

Take Home Exam for Integrated Logic Systems: Advanced WAM Module

May 13, 2009

The rules for this take-home test are as follows. Although the slides should be sufficient to answer all questions, you can consult any written or web-based material you like. However you may *not* ask questions of anyone except Prof. Swift, and you may *not* consult with each other. Failure to observe these rules will unconditionally result in a grade of 0 for this module.

Exams must be emailed in pdf format to tswift@cs.sunysb.edu no later than May 18 2009, 24:00.

Slides at <http://www.cs.sunysb.edu:tswift/webtalks/ils0.pdf> have been updated with additional material useful for this exam.

1. *Tabling Implementation: 2.5 points* In Lecture 4, the code for tabled `ancestor/2` makes use of a non-WAM instruction `getVn`. Explain in 200 words or less what this instruction does and why it is needed.

2. *Implementation: 5 points*

- Explain how tail recursion in the WAM (without tabling) saves space on the environment stack. Mention why a WAM `execute` instruction is used.
- Consider a tabled evaluation of the query `parent(1,X)` with left recursive form of `ancestor/2`

```
:- table ancestor/2
```

```
ancestor(X,Y):- ancestor(X,Z),parent(Z,Y)
```

```
ancestor(X,Y):- parent(X,Y)
```

and the right-recursive

```
:- table ancestor/2
```

```
ancestor(X,Y):- parent(X,Z),ancestor(Z,Y)
```

```
ancestor(X,Y):- parent(X,Y)
```

where `parent/2` facts have the form `parent(1,2),parent(2,3),...parent(999,1000)`. In other words, a chain of 1000 elements.

- How many tabled subgoals (i.e. how many different trees in the forest) will the left-recursive form create and how many will the right recursive form create?

- What is the total number of answers the left-recursive form will create for all tabled subgoals? How many will the right recursive form create?
- What is the total number of *consumer choice points* each form will create?
- What will be the differences, if any, in execution stack space between the left-recursive form and the right-recursive form if the evaluation uses local scheduling (which will give fully incremental completion)? You may restrict your discussion to the choice point stack in the split-stack model. Answer in 200 words or less.
- Now suppose call subsumption is used on the right-recursive form of `ancesotor/2`, via a declaration such as


```
:- table ancestor/2 as subsumptive.
```

 How many producer and subsumer subgoal frames will be created for the query `p(1,X)`? How many producer and subsumer subgoal frames will be created for the query `p(X,Y)`? What can you conclude about transformation of a predicate into left-recursion vs. using call subsumption?

3. *Negation 5 points* In Lecture 4, the definition of the SLG SIMPLIFICATION operation was provided with 4 different cases for applying SIMPLIFICATION. In this problem, you will need to indicate a query, program, and evaluation in which each case must be used.
- For the program, assume Prolog’s top-to-bottom clause selection strategy and left-to-right literal selection strategy.
 - Represent the evaluation by its final forest, indicating the order of operations by numbering each node. Make sure each (non-failure) node has the form

$$N = Ans \text{ :- } DelaySet|Goal_List$$

E.g. see the example on pages 139-141.

As long as these operations are performed, you may any number of programs and evaluations you like (4 separate programs and evaluations is probably easier) and the programs may be ground.

For the full 5 points, make sure the evaluations are delay minimal.

4. *Negation 2.5 points* The second condition of the COMPLETION operation is sometimes called *Early Completion*, and in Lecture 4 it is argued that Early Completion is needed in order to evaluate LRD-stratified programs without using the DELAYING operation. Indicate a LRD-stratified query and program that uses Early Completion to avoid DELAYING. Use the representation of an SLG evaluation as indicated above.
5. *Tabling Implementation 5 points* The lectures and slides discuss how call subsumption can have advantages for evaluating queries to certain programs. The paper *One Table Fits All* (J. Costa and R. Rocha, PADL 2009) also found on this web page, discusses another approach called *Global Tables*. This approach may offer advantages over the implementation of call variance, discussed in various Lectures, which uses predicate-based tries. Discuss the comparative advantages and disadvantages of call subsumption and global tables in terms of time and space. You need not provide example programs – just a clear comparison of strengths and weaknesses in 200 words or less.