language of a 1024-state ε-NFA then L is the language of a 1-state PDA.
5. If L is a superset of a non-regular language then L is non-regular.

 ... Continued on next page

Problem 5 (25%)

We now consider a parsing problem concerning representations of companies.

A *company* consists of a list of departments, where a *department* is characterized by a name, which is a string, an income, which is an integer, and a (possibly empty) list of sub-departments.

The corresponding datatype declaration for the abstract syntax of companies is:

```
datatype company    = Comp of department list
and      department = Dept of string * int * department list
```

and as input specification to mosmlyac we use:

```
%{ open Absyn %}

%token <int> INCOME
%token <string> NAME
%token EOF

%start COMP
%type <Absyn.company> COMP
%type <Absyn.department> DEPT
%type <Absyn.department list> DEPTList
%%

COMP: DEPTList EOF              { Comp $1 }         ;

DEPT: NAME INCOME DEPTList      { Dept($1,$2,$3) } ;

DEPTList:
                                { []       }
  | DEPT DEPTList               { $1 :: $2 }
;
```

Unfortunately, the grammar is ambigious and shift/reduce conflicts are reported.

1. Find the ambiguity in the grammar.
2. Change/extend the specification so that the ambiguity is resolved. Give a brief explanation of your decisions. (You have to change the language defined by the grammar.)
3. Extend the specification and the abstract syntax so that a non-empty list of employees is associated with every department. An employee is described by a name and an income. Take care that your specification is free of conflict.
4. We want to enforce that an employee is associated with at most one department in a company. Give a brief explanation of how you would handle this.

 ... Continued on next page

Problem 6 (10%)

In this problem we want to consider *exclusive union* of regular languages:

$$L_1 \oplus L_2 = (L_1 \cap \overline{L_2}) \cup (L_2 \cap \overline{L_1}) \qquad (1)$$

I.e. the elements of $L_1 \oplus L_2$ are all strings of L_1 which are not in L_2 and all strings of L_2 which are not in L_1.

Here we represent regular languages by deterministic finite automata as follows:

```
type ''a transFct = ''a * string -> ''a         (* transition function *)
type ''a dfa   = ''a list * string list * ''a transFct * ''a  * ''a list;
```

This representation together with the declarations of functions for union, intersection and complement were studied in the semester. For example, a function for intersection was declared as follows, based on a cartesian product of the states of the individual automata:

```
infix intersection;

fun (is1, ss, d1, i01, ff1) intersection  (is2, _ , d2, i02, ff2) =
  let val is = cartesian(is1, is2)
      fun d((i1, i2), s) = (d1(i1, s), d2(i2,s))
      val i0 = (i01, i02)
      val ff = List.filter
                 (fn (i1,i2) => member(ff1,i1) andalso member(ff2,i2))
                 is
  in  (is, ss, d, i0, ff) end;
```

where `cartesian` is an auxiliary function for computing the cartesian product of two sets, represented by lists:

$$\texttt{cartesian}([x_1, x_2, \ldots, x_m], [y_1, y_2, \ldots, y_n])$$
$$= [(x_1, y_1), (x_1, y_2), \ldots, (x_1, y_n), (x_2, y_1), \ldots, (x_m, y_n)]$$

A function for exclusive union can be declared using the functions for union, intersection and complement as in the above definition (1):

```
infix exUnion;

fun a1 exUnion a2 = (a1 intersection (complement a2))
                    union (a2 intersection (complement a1));
```

In this case an exclusive union operation will perform several product automaton constructions. At least one for each intersection.

The operation can, however, be performed by a direct construction involving precisely one construction of a cartesian product.

1. Explain this direct construction.
2. Declare an SML function for exclusive union based on 1.