Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages

Assignment 6

DUE: Thursday, April 27, 2023, 11:59PM Eastern Daylight Time

Please note that you can use at most three late days on a single assignment, i.e., the final due date is Sunday April 30, 2023, 11:59pm. If you collaborate with your classmates, please note who they are when you submit your answers.

Submission instructions: Please submit your homework via GradeScope. Some points to note:

- We encourage you to use LaTeX. If you produce your assignment using a different tool, please make sure your assignment is legible and follows these submission instructions.
- We will grade blind, i.e., graders should not know whose assignment they are grading. To support this, the first page of your assignment should state your name, HUID, and a list of collaborators (if any), and *all other pages of your assignment should not contain any identifying information*, i.e., they should not contain your name.
- We will provide (through Canvas) a LaTeX template you can use to write the answers to your assignment.
- When you submit your PDF on Gradescope, **please indicate which pages contain which questions**. This makes it much easier for the course staff to grade. 2 point deduction for failure to indicate pages!

1 Family Relations in Datalog (30 points)

In this question you will get familiar with Datalog by implementing some simple predicates.

(a) **First things first: get Datalog.** We are going to use AbcDatalog, which is our very own implementation. Go to `http://abcdatalog.seas.harvard.edu/` and download the graphical user interface (it's the second bullet in the "Downloads" section). To run the GUI, you will need to have Java 8 installed. To check which version of Java you have, run the command `java -version` in your terminal (assuming you're using OSX or Linux). If the version is in the 1.8 range, you are all set. If it's not, you will need to install Java 8. You can download an installer for your system at `http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`.

The "User's Guide" section of the AbcDatalog website has some information that you might find helpful. In particular, check out the subsections on "Datalog Syntax and Grammar" and "Using the GUI." Try out the simple example given on the website.

In the file `family.dtlg`, we have defined the predicates `person`(*name*) and `parent_of`(*parent*, *child*).

Download the file `family.dtlg`, available on Canvas. Please add your rules to this file inside the "BEGIN ANSWER" and "END ANSWER" comments, and upload it to Gradescope. You are free to modify the `family.dtlg` however you want so long as your final answer is between the "BEGIN ANSWER" and "END ANSWER" comments.

(b) **Siblings.** Define the predicate `siblings`(X, Y). If `alice` and `bob` are siblings, then `siblings(alice, bob)` and `siblings(bob, alice)` should both be true (but, e.g., `siblings(alice, alice)` should not).

(c) **Relations.** Define the predicate `related`(X, Y), which is true for any distinct X and Y that are related. Alice and Bob are related if Alice is the parent or ancestor of Bob, if they are siblings, or if they are cousins (i.e., if they have some common ancestor). For example, if Alice is the parent of Bob and Charlie, and Charlie is the parent of Dina, then Dina is related to Alice, Bob, and Charlie., and Bob is related to Alice, Charlie, and Dina.

Hint: you may find it useful to define additional predicates, such as `ancestor`(X, Y).

(d) **Same generation.** Define the predicate `same_generation`(X, Y), which is true for any distinct pair of persons that have a common ancestor and are of the same generation with respect to that ancestor. For example, if Alice is the parent of Bob and Charlie, and Bob is the parent of Dina, and Charlie is the parent of Elissa, then Bob and Charlie are the same generation, and Dina and Elissa are the same generation (and no other pairs are the same generation).

Hint: do not try to use the `related`(X, Y) predicate you defined above.

(e) **Unrelated.** Define the predicate `unrelated`(X, Y), which is true for any distinct pair of persons that are not related.

Hint: you will need to use negation for this. Use the `related`(X, Y) predicate you defined above.

2 Boolean Algebra in Datalog (30 points)

In this question you will use Datalog to build a calculator for boolean algebra. That is, you will write a Datalog program that takes as input the encoding of a boolean algebra expression and produces as output the encoding of the final answer to the boolean algebra computation specified by the input expression.

Boolean algebra expressions are described by the following grammar:

$$\begin{aligned} &e \in \textbf{BAExp} \\ &l \in \textbf{Label} \\ &e ::= T^l \mid F^l \mid (e_1 \land e_2)^l \mid (e_1 \lor e_2)^l \mid (\neg e)^l \end{aligned}$$

Each expression is associated with a unique label l, which we will use to identify subexpressions when we encode expression as sets of Datalog facts. For convenience, we write e^l to indicate that l is the label of the expression e.

Expressions T^l and F^l represent true and false respectively. Expression $(e_1 \land e_2)^l$ denotes the logical conjunction of e_1 and e_2. Expression $(e_1 \lor e_2)^l$ denotes the logical disjunction of e_1 and e_2. Expression $(\neg e)^l$ denotes the logical negation of e.

We encode a boolean algebra expression as a set of Datalog facts using the following recursive function.

$$\begin{aligned} \mathcal{F}[\![T^l]\!] &= \{\texttt{true\_constant}(l)\texttt{.}\} \\ \mathcal{F}[\![F^l]\!] &= \{\texttt{false\_constant}(l)\texttt{.}\} \\ \mathcal{F}[\![(e_1^{l_1} \land e_2^{l_2})^l]\!] &= \{\texttt{conjunction}(l, l_1, l_2)\texttt{.}\} \cup \mathcal{F}[\![e_1]\!] \cup \mathcal{F}[\![e_2]\!] \\ \mathcal{F}[\![(e_1^{l_1} \lor e_2^{l_2})^l]\!] &= \{\texttt{disjunction}(l, l_1, l_2)\texttt{.}\} \cup \mathcal{F}[\![e_1]\!] \cup \mathcal{F}[\![e_2]\!] \\ \mathcal{F}[\![(\neg e^{l'})^l]\!] &= \{\texttt{negation}(l, l')\texttt{.}\} \cup \mathcal{F}[\![e]\!] \end{aligned}$$

For example, $\mathcal{F}[\![((T^1 \lor F^2)^3 \land T^4)^5]\!]$, the encoding of the boolean algebra expression $((T^1 \lor F^2)^3 \land T^4)^5$, is:

$$\left\{ \begin{array}{c} \texttt{true\_constant}(1)\texttt{.},\ \texttt{false\_constant}(2)\texttt{.},\ \texttt{disjunction}(3,1,2)\texttt{.}, \\ \texttt{true\_constant}(4)\texttt{.},\ \texttt{conjunction}(5,3,4)\texttt{.} \end{array} \right\}$$

Using this encoding of a boolean algebra expression, we can implement an evaluator for boolean alegebra expressions. Specifically, you will define a predicate `eval`(l, v) such that `eval`$(l,$`true`$)$ holds if the expression labeled l evaluates to true, and `eval`$(l,$ `false`$)$ holds if the expression labeled l evaluates to false.

(a) **Your task is to write a set of Datalog rules that compute the final truth value of an encoded boolean algebra expression.** You can assume that your rules will be evaluated on a set of facts that are the correct encoding of a boolean algebra expression.

You must define rules for the predicate `eval`(l, v) such that `eval`$(l,$`true`$)$ holds if the expression labeled l evaluates to true, and `eval`$(l,$ `false`$)$ holds if the expression labeled l evaluates to false.

For example, for the example encoding of expression $((T^1 \vee F^2)^3 \wedge T^4)^5$ given above, the fact `eval(5, true)` should hold, and the fact `eval(5, false)` should not hold. (That is, if we ran the query `eval(5, X)?`, then the single fact `eval(5, true)`, would be returned.)

We have given you a head start in the file `algebra.dtlg`, available on Canvas. Please add your rules to this file inside the "BEGIN ANSWER" and "END ANSWER" comments, and upload it to Gradescope. You will want to test your rules on sample encodings, and you are free to modify the `algebra.dtlg` however you want (e.g., to try out your rules on different sample encodings) so long as your final answer is between the "BEGIN ANSWER" and "END ANSWER" comments.

3 Embedded EthiCS Module (40 points)

(Note that the word limits are upper limits; you are welcome—even encouraged—to write shorter answers!)

(a) In the Embedded EthiCS module, we discussed differences between *ex ante* and *ex post* risks in software development.

  (i) How does a case discussed in class demonstrate the difficulty in predicting *ex ante* risks? (100 words maximum.)

  (ii) What strategies could we use to better anticipate *ex ante* risks? Do you think they result in more responsible practices (why or why not?)? (100 words maximum.)

(b) We also discussed an argument from inductive risk for including stakeholder interests and values in software development.

  (i) Briefly summarize the argument. (100 words maximum.)

  (ii) Suppose the argument's conclusion is true. What do you think are additional considerations for responsibly integrating stakeholder values and interests into software design? (100 words maximum.)