Circuits for Query Provenance

Dan Suciu\textsuperscript{1}

University of Washington

\textsuperscript{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

\( \mathcal{F} \) is the provenance of some FO sentence \( Q \):

- Complexity of the Weighted Model Counting problem for \( \mathcal{F} \).
- The size of an OBDD, or FBDD, or Decision-DNNF for \( \mathcal{F} \).

Knowledge Compilation [Darwiche and Marquis, 2002].

- Glaring omission: SAT.

Main message: from Logic (\( Q \)) to Algorithms (for \( \mathcal{F} \))
Weighted Model Counting
Weighted Model Counting

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

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

\textbf{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 $P(F)$;

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

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

Independence: $P(F_1 \land F_2) = P(F_1) \cdot P(F_2)$, 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:

$$F_n[\forall x Q] \overset{\text{def}}{=} \bigwedge_{i=1, n} F_n[Q[i/x]]$$
Provenance/Lineage/Grounding

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

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

$$F_n[Q_1 \land Q_2] \overset{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$

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

Results in this talk: No negation, single quantifier type ($\exists \exists \cdots$ or $\forall \forall \cdots$).
Provenance/Lineage/Grounding

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

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

$$F_n[Q_1 \land Q_2] \overset{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$

$$F_n[\neg Q] \overset{\text{def}}{=} \neg F_n[Q]$$
Provenance/Lineage/Grounding

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

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

$$F_n[Q_1 \land Q_2] \overset{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$

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

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

Given $Q$, what is the complexity of $P(F_n[Q])$?
Provenance/Lineage/Grounding

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

$$F_n[\forall x Q] \overset{\text{def}}{=} \bigwedge_{i=1}^{n} F_n[Q[i/x]]$$
$$F_n[Q_1 \land Q_2] \overset{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$
$$F_n[\neg Q] \overset{\text{def}}{=} \neg F_n[Q]$$
$$F_n[R(i, j, \ldots)] \overset{\text{def}}{=} X_{ij} \ldots \text{a Boolean variable associated to this atom}$$

Example  

$$Q = \forall x \forall y (R(x) \lor S(x, y))$$
Provenance/Lineage/Grounding

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

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

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

$$F_n[Q_1 \land Q_2] \stackrel{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$

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

Example

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

$$F_n[Q] = \bigwedge_{i,j=1}^{n} (X_i \lor Y_{ij})$$
Provenance/Lineage/Grounding

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

$$F_n[\forall x Q] \overset{\text{def}}{=} \bigwedge_{i=1}^n F_n[Q[i/x]] \quad F_n[Q_1 \land Q_2] \overset{\text{def}}{=} F_n[Q_1] \land F_n[Q_2]$$

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

Example

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

$$F_n[Q] = \bigwedge_{i,j=1}^n (X_i \lor Y_{ij})$$

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

Results in this talk: No negation, single quantifier type ($\exists \exists \cdots$ or $\forall \forall \cdots$)
Syntactic Feature #1: Hierarchy

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

**Definition**

$Q$ is **hierarchical** if $at(x) \subseteq at(y)$, or $at(x) \supseteq at(y)$, or $at(x) \cap at(y) = \emptyset$. 
Syntactic Feature #1: Hierarchy

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

**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, $P(F_n[Q])$ in $PTIME$, otherwise $\#P$-hard.
Syntactic Feature #1: Hierarchy

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

**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, $P(F_n[Q])$ in PTIME, otherwise $\#P$-hard.

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

**Non-Hierarchical:** $\exists x \exists y (R(x) \land S(x, y) \land T(y))$ is $\#P$-hard.

---

$^2$Reduction from $\#F$ for $F = \bigvee_{(i,j) \in E} X_i \land Y_j$ [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 $Q$, $P(F_n[Q])$ 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:

**UCQ:** $\exists x \exists y (R(x) \land S(x, y)) \lor \exists u \exists v (S(u, v) \land T(v))$

**Dual:** $\forall x \forall y (R(x) \lor S(x, y)) \land \forall u \forall v (S(u, v) \lor T(v))$
Discussion

- Main take away: from static analysis on $Q$ to complexity of $\mathbf{P}(F_n[Q])$.

- Extension to UCQ\(^\infty\) (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

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

G=

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**

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

\[ G = \]

- \( X \) 1 1
- \( Y \) 0 1 0
- \( Z \) 1 0 0
- \( U \) 0 1 0
- \( V \) 1 1

Read-once property

**OBDD**: fixed variable order

**Decision-DNNF**

\[ F = \ldots \]

\[ G = \]

- \( X \) 1
- \( Y \) 0 1
- \( Z \) 0 0
- \( U \) 1 0
- \( W \) 0
- \( V \) 1

Decomposable \( \land \)-nodes.
Definitions: FBDDs, OBDDs, Decision-DNNFs

FBDD
\[ F = \overline{X} \overline{Y} Z + \overline{X} Y U + X \overline{Z} U + X Y Z. \]

G=

- Read-once property
- OBDD: fixed variable order

Decision-DNNF
\[ F = \cdots \]

G=

- Decomposable \( \land \)-nodes.
- \( \exists \) FBDD of size \( \leq 2|G|2^{\log^2 |G|} \)
Definitions: FBDDs, OBDDs, Decision-DNNF

**FBDD**

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

**OBDD:** fixed variable order

Read-once property

Decomposable \( \land \)-nodes.

\[ \exists \text{ FBDD of size } \leq 2|G|2^\log_2|G| \]

[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 \( F \mapsto \text{BDD for } F \) [Darwiche and Marquis, 2002].

Query compilation Fix \( Q \). \( n \mapsto \text{BDD for } F_n[Q] \) [Jha and Suciu, 2013].
OBDDs
OBDD

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

---

\(^3\)Product automaton.

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

\[ Q_1 = \forall x \forall y (R(x) \lor S(x, y)) \]
\[ F_2 = (R_1 \lor S_{11})(R_1 \lor S_{12})(R_2 \lor S_{21})(R_2 \lor S_{22}) \]
Example

\[ Q_1 = \forall x \forall y (R(x) \lor S(x, y)) \]

\[ F_2 = (R_1 \lor S_{11})(R_1 \lor S_{12})(R_2 \lor S_{21})(R_2 \lor S_{22}) \]

Size = \( O(n) \).
Example

\[ Q_1 = \forall x \forall y (R(x) \lor S(x, y)) \]

\[ F_2 = (R_1 \lor S_{11})(R_1 \lor S_{12})(R_2 \lor S_{21})(R_2 \lor S_{22}) \]

\[ Q_2 = \forall u \forall v (S(u, v) \lor T(u)) \]

Size = \( O(n) \).
Example

\[ Q_1 = \forall x \forall y (R(x) \lor S(x, y)) \]
\[ F_2 = (R_1 \lor S_{11})(R_1 \lor S_{12})(R_2 \lor S_{21})(R_2 \lor S_{22}) \]

Size = \( O(n) \).

\[ Q_2 = \forall u \forall v (S(u, v) \lor T(u)) \]

Size = \( O(n) \).
Example

\[ Q_1 = \forall x \forall y (R(x) \lor S(x, y)) \]
\[ F_2 = (R_1 \lor S_{11})(R_1 \lor S_{12})(R_2 \lor S_{21})(R_2 \lor S_{22}) \]

Size = \( O(n) \).

\[ Q_2 = \forall u \forall v (S(u, v) \lor T(u)) \]

Size = \( O(n) \).

Same variable order: synthesize OBDD for \( Q_1 \land Q_2 \) of size = \( O(n) \).
A Lower Bound

\[ H_0 \overset{\text{def}}{=} \forall x \forall y (R(x) \lor S(x, y) \lor T(y)) \]
A Lower Bound

\[ H_0 \overset{\text{def}}{=} \forall x \forall y (R(x) \lor S(x, y) \lor T(y)) \]

// drop \( \forall \cdots \)
A Lower Bound

\[ H_0 \overset{\text{def}}{=} R(x) \lor S(x, y) \lor T(y) \]  

// drop \( \forall \cdots \)
A Lower Bound

\[ H_0 \overset{\text{def}}{=} R(x) \lor S(x, y) \lor T(y) \]

\[ H_1 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor T(y_1)) \]
A Lower Bound

\[ H_0 \overset{\text{def}}{=} R(x) \lor S(x, y) \lor T(y) \] // drop \( \forall \cdots \)

\[ H_1 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor T(y_1)) \]

\[ H_2 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor T(y_2)) \]
A Lower Bound

\[ H_0 \overset{\text{def}}{=} R(x) \lor S(x, y) \lor T(y) \quad // \text{drop } \forall \cdots \]

\[ H_1 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor T(y_1)) \]

\[ H_2 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor T(y_2)) \]

\[ \cdots \]

\[ H_k \overset{\text{def}}{=} \ldots \]

\( H_k \) for \( k \geq 1 \) is hierarchical.
A Lower Bound

\[ H_0 \overset{\text{def}}{=} R(x) \lor S(x, y) \lor T(y) \]  
\[ H_1 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor T(y_1)) \]
\[ H_2 \overset{\text{def}}{=} (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor T(y_2)) \]
\[ \cdots \]
\[ H_k \overset{\text{def}}{=} \cdots \]

\( H_k \) for \( k \geq 1 \) is hierarchical.

**Theorem ([Beame et al., 2017])**

Any FBDD for \( H_k \) has size \( \geq \frac{(2^n - 1)}{n} \); 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:

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

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

**Example** every $H_k$ has a $k$-inversion.

$$H_2 = (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor T(y_2))$$

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 \( O(n^{\text{arity}}) \) (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 $P(Q)$ is in PTIME.

What about the converse?
The Inclusion/Exclusion Formula

If \( Q \) is a query without inversion then \( P(Q) \) is in PTIME.

What about the converse?

\[
Q_V \overset{\text{def}}{=} (R(x_0) \lor S(x_0, y_0)) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x) \lor T(y))
\]

Has inversion, yet \( P(Q_V) \) in PTIME:
### The Inclusion/Exclusion Formula

If $Q$ is a query without inversion then $P(Q)$ is in PTIME.

What about the converse?

$$Q_V \overset{\text{def}}{=} (R(x_0) \lor S(x_0, y_0)) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x) \lor T(y))$$

Has inversion, yet $P(Q_V)$ in PTIME:

$$Q_V = R(x)(R(x_0) \lor S(x_0, y_0))(S(x_1, y_1) \lor T(y_1)) \lor (R(x_0) \lor S(x_0, y_0))(S(x_1, y_1) \lor T(y_1))T(y)$$
The Inclusion/Exclusion Formula

If $Q$ is a query without inversion then $P(Q)$ is in PTIME.

What about the converse?

$$Q_V \overset{\text{def}}{=} (R(x_0) \lor S(x_0, y_0)) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x) \lor T(y))$$

Has inversion, yet $P(Q_V)$ in PTIME:

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

$$= R(x) \land (S(x_1, y_1) \lor T(y_1)) \lor (R(x_0) \lor S(x_0, y_0)) \land T(y)$$
The Inclusion/Exclusion Formula

If $Q$ is a query without inversion then $P(Q)$ is in PTIME.

What about the converse?

$$Q_V \overset{\text{def}}{=} (R(x_0) \lor S(x_0, y_0)) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x) \lor T(y))$$

Has inversion, yet $P(Q_V)$ in PTIME:

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

$$= R(x) \land (S(x_1, y_1) \lor T(y_1)) \lor (R(x_0) \lor S(x_0, y_0) \land T(y))$$

$$P(Q_V) = P(R(x) \land (S(x_1, y_1) \lor T(y_1))) + P((R(x_0) \lor S(x_0, y_0) \land T(y)))$$

$$- P((R(x) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x_0) \lor S(x_0, y_0)) \land T(y)))$$

$\equiv R(x) \land T(y)$
The Inclusion/Exclusion Formula

If $Q$ is a query without inversion then $P(Q)$ is in PTIME.

What about the converse?

$Q_V \overset{\text{def}}{=} (R(x_0) \lor S(x_0, y_0)) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x) \lor T(y))$

Has inversion, yet $P(Q_V)$ in PTIME:

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

$= R(x) \land (S(x_1, y_1) \lor T(y_1)) \lor (R(x_0) \lor S(x_0, y_0)) \land T(y)$

$P(Q_V) = P(R(x) \land (S(x_1, y_1) \lor T(y_1))) + P((R(x_0) \lor S(x_0, y_0)) \land T(y))$

$- P((R(x) \land (S(x_1, y_1) \lor T(y_1)) \land (R(x_0) \lor S(x_0, y_0))) \land T(y)))$

$\equiv R(x) \land T(y)$

All three queries inversion-free: $P(Q_V)$ 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 \( P(F_n[Q]) \) 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 $P(F_n[Q])$ 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

\[ H_3 = (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor S_3(x_2, y_2)) \land (S_3(x_3, y_3) \lor T(y_3)) \]

\[ \text{def} = h_{30} \quad \text{def} = h_{31} \quad \text{def} = h_{32} \quad \text{def} = h_{33} \]
Syntactic Feature #3: Cancellations

\[ H_3 = \left( R(x_0) \lor S_1(x_0, y_0) \right) \land \left( S_1(x_1, y_1) \lor S_2(x_1, y_1) \right) \land \left( S_2(x_2, y_2) \lor S_3(x_2, y_2) \right) \land \left( S_3(x_3, y_3) \lor T(y_3) \right) \]

\[ \text{def} \overset{h_{30}}{=} \quad \text{def} \overset{h_{31}}{=} \quad \text{def} \overset{h_{32}}{=} \quad \text{def} \overset{h_{33}}{=} \]

\[ Q_W \overset{\text{def}}{=} (h_{30} \land h_{32}) \lor (h_{30} \land h_{33}) \lor (h_{31} \land h_{33}) \]

Theorem ([Beame et al., 2017])

1. \( P(Q_W) \) in PTIME.
2. FBDD for \( Q_W \) has size \( 2^{\Omega(n)} \).
3. Decision-DNNF has size \( 2^{\Omega(\sqrt{n})} \).
Syntactic Feature #3: Cancellations

\[ H_3 = (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor S_3(x_2, y_2)) \land (S_3(x_3, y_3) \lor T(y_3)) \]

\[ Q_W \overset{\text{def}}{=} (h_{30} \land h_{32}) \lor (h_{30} \land h_{33}) \lor (h_{31} \land h_{33}) \]

Theorem ([Beame et al., 2017])

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

\[ P(Q_W) = P(h_{30} \land h_{32}) + P(h_{30} \land h_{33}) + P(h_{31} \land h_{33}) \]
\[ - P(h_{30} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{33}) \]
\[ + P(h_{30} \land h_{31} \land h_{32} \land h_{33}) \]
**Syntactic Feature #3: Cancellations**

\[
H_3 = (R(x_0) \lor S_1(x_0, y_0)) \land (S_1(x_1, y_1) \lor S_2(x_1, y_1)) \land (S_2(x_2, y_2) \lor S_3(x_2, y_2)) \land (S_3(x_3, y_3) \lor T(y_3))
\]

\[
\begin{align*}
    &\text{def } h_{30} = h_30 \\
    &\text{def } h_{31} = h_31 \\
    &\text{def } h_{32} = h_32 \\
    &\text{def } h_{33} = h_33
\end{align*}
\]

\[
Q_W \overset{\text{def}}{=} (h_{30} \land h_{32}) \lor (h_{30} \land h_{33}) \lor (h_{31} \land h_{33})
\]

**Theorem ([Beame et al., 2017])**

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

\[
P(Q_W) = P(h_{30} \land h_{32}) + P(h_{30} \land h_{33}) + P(h_{31} \land h_{33}) \\
- P(h_{30} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{33}) \\
+ P(h_{30} \land h_{31} \land h_{32} \land h_{33}) \quad // \text{ hard query } H_3 \text{ cancels out}
\]
Syntactic Feature #3: Cancellations

\[ H_3 = \left( R(x_0) \lor S_1(x_0, y_0) \right) \land \left( S_1(x_1, y_1) \lor S_2(x_1, y_1) \right) \land \left( S_2(x_2, y_2) \lor S_3(x_2, y_2) \right) \land \left( S_3(x_3, y_3) \lor T(y_3) \right) \]

\[ \text{def } = h_{30} \]

\[ \text{def } = h_{31} \]

\[ \text{def } = h_{32} \]

\[ \text{def } = h_{33} \]

\[ Q_W \overset{\text{def}}{=} \left( h_{30} \land h_{32} \right) \lor \left( h_{30} \land h_{33} \right) \lor \left( h_{31} \land h_{33} \right) \]

**Theorem ([Beame et al., 2017])**

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

\[
P(Q_W) = P(h_{30} \land h_{32}) + P(h_{30} \land h_{33}) + P(h_{31} \land h_{33})
- P(h_{30} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{32} \land h_{33}) - P(h_{30} \land h_{31} \land h_{33})
+ P(h_{30} \land h_{31} \land h_{32} \land h_{33})
\]

// hard query \( H_3 \) cancels out

FBDD for \( Q_W \) ⇒ multi-output FBDD for \( h_{30}, h_{31}, h_{32}, h_{33} \) ⇒ 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]
Binary decision diagrams.

Amarilli, A. and Ceylan, İ. İ. (2020).
A dichotomy for homomorphism-closed queries on probabilistic graphs.

Exact model counting of query expressions: Limitations of propositional methods.

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.

Complexity of generalized satisfiability counting problems.

The dichotomy of probabilistic inference for unions of conjunctive queries.

Decomposable negation normal form.
A knowledge compilation map.

Query processing on probabilistic data: A survey.

Dpll with a trace: From sat to knowledge compilation.
In *IJCAI*, pages 156–162.

The language of search.

Knowledge compilation meets database theory: Compiling queries to decision diagrams.

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

The complexity of counting cuts and of computing the probability that a graph is connected.

Probabilistic databases for all.
Probabilistic Databases. 

The complexity of enumeration and reliability problems. 

Branching programs and binary decision diagrams: theory and applications. 
SIAM, Philadelphia, PA, USA.