Exam

Uncertain Data Management
Université Paris-Saclay, M2 Data&Knowledge

January 30th, 2017

This is the final exam for the Uncertain Data Management class, which will determine your grade for this class. The duration of the exam is 2.5 hours. The exam consists of 4 independent exercises.

No additional explanations will be given during the exam, and no questions will be answered. If you think you have found an error in the problem statement, you should report on your answer sheet what you believe to be the error, and how you chose to interpret the intent of the question to recover from the alleged error.

You are allowed up to two A4 sheets of personal notes (i.e., four page sides), printed or written by hand, with font size of 10 points at most. If you use such personal notes, you must hand them in along with your answers. You may not use any other written material.

Write your name clearly on the top right of every sheet used for your exam answers. Number every page. Please use one sheet when answering Exercises 1 and 2, and a different sheet when answering Exercises 3 and 4.

The exam is strictly personal: any communication or influence between students, or use of outside help, is prohibited. No electronic devices such as calculators, computers, or mobile phones, are permitted. Any violation of the rules may result in a grade of 0 and/or disciplinary action.

Exercise 1: Number of Possible Worlds of c-tables (6 points)

Recall that a *c-table* is a relational table where tuples carry an *annotation*. The *annotation* is a Boolean formula on equalities and inequalities that involve constant values and variables (called *named nulls*). In this exercise, we consider c-tables where *nulls may only occur in annotations*, i.e., they do not occur elsewhere in the table.

Recall that a *valuation* of a c-table T is a function that maps each named null of the table to a value. Given a valuation ν of the c-table, we can apply it to the c-table by replacing each named null with its value in ν, evaluating the equalities and inequalities, evaluating the Boolean formulae, and keeping the tuples whose annotation evaluates to true. The result of this process is a relational table, called a *possible world* of T, and we say that it is *obtained* from ν. Remember that a given possible world may be obtained by several different valuations.

A *Boolean c-table* is a c-table whose nulls can only be replaced by two values, True and False; in other words, we only consider valuations whose domain is $\{\mathsf{True}, \mathsf{False}\}$. By contrast, a *general c-table* is a c-table where nulls can be replaced by arbitrary values. This exercise studies the *number of possible worlds* of c-tables.

We first consider Boolean c-tables. We give an example of a Boolean c-table T_1 below, which has two named nulls $\texttt{NULL}_1$ and $\texttt{NULL}_2$, and where two constant values (True and False) occur in the annotations:

T_1

att1	**att2**	
a	b	$\texttt{NULL}_1 = \mathsf{True}$
a	a	$\texttt{NULL}_2 = \mathsf{True} \vee \texttt{NULL}_1 = \mathsf{True}$
b	b	$\texttt{NULL}_1 = \mathsf{False}$

One example of a valuation is the function mapping $\texttt{NULL}_1$ to True and $\texttt{NULL}_2$ to False. The possible world obtained by this valuation is the relational table consisting of the first two rows of T_1 (without their annotation).

Question 1 (0.5 point). Give the possible worlds of T_1. For each possible world, give all the valuations that can be used to obtain that possible world. (No justification is required. There should be three possible worlds, and four valuations in total.)

Question 2 (0.5 point). Give an example of a Boolean c-table T_2 that contains 4 different named nulls and 4 tuples, and that has exactly 16 possible worlds. (There are multiple possible solutions. No justification is required. You do not need to write the 16 possible worlds explicitly.)

Question 3 (0.5 point). Give an example of a Boolean c-table T_3 that contains 4 different named nulls and 4 tuples, and that has exactly 1 possible world. (There are multiple possible solutions. No justification is required.)

Question 4 (0.5 point). Consider an arbitrary Boolean c-table T_4, and let $p \in \mathbb{N}$ be the number of tuples of T_4. Prove that the number of possible worlds of T_4 is at most 2^p. (We call this an *upper bound* on the number of possible worlds of T_4 as a function of p.)

Question 5 (1 point). Consider an arbitrary Boolean c-table T_5, and let $q \in \mathbb{N}$ be the number of different named nulls that occur in T_5. Express, as a function of q, the number of different valuations of T_5. Deduce an upper bound on the number of possible worlds of T_5 as a function of q.

We now move to general c-tables, i.e., c-tables where valuations can map named nulls to arbitrary values (but these nulls are still only allowed to occur in annotations). Given a general c-table T, we will write $q \in \mathbb{N}$ for the number of different named nulls that occur in the annotations of T, we will write $r \in \mathbb{N}$ for the number of different constant values that occur in the annotations of T, and we will write $w \in \mathbb{N}$ for the number of possible worlds of T.

Question 6 (1 point). Give an example of a general c-table T_6 with two named nulls ($q = 2$), one constant value ($r = 1$), and which has exactly 5 possible worlds ($w = 5$). (There are multiple possible solutions. No justification is required.)

Question 7 (1 point). Consider an arbitrary general c-table T_7 with only one named null ($q = 1$). Show that $w \leq r + 1$.

Question 8 (1 point). Consider an arbitrary general c-table T_8 where no constant values occur in annotations ($r = 0$). Show that $w \leq 2^{q^2}$.

Bonus question (optional). Explain why the bound of question 8 is not tight. Can you find a better bound (still assuming $r = 0$)?

Exercise 2: Games and Prizes (4 points)

We consider two relational tables given below: one table *Game*, indicating which games of chance you have won, and one table *Prize*, indicating the prizes at stake for each game. We further consider the query Q asking what are the prizes at stake for the games that you won: its result $Q(Game, Prize)$ on *Game* and *Prize* is also given below.

Game

game
Loto
Bingo

Prize

game	**prize**
Loto	Money
Bingo	Car
Bingo	Castle

$Q(Game, Prize)$

prize
Money
Car
Castle

Question 1 (1 point). Express the query Q in the relational algebra, and in the relational calculus.

Consider now the probabilistic version of the *Game* table given by the following tuple-independent database (TID), indicating the probability that you win each game:

*Game*2

game	
Loto	0.1
Bingo	0.3

Consider the probabilistic relation T defined by evaluating the query Q on the TID relation *Game*2 and the non-probabilistic relation *Prize* above, i.e., $T := Q(Game2, Prize)$. This relation describes the possible prizes that you will win, assuming the distribution of *Game*2 on the games that you win.

Question 2 (1 point). Give the four possible worlds of T and the probability of each possible world.

Question 3 (1 point). Show that T cannot be expressed as a TID table.

Question 4 (1 point). Remember that a *pc-table* is a Boolean c-table where named nulls occurs only in annotations (see Exercise 1), and each named null has a given probability of being true: we call the named nulls *variables*. Give a pc-table that expresses T, i.e., which represents the probabilistic instance T. (There are multiple possible solutions. No justification is required.)

Exercise 3: Lineages, Queries, and Probabilities (4 points)

Consider the following Boolean formula, where the variables $x_1, x_2, y_1, y_2, y_3, y_4$ represent independent probabilistic events:

$$\Phi = x_1y_3 \vee x_2y_4 \vee x_1y_1 \vee x_2y_2.$$

Question 1 (1 point). Remember that a read-once formula is a Boolean formula where each variable occurs at most once. Rewrite Φ to an equivalent read-once formula, Use this to compute the probability $\mathrm{P}(\Phi)$, as a function of the probabilities $\mathrm{P}(x_1), \mathrm{P}(x_2), \mathrm{P}(y_1), \mathrm{P}(y_2), \mathrm{P}(y_3), \mathrm{P}(y_4)$ of each variable.

In the rest of this exercise, we will work on pc-tables (see Exercise 2 for a reminder of what this is), whose variables are chosen among $x_1, x_2, y_1, y_2, y_3, y_4$. An *atomic* pc-table on variables $x_1, \ldots, x_m$ is a pc-table where each fact is annotated with a condition of the form $x_i = \mathsf{True}$ for some $i \in \{1, \ldots, m\}$, and where each condition is used at most once (so in particular there are at most m tuples). An *atomic*

pc-table on variables $y_1, \ldots, y_m$ is defined analogously. Remember that the *lineage* of a Boolean query on some pc-tables is a Boolean formula expressed on the variables of the pc-tables that precisely describes the conditions for which the query evaluates to true.

Question 2 (1 point). Consider the query $Q() := \exists x, y\ A(x) \wedge B(x, y)$, expressed in the relational calculus. Write an atomic pc-table A on variables x_1, x_2 and an atomic pc-table B on variables y_1, y_2, y_3, y_4 such that the lineage of Q on A and B is Φ.

Question 3 (2 points). Write Q in the relational algebra. Detail the steps of a *safe extensional plan* for the query Q and for the tables A and B from Question 2. What is the final probability formula and how does it compare to the one obtained in Question 1?

Exercise 4: Probabilistic Queries Continued (6 points)

For this exercise, we will consider variants of the H_0 query:

$$H_0() \ := \ \exists x, y\ R(x) \wedge S(x, y) \wedge T(y)$$

a query that is known to be hard to evaluate on TID instances. Moreover, we will consider databases that *conform to the structure* $\mathcal{S}$, which means the following:

- R is an atomic pc-table on $x_1, \ldots, x_m$ (see the definition in Exercise 3), containing m tuples, with one attribute $\mathbf{x}$.
- T is an atomic pc-table on $y_1, \ldots, y_n$, containing n tuples, with one attribute $\mathbf{y}$.
- S is a deterministic instance that has two attributes $\mathbf{x}$ and $\mathbf{y}$, and which contains *all* tuples of the form (a, b) such that a appears in a tuple of R and b appears in a tuple of T. Hence S always contains exactly $m \times n$ tuples.

An example of a database $\mathcal{D}$ that conforms to the structure $\mathcal{S}$ is given below, where $m = n = 2$:

R

x	
a	$x_1 = \mathsf{True}$
b	$x_2 = \mathsf{True}$

S

x	y
a	c
a	d
b	c
b	d

T

y	
c	$y_1 = \mathsf{True}$
d	$y_2 = \mathsf{True}$

Question 1 (1.5 points). First, we consider a non-Boolean form of H_0:

$$H_0'(x) \ := \ \exists y\ R(x) \wedge S(x, y) \wedge T(y)$$

and the database instance $\mathcal{D}$ above. Show the lineage formula for the Boolean query $H_0'(a)$, derive its probability formula and draw the resulting *read-once circuit*.

Question 2 (1.5 points). Show that, for any database $\mathcal{D}'$ that conforms to the structure $\mathcal{S}$, and for any element b that occurs in a tuple of R, the lineage for $H_0'(b)$ is always *read-once*. Write down a general formula for its probability, and draw the general form of its *read-once circuit*.

Question 3 (2 points). Moving back to the Boolean form of $H_0()$, explain why, *in general*, its lineage on database instances that conform to the structure $\mathcal{S}$ is *not read-once*.

Question 4 (1 point). Consider a database $\mathcal{D}''$ that conforms to the structure $\mathcal{S}$, with $m \geq 1$ and $n \geq 1$. Give a condition for $\mathcal{D}''$ for which $H_0()$ can be *read-once*. No justification needed.