Name: Matriculation Number:

Final Exam
General CS 1 (320101)
December 20. 2006

You have two hours(sharp) for the test;
Write the solutions to the sheet.

The estimated time for solving this exam is 0 minutes, leaving you 120 minutes for revising your exam.

You can reach 0 points if you solve all problems. You will only need 52 points for a perfect score, i.e. -52 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here	
prob.	Sum	grade
total	0	
reached		

Good luck to all students who take this test

1 Mathematical Foundations

3pt

Problem 1.1 (Function Definition)
Let A and B be sets. State the definition of the concept of a partial function with domain A and codomain B. Also state the definition of a total function with domain A and codomain B.

Solution: Let A and B be sets, then a relation $R \subseteq AB$ is called a **partial/total function**, iff for each $a \in A$, there is at most/exactly one $b \in B$, such that $\langle a, b\rangle \in R$.

Problem 1.2 (Invariance of Equivalence Relations) 5pt

Let A be a set and $R, S \in A^2$ be equivalence relations. Prove or refute that $R \cup S$ is an equivalence relation too.

Solution: The claim is not valid: $R \cup S$ is reflexive, symmetric, but not transitive: if we have $\langle a, b\rangle \in R$ and $\langle b, c\rangle \in S$, then we do not know that $\langle a, c\rangle \in (R \cup S)$.

2 Standard ML

6pt

Problem 2.1 (SML Function for Coordinate Transformation)
Declare SML datatypes `cartesian` and `polar` representing points in two dimensional space in Cartesian and polar coordinates respectively.

Moreover define an SML function `cartesianToPolar : cartesian -> polar` which does the intended coordinate transformation:

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \sqrt{\sin(x)^2 + \cos y^2} \\ \arctan x/y \end{pmatrix}$$

It should raise an exception for $y = 0$.

Hint: You can assume `sqrt`, `square`, `sin`, `cos`, `arctan` as built-in functions of type `real -> real`.

Use SML syntax for the whole problem.

Solution:

```
datatype polar = polar of real * real;
datatype cartesian = cartesian of real * real;

exception Undefined;

cartesianToPolar (cartesian _ 0) = raise Undefined;
cartesianToPolar (cartesian x y) =
  polar (sqrt (square (sin x)) (square (cos x))) (arctan (x/y));
```

Problem 2.2 (Higher-Order Functions) 6pt

Write three higher-order functions that take a predicate p (a function with result type `bool`) and a list l.

- `myfilter` that returns the list of all members a of l where $p(a)$ evaluates to `true`.
- `myexists` that returns `true` if there is at least one element a in l, such that $p(a)$ evaluates to `true`.
- `myforall` that returns `true` if $p(a)$ evaluates to `true` on all elements of l.

Hint: If you are in need of a test predicate, you can work on a list l of `int`s and use the "even number" predicate:

```
fun even n = n mod 2 = 0;
```

Solution:

```
fun myfilter p nil = nil
  | myfilter p h::t =
    if (p h) then h :: myfilter p t else myfilter p t

fun myexists p nil = false
  | myexists p h::t =
    if (p h) then true else myexists p t

fun myforall p nil = true
  | myforall p h::t =
    if (p h) then myforall p t else false
```

For the last two, we can make use of the predefined functions `orelse` and `andalso`, which are useful abbreviations: e_1 `andalso` e_2 abbreviates `if` e_1 `then` e_2 `else false` and e_1 `orelse` e_2 abbreviates `if` e_1 `then true else` e_2. Using this, we can write

```
fun myforall f nil = true
  | myforall f (x::xr) = f x andalso myforall f xr

fun myexists f nil = false
  | myexists f (x::xr) = f x orelse myexists f xr
```

---

3 Abstract Data Types and Procedures

6pt

Problem 3.1 (Abstract Procedure for Addition and Multiplication)
Given the abstract data type of unary natural numbers: $\langle\{\mathbb{N}\}, \{[o\colon \mathbb{N}], [s\colon \mathbb{N} \to \mathbb{N}]\}\rangle$ Write two abstract procedures *plus* and *mult* on this abstract data type which compute the ordinary addition and multiplication respectively of unary numbers. Trace the evaluation of $mult(s(s(o)), s(s(o)))$.

Solution:

$$\mathcal{F} := \langle plus{::}\mathbb{N} \times \mathbb{N} \to \mathbb{N}\,;\ \{plus(o,n) \rightsquigarrow n, plus(s(m),n) \rightsquigarrow plus(m,s(n))\}\rangle$$
$$\mathcal{G} := \langle mult{::}\mathbb{N} \times \mathbb{N} \to \mathbb{N}\,;\ \{mult(o,n) \rightsquigarrow o, mult(s(m),n) \rightsquigarrow plus(mult(m,n),n)\}\rangle$$

Trace of evaluation:

$$mult(s(s(o)), s(s(o))) \rightsquigarrow plus(mult(s(o), s(s(o))), s(s(o))) \rightsquigarrow plus(s(s(o)), s(s(o)))$$
$$\rightsquigarrow plus(s(o), s(s(s(o)))) \rightsquigarrow plus(o, s(s(s(s(o))))) \rightsquigarrow s(s(s(s(o))))$$

---

4 Formal Languages and Codes

5pt

Problem 4.1 (Minimal Word Length)
Let $\mathcal{A}$ and $\mathcal{B}$ alphabets with $\#(\mathcal{A}) > \#(\mathcal{B})$. What is the minimal length of the longest codeword from $\mathcal{B}^+$ such that $c\colon \mathcal{A} \to \mathcal{B}^+$ is a character code?

Hint: Think of how you can use the properties of character codes and the cardinalities of your alphabets to figure out the answer. Also, it is always a good idea to test your final value for some simple cases. Consider for example $\#(\mathcal{A}) = 12$ and $\#(\mathcal{B}) = 2$ or $\#(\mathcal{A}) = 5$ and $\#(\mathcal{B}) = 3$ as initial values.

Solution: Old version:

Let c be a character code, i.e. an injective function $c\colon \#(\mathcal{A}) \to \mathcal{B}^+$. Then the cardinality of the image of c must be greater than that of its domain $\mathcal{A}$. In terms of word length n we have the constraint $\#(\mathcal{B}^n) \geq \#(\mathcal{A})$. This can be transformed to $n \geq \frac{\#(\mathcal{A})}{\#(\mathcal{B})}$.

New version:

The constraint used for c being a character code can be formulated as

$$\#(\mathcal{A}) \leq \#(\mathcal{B}) + \#(\mathcal{B})^2 + ... + \#(\mathcal{B})^{min}$$

We can now use the formula for geometric progressions and find the inequality:

$$\#(\mathcal{A}) \leq \frac{\#(\mathcal{B})^{min+1} - 1}{\#(\mathcal{B}) - 1} - 1$$

By taking to the common denominator we get

$$\#(\mathcal{A}) \leq \frac{\#(\mathcal{B})^{min+1} - \#(\mathcal{B})}{\#(\mathcal{B}) - 1}$$

From this point on standard mathematical operations are used to deduce the value of min.

$$\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) \leq \#(\mathcal{B})^{min+1} - \#(\mathcal{B})$$

$$\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) + \#(\mathcal{B}) \leq \#(\mathcal{B})^{min+1}$$

$$log_{\#(\mathcal{B})}(\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) + \#(\mathcal{B})) \leq log_{\#(\mathcal{B})}(\#(\mathcal{B})^{min+1})$$

$$log_{\#(\mathcal{B})}(\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) + \#(\mathcal{B})) \leq min + 1$$

$$log_{\#(\mathcal{B})}(\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) + \#(\mathcal{B})) - 1 \leq min$$

So in conclusion, since we need an integer (actually natural) number for this purpose, the final answer will be of the form:

$$min \geq \lceil log_{\#(\mathcal{B})}(\#(\mathcal{A}) \cdot (\#(\mathcal{B}) - 1) + \#(\mathcal{B})) - 1 \rceil$$

Problem 4.2 (Lexical Ordering) 3pt

Let $A := \{x, :, +, R, a\}$ and $\prec$ be the ordering relation on A with $a \prec R \prec + \prec : \prec x$. Order the following strings in A^* in the lexical ordering $\prec_{\text{lex}}$ induced by $\prec$.

$s_1 = RRRR$	$s_2 = RR + RRx$	$s_3 = \epsilon$
$s_4 = RR : RRa$	$s_5 = xRRRxR$	$s_6 = RRRR :$

Solution: $s_3 \prec_{\text{lex}} s_1 \prec_{\text{lex}} s_6 \prec_{\text{lex}} s_2 \prec_{\text{lex}} s_4 \prec_{\text{lex}} s_5$

Problem 4.3 (Character Code) 3pt

Consider the character code

$$c := \{l \mapsto wi, s \mapsto \_y, \ddot{a} \mapsto w, t \mapsto ou, a \mapsto sh, T \mapsto e\_\}$$

and the mapping

$$d := ish \mapsto hol, ou \mapsto ys!, we \mapsto hap, \_y \mapsto ida, \_w \mapsto py\_.$$

Let c' be the extension of c. Write down the values of $c'(\ddot{a}Tlast)$ and $d(c'(\ddot{a}Tlast))$

Solution:

- $c'(\ddot{a}Tlast) = we\_wish\_you$
- $d(c'(\ddot{a}Tlast)) = happy\_holidays$

5 Boolean Expressions

4pt

Problem 5.1 (Example for Validity and (Un-)Satisfiability)
Given the schema $(x_1 \vee e_1) \wedge e_2$ of a Boolean expression, where e_1 and e_2 stand for arbitrary other Boolean expressions. Generate three different Boolean expressions by instantiation of e_1 and e_2 such that the result expression becomes

1. valid,
2. unsatisfiable,
3. satisfiable and falsifiable at the same time.

For the last case provide two assignments, one satisfying your expression and the other falsifying it.

Solution:

1. valid: $(x_1 \vee \neg x_1) \wedge (x_1 \vee \neg x_1)$
2. unsatisfiable: $(x_1 \vee x_1) \wedge \neg x_1$
3. satisfiable and falsifiable: $(x_1 \vee x_2) \wedge x_1$

The last expression can be

- satisfied by $x_1 \mapsto \mathsf{T}$ and
- falsified by $x_1 \mapsto \mathsf{F}$

independently how x_2 is assigned.

6 Complexity

3pt

Problem 6.1 (Sorting Landau Sets)
For two functions f and g let us define $f \sim g$ iff $f \in \Theta(g)$ and $f \prec g$ iff $f \in O(g)$. Write down in terms of these two ordering relations how the following functions are related to each other.

Hint: Since these relations induce a total ordering, it is not necessary to write down explicitly every relation for each pair of functions. Instead you can determine the whole ordering in one line; e.g. in the form of $\ldots \prec f \sim g \prec h \prec \ldots$.

1. $f_1(n) := 2^{n+3}$
2. $f_2(n) := log(n^2)$
3. $f_3(n) := n^4$
4. $f_4(n) := 2^n$
5. $f_5(n) := log(n/2)$
6. $f_6(n) := 3^n$
7. $f_7(n) := n + 2^4$

Solution: $f_2 \sim f_5 \prec f_3 \sim f_7 \prec f_4 \sim f_1 \prec f_6$

7 The Quine-McCluskey Algorithm

6pt

Problem 7.1 (Quine-McCluskey Algorithm)
Use the algorithm of Quine-McCluskey to determine the minimal polynomial of the following function:

$x1$	$x2$	$x3$	f
F	F	F	T
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	T
T	T	T	T

8 Machine-Oriented Calculi

5pt

Problem 8.1 (Basics of Resolution)
What are the principal steps when you try to prove the validity of a propositional formula by means of resolution calculus? In case you succeed deriving the empty clause, why does this mean you have found a proof for the validity of the initial formula?