CSE-6490B Assignment #1

1. **Queries in Datalog.** *Enroll now in Datalog U.!* (5 points)

   Consider the following schema.

```
student(s#, sname, dob, d#)
    FK (d#) refs dept           // Student's major
prof(p#, pname, d#)
    FK (d#) refs dept           // Professor's home deparmtent
dept(d#, dname, building, p#)
    FK (p#) refs prof           // Department's chair
course(d#, no, title)
    FK (d#) refs dept           // Course offered by this deparmtent
class(d#, no, term, year, section, room, time, p#)
    FK (d#, no) refs course     // Class is an offering of this course
    FK (p#) refs prof           // Instructor of class
enroll(s#, d#, no, term, year, section, grade)
    FK (s#) refs student        // This student is enrolled in
    FK (d#, no, term, year, section) refs class // this class
```

   'FK' above stands for *foreign key.* These indicate foreign-key constraints in the schema.

   Write the following queries in Datalog (and Datalog¬). You may use auxiliary predicates and rules. (You may reuse auxiliary predicates and rules in following sub-questions.)

   A common convention is to use '_' as a variable name when the variable is unimportant for the query; e.g., *class* (*D, N,* _, _, _, _, _, _). By convention, two occurrences of '_' are *different* variables and may take on different values (even though they seem to have the same "name"). You may find this convention useful.

   Be careful that all your rules are *safe*, including rules that you write that use negation.

   a. Which students have taken some course twice?

   b. Which students have taken a course with *a* department chair?
   Note that a professor may teach classes outside of his or her department. Also note that a student may take classes in a department outside of his or her major's department.

   c. Which students have never taken a course in his or her major (dept)?

   d. Which students have taken all of the courses offered by a department?

   e. (somewhat hard) Which students have taken at least five courses in their major (dept)?
   You shall need to use arithmetics (e.g., '$\neq$', '$<$') here. Assume that course numbers (no) can be compared; e.g., $M < N$. Use the predicate *is* to equate numbers; e.g., J is $I + 1$.

2. **Datalog Modeling.** *As easy as rolling off a log.* (5 points)

   The puzzle *Sū Doku*—or just *sudoku*—is to fill in the blank cells of a 9×9 matrix with the numerals 1, . . . , 9 such that no row has the same numeral twice, no column has the same numeral twice, and no *block* has the same numeral twice. The 9×9 matrix is tiled by nine 3×3 matrices, each called a block.

   A typical sudoku puzzle has some of the cells already filled in (the *givens*) so that there exists exactly one solution. For example,

5	**3**			**7**				
6			**1**	**9**	**5**			
	9	**8**					**6**	
8				**6**				**3**
4			**8**		**3**			**1**
7				**2**				**6**
	6					**2**	**8**	
			4	**1**	**9**			**5**
				8			**7**	**9**

$\Rightarrow$

5	**3**	4	6	**7**	8	9	1	2
6	7	2	**1**	**9**	**5**	3	4	8
1	**9**	**8**	3	4	2	5	**6**	7
8	5	9	7	**6**	1	4	2	**3**
4	2	6	**8**	5	**3**	7	9	**1**
7	1	3	9	**2**	4	8	5	**6**
9	**6**	1	5	3	7	**2**	**8**	4
2	8	7	**4**	**1**	**9**	6	3	**5**
3	4	5	2	**8**	6	1	**7**	**9**

   with the solution shown on the right.

   Write a Datalog program for sudoku. Let each cell in the sudoku matrix be represented by a variable:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}
X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}
X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}
X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	X_{41}	X_{42}	X_{43}	X_{44}
X_{45}	X_{46}	X_{47}	X_{48}	X_{49}	X_{50}	X_{51}	X_{52}	X_{53}
X_{54}	X_{55}	X_{56}	X_{57}	X_{58}	X_{59}	X_{60}	X_{61}	X_{62}
X_{63}	X_{64}	X_{65}	X_{66}	X_{67}	X_{68}	X_{69}	X_{70}	X_{71}
X_{72}	X_{73}	X_{74}	X_{75}	X_{76}	X_{77}	X_{78}	X_{79}	X_{80}

   Do not use negation, arithmetics (e.g., “$\neq$”, “$<$”), or function symbols (which are not in Datalog proper anyway).

   One predicate should be *sudoku* that takes arguments $X_0, \ldots, X_{80}$. For a given puzzle, one could then query for the solution; e.g.,

$$\leftarrow \textit{sudoku}\ (5,\ 3,\ X_2,\ X_3,\ \ldots,\ 9).$$

   Note that your sudoku “program” need not be efficient in any way. It just needs to *specify* logically and correctly the problem. So try to keep it quite simple. You may use ellipses (e.g., “. . .”, “$\vdots$”) where appropriate and when easily understood to make your answer briefer.

   Hint: Be clever in defining the permutations of 1, . . . , 9.

3. **Query Containment.** *I can't contain myself.* (5 points)

   Consider the following conjunctive queries, Ullman-style. Note that 'c' in $\mathcal{Q}_4$ is a constant, not as variable.

$$
\begin{aligned}
\mathcal{Q}_1&: p\ (X,\ Y) \leftarrow q\ (X,\ A),\ q\ (A,\ B),\ q\ (B,\ Y).\\
\mathcal{Q}_2&: p\ (X,\ Y) \leftarrow q\ (X,\ A),\ q\ (A,\ B),\ q\ (B,\ C),\ q\ (C,\ Y).\\
\mathcal{Q}_3&: p\ (X,\ Y) \leftarrow q\ (X,\ A),\ q\ (B,\ C),\ q\ (D,\ Y),\\
&\qquad\qquad\qquad q\ (X,\ B),\ q\ (A,\ C),\ q\ (C,\ Y).\\
\mathcal{Q}_4&: p\ (X,\ Y) \leftarrow q\ (X,\ A),\ q\ (A,\ c),\ q\ (c,\ B),\ q\ (B,\ Y).
\end{aligned}
$$

a. Find all containments and equivalences between $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$.

b. For each of $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$, simplify it. This means find the minimal clause that is equivalent to $\mathcal{Q}'_i$, in each case.

   Simplify $\mathcal{Q}'_1 \cup \mathcal{Q}'_2 \cup \mathcal{Q}'_3 \cup \mathcal{Q}'_4$ (where $\mathcal{Q}'_i$ is your simplified $\mathcal{Q}_i$). This means eliminate any of the rules contained in any other, because these do not contribute anything additionally to $p\ (X,\ Y)$.

Inequalities.

$$
\begin{aligned}
\mathcal{Q}_1&: p\ (X,\ Y) \leftarrow e\ (X,\ A),\ e\ (A,\ Y),\ e\ (X,\ B),\\
&\qquad\qquad\qquad e\ (B,\ Y),\ X < A,\ B < Y.\\
\mathcal{Q}_2&: p\ (X,\ Y) \leftarrow e\ (X,\ Z),\ e\ (Z,\ Y),\ X < Z,\ Z < Y.
\end{aligned}
$$

c. A containment mapping is sufficient and necessary to show containment for Datalog conjunctive queries without inequalities. It is still necessary but not sufficient to show containment for Datalog conjunctive queries with inequalities.

   What do you need to additionally show in these cases to prove containment?

d. For $\mathcal{Q}_1$ and $\mathcal{Q}_2$ above, show whether each is contained in the other.

e. Name a reasonable restriction on conjunctive queries (Ullman's definition) that makes the query containment problem—that one conjunctive query is contained by another—decidable in polynomial time. Be very precise about what the restriction is.

4. **Integrity Constraints.** *You can't say that!* (5 points)

Most schema also have integrity constraints. We can extend our Datalog databases to include integrity constraints (ICs), and our notion of containment to account for ICs.

An integrity constraint can be written as a query, with the mandate that the IC "query" must evaluate to have *no* answers. A common convention is to use '$Leftarrow$' instead of '$leftarrow$' when writing an IC instead of a query, to distinguish ICs and queries.

---

a. (2 points) Write an IC to represent the constraint that s# is the primary key of **student** (as in the schema in Question 1); that is, a s# value can only appear once.

Write an IC to represent the constraint that d# is a foreign key of **student** referencing **dept**; that is, any d# value in **student** must be also a value in **dept**.

---

b. (2 points) Consider

$$\begin{array}{l} \Leftarrow e\,(A,\ C),\ e\,(B,\ C),\ A \neq B. \\ d\,(A,\ C) \leftarrow e\,(A,\ B),\ e\,(B,\ C). \\ t\,(A,\ D) \leftarrow e\,(A,\ B),\ e\,(B,\ C),\ e\,(C,\ D). \end{array}$$

Given d and t as resources, and assuming the IC for both, find the maximal contained foldings for

$$\leftarrow e\,(X,\ Y).$$

---

c. (1 point) Consider

$$\begin{array}{l} \Leftarrow e\,(A,\ B),\ e\,(B,\ A). \\ tri\,(A,\ B,\ C) \leftarrow e\,(A,\ B),\ e\,(B,\ C),\ e\,(C,\ A). \\ nontrans\,(A,\ B,\ C) \leftarrow e\,(A,\ B),\ e\,(B,\ C),\ \mathsf{not}\ e\,(A,\ C). \end{array}$$

Is *tri* contained by *nontrans*, given the IC?

Is *nontrans* contained by *tri*, given the IC?

5. **Complexity of Containment.** *Does it have to be so hard?* (5 points)

   3-SAT is an NP-complete problem. Given a collection of propositional clauses (a disjunction of propositional variables, some negated), each consisting of exactly *three* propositional variables, is the set of clauses mutually satisfiable? That is, is there a model of the set of clauses (an assignment of *true* or *false* to each variable that makes all the clauses *true*)?

   We can represent a 3-SAT-compatible clausal theory as follows. Let $\mathcal{C}_0, \ldots, \mathcal{C}_{k-1}$ be the set of clauses over the propositional variables $p_0, \ldots, p_{n-1}$. Let each clause be a set of three of the variables, some negated; e.g., $\mathcal{C}_i = \{p_3, \neg p_5, \neg p_7\}$.

   Your job is to design a mapping of 3-SAT problems into Datalog in order to establish complexity bounds on the containment problem. You are not allowed recursion, negation, or arithmetics (e.g., '$\neq$', '$<$').

   Predicates that you use in the head of rules *and* in the body of rules can be zero-ary (that is, have no arguments). Note that a zero-ary predicate is simply a propositional variable (which is perfectly fine in the predicate calculus and in Datalog). In fact, you may not need any variables for this task.

   A "database", given a collection of zero-ary predicates, is a subset of those predicates. If the predicate appears in the database, it is *true*; if it is missing, it is *false* (by the closed world assumption).

   Using 3-SAT, establish that Datalog $\sqsubseteq$ Datalog (for Datalog without recursion, negation, or arithmetics) is co-NP-hard. That is, the class of a Datalog program contained in a Datalog program.

   A problem is *NP-hard* if one can show a (polynomial) mapping of an NP-complete problem into it, but we have not necessarily shown the problem to be in NP. So we may not be able to establish that, given a solution to the problem, we can verify the solution easily (in polynomial time). If we *could* also establish the problem to be in NP, the we would have established it to be NP-complete.

   A problem is co-NP-hard (or co-NP-complete) if we can show that the co-problem (or "dual problem") is NP-hard (or co-NP-complete). Here, the co-problem of 3-SAT is to identify those 3-SAT theories that are *not* satisfiable.

   Clearly, given your mapped 3-SAT problem as a Datalog program, verifying the (co-problem) solution would be in NP. However, for the Datalog-in-Datalog containment problem in general, it is hard to determine whether this would always be doable. So we just shoot for co-NP-hard.