Πgora: An Integration System for Probabilistic Data

Dan Olteanu[1] and Lampros Papageorgiou[1] and Sebastiaan J. van Schaik[1,2]

[1]*Department of Computer Science, University of Oxford, UK*
[2]*Oxford e-Research Centre, University of Oxford, UK*
{Dan.Olteanu,Lampros.Papageorgiou,Sebastiaan.van.Schaik}@cs.ox.ac.uk

***Abstract*—Πgora is an integration system for probabilistic data modelled using different formalisms such as pc-tables, Bayesian networks, and stochastic automata. User queries are expressed over a global relational layer and are evaluated by Πgora using a range of strategies, including data conversion into one probabilistic formalism followed by evaluation using a formalism-specific engine, and hybrid plans, where subqueries are evaluated using engines for different formalisms.**

This demonstration allows users to experience Πgora on real-world heterogeneous data sources from the medical domain.

I. Integrating Probabilistic Data

Real-world applications model probabilistic data using a plethora of different formalisms. The reason for this diversity stems from the fact that each formalism has its own benefits and better fits particular scenarios. We next focus on three such formalisms. Bayesian networks are a natural fit for managing expert knowledge, where the probabilistic relationship between input random variables, which are observable quantities, unknown parameters, or hypotheses, exhibits conditional independence. The pc-tables formalism excels at managing uncertain relational data, such as NELL tables [1], which consist of records extracted from hundreds of millions of web pages, and Google Squared tables that aggregate unstructured, possibly contradictory information representing answers to keyword web search queries [2]. Finite State Transducers (FSTs) are stochastic automata used by state-of-the-art optical character recognition programs, such as those powering Google Books, to capture probability distributions over all possible strings that could be represented in a given image [5].

These formalisms naturally support probabilistic processing to varying degrees. The pc-tables formalism supports select-project-join queries whose answers and their probabilities can be represented as pc-tables; Bayesian networks support inference queries that ask for the conditional probability of an event given another event; FSTs support selection queries that ask for the probability that a certain string occurs in their possible runs.

In order to harness the value of heterogeneous probabilistic data sources, it becomes imperative to provide a uniform interface to them. Such an interface would allow for their integration and enable expressive SQL-like querying across them. This is possible since their underlying formalisms have a common denominator: they all admit a sound interpretation via the possible worlds semantics [11]. Under this semantics, pc-tables, Bayesian networks, and FSTs represent finite probability distributions over sets of possible tables, sets of correlated observations, and respectively sets of possible strings represented in an image.

Our system Πgora (to be pronounced pi-gora: probabilistic agora) provides such a uniform interface over Bayesian networks, pc-tables, and FSTs. In addition, it provides a query evaluation mechanism over the interface, whose strategies take the native querying capabilities of the underlying formalisms into account, to devise an efficient query plan defined either by a sequence of sub-plans to be evaluated by engines for different formalisms, or by transformations of sources to one of the existing formalisms followed by evaluation using a single query engine. Further common aspects of data integration systems, such as declarative specifications of capabilities of each data source (in addition to those of their underlying formalisms), automatically rewriting the user query to use the views representing local sources [6], and dealing with many possible rewritings in case several sources publish similar or same data, are not yet considered by Πgora.

II. The Architecture of Πgora

Πgora presents a unifying relational interface over heterogeneous sources called *mediated schema*. The users phrase their queries over this mediated schema. The components of the system, as well as the data and control flow, are shown in Figure 1. The system works as follows. Each local source is registered to the system with a relational schema that becomes part of the mediated schema. The user queries are expressed as select-project-join SQL queries extended with an exact and approximate probability computation aggregate and with a *given* clause, which allows to formulate conditionals and ask for the probability of an event given another event. For instance, one could ask for each age group and sex how probable it is that a person suffers from diabetes and their diabetes medication causes exhaustion. This query joins (1) a Bayesian network[1] expressing probabilistic relationships between, among others, the existence of diabetes, age, sex, and pregnancy, (2) a NELL pc-table relating relationships between drugs and side effects, and (3) a NELL pc-table relating drugs and diseases. One could ask for the probability that a pregnant woman suffers from a left breast tumour given that she suffers from hypothyroidism. This query is a join of two Bayesian networks with relationships between age, breast cancer, pregnancy, and hypothyroidism.

[1]Available from the UCI Machine Learning Repository at http://archive.ics.uci.edu/ml/datasets.html.

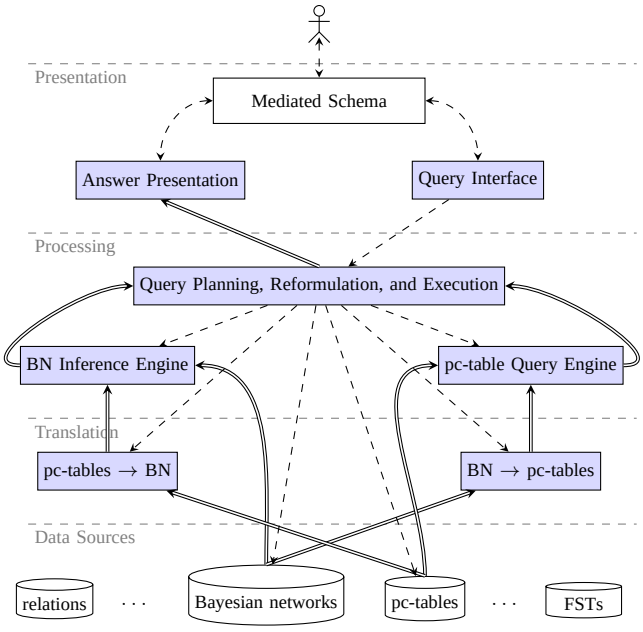

Fig. 1. The architecture of ΠΙgora. Double and dashed arrows denote data and control flow, respectively.

We next detail the components of the system.

Data sources. Each local source defines a relational schema that is part of the mediated schema. A pc-table is a relation extended with a special column that encodes the uncertainty of the records using probabilistic events, which are propositional formulas over random variables. Tuple correlations can be encoded by such events. A special case is that of tuple-independent pc-tables, where the events are pairwise independent. An excerpt of the NELL pc-table relating drugs and diseases is given below:

Drug	Disease	P	E
avandia	diabetes	1	x_1
tamoxifen	breast_cancer	1	x_2
actos	diabetes	0.998	x_3
glucophage	diabetes	0.996	x_4

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph [9]. Figure 2 shows a Bayesian network representing dependencies between diabetes and age, sex, and pregnancy (taken from the UCI ML Repository). Its relational schema consists of one attribute per node in the network. The network thus corresponds to a relation Diabetes with attributes `age`, `diabetes`, `sex`, `pregnant`, and so on; its schema is that of the relation representing the join of all conditional probability tables in the network.

A finite state transducer is an automaton that converts strings from an input alphabet to an output alphabet and where there is a probability distribution over the transitions from each node.

We use MayBMS to manage pc-tables [4]. FSTs are managed by Staccato [5]. Bayesian networks are represented in the XML-based BayesNets Interchange Format[2].

[2] http://www.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/

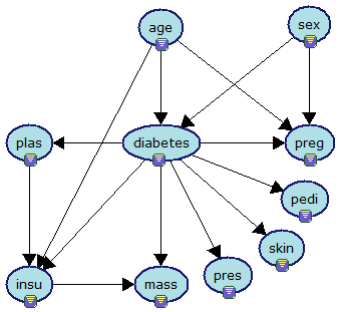

Fig. 2. Bayesian network for diabetes from the UCI machine learning repository (the conditional probability tables at nodes are not shown).

Translation layer. The possible worlds semantics represents a bridge between the different formalisms and enables sound, equivalence-preserving translations between their instances [8]. These translations are needed when ΠΙgora's query evaluation strategy requires to translate all sources into one formalism and executes the query using one dedicated engine.

We next exemplify the translation into pc-tables of the nodes `age`, `sex`, and `diabetes` from the Bayesian network in Figure 2. The nodes correspond to Boolean random variables (random variables with a domain of size n can be expressed by n Boolean variables); for space reasons, we assume that `age` is Boolean: below 40 and above 40. The conditional probability tables of the nodes are as follows:

Sex	P
Male	0.48
Female	0.52

Age	P
≤ 40	0.314
> 40	0.686

Age	Sex	Diabetes	P
≤ 40	Female	true	0.349
≤ 40	Female	false	0.651
≤ 40	Male	true	0.255
≤ 40	Male	false	0.745
> 40	Female	true	0.355
> 40	Female	false	0.645
> 40	Male	true	0.249
> 40	Male	false	0.751

Each conditional probability table is translated into an equivalent pc-table, where the conditional probabilities as well as the dependencies of the node to its ancestors in the network are captured by probabilistic events:

Sex	P	E_{sex}
Male	0.48	V^0_{sex}
Female	0.52	$\neg V^0_{sex}$

Age	P	E_{age}
≤ 40	0.314	V^0_{age}
> 40	0.686	$\neg V^0_{age}$

Age	Sex	Diabetes	P	$E_{diabetes}$
≤ 40	Female	true	0.349	$V^0_{age} \wedge V^0_{sex} \wedge V^0_{diabetes}$
≤ 40	Female	false	0.651	$V^0_{age} \wedge V^0_{sex} \wedge \neg V^0_{diabetes}$
≤ 40	Male	true	0.255	$V^0_{age} \wedge \neg V^0_{sex} \wedge V^1_{diabetes}$
≤ 40	Male	false	0.745	$V^0_{age} \wedge \neg V^0_{sex} \wedge \neg V^1_{diabetes}$
> 40	Female	true	0.355	$\neg V^0_{age} \wedge V^0_{sex} \wedge V^2_{diabetes}$
> 40	Female	false	0.645	$\neg V^0_{age} \wedge V^0_{sex} \wedge \neg V^2_{diabetes}$
> 40	Male	true	0.249	$\neg V^0_{age} \wedge \neg V^0_{sex} \wedge V^3_{diabetes}$
> 40	Male	false	0.751	$\neg V^0_{age} \wedge \neg V^0_{sex} \wedge \neg V^3_{diabetes}$

At each node, for each combination of values of the parent nodes, we introduce a Boolean random variable to capture the conditional probability distribution at that node. For instance,

the Boolean random variable V^0_{age} encodes whether the age is under 40 or over 40, and its probability distribution is that of the node `age`. The first entry in the conditional probability table for the node `diabetes` states that there is a diabetes probability of 0.349 for females under 40. This dependency is encoded by the conjunction $V^0_{age} \wedge V^0_{sex} \wedge V^0_{diabetes}$, where the new Boolean random variable $V^0_{diabetes}$ captures the probability distribution of the node `diabetes` in the network given that `sex` is female and `age` is under 40.

This translation can lead to an exponential blowup for networks that are not tree-shaped. We mitigate this problem by allowing *let definitions* in our pc-table formalism, whereby events can be named and re-used several times.

Inference and Query Engines. ΠgΟra is implemented in Java on top of the probabilistic management system MayBMS [4] with the SPROUT query engine [7] for queries on pc-tables, SMILE [3] for Bayesian inference, and Staccato for selection queries on FSTs [5].

Query Planning, Reformulation, and Execution. The task of this component is to decide how to evaluate the user query. ΠgΟra uses two broad strategies for query evaluation. These are exemplified in the next section.

The default strategy is to identify subqueries that are naturally supported by the formalisms of the sources referenced in these subqueries, *i.e.*, inference queries for Bayesian networks, selection queries for finite state transducers, and select-project-joins for pc-tables. We then use the engines associated with the formalism of the data sources to answer the subqueries. All remaining processing steps, e.g., joining subqueries that are evaluated using different query engines, are supported by translation to pc-tables. Besides the identification of such subqueries, a further challenge of this strategy is thus to compute the final query result using the results of the subqueries.

A further strategy is to (possibly, offline) convert all data sources used by the query into either the pc-tables or Bayesian networks formalism, followed by evaluation using either a query or an inference engine respectively, after reformulating the query over the corresponding formalism. A possible optimisation is to specialise the query to only use those parts of the data sources that are needed for the evaluation. For instance, in case only a few nodes of a Bayesian network are needed for the evaluation, we could rewrite the query to only use their corresponding pc-tables. For relational evaluation, if the query has a conditional clause, we can then rewrite it, by following the definition of conditional probability, into a query for numerator and one for denominator, and a third query that uses the results of the first two queries to compute the final result. For evaluation via Bayesian inference, the user query is rewritten into a sequence of inference queries that are optimised by identifying independence and temporary results that can be reused several times.

III. Demonstration Scenario: Medical Data

We demonstrate ΠgΟra using real-world data sources in the medical domain from NELL, the UCI machine learning repository, and a repository of FSTs from Google Books.

Let us consider the query in Figure 3 (left). It asks for the probability (note the construct `conf()` in the select clause) of a pregnant woman suffering from a left breast tumour, given that she also suffers from hypothyroidism. Correlations between the two medical conditions are reported in literature [10]. Possible data sources are the Bayesian networks `Hypothyroid` and `Breast_cancer`. We join these sources on `age`, since both conditions depend on age.

ΠgΟra chooses a purely Bayesian evaluation, since both data sources are Bayesian networks. In this case, we phrase the SQL query as a sum of inference queries:

$$\sum_{\text{B.age}} (P(\text{B.tumor = true } \wedge \text{ B.breast = left } \wedge \text{ H.tumor = true } \wedge \text{ H.pregnant = true } | \text{ (B.age = H.age } \wedge \text{ H.hypothyroid = primary)))}$$

For a given value x for `age`, we have the inference query:

$$P(\text{B.tumor = true } \wedge \text{ B.breast = left } \wedge \text{ H.tumor = true } \wedge \text{ H.pregnant = true } | \text{ (B.age} = x \text{ } \wedge \text{ H.age} = x \text{ } \wedge \text{ H.hypothyroid = primary))}$$

Since the two Bayesian networks are independent, we can regroup as follows:

$$P(\text{B.tumor = true } \wedge \text{ B.breast = left } | \text{ B.age} = x) * P(\text{H.tumor = true } \wedge \text{ H.pregnant = true } | \text{ (H.age} = x \text{ } \wedge \text{ H.hypothyroid = primary))}$$

Next, we present two further possible scenarios. Figure 3 (right) depicts the plan for relational evaluation. We first apply the conditional probability formula: $P(A \,|\, B) = \frac{P(A \wedge B)}{P(B)}$. Expressed in SQL, this means that for each `age` value, we compute (1) the probability of the query with a conjunction of the conditions in the `where` and `given` clauses, (2) the probability of the query representing only the `given` clause, (3) the division of the two probabilities, and finally (4) sum up the probabilities of all `age` values. Further optimisations are applicable, such as specialising the relations `Hypothyroid` and `Breast_cancer` to those constituent pc-tables strictly needed for query evaluation.

Let us now assume that `Breast_cancer` is a pc-table. The default strategy would then split the query into the subquery referring to the `Hypothyroid` network and the subquery referring to the `Breast_cancer` relation.

For each value of x for `age` we have the inference query:

$$\forall x : P_H(x) = P(\text{H.tumor = true } \wedge \text{ H.pregnant = true } | \text{ H.age } = x \text{ } \wedge \text{ H.hypothyroid = primary})$$

The subquery that refers to Breast_cancer is now rewritten following the conditional probability formula:

create table T_1 **as select** B.age, conf() as p1
from Breast_cancer B
where B.tumor='true' **and** B.breast='left' **group by** B.age;

create table T_2 **as select** B.age, conf() as p2
from Breast_cancer B **group by** B.age;

create table T_3 **as select** T_1.age, p1/p2 as P_B
from T_1, T_2 **where** T_1.age = T_2.age;

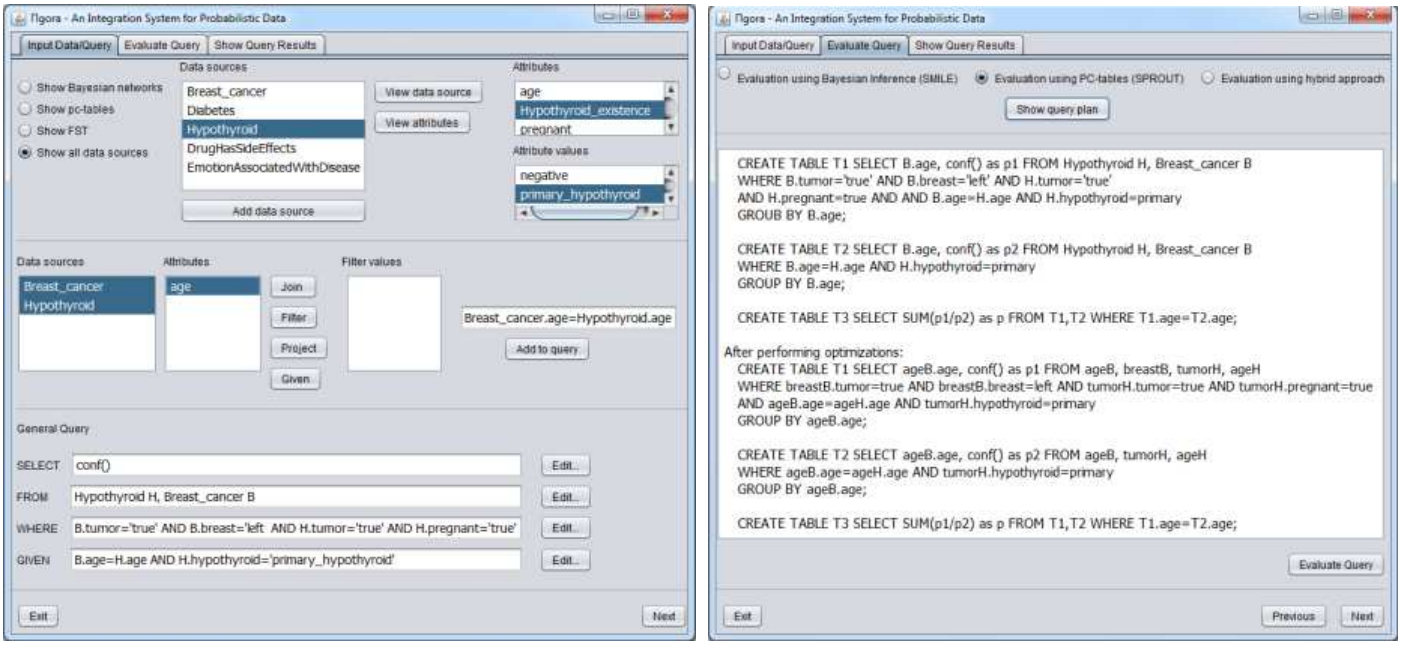

Fig. 3. IIgora Graphical User Interface: Input data and query (left) and Visualisation of the chosen evaluation strategy (right).

We finally obtain the query answer by joining the independent intermediate results P_H and T_3: $\sum_{age} P_B(age) * P_H(age)$, where $P_B(age)$ denotes P_B for the tuple (age, P_B) in T_3.

We note that one further data source for this query could be a collection of finite state transducers modelling possible strings represented in images of scanned book pages referring to these medical conditions. Then, further evidence of correlation between these medical conditions can be signalled by multiple co-occurrences of the names of the two conditions within paragraphs on the same pages, and hence by large co-occurrence probability.

IV. User Interaction

The users can interact with IIgora via its graphical user interface that allows to view and register data sources, compose and execute queries, inspect query evaluation strategies, and see query results.

Figure 3 depicts two screen shots of IIgora at work. The left screen shot corresponds to the input data and query tab: it presents a list of data sources and shows how the user can compose queries.

The right screen shot corresponds to the evaluation tab, where the user can choose one of the supported evaluation strategies, *i.e.*, evaluation via pc-tables, via Bayesian inference, or the default mixed evaluation where formalism-specific engines are used to evaluate subqueries of the input query. This tab also depicts the execution plan chosen by the system to evaluate the input query.

The right screen shot also depicts a relational plan used by a strategy based on pc-tables to evaluate the query mentioned in the previous section. The plan consists of several relational queries that together encode the conditioning expressed in the original query. It is optimised such that it only refers to those pc-tables obtained by translating the nodes in the input Bayesian networks for hypothyroidism and breast cancer that are necessary to express the query.

The visitors of our demonstration will also be encouraged to inspect the translations of Bayesian networks into pc-tables, which are of independent interest.

Acknowledgement

This research was partially supported by ERC grant agreements FOX number 233599 and HiPerDNO number 248235, and EPSRC grant agreement ADEPT EP/I000194/1.

References

[1] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. J. Hruschka, and T. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
[2] D. Crow. Google Squared web scale, open domain information extraction and presentation. In *ECIR*, 2010.
[3] M. Druzdzel. SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In *AIII*, 1999.
[4] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: A probabilistic database management system. In *SIGMOD*, 2009.
[5] A. Kumar and C. Ré. Probabilistic management of OCR data using an RDBMS. *PVLDB*, 5(4), 2011.
[6] I. Manolescu, D. Florescu, D. Kossmann, F. Xhumari, and D. Olteanu. Agora: Living with XML and relational. In *VLDB*, 2000.
[7] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, 2010.
[8] L. Papageorgiou. IIgora: An integration system for probabilistic data. Master's thesis, University of Oxford, 2012.
[9] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1989.
[10] P. P. Smyth. The thyroid, iodine and breast cancer. *Breast Cancer Res.*, 5(5), 2003.
[11] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.