Name: Matriculation Number:

General CS II (320201) Final Exam
May 23. 2007

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

The estimated time for solving this exam is 116 minutes, leaving you 4 minutes for revising your exam.

You can reach 80 points if you solve all problems. You will only need 72 points for a perfect score, i.e. 8 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.	1.1	2.1	2.2	3.1	3.2	4.1	5.1	5.2	6.1	7.1	Sum	grade
total	4	4	6	12	8	10	10	8	12	6	80	
reached												

Good luck to all students who take this test

1 An Old GenCS Favourite ;-)

4pt
3min

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 $\langle a, b\rangle \in R$.

2 Graphs

4pt
5min

Problem 2.1 (Alternative Definition of a Tree)

Prof. Simplovsky approaches Prof. Kohlhase at a conference in Saskatchewan and suggests a different definition of trees which he claims is simpler than Prof. Kohlhase's (in the slides) and yet it fully captures the notion of trees too:

"A tree is a directed, connected acyclic graph with exactly one node with indegree zero, which we call the root node."

Is Prof. Simplovsky right? Explain your answer by proving the equivalence of the respective definitions or giving an example that differentiates them.

Note: We call a directed graph connected, iff for any two nodes n_1 and n_2 there is a path in the underlying *undirected* graph (that we get by disregarding all directions of the edges) starting at n_1 and ending at n_2. (If this property holds for the directed graph itself, it is called *strongly* connected instead.)

Solution: Prof. Simplovsky is wrong; counter-example:

$a \to b$
$a \to c$
$b \to d$
$c \to d$

That is, we need the additional condition that every node except the root has an in-degree of 1.

(Thanks to Darko Pesikan for suggesting this problem.)

Problem 2.2 (Another Alternative Definition of a Tree) 6 pt 8 min

Prof. Trivialczyk approaches Prof. Kohlhase at a conference on Hawaii and notes that a fully balanced binary tree can be defined without introducing balanced trees before, but simply as *"a binary tree with the numbers of nodes at each depth equal to powers of two, and with a total number of nodes equal to* 2^{n-1} *for some* $n \geq 1$*."*

Is Prof. Trivialczyk right? Explain your answer by proving the equivalence of the respective definitions or giving an example that differentiates them.

Solution: Prof. Trivialczyk is wrong; counter-example: a binary tree with 7 nodes, depth 3 and 2 nodes at each level.

(Thanks to Darko Pesikan for suggesting this problem.)

3 Combinatorial Circuits and Memory

12pt
20min

Problem 3.1 (A Vending Machine Circuit)

Given is a vending machine for candies that accepts coins of 5 and 10 cents only and stores the current credit c. One candy costs 15 cents. If the credit reaches 15 cents or more ($c \geq 15$), the machine dispenses one candy (you don't need to implement this! ;-) and reduces the credit by 15 cents, i. e. the new credit c' is $c - 15$.

Assume that no two coins can be thrown into the machine at the same time, and that a clock signal is only generated when a coin is thrown into the machine, i. e. there are no transitions like $c' = c + 0$.

Note: Take the clock signal as a "magic" input from outside. You do not need to generate or manipulate it yourself.

We shall model this machine as a black box that accepts an input for the coins (appropriately encoded) and gives two output bits that represent the current credit.

1. Devise an appropriate encoding for storing the credit and representing coins as input bits.

2. Draw a state diagram, i. e. a graph whose nodes are the possible states of the machine (here, a state is the value of c) and whose edges are possible transitions between the states that are taken on a certain input.

3. Now draw the sequential logic circuit. You may use D-flipflops and any combinational circuits or gates that have been introduced in this course. Make sure you remove any ambiguity in your drawing by properly labeling wires and/or circuit elements and sufficiently explaining the layout.

Solution:

1. One possible encoding is: the state "0 cents credit" represented as 00, "5 cents" represented as 01, and "10 cents" represented as 10. An input of 5 cents is represented as 01, 10 cents are represented as 10. The state 11 is not possible, neither are the inputs 00 and 11.

2. ...

Note regarding the input of coins: If you take the clock as "magic", one bit is sufficient to distinguish 5 from 10 cents. If a student generates or manipulates the clock signal (as he might know more about clocks than we learned in GenCS), a two-bit encoding is needed and should also be accepted as a solution.

(Thanks to Snežana Jovanoška and Darko Pesikan for suggesting this problem.)

Problem 3.2 (Conditional Sum Adder) 8pt 10min

Draw the circuit of a Conditional Sum Adder (CSA) that adds two four-bit numbers. Go down to the level of elementary gates.

Hint: To spend less time on drawing gates, you can draw a complex circuit that is needed for the CSA (e. g. a half adder) once from elementary gates and then re-use it as a building block with appropriate inputs and outputs. Be sure to *explain* the layout of the whole circuit.

4 Virtual Machines

10pt
15min

Problem 4.1 (While Loop in $\mathcal{L}(\texttt{VM})$)

Write a program in the Simple While language that takes two numbers A and B, given at the memory addresses 1 and 2, and returns $(A + B)^{42}$. Show how the compiled version of it looks like in the Virtual Machine Language $\mathcal{L}(\texttt{VM})$ (concrete, not abstract syntax).

Solution:

```
var n := 1; var a := A; var b := B;
var c := a+b;
while n >= 42 do (
  p := c * p;
  n := n + 1;
)
return p;
```

```
con 1 peek 1 peek 2
add con 1
peek 0 con 43 leq cjp halt
peek 1 peek 2 mul poke 2
peek 0 con 1 add poke 0
jp back
halt
```

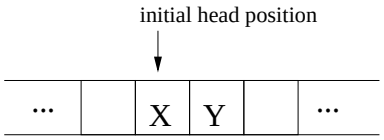
---

5 Turing Machines

10pt
15min

Problem 5.1 (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.

Hint: You only need to consider the two cells "X" and "Y". It does not matter where the head stays when the TM terminates.

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.

Solution: There are lots of possible solutions.

The following four-state solution (including the final state) by Dmakreshanski Pesikan does not use any additional alphabet symbols:

Old	Read	Write	New	Move
s_1	0	0	s_2	right
s_1	1	1	s_2	right
s_2	0	0	s_3	left
s_3	1	0	s_4	left
s_3	0	1	s_4	none

Tanmay Pradhan presented a solution that uses additional symbols and only needs *two states*. This is supposed to be optimal.

Old	Read	Write	New	Move
s_1	0	W	s_2	right
s_1	1	Y	s_2	right
s_2	0	L	s_1	left
s_2	1	L	s_2	left
s_1	W	1	s_1	right
s_1	Y	0	s_1	right
s_2	W	0	s_1	right
s_2	Y	1	s_1	right

If we require that the head must stop on "X" – but we don't , as Darko pointed out! – , it gets more complicated. The following solution by Christoph Lange is quite straight-forward, but not optimal:

Old	Read	Write	New	Move
a	0	0	b	right
a	1	1	b	right
b	0	0	c_0	left
b	1	1	c_1	left
c_0	0	1	$\perp$	none
c_0	1	0	$\perp$	none
c_1	0	0	$\perp$	none
c_1	1	1	$\perp$	none

Andrei Aiordachioaie supposed this four-state solution:

Old	Read	Write	New	Move
s_0	1	X	right	s_x
s_0	0	Y	right	s_y
s_x	1	1	left	s_x
s_x	X	1	none	$\perp$
s_x	0	(anything)	left	s_y
s_x	Y	0	none	$\perp$
s_y	0	0	L	s_y
s_y	Y	1	none	$\perp$
s_y	1	(anything)	left	s_x
s_y	X	0	none	$\perp$

Problem 5.2 (Halting Problem) 8pt 10min

Define the Halting Problem:

- What does it state about the computational power of Turing machines?
- Outline (informally) how one would prove this statement.
- Why is the halting problem interesting; what is its practical relevance?

6 Problem Solving and Search

12pt
20min

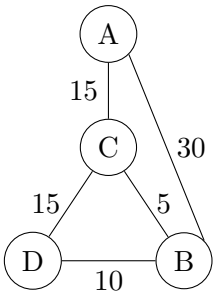
Problem 6.1 (Interpreting Search Results)

The state of Ingushetia has only four cities (A, B, C, and D) and a few two-way roads between them, so that it can be modeled as an undirected graph with four nodes. The task is to go from city A to city D. The UCS algorithm finds a solution to this task that is 10km shorter than the one BFS finds. The solution of BFS in turn is 10km shorter than the one of the DFS algorithm.

Draw a map of Ingushetia with roads and their distances that satisfies both conditions. What paths between A and D in your map will be found as solutions by each of those algorithms?

Note: All algorithms had repetition checking implemented, so that when a node is expanded, all its children that belong to a list of previously expanded nodes during the execution of that algorihtm are ignored. In addition, when no order of choosing a node for expansion is specified by an algorithm, expansion in alphabetical order takes place.

Solution: One possible solution:



The paths found are:

UCS ACBD (30km)

BFS ABD (40km)

DFS ABCD (50km)

A second solution: All possible edges should be present in the graph, except AD. The weigths are given as : AB = 10, AC = 10, AD is not in the graph, BC = 20, BD = 20, CD = 10. The solutions are then: DFS: ABCD , with cost = 40; BFS: ABD, with cost = 30; UCS: ACD, with cost = 20.

(Thanks to Darko Pesikan for suggesting this problem.)

---

7 Prolog

6pt
10min

Problem 7.1 (Paths in a Graph)

Given a directed graph, represented by `edge(from, to)` facts, write a predicate `trip(A, B, L)` that succeeds if the node `B` is accessible from `A` via the intermediate nodes `L` (an ordered list of nodes).

Note: It is not required to avoid cyclic trips.

Solution:

```
trip(A, B, []) :- edge(A, B).
trip(A, B, [H | T]) :- edge(A, H), trip(H, B, T).
```
