Deterministic Decomposable Circuits for Safe Queries

Mikaël Monet, Dan Olteanu

Probabilistic Query Evaluation

Tuple Independent Database (D,π):

S		
a	a	.5
b	c	.2

represents

.5 × .2	.5 × (1 - .2)	(1 - .5) × .2	(1 - .5) × (1 - .2)
S	S	S	S
a a	a a	b c	
b c			

Probabilistic Query Evaluation PQE(q):

- **INPUT: TID (D,π)**
- **OUTPUT:** probability that **(D,π)** satisfies **q**

Pr($(D,\pi) \models \exists x,y\ S(x,y) \wedge x \neq y$**)** = **.5 × .2 + (1 - .5) × .2**

For **Unions of Conjunctive Queries**, [Dalvi & Suciu] imply:

- There is a class of **safe** UCQs such that

→ PQE(q) is **PTIME** if q is **safe**

→ PQE(q) is **#P-hard** q is **not safe**

Two Techniques to Compute the Probability

Independence:

- **Pr(**$q_1 \wedge q_2$**)** = **Pr(**q_1**)** × **Pr(**q_2**)**
- **Pr(**$q_1 \vee q_2$**)** = 1- (1-**Pr(**q_1**)**) × (1-**Pr(**q_2**)**)

When q_1 and q_2 depend on disjoint sets of tuples

Inclusion-Exclusion:

- **Pr(**$q_1 \wedge q_2 \wedge q_3$**)** = **Pr(**$q_1$**)** + **Pr(**$q_2$**)** - **Pr(**$q_1 \vee q_2$**)** - **Pr(**$q_2 \vee q_3$**)** - **Pr(**$q_1 \vee q_3$**)** + **Pr(**$q_1 \vee q_2 \vee q_3$**)**

Works for any safe query [Dalvi & Suciu]

Independence:

- **Pr(**$q_1 \wedge q_2$**)** = **Pr(**q_1**)** × **Pr(**q_2**)**
- **Pr(**$q_1 \vee q_2$**)** = 1- (1-**Pr(**q_1**)**) × (1-**Pr(**q_2**)**)

Determinism:

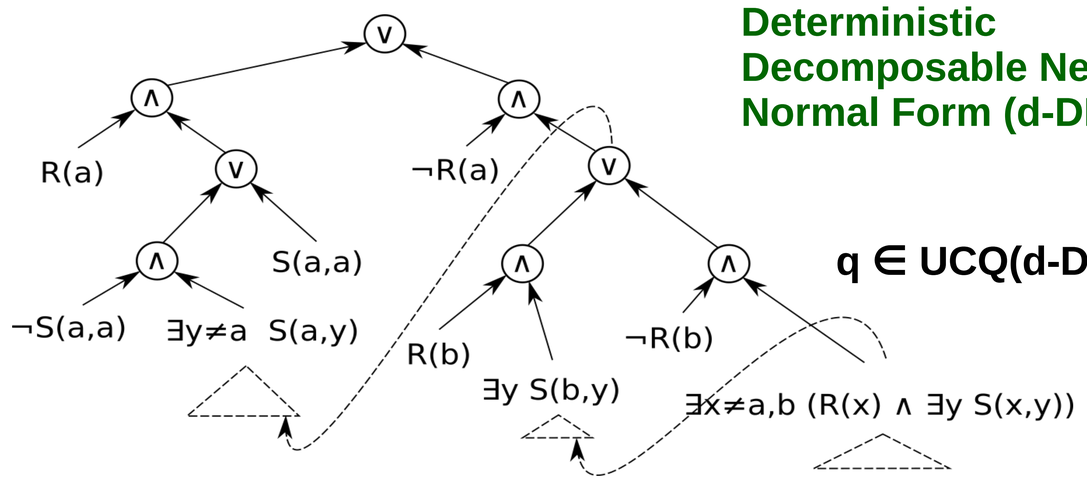
- **Pr(**$q_1 \vee q_2$**)** = **Pr(**q_1**)** + **Pr(**q_2**)**

When q_1 and q_2 are mutually exclusive

Question: does this work for any safe query?

Example: q = ∃ x,y R(x) ∧ S(x,y)

- $q = \bigvee_{a \in Dom(D)} [R(a) \wedge \exists y\ S(a,y)]$
- **Pr(**q**)** = $1 - \prod_{a \in Dom(D)} [1 - \mathbf{Pr}(R(a) \wedge \exists y\ S(a,y))]$
- **Pr(**R(a) ∧∃y S(a,y)**)** = **Pr(**R(a)**)** × **Pr(**∃y S(a,y)**)**
- **Pr(**∃y S(a,y)**)** = $1 - \prod_{b \in Dom(D)} [1 - \mathbf{Pr}(S(a,b))]$



Deterministic Decomposable Negation Normal Form (d-DNNF)

q ∈ UCQ(d-DNNF)

Problem

Safe UCQs $\stackrel{?}{=}$ UCQ(d-DNNF)

The H_k query classes

Fix k ≥ 1, define:

- 0 = ∃x,y R(x) ∧S(x,y)
- i = ∃x,y S_i(x) ∧S_{i+1}(x,y) for 1≥ i ≥k
- k = ∃x,y S_k(x,y) ∧T(y)

H_k query class: monotone Boolean combination of the queries {0,1,…,k}

- Example : take k = 3 and define q' = (2∨3) ∧ (1∨3) ∧ (0∨3) ∧ (0∨1∨2)
- **Pr(**q'**)** = **Pr(**2∨3**)** + **Pr(**1∨3**)** +**Pr(**0∨3**)** +**Pr(**0∨1∨2**)**
  - **Pr(**2∨3∨1**)** - **Pr(**2∨3∨0**)** - ~~**Pr(**2∨3∨0∨1**)**~~ - **Pr(**1∨3∨0**)** - ~~**Pr(**1∨3∨0∨2**)**~~ - ~~**Pr(**0∨3∨1∨2**)**~~
  + ~~**Pr(**2∨3∨1∨0**)**~~ + ~~**Pr(**2∨3∨0∨1**)**~~ + ~~**Pr(**2∨3∨0∨1**)**~~ + ~~**Pr(**1∨3∨0∨2**)**~~
  - ~~**Pr(**2∨3∨1∨0**)**~~

If the term **0∨1∨…∨k** cancels out, the query is **safe**, otherwise it is **not safe**

If the Boolean function representing the query does not depend on all the variables {0,..,k}, then the query is in **UCQ(d-DNNF)**

Our Method

We **rewrite** q' as $q_0 \vee q_1 \vee q_2 \vee q_3$, where :

- q_0 = 0 ∧ ¬2 ∧ 3 which is in UCQ(d-DNNF)
- q_1= ¬1 ∧ 2 ∧ 3 which is in UCQ(d-DNNF)
- q_2 = ¬0 ∧ 1 ∧ 3 which is in UCQ(d-DNNF)
- q_3 = 0 ∧ 1 ∧ 2 which is in UCQ(d-DNNF)

The disjunctions are deterministic, hence q' is in UCQ(d-DNNF)

We say that q' is **nice**

Results

- We **generated** all the non-trivial safe queries in H_k for 1 ≥k ≥6 (about 20M queries)
- We determined if they are **nice** using a **sat-solver**
- Turns out they are almost all nice, so in **UCQ(d-DNNF) !**
- The 69 queries that are not nice are **co-nice** (their negation is nice), so they are in **UCQ(d-D)**
- This is encouraging experiments in favor of Safe UCQs = UCQ(d-D)
- Could the queries that are not nice but co-nice separate UCQ(d-DNNF) from UCQ(d-D) ?