SPROUT^2: A Squared Query Engine for Uncertain Web Data

Robert Fink[1], Andrew Hogue[2], Dan Olteanu[1], Swaroop Rath[1]

[1]Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK [2]Google Inc., New York, NY, USA

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF OXFORD

SPROUT^2 — Structured Queries over Unstructured Data

Challenges

- Inherently uncertain Web data: NELL, Google Squared
- Scalable evaluation of any relational algebra query on expressive uncertain databases of Web data

Contributions

- Database wrappers for Web data
  - Probabilistic databases from Google Squared, NELL
  - Deterministic databases from Google Fusion
- Scalable query processing for any relational algebra query
  - Exact/Approximate evaluation for tractable/hard queries
  - Done by the SPROUT query engine (part of MayBMS)
- Graphical user interface in SPROUT^2's for selecting data sources, composing queries, and browsing results

Uncertain Data on the Web

- Traditional databases: Data must adhere to rigid structure and must be complete and trustworthy.
- Internet applications witness an unprecedented shift towards unstructured and user-generated content.
- Example scenario: Business Intelligence
  - Extract valuable business information from sources such as twitter feeds, blogs, or email messages.
  - Join those data with offline databases, e.g. on products
  - .. to obtain early feedback about a product's quality.

Selected Publications on SPROUT and SPROUT^2

- **SPROUT^2: A Squared Query Engine for Uncertain Web Data.** R. Fink, D. Olteanu, S. Rath. In *SIGMOD*, 2011.
- **On the Optimal Approximation of Queries Using Tractable Propositional Languages.** R. Fink, D. Olteanu. In *ICDT*, 2011.
- **Providing Support for Full Relational Algebra in Probabilistic Databases.** R. Fink, D. Olteanu, S. Rath. In *ICDE*, 2011.
- **Approximate Confidence Computation for Probabilistic Databases.** D. Olteanu, J. Huang, C. Koch. In *ICDE*, 2010.
- **Secondary-Storage Confidence Computation for Conjunctive Queries with Inequalities.** D. Olteanu, J. Huang. In *SIGMOD*, 2009.
- **SPROUT: Lazy vs. Eager Query Plans for Tuple-independent Probabilistic Databases.** Olteanu, Huang, Koch. In *ICDE*, 2009.
- **Using OBBDs for Efficient Query Evaluation on Probabilistic Databases.** D. Olteanu, J. Huang. In *SUM*, 2008.

New Book on Probabilistic Databases

Probabilistic Databases. D. Suciu, D. Olteanu, C. Ré, C. Koch. May 2011. Morgan & Claypool Publishers.

Query Answering in SPROUT

Query evaluation in two steps (logically distinct but possibly intertwined):

1. Compute the tuples in the query result together with their lineage

   The lineage of a result tuple t is a propositional formula over the tuples in the input database and says which input tuples must be present in order for the query to return t.

2. Compute the probabilities of result tuples by incremental lineage compilation:

Independent or (Φ and Ψ are syntactically independent)	$P(\Phi \vee \Psi) = 1 - (1 - P(\Phi))(1 - P(\Psi))$
Independent and (Φ and Ψ are syntactically independent)	$P(\Phi \wedge \Psi) = P(\Phi) \cdot P(\Psi)$
Shannon expansion (x is a variable in Φ)	$P(\Phi) = P(x)P(\Phi\|_x) + P(\neg x)P(\Phi\|_{\neg x})$

Example: Relational division query "Which supplier stocks all products?"

Supplier S

Supplier	Product	E
1	1	x_1
1	2	x_2
1	7	x_3
2	1	x_4
2	5	x_5

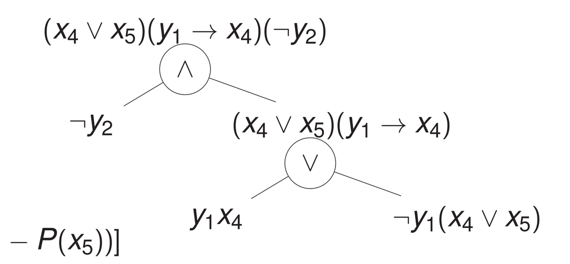
Product P

Product	E
1	y_1
2	y_2

Supplier of all products: $S \div P$

Supplier	E
1	$(x_1 \vee x_2 \vee x_3)(y_1 \rightarrow x_1)(y_2 \rightarrow x_2)$
2	$\mathbf{(x_4 \vee x_5)(y_1 \rightarrow x_4)(\neg y_2)}$

$$P((x_4 \vee x_5)(y_1 \rightarrow x_4)(\neg y_2)) = P(\neg y_2)[P(y_1)P(x_4) + P(\neg y_1)[1 - (1 - P(x_4))(1 - P(x_5))]$$



Novel Techniques for Exact and Approximate Probability Computation

- Incremental decomposition of lineage into d-trees using the above three rules
- After each decomposition step, compute rough lower and upper bounds on the probabilities of the residual formulas at the leafs of the decomposition tree
  - Approach 1: Lower bound is the largest probability of a clause in Φ; Upper bound is the sum of probabilities of all clauses in Φ.
  - Approach 2: Compute read-once formulas, whose probabilities represent lower and upper bounds.
- Using the bounds at the leaves, compute lower and upper bounds for the whole lineage
- Stop when the desired precision is reached or the time budget is exhausted
- Underlying idea: Leaves deeper in the d-tree contribute little to the overall probability mass, hence a good approximation can be found quickly

Complete decomposition

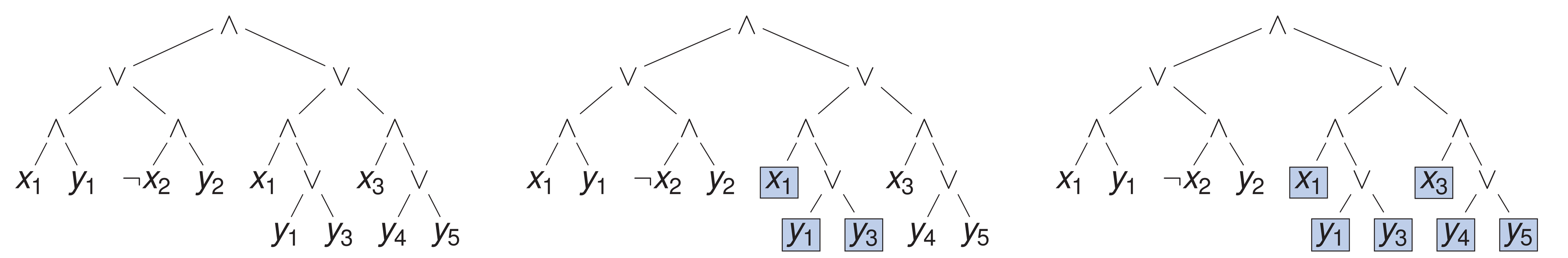
- Corresponds to exact probability computation
- Can be done in polynomial time for tractable query & data instances
  - relational algebra queries without repeating symbols and with read-once lineage
  - a class of conjunctive queries with inequality ($<, \neq$) joins

Partial decomposition

- Corresponds to approximate probability computation with error guarantees
- Applicable for hard query and data instances

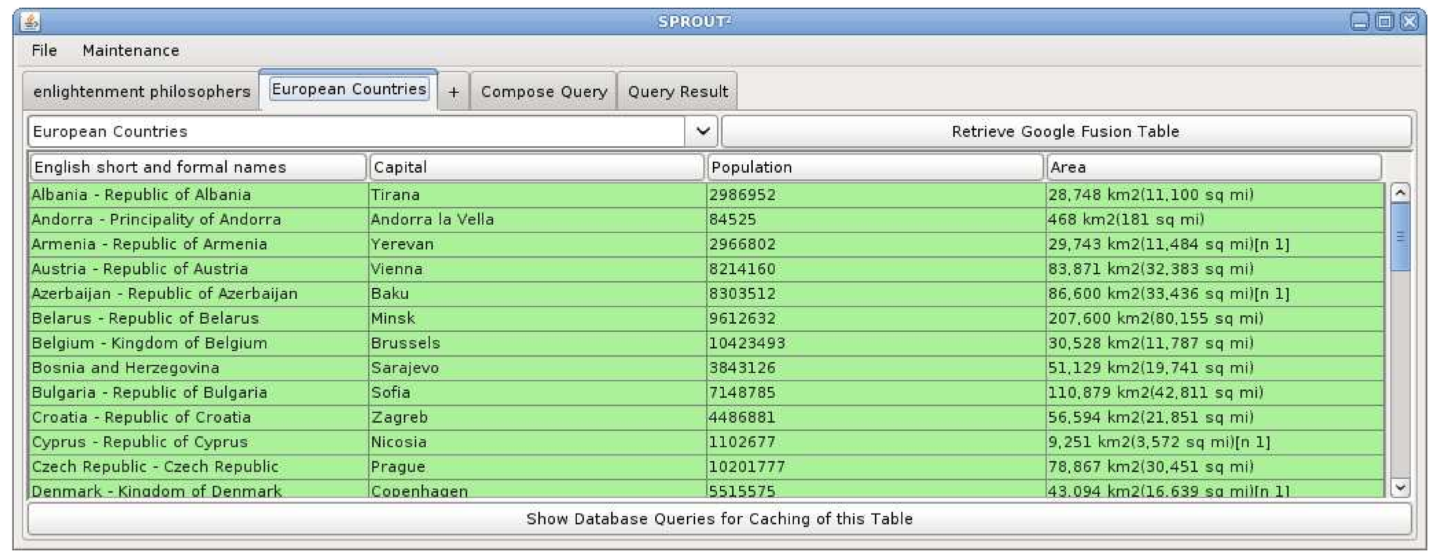
Example: Efficient computation of bounds that are read-once formulas

- Left: original formula; middle: lower bound; right: upper bound
- Lower/Upper bounds obtained by setting the marked literals to false/true

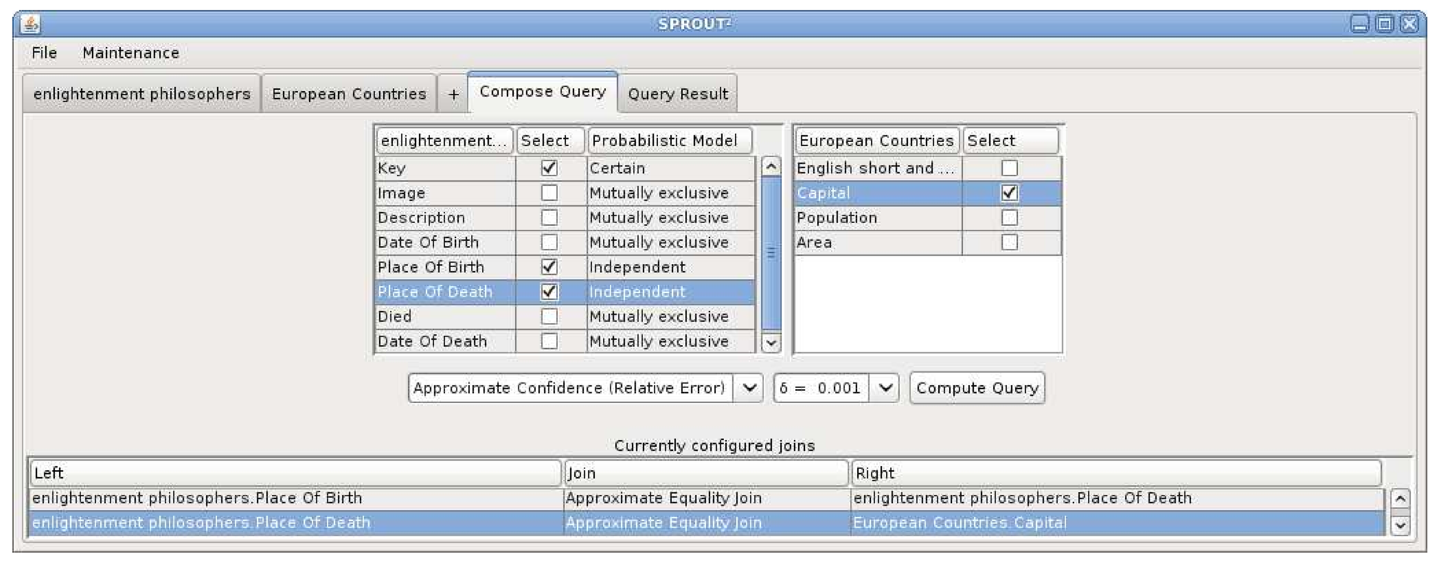


$(x_1y_1 \vee \neg x_2y_2)[x_1(y_1 \vee y_3) \vee x_3(y_4 \vee y_5)]$ Lower bound $(x_1y_1 \vee \neg x_2y_2)x_3(y_4 \vee y_5)$ Upper bound $x_1y_1 \vee \neg x_2y_2$

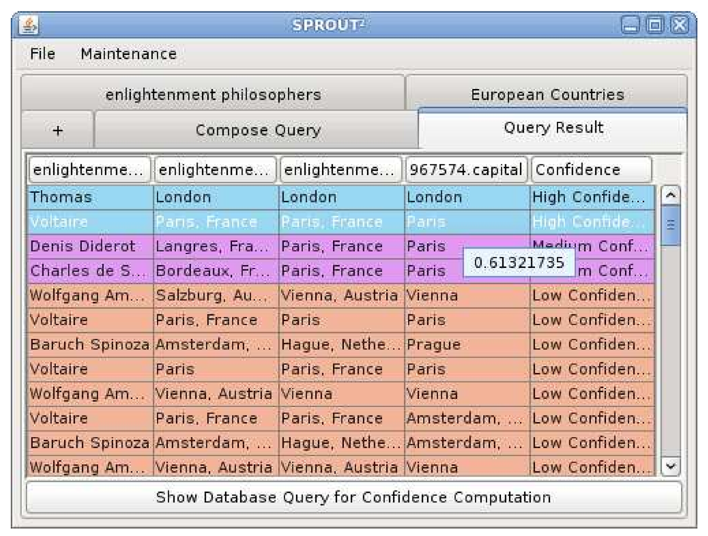
SPROUT^2: Selecting Web or Offline Data Sources

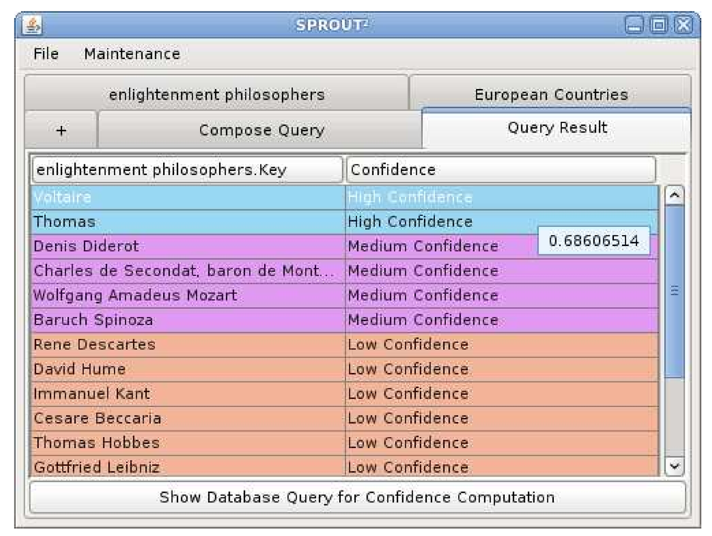


SPROUT^2: Composing Queries over Selected Sources



SPROUT^2: Browsing Ranked Query Results





Publicly-available Prototype

- SPROUT is implemented as an extension of PostgreSQL 8.4 backend
- Public version downloadable with MayBMS from CVS at `maybms.sourceforge.net`