Circuits for Query Provenance

Dan Suciu[1]

University of Washington

[1]Joint work with Paul Beame, Nilesh Dalvi, Abhay Jha, Jerry Li, Sudeepa Roy

Motivation

- Consider some problem on Boolean formulas F: SAT, model counting, circuit (BDD) construction, etc, etc.
- In general, the complexity is exponential in F.
- Now assume that F is the provenance (lineage/grounding) of an FO sentence Q over some input domain.
- For fixed Q, what is the problem complexity as a function of $|\text{input}|$?

This Talk

F is the provenance of some FO sentence Q:

- Complexity of the Weighted Model Counting problem for F.
- The size of an OBDD, or FBDD, or Decision-DNNF for F.
  Knowledge Compilation [Darwiche and Marquis, 2002].
- Glaring omission: SAT.

Main message: from Logic (Q) to Algorithms (for F)

Weighted Model Counting

Weighted Model Counting

Boolean formula F; Model count $\#F$ is #P-complete [Valiant, 1979]

For each variable X_i, a probability $p_i \in [0, 1]$: Weighted model count $\mathbf{P}(F)$;

Weighted Model Counting

Boolean formula F; Model count $\#F$ is #P-complete [Valiant, 1979]

For each variable X_i, a probability $p_i \in [0,1]$: Weighted model count $\mathbf{P}(F)$;

Subfunctions become easier: $\text{Time}(\mathbf{P}(F[\theta])) \leq \text{Time}(\mathbf{P}(F))$.

Weighted Model Counting

Boolean formula F; Model count $\#F$ is #P-complete [Valiant, 1979]

For each variable X_i, a probability $p_i \in [0,1]$: Weighted model count $\mathbf{P}(F)$;

Subfunctions become easier: $\text{Time}(\mathbf{P}(F[\theta])) \leq \text{Time}(\mathbf{P}(F))$.

Shannon expansion: $\boxed{\mathbf{P}(F) = (1-p_i)\cdot \mathbf{P}(F[X_i := 0]) + p_i \cdot \mathbf{P}(F[X_i := 1])}$

Independence: $\boxed{\mathbf{P}(F_1 \wedge F_2) = \mathbf{P}(F_1)\cdot \mathbf{P}(F_2)}$, if $\text{Vars}(F_1) \cap \text{Vars}(F_2) = \emptyset$.

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]]$$

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

$$F_n[\neg Q] \stackrel{\text{def}}{=} \neg F_n[Q]$$

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

$$F_n[\neg Q] \stackrel{\text{def}}{=} \neg F_n[Q] \qquad F_n[R(i,j,\ldots)] \stackrel{\text{def}}{=} X_{ij\ldots}$$ a Boolean variable associated to this atom

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

$$F_n[\neg Q] \stackrel{\text{def}}{=} \neg F_n[Q] \qquad F_n[R(i, j, \ldots)] \stackrel{\text{def}}{=} X_{ij\ldots}$$ a Boolean variable associated to this atom

Example $Q = \forall x \forall y (R(x) \vee S(x, y))$

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

$$F_n[\neg Q] \stackrel{\text{def}}{=} \neg F_n[Q] \qquad F_n[R(i,j,\ldots)] \stackrel{\text{def}}{=} X_{ij\ldots}$$ a Boolean variable associated to this atom

Example $Q = \forall x \forall y (R(x) \vee S(x,y))$ $\quad F_n[Q] = \bigwedge_{i,j=1,n}(X_i \vee Y_{ij})$

Provenance/Lineage/Grounding

FO sentence Q. The provenance of Q on a domain of size n, $F_n[Q]$ is:

$$F_n[\forall x Q] \stackrel{\text{def}}{=} \bigwedge_{i=1,n} F_n[Q[i/x]] \qquad F_n[Q_1 \wedge Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \wedge F_n[Q_2]$$

$$F_n[\neg Q] \stackrel{\text{def}}{=} \neg F_n[Q] \qquad F_n[R(i,j,\ldots)] \stackrel{\text{def}}{=} X_{ij\ldots}$$ a Boolean variable associated to this atom

Example $Q = \forall x \forall y (R(x) \vee S(x,y))$ $F_n[Q] = \bigwedge_{i,j=1,n} (X_i \vee Y_{ij})$

Given Q, what is the complexity of $\mathbf{P}(F_n[Q])$?

Results in this talk: No negation, single quantifier type ($\exists\exists\cdots$ or $\forall\forall\cdots$)

Syntactic Feature #1: Hierarchy

Fix Q; $at(x) \stackrel{\text{def}}{=}$ the set of atoms containing variable x.

Definition

Q is hierarchical if $at(x) \subseteq at(y)$, or $at(x) \supseteq at(y)$, or $at(x) \cap at(y) = \emptyset$.

2

Syntactic Feature #1: Hierarchy

Fix Q; $at(x) \stackrel{\text{def}}{=}$ the set of atoms containing variable x.

Definition

Q is hierarchical if $at(x) \subseteq at(y)$, or $at(x) \supseteq at(y)$, or $at(x) \cap at(y) = \emptyset$.

Theorem

CQ w/o self-joins: if hierarchical, $\mathbf{P}(F_n[Q])$ *in PTIME, otherwise #P-hard.*

2

Syntactic Feature #1: Hierarchy

Fix Q; $at(x) \stackrel{\text{def}}{=}$ the set of atoms containing variable x.

Definition

Q is hierarchical if $at(x) \subseteq at(y)$, or $at(x) \supseteq at(y)$, or $at(x) \cap at(y) = \emptyset$.

Theorem

CQ w/o self-joins: if hierarchical, $\mathbf{P}(F_n[Q])$ *in PTIME, otherwise #P-hard.*

Hierarchical: $\exists x \exists y (R(x) \wedge S(x, y))$ is in PTIME.

Non-Hierarchical: $\exists x \exists y (R(x) \wedge S(x, y) \wedge T(y))$ is #P-hard.[2]

[2] Reduction from $\#F$ for $F = \bigvee_{(i,j) \in E} X_i \wedge Y_j$ [Provan and Ball, 1983].

Dichotomy

What about CQs with self-joins? Or UCQs?

Dichotomy

What about CQs with self-joins? Or UCQs?

Theorem ([Dalvi and Suciu, 2012])

For any Q, $\mathbf{P}(F_n[Q])$ is either in PTIME, or it is #P-hard.

Dichotomy

What about CQs with self-joins? Or UCQs?

Theorem ([Dalvi and Suciu, 2012])

For any Q, $\mathbf{P}(F_n[Q])$ is either in PTIME, or it is #P-hard.

Hierarchy is necessary but not sufficient condition for PTIME:

Example hierarchical, yet #P-hard:

$$\text{UCQ: } \boxed{\exists x \exists y (R(x) \wedge S(x,y)) \vee \exists u \exists v (S(u,v) \wedge T(v))}$$

$$\text{Dual: } \boxed{\forall x \forall y (R(x) \vee S(x,y)) \wedge \forall u \forall v (S(u,v) \vee T(v))}$$

Discussion

- Main take away: from static analysis on Q to complexity of $\mathbf{P}(F_n[Q])$.
- Extension to UCQ^∞ (includes datalog) [Amarilli and Ceylan, 2020].
- Open: beyond UCQ/dualUCQ?
- #SAT Dichotomy theorem [Creignou and Hermann, 1996] based on type of clauses (affine or not); dichotomy for UCQ based on structure.

Next: size of a BDD for $F_n[Q]$.

Background: Binary Decision Diagrams

Overview: BDDs

Monography on BDDs [Wegener, 2000].

This talk:

Free Binary Decision Diagrams, FBDDs:

- Read-Once Branching Programs
- Binary Decision Diagrams [Akers, 1978] or Branching Programs [Masek, 1976]), subject to the read-once rule.

Ordered Binary Decision Diagrams, OBDD [Bryant, 1986].

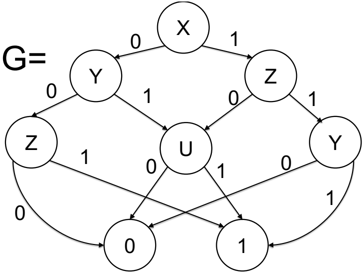
Decision-DNN [Huang and Darwiche, 2005, Huang and Darwiche, 2007]:

- Special case of AND-FBDDs [Wegener, 2000].
- Special case of d-DNNF [Darwiche, 2001].

Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD

$F = \bar{X}\bar{Y}Z + \bar{X}YU + X\bar{Z}U + XYZ.$

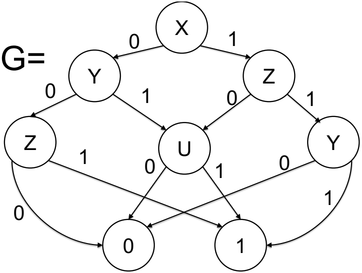


Read-once property

Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD

$F = \bar{X}\bar{Y}Z + \bar{X}YU + X\bar{Z}U + XYZ.$



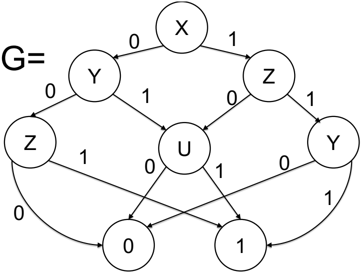
Read-once property

OBDD: fixed variable order

Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD

$F = \bar{X}\bar{Y}Z + \bar{X}YU + X\bar{Z}U + XYZ.$

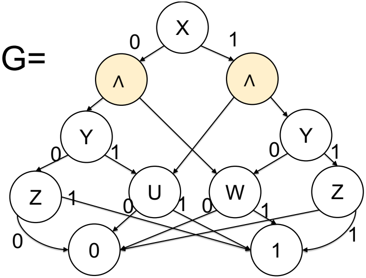


Read-once property

OBDD: fixed variable order

Decision-DNNF

$F = \cdots$

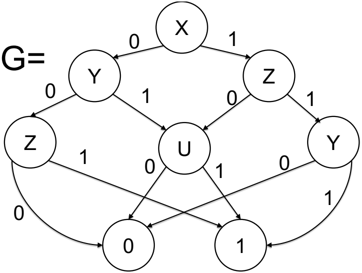


Decomposable $\bigwedge$-nodes.

Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD

$F = \bar{X}\bar{Y}Z + \bar{X}YU + X\bar{Z}U + XYZ.$

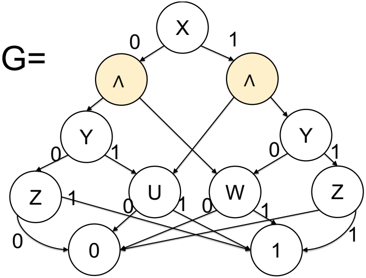


Read-once property

OBDD: fixed variable order

Decision-DNNF

$F = \cdots$



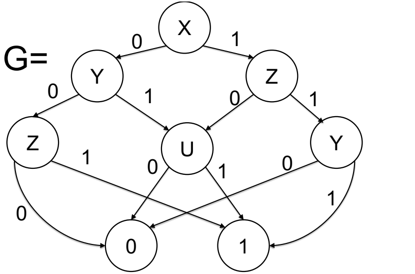
Decomposable $\bigwedge$-nodes.

$\exists$ FBDD of size $\leq 2|G|2^{\log^2|G|}$

Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD

$F = \bar{X}\bar{Y}Z + \bar{X}YU + X\bar{Z}U + XYZ.$

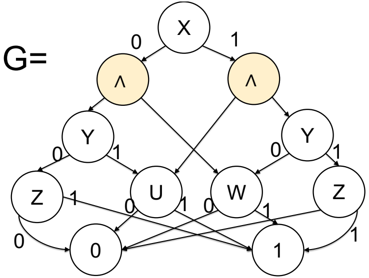


Read-once property

OBDD: fixed variable order

Decision-DNNF

$F = \cdots$



Decomposable $\bigwedge$-nodes.

$\exists$ FBDD of size $\leq 2|G|2^{\log^2|G|}$

[Wegener, 2000]

- WMC in linear time: $\text{Time}(\mathbf{P}(F)) = O(|G|)$
- BDDs for subfunctions become smaller: $|G(F[\theta])| \leq |G(F)|$

Knowledge Compilation v.s. Query Compilation

Knowledge compilation $F \mapsto$ BDD for F [Darwiche and Marquis, 2002].

Query compilation Fix Q. $n \mapsto$ BDD for $F_n[Q]$ [Jha and Suciu, 2013].

OBDDs

OBDD

- OBDD = an FBDD that follows a fixed variable order Π.
- Similar to a DFA [Wegener, 2000].
- Synthesis:[3] Given OBDDs G_1, G_2 for F_1, F_2 using same order Π, can synthesize an OBDD for $F_1 \wedge F_2$ or $F_1 \vee F_2$, of size $\leq |G_1| \cdot |G_2|$.

Given Q, what is the size of the OBDD for $F_n[Q]$?

[3] Product automaton.

Example

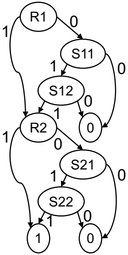
$$\boxed{Q_1 = \forall x \forall y (R(x) \vee S(x,y))}$$

$$F_2 = (R_1 \vee S_{11})(R_1 \vee S_{12})(R_2 \vee S_{21})(R_2 \vee S_{22})$$

Example

$$\boxed{Q_1 = \forall x \forall y (R(x) \vee S(x,y))}$$

$$F_2 = (R_1 \vee S_{11})(R_1 \vee S_{12})(R_2 \vee S_{21})(R_2 \vee S_{22})$$

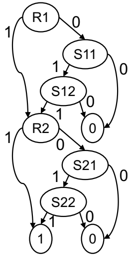


Size $= O(n)$.

Example

$Q_1 = \forall x \forall y (R(x) \vee S(x, y))$ $Q_2 = \forall u \forall v (S(u, v) \vee T(u))$.

$F_2 = (R_1 \vee S_{11})(R_1 \vee S_{12})(R_2 \vee S_{21})(R_2 \vee S_{22})$

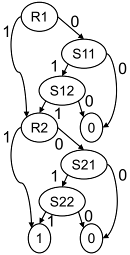


Size $= O(n)$.

Example

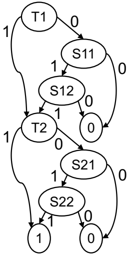
$Q_1 = \forall x \forall y (R(x) \vee S(x, y))$

$F_2 = (R_1 \vee S_{11})(R_1 \vee S_{12})(R_2 \vee S_{21})(R_2 \vee S_{22})$



Size $= O(n)$.

$Q_2 = \forall u \forall v (S(u, v) \vee T(u))$.



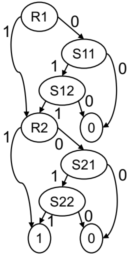
Size $= O(n)$.

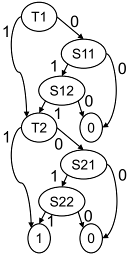
Example

$Q_1 = \forall x \forall y (R(x) \vee S(x,y))$

$F_2 = (R_1 \vee S_{11})(R_1 \vee S_{12})(R_2 \vee S_{21})(R_2 \vee S_{22})$

$Q_2 = \forall u \forall v (S(u,v) \vee T(u))$.





Size $= O(n)$.

Size $= O(n)$.

Same variable order: synthesize OBDD for $Q_1 \wedge Q_2$ of size $= O(n)$.

A Lower Bound

$$H_0 \stackrel{\text{def}}{=} \forall x \forall y (R(x) \vee S(x, y) \vee T(y))$$

A Lower Bound

$H_0 \stackrel{\text{def}}{=} \forall x \forall y (R(x) \vee S(x, y) \vee T(y))$ // drop $\forall \cdots$

A Lower Bound

$H_0 \stackrel{\text{def}}{=} R(x) \vee S(x, y) \vee T(y)$ // drop $\forall \cdots$

A Lower Bound

$$H_0 \stackrel{\text{def}}{=} R(x) \vee S(x,y) \vee T(y) \qquad // \text{ drop } \forall \cdots$$

$$H_1 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee T(y_1))$$

A Lower Bound

$$H_0 \stackrel{\text{def}}{=} R(x) \vee S(x,y) \vee T(y) \qquad // \text{ drop } \forall \cdots$$

$$H_1 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee T(y_1))$$

$$H_2 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge (S_2(x_2,y_2) \vee T(y_2))$$

A Lower Bound

$$H_0 \stackrel{\text{def}}{=} R(x) \vee S(x,y) \vee T(y) \qquad // \text{ drop } \forall \cdots$$

$$H_1 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee T(y_1))$$

$$H_2 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge (S_2(x_2,y_2) \vee T(y_2))$$

$$\cdots$$

$$H_k \stackrel{\text{def}}{=} \ldots$$

H_k for $k \geq 1$ is hierarchical.

A Lower Bound

$$H_0 \stackrel{\text{def}}{=} R(x) \vee S(x,y) \vee T(y) \qquad // \text{ drop } \forall \cdots$$

$$H_1 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee T(y_1))$$

$$H_2 \stackrel{\text{def}}{=} (R(x_0) \vee S_1(x_0,y_0)) \wedge (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge (S_2(x_2,y_2) \vee T(y_2))$$

$$\ldots$$

$$H_k \stackrel{\text{def}}{=} \ldots$$

H_k for $k \geq 1$ is hierarchical.

Theorem ([Beame et al., 2017])

Any FBDD for H_k has size $\geq (2^n - 1)/n$; Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

Syntactic Feature #2: Inversions

Definition

A *k-inversion* in a sentence Q is a sequence of atoms:

$$S_1(..,x_0,..,y_0..), S_1(..,x_1,..,y_1..), S_2(..,x_1,..,y_1..), S_2(..,x_2,..,y_2,..), \ldots, S_k(..,x_k,..,y_k,..)$$

Such that $at(x_0) \supsetneq at(y_0)$ and $at(x_k) \subsetneq at(y_k)$.

Example every H_k has a k-inversion.

$$H_2 = (R(x_0) \vee S_1(\underline{x_0,y_0})) \wedge (S_1(\underline{x_1,y_1}) \vee S_2(\underline{x_1,y_1})) \wedge (S_2(\underline{x_2,y_2}) \vee T(y_2))$$

Inversions prevent us from finding a good order for the OBDD.

Dichotomy

Theorem ([Jha and Suciu, 2013, Beame et al., 2017])

1. *If Q has no inversions, then $F_n[Q]$ has an OBDD of size $O(n^{arity})$ (linear).*
2. *If Q has a k-inversion, then the OBDD for $F_n[Q]$ has size $2^{\Omega(n/(k+1))}$.*

1. Order the Boolean variables consistent with the hierarchy $at(x)$: "no inversion" makes this possible. Build the OBDD using synthesis.
2. OBDD G for Q $\Rightarrow$ $k+1$ subfunction OBDDs for the clauses of H_k $\Rightarrow$ synthesis OBDD for H_k of size $O(|G|^{k+1} \geq (2^n - 1)/n$.

Both proofs fail for FBDD: no synthesis.

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

$$Q_V \stackrel{\text{def}}{=} (R(x_0) \vee S(x_0, y_0)) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x) \vee T(y))$$

Has inversion, yet $\mathbf{P}(Q_V)$ in PTIME:

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

$$Q_V \stackrel{\text{def}}{=} (R(x_0) \vee S(x_0, y_0)) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x) \vee T(y))$$

Has inversion, yet $\mathbf{P}(Q_V)$ in PTIME:

$$Q_V = R(x)(R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1))T(y)$$

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

$$Q_V \stackrel{\text{def}}{=} (R(x_0) \vee S(x_0, y_0)) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x) \vee T(y))$$

Has inversion, yet $\mathbf{P}(Q_V)$ in PTIME:

$$\begin{aligned} Q_V &= R(x)(R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1))T(y) \\ &= R(x) \wedge (S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0)) \wedge T(y) \end{aligned}$$

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

$$Q_V \stackrel{\text{def}}{=} (R(x_0) \vee S(x_0, y_0)) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x) \vee T(y))$$

Has inversion, yet $\mathbf{P}(Q_V)$ in PTIME:

$$Q_V = R(x)(R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1))T(y)$$

$$= R(x) \wedge (S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0)) \wedge T(y)$$

$$\mathbf{P}(Q_V) = \mathbf{P}(R(x) \wedge (S(x_1, y_1) \vee T(y_1))) + \mathbf{P}((R(x_0) \vee S(x_0, y_0)) \wedge T(y))$$

$$- \mathbf{P}(\underbrace{(R(x) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x_0) \vee S(x_0, y_0)) \wedge T(y))}_{\equiv R(x) \wedge T(y)})$$

The Inclusion/Exclusion Formula

If Q is a query without inversion then $\mathbf{P}(Q)$ is in PTIME.

What about the converse?

$$Q_V \stackrel{\text{def}}{=} (R(x_0) \vee S(x_0, y_0)) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x) \vee T(y))$$

Has inversion, yet $\mathbf{P}(Q_V)$ in PTIME:

$$Q_V = R(x)(R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0))(S(x_1, y_1) \vee T(y_1))T(y)$$

$$= R(x) \wedge (S(x_1, y_1) \vee T(y_1)) \quad \bigvee \quad (R(x_0) \vee S(x_0, y_0)) \wedge T(y)$$

$$\mathbf{P}(Q_V) = \mathbf{P}(R(x) \wedge (S(x_1, y_1) \vee T(y_1))) + \mathbf{P}((R(x_0) \vee S(x_0, y_0)) \wedge T(y))$$

$$- \mathbf{P}(\underbrace{(R(x) \wedge (S(x_1, y_1) \vee T(y_1)) \wedge (R(x_0) \vee S(x_0, y_0)) \wedge T(y))}_{\equiv R(x) \wedge T(y)})$$

All three queries inversion-free: $\mathbf{P}(Q_V)$ in PTIME, OBDD $2^{\Omega(n)}$

Discussion

- From static analysis on Q to OBDD size for $F_n[Q]$.
- OBDDs are "incomplete".
- [Beame and Liew, 2015] prove the same linear/exponential dichotomy for SDDs (a strict generalization of OBDDs)

Are FBDDs/Decision-DNNFs complete?

FBDDs and Decision-DNNFs

The Quest of a "Complete" Family of Circuits

If $\mathbf{P}(F_n[Q])$ is in PTIME, does $F_n[Q]$ have a polynomial size FBDD? Or Decision-DNNF

In other words, are FBDDs/Decision-DNNF "complete" for tractable UCQs?

The Quest of a "Complete" Family of Circuits

If $\mathbf{P}(F_n[Q])$ is in PTIME, does $F_n[Q]$ have a polynomial size FBDD? Or Decision-DNNF

In other words, are FBDDs/Decision-DNNF "complete" for tractable UCQs?

Will show both are incomplete

Syntactic Feature #3: Cancellations

$$H_3 = \underbrace{(R(x_0) \vee S_1(x_0, y_0))}_{\stackrel{\text{def}}{=} h_{30}} \wedge \underbrace{(S_1(x_1, y_1) \vee S_2(x_1, y_1))}_{\stackrel{\text{def}}{=} h_{31}} \wedge \underbrace{(S_2(x_2, y_2) \vee S_3(x_2, y_2))}_{\stackrel{\text{def}}{=} h_{32}} \wedge \underbrace{(S_3(x_3, y_3) \vee T(y_3))}_{\stackrel{\text{def}}{=} h_{33}}$$

Syntactic Feature #3: Cancellations

$$H_3 = \underbrace{(R(x_0) \vee S_1(x_0, y_0))}_{\stackrel{\text{def}}{=} h_{30}} \wedge \underbrace{(S_1(x_1, y_1) \vee S_2(x_1, y_1))}_{\stackrel{\text{def}}{=} h_{31}} \wedge \underbrace{(S_2(x_2, y_2) \vee S_3(x_2, y_2))}_{\stackrel{\text{def}}{=} h_{32}} \wedge \underbrace{(S_3(x_3, y_3) \vee T(y_3))}_{\stackrel{\text{def}}{=} h_{33}}$$

$$Q_W \stackrel{\text{def}}{=} (h_{30} \wedge h_{32}) \vee (h_{30} \wedge h_{33}) \vee (h_{31} \wedge h_{33})$$

Theorem ([Beame et al., 2017])

(1) $\mathbf{P}(Q_W)$ *in PTIME.*

(2) FBDD for Q_W *has size* $2^{\Omega(n)}$

Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

Syntactic Feature #3: Cancellations

$$H_3 = \underbrace{(R(x_0) \vee S_1(x_0, y_0))}_{\stackrel{\text{def}}{=} h_{30}} \wedge \underbrace{(S_1(x_1, y_1) \vee S_2(x_1, y_1))}_{\stackrel{\text{def}}{=} h_{31}} \wedge \underbrace{(S_2(x_2, y_2) \vee S_3(x_2, y_2))}_{\stackrel{\text{def}}{=} h_{32}} \wedge \underbrace{(S_3(x_3, y_3) \vee T(y_3))}_{\stackrel{\text{def}}{=} h_{33}}$$

$$Q_W \stackrel{\text{def}}{=} (h_{30} \wedge h_{32}) \vee (h_{30} \wedge h_{33}) \vee (h_{31} \wedge h_{33})$$

Theorem ([Beame et al., 2017])

(1) $\mathbf{P}(Q_W)$ *in PTIME.*
(2) FBDD for Q_W *has size* $2^{\Omega(n)}$
Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

$$\begin{aligned}\mathbf{P}(Q_W) = & \mathbf{P}(h_{30} \wedge h_{32}) + \mathbf{P}(h_{30} \wedge h_{33}) + \mathbf{P}(h_{31} \wedge h_{33}) \\ & - \mathbf{P}(h_{30} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{33}) \\ & + \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33})\end{aligned}$$

Syntactic Feature #3: Cancellations

$$H_3 = \underbrace{(R(x_0) \vee S_1(x_0,y_0))}_{\stackrel{\text{def}}{=} h_{30}} \wedge \underbrace{(S_1(x_1,y_1) \vee S_2(x_1,y_1))}_{\stackrel{\text{def}}{=} h_{31}} \wedge \underbrace{(S_2(x_2,y_2) \vee S_3(x_2,y_2))}_{\stackrel{\text{def}}{=} h_{32}} \wedge \underbrace{(S_3(x_3,y_3) \vee T(y_3))}_{\stackrel{\text{def}}{=} h_{33}}$$

$$Q_W \stackrel{\text{def}}{=} (h_{30} \wedge h_{32}) \vee (h_{30} \wedge h_{33}) \vee (h_{31} \wedge h_{33})$$

Theorem ([Beame et al., 2017])

(1) $\mathbf{P}(Q_W)$ in PTIME.
(2) FBDD for Q_W has size $2^{\Omega(n)}$ Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

$$\begin{aligned}\mathbf{P}(Q_W) = {} & \mathbf{P}(h_{30} \wedge h_{32}) + \mathbf{P}(h_{30} \wedge h_{33}) + \mathbf{P}(h_{31} \wedge h_{33}) \\ & - \mathbf{P}(h_{30} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{33}) \\ & + \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33}) \qquad \text{// hard query } H_3 \text{ cancels out}\end{aligned}$$

Syntactic Feature #3: Cancellations

$$H_3 = \underbrace{(R(x_0) \vee S_1(x_0,y_0))}_{\stackrel{\text{def}}{=} h_{30}} \wedge \underbrace{(S_1(x_1,y_1) \vee S_2(x_1,y_1))}_{\stackrel{\text{def}}{=} h_{31}} \wedge \underbrace{(S_2(x_2,y_2) \vee S_3(x_2,y_2))}_{\stackrel{\text{def}}{=} h_{32}} \wedge \underbrace{(S_3(x_3,y_3) \vee T(y_3))}_{\stackrel{\text{def}}{=} h_{33}}$$

$$Q_W \stackrel{\text{def}}{=} (h_{30} \wedge h_{32}) \vee (h_{30} \wedge h_{33}) \vee (h_{31} \wedge h_{33})$$

Theorem ([Beame et al., 2017])

(1) $\mathbf{P}(Q_W)$ *in PTIME.*
(2) FBDD for Q_W *has size* $2^{\Omega(n)}$
Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

$$\mathbf{P}(Q_W) = \mathbf{P}(h_{30} \wedge h_{32}) + \mathbf{P}(h_{30} \wedge h_{33}) + \mathbf{P}(h_{31} \wedge h_{33})$$
$$- \mathbf{P}(h_{30} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33}) - \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{33})$$
$$+ \mathbf{P}(h_{30} \wedge h_{31} \wedge h_{32} \wedge h_{33}) \quad \text{// hard query } H_3 \text{ cancels out}$$

FBDD for Q_W $\Rightarrow$ multi-output FBDD for $h_{30}, h_{31}, h_{32}, h_{33}$ $\Rightarrow$ FBDD for H_3.

Discussion

- FBDDs and Decision-DNNFs are Incomplete.
- The theorem generalizes from Q_W to arbitrary Boolean combinations of the clauses of H_k [Beame et al., 2017].
- Inclusion/exclusion with cancellation: a powerful syntactic feature.

Summary and Open Problems

Logic to Algorithms:
Statics analysis on the FO sentence Q to complexity analysis of $F_n[Q]$.

Syntactic Features: hierarchy, inversions, cancellations

- Open: beyond UCQs and their duals?
  - Add quantifier alternation, or negation, or ...
- Open: dichotomy for full FO?
  By Trakhentbrot's theorem we won't be able to decide the complexity.
- Open: complexity of SAT($F_n[Q]$)?
- Open: is there a "complete" family of circuits for UCQs? (Next talk)

Summary and Open Problems

Logic to Algorithms:
Statics analysis on the FO sentence Q to complexity analysis of $F_n[Q]$.

Syntactic Features: hierarchy, inversions, cancellations

- Open: beyond UCQs and their duals?
  - Add quantifier alternation, or negation, or ...
- Open: dichotomy for full FO?
  By Trakhentbrot's theorem we won't be able to decide the complexity.
- Open: complexity of SAT($F_n[Q]$)?
- Open: is there a "complete" family of circuits for UCQs? (Next talk)

Thank You

Surveys

[Suciu et al., 2011]

[den Broeck and Suciu, 2017]

[Suciu, 2020]

Akers, S. B. (1978).
Binary decision diagrams.
IEEE Trans. Computers, 27(6):509–516.

Amarilli, A. and Ceylan, İ. İ. (2020).
A dichotomy for homomorphism-closed queries on probabilistic graphs.
In Lutz, C. and Jung, J. C., editors, *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, volume 155 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Beame, P., Li, J., Roy, S., and Suciu, D. (2017).
Exact model counting of query expressions: Limitations of propositional methods.
ACM Trans. Database Syst., 42(1):1:1–1:46.

Beame, P. and Liew, V. (2015).
New limits for knowledge compilation and applications to exact model counting.
In *UAI*, pages 131–140.

Bryant, R. E. (1986).
Graph-based algorithms for boolean function manipulation.
IEEE Trans. Computers, 35(8):677–691.

Creignou, N. and Hermann, M. (1996).
Complexity of generalized satisfiability counting problems.
Inf. Comput., 125(1):1–12.

Dalvi, N. N. and Suciu, D. (2012).
The dichotomy of probabilistic inference for unions of conjunctive queries.
J. ACM, 59(6):30.

Darwiche, A. (2001).
Decomposable negation normal form.
J. ACM, 48(4):608–647.

Darwiche, A. and Marquis, P. (2002).
A knowledge compilation map.
J. Artif. Int. Res., 17(1):229–264.

den Broeck, G. V. and Suciu, D. (2017).
Query processing on probabilistic data: A survey.
Found. Trends Databases, 7(3-4):197–341.

Huang, J. and Darwiche, A. (2005).
Dpll with a trace: From sat to knowledge compilation.
In *IJCAI*, pages 156–162.

Huang, J. and Darwiche, A. (2007).
The language of search.
JAIR, 29:191–219.

Jha, A. K. and Suciu, D. (2013).
Knowledge compilation meets database theory: Compiling queries to decision diagrams.
Theory Comput. Syst., 52(3):403–440.

Masek, W. J. (1976).
A fast algorithm for the string editing problem and decision graph complexity.
Master's thesis, MIT.

Provan, J. S. and Ball, M. O. (1983).
The complexity of counting cuts and of computing the probability that a graph is connected.
SIAM J. Comput., 12(4):777–788.

Suciu, D. (2020).
Probabilistic databases for all.
In Suciu, D., Tao, Y., and Wei, Z., editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 19–31. ACM.

Suciu, D., Olteanu, D., Ré, C., and Koch, C. (2011).
Probabilistic Databases.
Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Valiant, L. G. (1979).
The complexity of enumeration and reliability problems.
SIAM J. Comput., 8(3):410–421.

Wegener, I. (2000).
Branching programs and binary decision diagrams: theory and applications.
SIAM, Philadelphia, PA, USA.