# General CS II (320102) Final Exam
# May 23. 2006

NAME:

MATRICULATION NUMBER:

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

You can reach 72 points if you solve all problems. You will only need 70 points for a perfect score, i.e. 2 points are bonus points.

*You have ample time, so take it slow and avoid rushing to mistakes!*

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

| To be used for grading, do not write into +this box | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| prob. | 1.1 | 2.1 | 3.1 | 3.2 | 4.1 | 5.1 | 5.2 | 6.1 | Sum | grade |
| total | 4 | 12 | 8 | 10 | 8 | 12 | 8 | 10 | 72 | |
| reached | | | | | | | | | | |

# 1 Computational Logic

4pt
8min

**Problem 1.1 (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?

# 2 Virtual Machines

12pt
15min

**Problem 2.1 (Binary Conversion in $\mathcal{L}(\texttt{VM})$)**
Write a $\mathcal{L}(\texttt{VM})$ program that converts a binary natural number into a decimal natural number. Suppose that $n$, the number of digits, is stored in `stack[2]` and $n$ numbers 0 or 1 above it follow, where the top of stack is the least significant bit. `stack[0]` and `stack[1]` are available for your use. Your program should leave only the converted number on the stack (in `stack[0]`). You are allowed to use labels for (conditional) jumps.

For instance an initial stack  should give the result stack  .

# 3 Combinational Circuits

8pt
10min

**Problem 3.1 (Shift and Duplication on PNS)**
Consider for this problem the signed bit number system and the two's complement number system. Given a binary string $b = a_n \ldots a_0$. We define

1. the duplication function *dupl* that duplicates the leading bit; i.e. it maps the $n+1$-bit number $a_n \ldots a_0$ to the $n+2$-bit number $a_n a_n \ldots a_0$ and
2. the shift function $shift$ that maps the $n+1$-bit number $a_n \ldots a_0$ to the $n+2$-bit number $a_n \ldots a_0 0$

Prove or refute the following two statements

- The $shift$ function has the same effect in both number systems; i.e. for any integer $z$:
$$\langle\!\langle shift(B(z)) \rangle\!\rangle^- = \langle\!\langle shift(B_n^{2s}(z)) \rangle\!\rangle_{n+1}^{2s}$$
- The *dupl* function has the same effect in both number systems; i.e. for any integer $z$:
$$\langle\!\langle dupl(B(z)) \rangle\!\rangle^- = \langle\!\langle dupl(B_n^{2s}(z)) \rangle\!\rangle_{n+1}^{2s}$$

10pt
15min

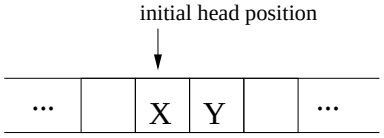
**Problem 3.2 (Carry Chain and Conditional Sum Adder)**
Determine depth and cost of a 4-bit Carry Chain Adder, a 4-bit Conditional Sum Adder, and a combination of both where two 2-bit Carry Chain Adders are connected with by one Mux.

Which adder has lowest cost and depth respectively?

# 4 Turing Machines

11pt
20min

**Problem 4.1 (Boolean Equivalence)**
Consider a tape arbitrarily filled with ones and zeros and the head initially positioned over some cell "X" as depicted below




Define a transition table for an always terminating Turing machine TM that computes the boolean equivalence of "X" and "Y": Upon halting, your TM should return the value 1 in cell "X" if the values of the cells "X" and "Y" were initially equal and otherwise 0.

Try to use as few states as possible. The number of points you can obtain for this exercise is $\max(0, 14 - x)$, where $x$ is the number of states of your working TM.

**Note:**

1. Admissible moves are *left*, *right*, and *none* with the obvious meaning.
2. You are free to overwrite the initial value of "Y" and to introduce additional symbols in the alphabet, if you need it for your solution.

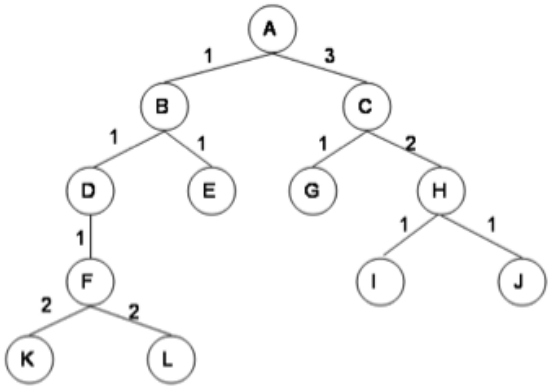
# 5 Problem Solving and Search

12pt
15min

**Problem 5.1 (Monotone heuristics)**
Let $c(n, a, n')$ be the cost for a step from node $n$ to a successor node $n'$ for an action $a$. A heuristic $h$ is called *monotone* if $h(n) \leq h(n') + c(n, a, n')$. Prove that if a heuristic is monotone, it must be admissible. Construct a search problem and a heuristic that is admissible but not monotone. Note: For the goal node $g$ it holds $h(g) = 0$. Moreover we require that the goal must be reachable.

8pt
10min

**Problem 5.2 (Search Strategy Comparison on Tree Search)**
Consider the tree shown below. The numbers on the arcs are the arc lengths.




Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state $G$. No visited or expanded lists are used. What order would the states be expanded by each type of search? Stop when you expand $G$. Write only the sequence of states expanded by each search.

| Search Type | Sequence of States |
|---|---|
| Breadth First | |
| Depth First | |
| Iterative Deepening (step size 1) | |
| Uniform Cost | |

# 6 Prolog

10pt
15min

**Problem 6.1 (Greatest Common Divisor)**

Write a `ProLog` program with a ternary predicate `gcd`, such that `gcd(A,B,D)` returns "Yes" if `D` is the greatest common divisor of `A` and `B`. Otherwise it should either return "No" or it should never terminate.

You can make use of any of the following built-in predicates: `<`, `>`, `=<`,`>=`, and `=`, as well as the basic arithmetic operations. But you have to define `mod` on your own if you need it.

**Note:** The Euclidean algorithm for the look like:

```
fun gcd(x, y) = if y = 0 then x else gcd(y, x mod y);
```