

# Third Exam

## Logic Programming, LVA 703113

October 27, 2017

---

Name:

Student Number:

---

The exam consists of 5 exercises with a total of 100 points. Please fill out your name and credentials *before* you start the exam.

1	2	3	4	5	Sum
<input type="checkbox"/>					

0-49: 5	50-59: 4	60-74: 3	75-89: 2	90-100: 1
---------	----------	----------	----------	-----------

1. Consider the directed graph  $G = (\{a, b, c, d, e, f, g, h\}, E)$  with the following set of edges:

$$E = \{(a, a), (a, b), (b, b), (b, c), (c, d), (d, e)\} .$$

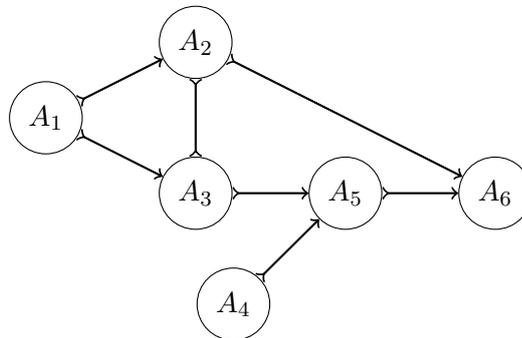
- a) Represent  $G$  in Prolog and implement a relation *connected/2* that expresses that two nodes are connected in  $G$ . (5 pts)
- b) Show that for any directed acyclic graph  $G$  the *height* of the SLD tree is bounded by the number of vertices in  $G$ . (10 pts)
- c) Is the above estimate on the height of the search tree also true for non-ground queries? (5 pts)
2. Consider two lists  $Xs, Ys$  and implement a predicate *common\_subsequence(Xs,Ys,Zs)*, which yields in its third argument the length of the longest common subsequence of  $Xs$  and  $Ys$ :

$$:- \text{common\_subsequence}([1, 2, 3, 4], [2, 3], Xs), Xs = 2. \quad (20 \text{ pts})$$

3. Implement the concatenation of three lists *append\_append\_append(Xs,Ys,Zs,Rs)* using DCGs.

$$:- \text{append\_append\_append}([1, 2], [2, 3], [3, 4], Rs), \\ Rs = [1, 2, 2, 3, 3, 4]. \quad (20 \text{ pts})$$

4. Consider the following dependency graph of six tasks. Here the edge between  $A_2$  and  $A_3$  denotes that these two task must not be scheduled at the same time.



Implement a predicate *plan([A1,A2,A3,A4,A5,A6])*, succeeding, if  $[A1, A2, A3, A4, A5, A6]$  denotes a valid sequence of all task.

$$:- \text{plan}(Xs), Xs = [1, 2, 3, 1, 4, 5]. \quad (20 \text{ pts})$$

5. Determine whether the following statements are true or false. Every correct answer is worth 2 points, every wrong answer -1 points. (The worst that can happen is that you get zero points for this exercise.) (20 pts)

<b>statement</b>	<b>yes</b>	<b>no</b>
An existential fact is a fact that contains existentially quantified variables.	<input type="checkbox"/>	<input type="checkbox"/>
A rule is a universally quantified logical formula of the form $A : \neg B_1, B_2, \dots, B_n$ , where $A$ is a goal and for all $i = 1, \dots, n$ : $B_i$ is a goal.	<input type="checkbox"/>	<input type="checkbox"/>
Data is structured in logic programs to obtain for example (i) better modularity or (ii) better organisation of the data.	<input type="checkbox"/>	<input type="checkbox"/>
Almost all, but not all basic elements of a relation database model can be expressed in Prolog.	<input type="checkbox"/>	<input type="checkbox"/>
Consider the standard implementation of <i>member/2</i> . Then any call to <i>member</i> terminates iff the second argument is a complete list.	<input type="checkbox"/>	<input type="checkbox"/>
A Prolog clause is called <i>iterative</i> if it has one recursive call and zero or more calls to system predicates that appear before the recursive call.	<input type="checkbox"/>	<input type="checkbox"/>
A cut fixes all choices between (and including) the moment of matching the rule's head with parent goal and the cut. If backtracking should reaches the cut, then the cut succeeds and the execution is continued with the clause <i>after</i> the clause containing the cut.	<input type="checkbox"/>	<input type="checkbox"/>
Constraint logic programs typically fall into two parts. In one part the pure relation is formulated, possible using constraints. In the second part the constraints are instantiated.	<input type="checkbox"/>	<input type="checkbox"/>
The predicate <i>bagof(Template,Goal,Bag)</i> unifies <i>Bag</i> with the alternatives of <i>Goal</i> that meet <i>Template</i> .	<input type="checkbox"/>	<input type="checkbox"/>
Time complexity of Prolog programs is measured via the number of inferences performed and real time consumption depends on the time used for unification attempts.	<input type="checkbox"/>	<input type="checkbox"/>