Deterministic Decomposable Circuits for Safe Queries

Mikaël Monet[1,2] and Dan Olteanu[3]

May 26th, 2018

[1]LTCI, Télécom ParisTech, Université Paris-Saclay; Paris, France

[2]Inria Paris; Paris, France

[3]University of Oxford; Oxford, UK

Tuple-independent databases (TID)

- Probabilistic databases: model uncertainty about data
- Simplest model: tuple-independent databases (TID)
  - A relational database D
  - A probability valuation π mapping each fact of D into $[0, 1]$
- Semantics of a TID (D, π): a probability distribution on 2^D:
  - Each fact $F \in D$ is either present or absent with probability $\pi(F)$
  - Assume independence across facts

$\rightarrow$ For $D' \subseteq D$, $\Pr(D') = (\prod_{F \in D'} \pi(F)) \times (\prod_{F \in D \setminus D'} (1 - \pi(F)))$

Tuple-independent databases (TID)

$D =$

S	
a	a
b	c

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$.5 \times .2$

S	
a	a
b	c

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$.5 \times .2$

S	
a	a
b	c

$.5 \times (1 - .2)$

S	
a	a

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$.5 \times .2$

S	
a	a
b	c

$.5 \times (1 - .2)$

S	
a	a

$(1 - .5) \times .2$

S	
b	c

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$.5 \times .2$

S	
a	a
b	c

$.5 \times (1 - .2)$

S	
a	a

$(1 - .5) \times .2$

S	
b	c

$(1 - .5) \times (1 - .2)$

S

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \land x \neq y$

$.5 \times .2$

S	
a	a
b	c

$.5 \times (1 - .2)$

S	
a	a

$(1 - .5) \times .2$

S	
b	c

$(1 - .5) \times (1 - .2)$

S

$\Pr(D \models q) =$

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

✓

$.5 \times (1 - .2)$

S	
a	a

$(1 - .5) \times .2$

S	
b	c

$(1 - .5) \times (1 - .2)$

S

$\Pr(D \models q) =$

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

$.5 \times (1 - .2)$

S	
a	a

$(1 - .5) \times .2$

S	
b	c

$(1 - .5) \times (1 - .2)$

S	

$\Pr(D \models q) = .5 \times .2$

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

✓

$.5 \times (1 - .2)$

S	
a	a

✗

$(1 - .5) \times .2$

S	
b	c

$(1 - .5) \times (1 - .2)$

S

$\Pr(D \models q) = .5 \times .2$

$(D, \pi) =$

S		
a	a	$.5$
b	c	$.2$

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

✓

$.5 \times (1 - .2)$

S	
a	a

✗

$(1 - .5) \times .2$

S	
b	c

✓

$(1 - .5) \times (1 - .2)$

S

$\Pr(D \models q) = .5 \times .2$

Tuple-independent databases (TID)

$(D, \pi) =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

✓

$.5 \times (1 - .2)$

S	
a	a

✗

$(1 - .5) \times .2$

S	
b	c

✓

$(1 - .5) \times (1 - .2)$

S

$\Pr(D \models q) = .5 \times .2 + (1 - .5) \times .2$

$(D, \pi) =$

S		
a	a	$.5$
b	c	$.2$

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$

S	
a	a
b	c

✔

$.5 \times (1 - .2)$

S	
a	a

✘

$(1 - .5) \times .2$

S	
b	c

✔

$(1 - .5) \times (1 - .2)$

S

✘

$\Pr(D \models q) = .5 \times .2 + (1 - .5) \times .2$

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation PQE(q):

- **INPUT:** a TID (D, π)

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation PQE(q):

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

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation PQE(q):

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

Dichotomy for UCQs **[Dalvi & Suciu, 2012]:**

- There is a class of safe queries $\subseteq$ UCQs

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation PQE(q):

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

Dichotomy for UCQs **[Dalvi & Suciu, 2012]:**

- There is a class of safe queries $\subseteq$ UCQs

$\rightarrow$ q is **safe** $\implies$ PQE(q) is PTIME

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation $\mathrm{PQE}(q)$:

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

Dichotomy for UCQs **[Dalvi & Suciu, 2012]:**

- There is a class of safe queries $\subseteq$ UCQs

$\rightarrow$ q is **safe** $\implies$ $\mathrm{PQE}(q)$ is PTIME

$\rightarrow$ q is **not safe** $\implies$ $\mathrm{PQE}(q)$ is **#P-hard**

Probabilistic query evaluation (PQE)

Let us fix a Boolean query q (e.g., CQ, FO, Datalog...)

Probabilistic query evaluation PQE(q):

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

Dichotomy for UCQs **[Dalvi & Suciu, 2012]:**

- There is a class of safe queries $\subseteq$ UCQs

$\rightarrow$ q is **safe** $\implies$ PQE(q) is PTIME

$\rightarrow$ q is **not safe** $\implies$ PQE(q) is **#P-hard**

(In the rest of this talk, we will asume PTIME $\neq$ **#P**)

Algorithm for safe queries

In essence, the algorithm of **[Dalvi & Suciu, 2012]** uses:

Algorithm for safe queries

In essence, the algorithm of **[Dalvi & Suciu, 2012]** uses:

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

Algorithm for safe queries

In essence, the algorithm of **[Dalvi & Suciu, 2012]** uses:

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2)]$

Algorithm for safe queries

In essence, the algorithm of **[Dalvi & Suciu, 2012]** uses:

- Independence

→ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

→ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2)]$

- Inclusion-Exclusion

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

Algorithm for safe queries

In essence, the algorithm of **[Dalvi & Suciu, 2012]** uses:

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2)]$

- Inclusion-Exclusion

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

This works for any safe query!

Alternative Approach

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2))]$

Alternative Approach

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2))]$

- Mutual-exclusiveness

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = \Pr(D \models q_1) + \Pr(D \models q_2)$

Alternative Approach

- Independence

$\rightarrow$ $\Pr(D \models q_1 \wedge q_2) = \Pr(D \models q_1) \times \Pr(D \models q_2)$

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = 1 - [(1 - \Pr(D \models q_1)) \times (1 - \Pr(D \models q_2)]$

- Mutual-exclusiveness

$\rightarrow$ $\Pr(D \models q_1 \vee q_2) = \Pr(D \models q_1) + \Pr(D \models q_2)$

Does this work for all the safe queries?

Question

Independence $+$ **Inclusion-exclusion**

$$\stackrel{?}{=}$$

Independence $+$ **mut.-exclusiveness**

Example: independence

- $q := \exists x, y \; R(x) \wedge S(x, y)$

Example: independence

- $q := \exists x, y\ R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y\ S(a, y))$

Example: independence

- $q := \exists x, y \; R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y \; S(a, y))$

Example: independence

- $q := \exists x, y \; R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y \; S(a, y))$

$\rightarrow$ $\Pr(D \models q) = 1 - \prod_{a \in \mathrm{Dom}(D)} (1 - \Pr(D \models R(a) \wedge \exists y \; S(a, y)))$

Example: independence

- $q := \exists x, y\ R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y\ S(a, y))$

$\rightarrow$ $\Pr(D \models q) = 1 - \prod_{a \in \mathrm{Dom}(D)} (1 - \Pr(D \models R(a) \wedge \exists y\ S(a, y)))$

$\rightarrow$ $\Pr(D \models R(a) \wedge \exists y\ S(a, y)) =$

Example: independence

- $q := \exists x, y\ R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y\ S(a, y))$

$\rightarrow$ $\Pr(D \models q) = 1 - \prod_{a \in \mathrm{Dom}(D)} (1 - \Pr(D \models R(a) \wedge \exists y\ S(a, y)))$

$\rightarrow$ $\Pr(D \models R(a) \wedge \exists y\ S(a, y)) =$

Example: independence

- $q := \exists x, y\ R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y\ S(a, y))$

$\rightarrow$ $\Pr(D \models q) = 1 - \prod_{a \in \mathrm{Dom}(D)} (1 - \Pr(D \models R(a) \wedge \exists y\ S(a, y)))$

$\rightarrow$ $\Pr(D \models R(a) \wedge \exists y\ S(a, y)) = \Pr(D \models R(a)) \times \Pr(D \models \exists y\ S(a, y))$

Example: independence

- $q := \exists x, y\ R(x) \wedge S(x, y)$
- $q \equiv \bigvee_{a \in \mathrm{Dom}(D)} (R(a) \wedge \exists y\ S(a, y))$

$\rightarrow$ $\Pr(D \models q) = 1 - \prod_{a \in \mathrm{Dom}(D)} (1 - \Pr(D \models R(a) \wedge \exists y\ S(a, y)))$

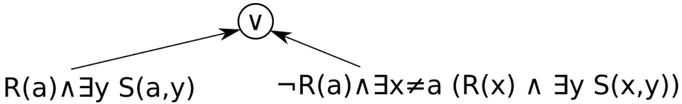
$\rightarrow$ $\Pr(D \models R(a) \wedge \exists y\ S(a, y)) = \Pr(D \models R(a)) \times \Pr(D \models \exists y\ S(a, y))$

$\rightarrow$ $\Pr(D \models \exists y\ S(a, y)) = 1 - \prod_{b \in \mathrm{Dom}(D)} (1 - \Pr(D \models S(a, b)))$

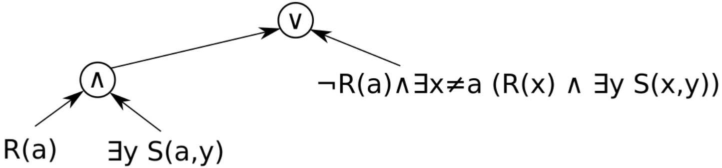
Example: independence + mutually-exclusiveness

$\exists x,y\ (R(x) \wedge S(x,y))$

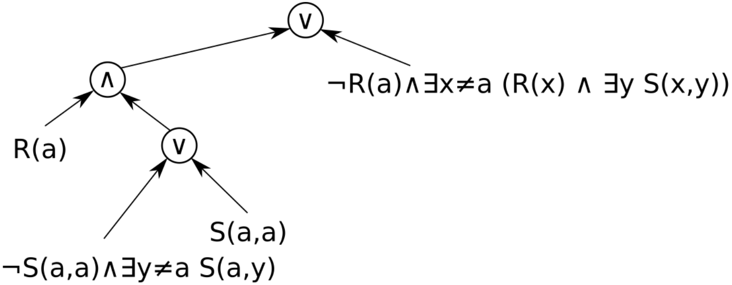
Example: independence + mutually-exclusiveness



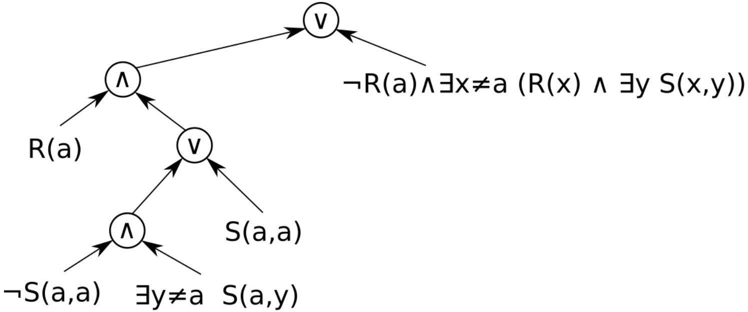
Example: independence + mutually-exclusiveness



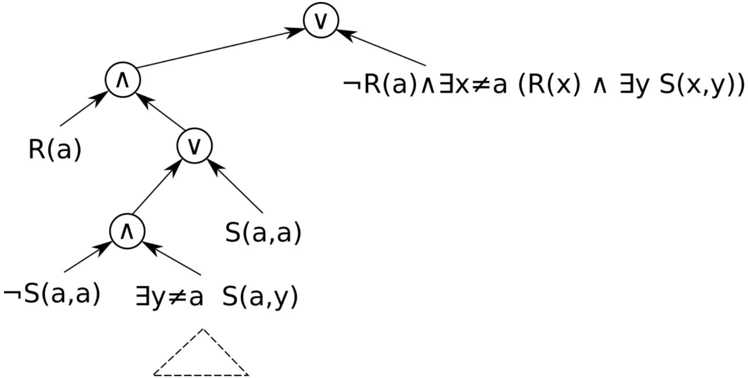
Example: independence + mutually-exclusiveness



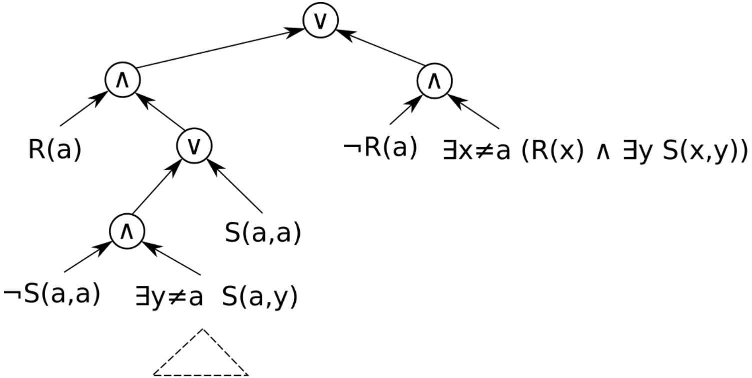
Example: independence + mutually-exclusiveness



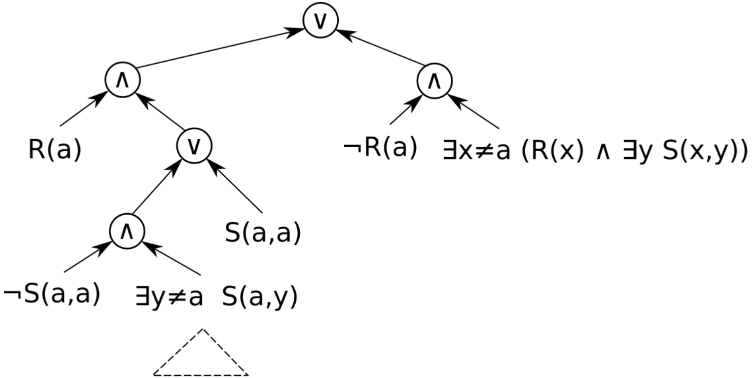
Example: independence + mutually-exclusiveness



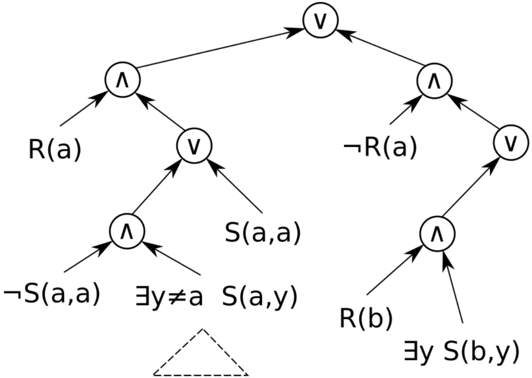
Example: independence + mutually-exclusiveness



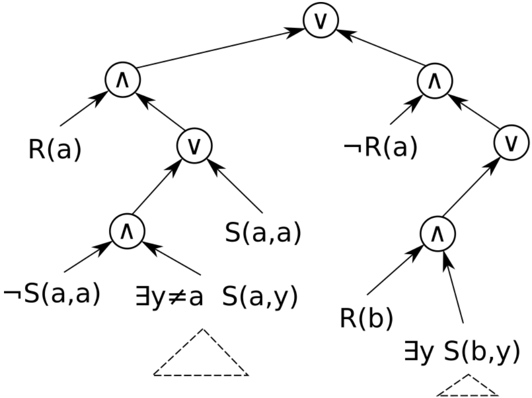
Example: independence + mutually-exclusiveness



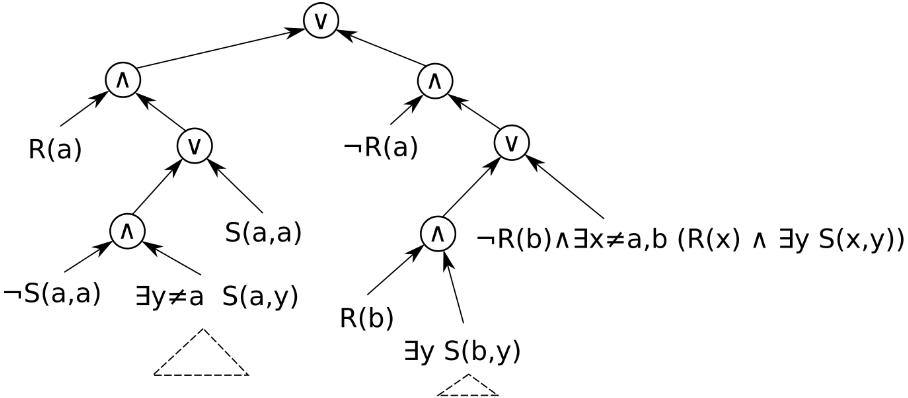
Example: independence + mutually-exclusiveness



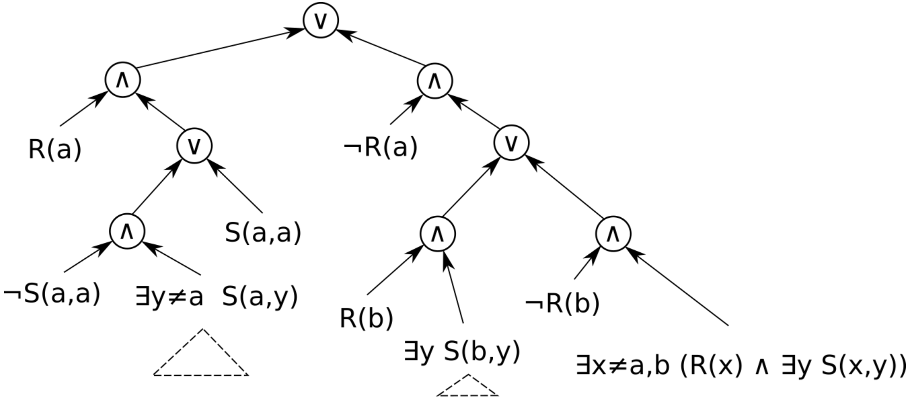
Example: independence + mutually-exclusiveness



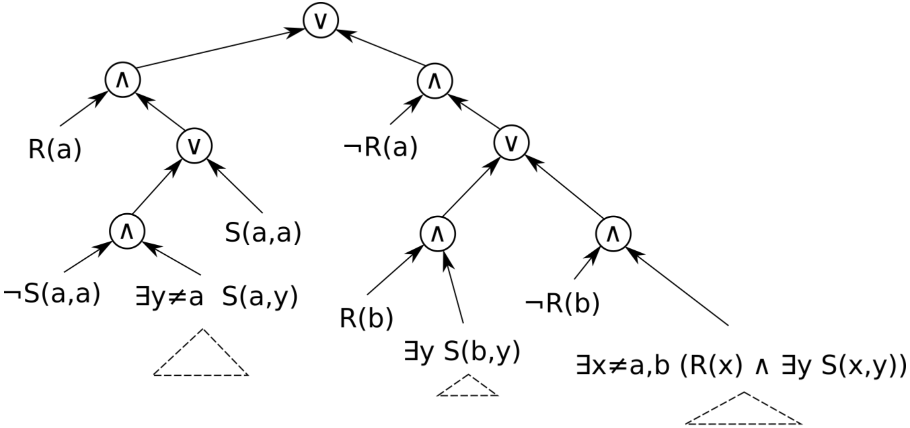
Example: independence + mutually-exclusiveness



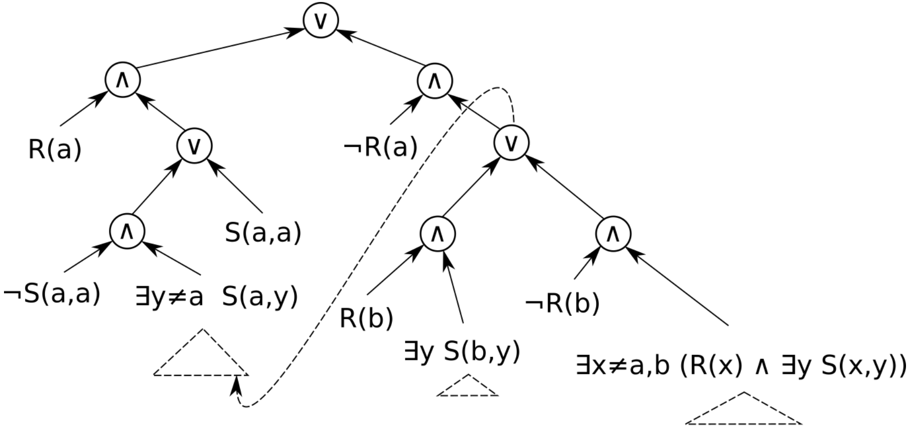
Example: independence + mutually-exclusiveness



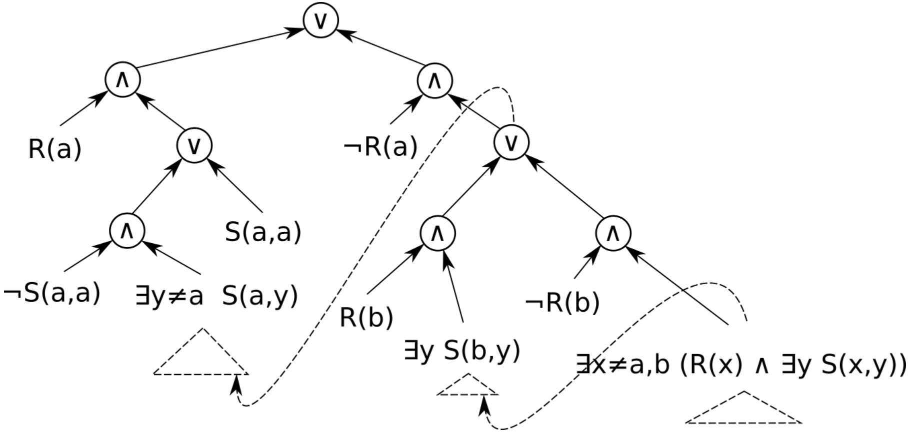
Example: independence + mutually-exclusiveness



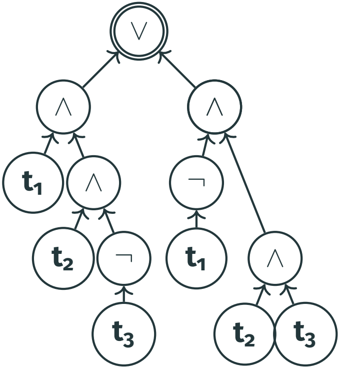
Example: independence + mutually-exclusiveness



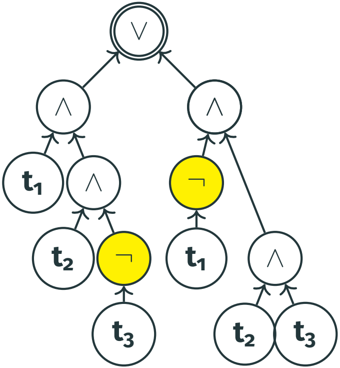
Example: independence + mutually-exclusiveness



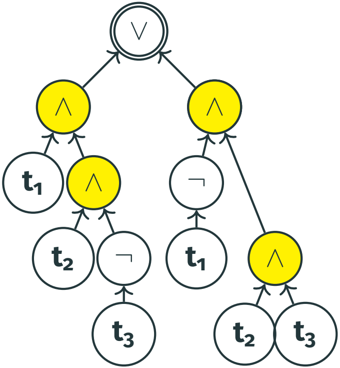
d-DNNF



d-DNNF

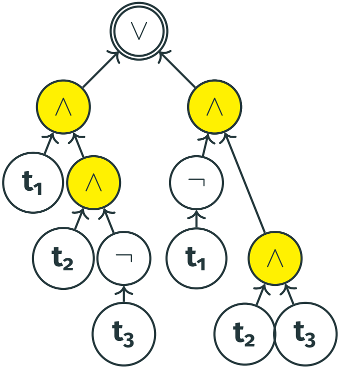


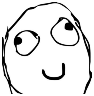
- Negation Normal Form: negations only applied to the leaves



- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)

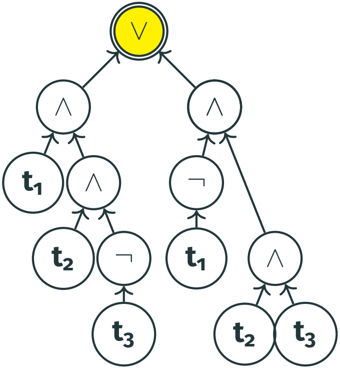
d-DNNF

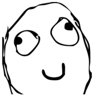




- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  - → **SAT** can be solved efficiently

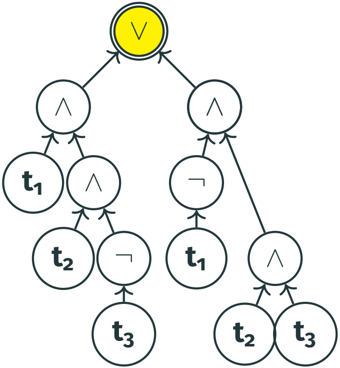
d-DNNF

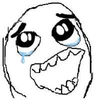




- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  - → **SAT** can be solved efficiently
- Deterministic: inputs of ∨-gates are **mutually exclusive**

d-DNNF





- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  - → **SAT** can be solved efficiently
- Deterministic: inputs of ∨-gates are **mutually exclusive**
  - → **#SAT** and **probability evaluation**

Conjecture

- UCQ $q \in$ **UCQ(d-DNNF)** $\equiv$ for every database D, we can compute in PTIME a d-DNNF representing the result

Conjecture

- UCQ $q \in$ **UCQ(d-DNNF)** $\equiv$ for every database D, we can compute in PTIME a d-DNNF representing the result
- **Question: tractable UCQs $=$ safe UCQs $\stackrel{?}{=}$ UCQ(d-DNNF)**

Conjecture

- UCQ $q \in$ **UCQ(d-DNNF)** $\equiv$ for every database D, we can compute in PTIME a d-DNNF representing the result
- **Question: tractable UCQs = safe UCQs $\stackrel{?}{=}$ UCQ(d-DNNF)**
- Independence + IE $\stackrel{?}{=}$ independence + mut.-exclusiveness

Conjecture

- UCQ $q \in$ **UCQ(d-DNNF)** $\equiv$ for every database D, we can compute in PTIME a d-DNNF representing the result
- **Question: tractable UCQs = safe UCQs $\stackrel{?}{=}$ UCQ(d-DNNF)**
- Independence + IE $\stackrel{?}{=}$ independence + mut.-exclusiveness

$\rightarrow$ We already have: safe UCQs $\supseteq$ UCQ(d-DNNF)

Conjecture

- UCQ $q \in$ **UCQ(d-DNNF)** $\equiv$ for every database D, we can compute in PTIME a d-DNNF representing the result
- **Question: tractable UCQs = safe UCQs $\stackrel{?}{=}$ UCQ(d-DNNF)**
- Independence + IE $\stackrel{?}{=}$ independence + mut.-exclusiveness

$\rightarrow$ We already have: safe UCQs $\supseteq$ UCQ(d-DNNF)

$\rightarrow$ So the question is: safe UCQs $\subseteq$ UCQ(d-DNNF)?

- Asked by Dalvi, Jha and Suciu

$\mathcal{H}$ query class

- We focussed on a subset of the safe queries, which we will call the $\mathcal{H}$-queries

$\mathcal{H}$ query class

- We focussed on a subset of the safe queries, which we will call the $\mathcal{H}$-queries
- These $\mathcal{H}$-queries have already been studied before, and:

$\rightarrow$ It is not known if $\mathcal{H}$-queries $\subseteq$ UCQ(d-DNNF)

$\mathcal{H}$ query class

- We focussed on a subset of the safe queries, which we will call the $\mathcal{H}$**-queries**
- These $\mathcal{H}$-queries have already been studied before, and:

→ It is not known if $\mathcal{H}$-queries $\subseteq$ UCQ(d-DNNF)

→ It is known that $\mathcal{H}$-queries $\subsetneq$ UCQ(**decision**-DNNF)

$\mathcal{H}$ query class

- We focussed on a subset of the safe queries, which we will call the $\mathcal{H}$**-queries**
- These $\mathcal{H}$-queries have already been studied before, and:

→ It is not known if $\mathcal{H}$-queries $\subseteq$ UCQ(d-DNNF)

→ It is known that $\mathcal{H}$-queries $\subsetneq$ UCQ(**decision**-DNNF)

→ It is known that $\mathcal{H}$-queries $\subsetneq$ UCQ(**structured** d-DNNF)

Our contribution

- We defined a notion of an $\mathcal{H}$-query q being nice, such that:

$\rightarrow$ If q is nice, then $q \in$ UCQ(d-DNNF)

Our contribution

- We defined a notion of an $\mathcal{H}$-query q being nice, such that:

$\rightarrow$ If q is nice, then $q \in$ UCQ(d-DNNF)

- We generated all the $\mathcal{H}$-queries up to a certain size parameter (amounting to about 60 million queries)

Our contribution

- We defined a notion of an $\mathcal{H}$-query q being nice, such that:

$\rightarrow$ If q is nice, then $q \in$ UCQ(d-DNNF)

- We generated all the $\mathcal{H}$-queries up to a certain size parameter (amounting to about 60 million queries)
- We checked whether these queries are nice using a sat solver

Our contribution

- We defined a notion of an $\mathcal{H}$-query q being nice, such that:

$\rightarrow$ If q is nice, then $q \in$ UCQ(d-DNNF)

- We generated all the $\mathcal{H}$-queries up to a certain size parameter (amounting to about 60 million queries)
- We checked whether these queries are nice using a sat solver

$\rightarrow$ Turns out almost all of them are nice, so they are in UCQ(d-DNNF)

$\rightarrow$ For those that were not nice, we showed that their negation is nice, so they are in UCQ(d-D)

Our contribution

- We defined a notion of an $\mathcal{H}$-query q being nice, such that:

$\rightarrow$ If q is nice, then $q \in$ UCQ(d-DNNF)

- We generated all the $\mathcal{H}$-queries up to a certain size parameter (amounting to about 60 million queries)
- We checked whether these queries are nice using a sat solver

$\rightarrow$ Turns out almost all of them are nice, so they are in UCQ(d-DNNF)

$\rightarrow$ For those that were not nice, we showed that their negation is nice, so they are in UCQ(d-D)

- Maybe **safe UCQs** = **UCQ(d-D)**?

Conclusion

$$[(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, S_4(x,y) \wedge S_5(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))]$$

Conclusion

$[(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, S_4(x,y) \wedge S_5(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))]$

This query is obviously in UCQ(d-D), but is it in UCQ(d-DNNF)?

Conclusion

$[(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_3(x,y) \wedge S_4(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_3(x,y) \wedge S_4(x,y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_2(x,y) \wedge S_3(x,y))] \wedge [(\exists x \exists y\, S_2(x,y) \wedge S_3(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))] \wedge [(\exists x \exists y\, R(x) \wedge S_1(x,y)) \vee (\exists x \exists y\, S_1(x,y) \wedge S_2(x,y)) \vee (\exists x \exists y\, S_4(x,y) \wedge S_5(x,y))] \wedge [(\exists x \exists y\, S_4(x,y) \wedge S_5(x,y)) \vee (\exists x \exists y\, S_5(x,y) \wedge T(y))]$

This query is obviously in UCQ(d-D), but is it in UCQ(d-DNNF)?

Thanks for your attention!