Query Compilation: the View from the Database Side

Dan Suciu
University of Washington

Joint with: P. Beame, N. Dalvi, A. Jha, J. Li, S. Roy

The View from the Database Side

- The large Boolean formula F is generated by some much smaller program Q

- Each Q defines a different problem:
  - KC, or SAT-solver, or … for formulas F produced by Q
  - *Data complexity*: query Q, database D

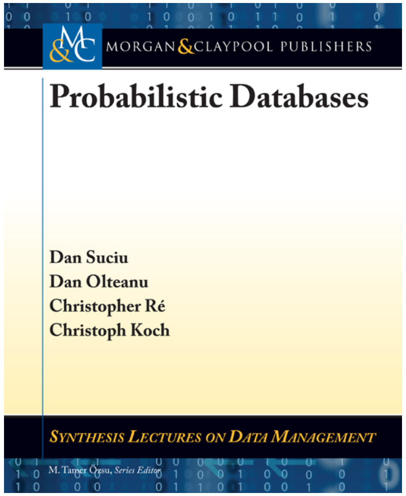
- This talk:
  - Q is a sentence in FO($\wedge, \vee, \forall$)
  - The problem is model counting #F and KC

- Color code: blue=fixed, red=input

Sources

- Jha, S., ICDT 2011
- Dalvi, S., JACM 2012
- Beame, Li, Roy, S. UAI'2013
- Beame, Li, Roy, S. ICDT'2014

- Background on probabilistic databases:



Model Counting

- Given Boolean formula F, compute the number of models #F

Example:
F = X1 X2 ∨ X2 X3 ∨ X3 X1

#F = 4

X1	X2	X3	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

[Valiant] #P-hard, even for 2CNF

Probability of a Formula

- Each variable X has a probability p(X);
- P(F) = probability that F=true

Example:
$F = X1\ X2 \lor X2\ X3 \lor X3\ X1$

$$P(F) = (1-p1)*p2*p3 + p1*(1-p2)*p3 + p1*p2*(1-p3) + p1*p2*p3$$

$P(F) = \#F / 2^n$, when $p(X) = ½$ for all X

X1	X2	X3	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Grounding of an FO Sentence

Let Q, be an FO sentence, n a natural number.

Def The grounding, $F_n(Q)$ is:

- $F_n(\forall x Q) = \bigwedge_{i \in [n]} F_n(Q[i/x])$
- $F_n(\exists x Q) = \bigvee_{i \in [n]} F_n(Q[i/x])$
- $F_n(Q_1 \text{ op } Q_2) = F_n(Q_1) \text{ op } F_n(Q_1)$ op = $\wedge, \vee, \neg$

Example: $Q = \forall d\ (\mathrm{Rain}(d) \Rightarrow \mathrm{Cloudy}(d))$ $n = 7$

$F_7(Q) = (\mathrm{Rain}_1 \Rightarrow \mathrm{Cloudy}_1) \wedge \ldots \wedge (\mathrm{Rain}_7 \Rightarrow \mathrm{Cloudy}_7)$

Probabilistic Databases, Markov Logic Networks, …

Research Question

Given an FO sentence Q determine the complexity of $P(F_n(Q))$; PTIME? #P-hard?

Data complexity: assume fixed Q, input given by n

In practice, Q is small:

- SQL query: 10-20 joins
- MLN's: 10-15 rules

Next: knowledge compilation for $F_n(Q)$

Outline

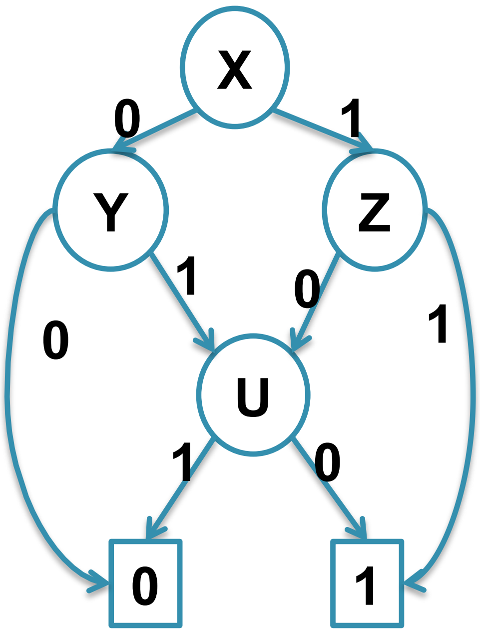
- Problem statement
- Review: FBDD, Decision-DNNF
- Hard Queries
- Easy Queries
- Hard/Easy Queries
- Conclusion

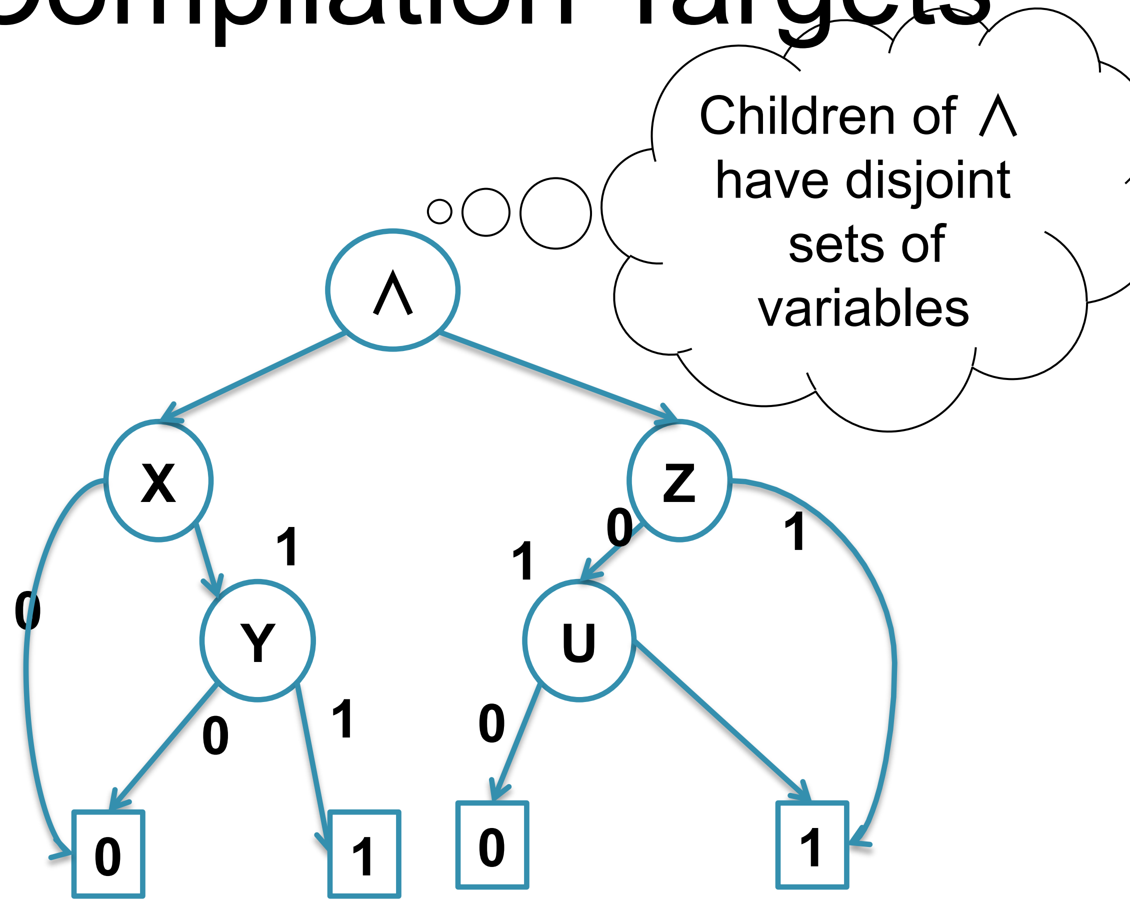
Knowledge Compilation Targets

- FBDD (Free Binary Decision Diagram)
- Decision-DNNF (Decomposable Negation Normal Form)

P(F) computable in linear time in the KC

Knowledge Compilation Targets





FBDD:
Decision-, sink-nodes

Decision-DNNF
add: decomposable-∧-nodes

DPLL and Knowledge Compilation

Fact: Trace of full-search DPLL → KC:

- Basic DPLL
  → decision trees
- DPLL + caching
  → FBDD
- DPLL + caching + components
  → decision-DNNF

Our interest in KC: lower bounds for DPLL.

Research Question

Given an FO sentence Q

Determine the complexity of $P(F_n(Q))$; PTIME? #P-hard?
Determine the size of KC for $F_n(Q)$

"Data complexity": fixed Q, input given by n

Outline

- Problem statement
- Review: FBDD, Decision-DNNF
- Hard Queries
- Easy Queries
- Hard/Easy Queries
- Conclusion

Background

Theorem [Bollig&Wegener'98] Any FBDD for $F = \bigwedge_{(i,j) \in E} (R_i \vee T_j)$ has size $2^{\Omega(\sqrt{n})}$

Where

- $R_1, \ldots, R_n, T_1, \ldots, T_n$ = Boolean Variables
- E = …(some complex relation $\subseteq [n] \times [n]$)

For p = a prime, $n = p^2$,
$E = \{(1+i, 1+j) \mid i = a+bp, j=c+dp, c=a+bd \bmod p\}$
$|E| = p^3 = n^{3/2}$

H_0 is Hard for FBDDs

$$H_0 = \forall x \forall y\, (R(x) \vee S(x,y) \vee T(y))$$

$$F_n(H_0) = \bigwedge_{i \in [n],\, i \in [n]} (R_i \vee S_{ij} \vee T_j)$$

By [B&W], any FBDD has size $2^{\Omega(\sqrt{n})}$. We strengthen:

Th. [Beame'14] Any FBDD for $F_n(H_0)$ has size $\geq 2^{n-1}/n$.

What about Decision-DNNFs?

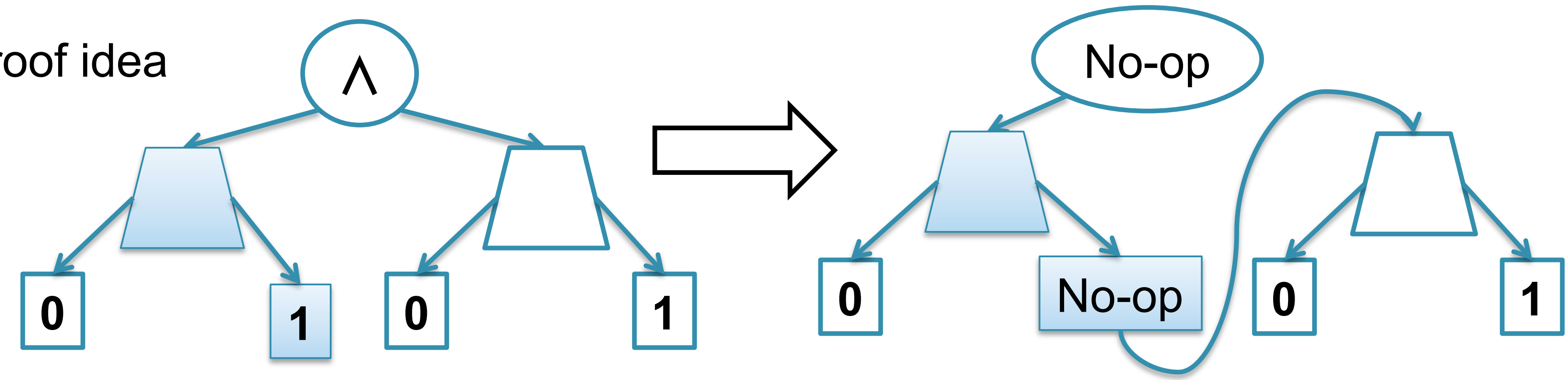
Decision-DNNF to FBDD

Optimal [Razgon]

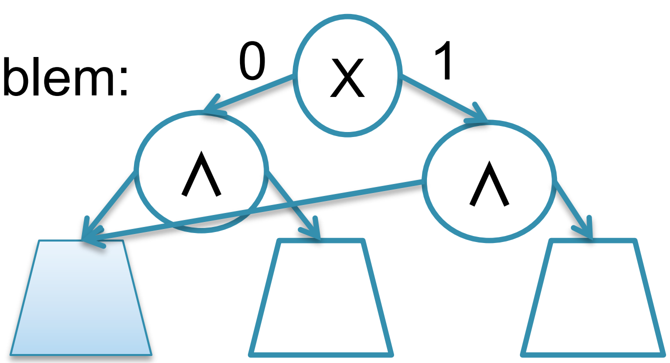
We proved this in [Beame'13]:

Theorem If F has a Decision-DNNF with N nodes, then F has an FBDD with at most $N^{1+\log(N)}$ nodes.

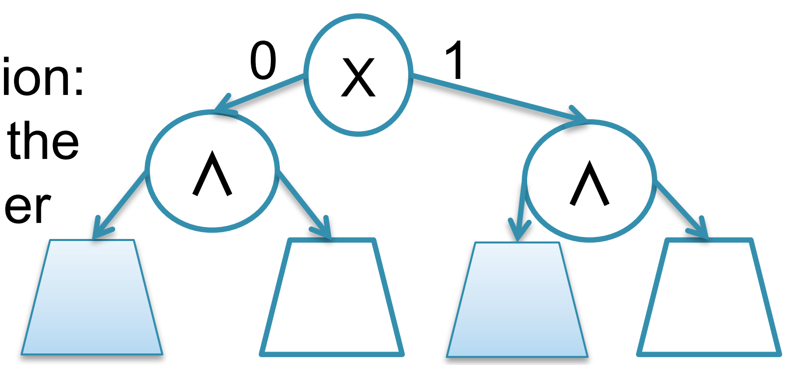
Proof idea



Problem:



Solution: copy the smaller child



H_0 is Hard for Decision-DNNFs

Corollary Any Decision-DNNF for $F_n(H_0)$ has size $2^{\Omega(\sqrt{n})}$

Proof. N-node Decision-DNNF to $N^{1+\log(N)}$ nodes FBDD.

$$N^{1+\log(N)} > 2^{n-1}/n \ ,$$
$$\log(N) + \log^2(N) > n - 1 - \log(n)$$
$$\log^2(N) = \Omega(n)$$
$$\log(N) = \Omega(\sqrt{n})$$

Generalization

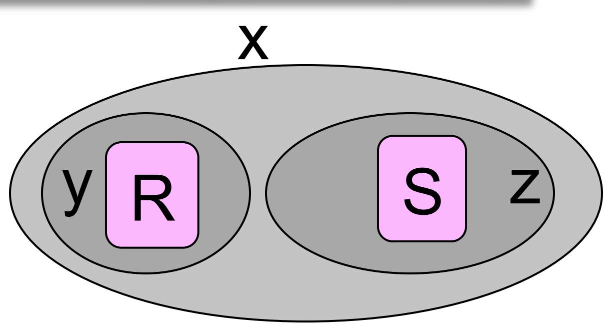
C = a positive clause;   at(x) = set of atoms containing variable x

Definition C is hierarchical if forall x, y:
$at(x) \subseteq at(y)$ or $at(x) \supseteq at(y)$ or $at(x) \cap at(y) = \emptyset$

A query Q in $FO(\wedge, \vee, \forall)$ is hierarchical if all its clauses are

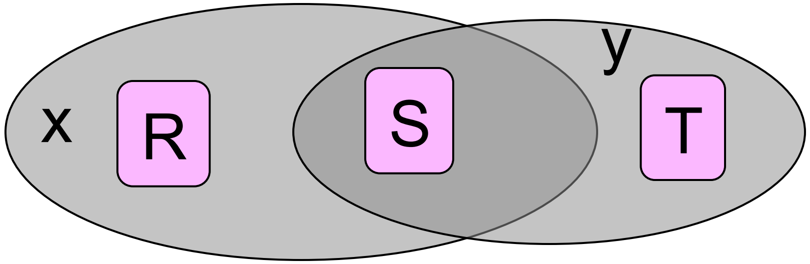
Hierarchical

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



Non-hierarchical

$H_0 = R(x) \vee S(x,y) \vee T(y)$



Thrm. If Q is non-hierarchical, any Decision-DNNF has size $2^{\Omega(\sqrt{n})}$.

Discussion

Exponential size of KC not surprising, because:

Theorem $\#F_n(H_0)$ is #P-hard.
(Same holds for any non-hierarchical Q)

Proof:
[Provan&Ball'82] PP2CNF is #P-complete:

$$F = \bigwedge_{(i,j)\in E} (R_i \vee T_j)$$

Research Question

Given an FO sentence Q in FO(∧, ∨, ∀)

	Non-hierarchical Q (e.g. H_0)
Is $P(F_n(Q))$ in PTIME? Or #P-hard?	#P-hard
How large is Knowledge Compilation for $F_n(Q)$?	decision-DNNF has size $2^{\Omega(\sqrt{n})}$

What about hierarchical queries ?

Outline

- Problem statement
- Review: FBDD, Decision-DNNF
- Hard Queries
- Easy Queries
- Hard/Easy Queries
- Conclusion

Easy Queries

- Let Q in FO($\wedge, \vee, \forall$). Then F_n(Q) has a polynomial-size OBDD iff it is both hierarchical *and* inversion-free.

- Recall: OBDD = FBDD with fixed variable order Π

Inversion-Free Queries

Definition An inversion in Q is a sequence of co-occurring vars:

$(x_0,y_0), (x_1,y_1), \ldots, (x_k,y_k)$, such that:

- $at(x_0) \not\subseteq at(y_0)$, $at(x_1)=at(y_1), \ldots, at(x_{k-1})=at(y_{k-1})$, $at(x_k) \not\supseteq at(y_k)$
- For all i=1,..,k-1 there exists two atoms in Q of the form:
  $S_i(\ldots,x_{i-1},\ldots,y_{i-1},\ldots)$ and $S_i(\ldots,x_i, \ldots, y_i, \ldots)$

Inversion-free implies hierarchical, but converse fails

$Q=[R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(x_1)]$

Inversion-free

Inversion

$H_1=[R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

Easy Queries

Theorem [Jha&S.11] Let Q in FO($\wedge, \vee, \forall$)

1. If Q has inversion then OBDD for $F_n(Q)$ has size $2^{\Omega(n)}$
2. Else, $F_n(Q)$ has OBDD of width $2^{\#atoms(Q)}$ (linear size)

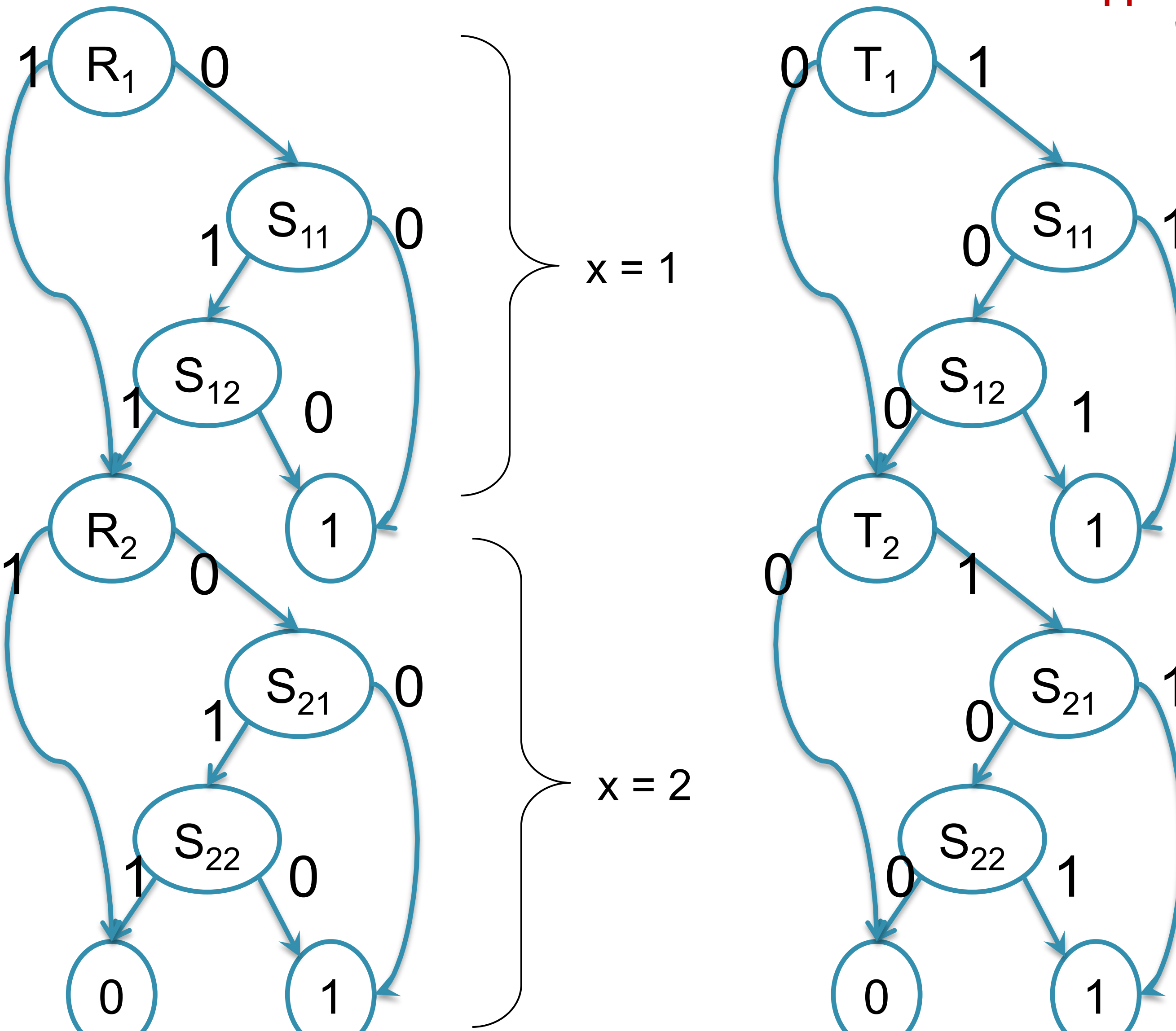
Proof (part 2 only – next slide)

$C_1 = R(x) \vee S(x,y)$ $\wedge$ $C_2 = T(x') \wedge S(x',y')$ $=$ $Q = [R(x) \vee S(x,y)] \wedge [T(x') \vee S(x',y')]$

$F(C_1) = (R_1 \vee S_{11}) \wedge (R_1 \vee S_{12}) \wedge (R_2 \vee S_{21}) \wedge (R_2 \vee S_{22})$

$n = 2$

$\Pi = \underbrace{R_1 T_1 S_{11} S_{12}}_{x=1} \underbrace{R_2 T_2 S_{21} S_{22}}_{x=2}$



Same variable order Π in both OBDDs!

OBDD for $Q = C_1 \wedge C_2$ has width = width1 × width2

Research Question

Given an FO sentence Q in FO(∧, ∨, ∀)

	Non-hierarchical Q (e.g. H_0)	Inversion-free Q
Is P(F_n(Q)) in PTIME? Or #P-hard?	#P-hard	PTIME
How large is Knowledge compilation for F_n(Q)?	decision-DNNF has size $2^{\Omega(\sqrt{n})}$	Poly-size

What about hierarchical queries w/ inversion?

Outline

- Problem statement
- Review: FBDD, Decision-DNNF
- Hard Queries
- Easy Queries
- Hard/Easy Queries
- Conclusion

Easy/Hard Queries

Will describe a class of queries Q such that:

- Computing probability is easy ($P(F_n(Q))$ in PTIME)
- Compiling $F_n(Q)$ is hard (Exponential-size Decision-DNNF)

- Implication: inherent limitation of DPLL-based algorithms for model counting

The Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

Non-hierarchical

Hierarchical

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

Inversion: $at(x_0) \supset at(y_0)$, $at(x_1) \subset at(y_1)$

$H_2 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee T(y_2)]$

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

Longer inversion:
$at(x_0) \supset at(y_0)$, $at(x_1) = at(y_1)$, $at(x_2) = at(y_2)$, $at(x_3) \subset at(y_3)$:

. . .

Easy/Hard Queries

The clauses of H_k (dropping $\forall$)

$$
\begin{aligned}
H_{k0} &= R(x_0) \vee S_1(x_0,y_0) \\
H_{k1} &= S_1(x_1,y_1) \vee S_2(x_1,y_1) \\
H_{k2} &= S_2(x_2,y_2) \vee S_3(x_2,y_2) \\
&\dots \\
&\dots \\
H_{kk} &= S_k(x_k,y_k) \vee T(y_k)
\end{aligned}
$$

$f(Z_0, Z_1, \dots, Z_k)$ = a Boolean function in k+1 variables
$Q = f(H_{k0}, H_{k1}, \dots, H_{kk})$,

Example: $f = Z_0 \wedge Z_1 \wedge \dots \wedge Z_k$ then $f(H_{k0}, H_{k1}, \dots, H_{kk}) = H_k$

Easy/Hard Queries

$f(Z_0, Z_1, \ldots, Z_k)$ = Boolean function in k+1 vars

$Q = f(H_{k0}, H_{k1}, \ldots, H_{kk})$

Theorem [Beame'14] Any FBDD for $F_n(Q)$ has size $2^{\Omega(n)}$
Any Decision-DNNF has size $\geq 2^{\Omega(\sqrt{n})}$.

Theorem [Dalvi'12] Assume f is monotone, let L be its DNF lattice, μ its Möbius function

- If $\mu(\hat{0}, \hat{1}) = 0$ then $P(F_n(Q))$ is in PTIME
- If $\mu(\hat{0}, \hat{1}) \neq 0$ then $P(F_n(Q))$ is in #P-hard

Proof Highlights

Theorem [Beame'14] Any FBDD for $F_n(Q)$ has size $2^{\Omega(n)}$

Proof part 1: any FBDD for $F_n(H_k)$ has size $\geq 2^{n-1}/n$

Proof part 2:
Convert a N-node FBDD for $F_n(f(H_{k0}, H_{k1}, \ldots, H_{kk}))$,
to a $O(n^3 N)$-node multi-output FBDD
for k+1 functions: $F_n(H_{k0})$, $F_n(H_{k1})$,…, $F_n(H_{kk})$

Convert the latter to an FBDD for $F_n(H_k)$

Proof Highlights

Theorem [Dalvi'12] If μ = 0 then $P(F_n(Q))$ is in PTIME

By example on $f = Z_0 \wedge Z_2 \vee Z_0 \wedge Z_3 \vee Z_1 \wedge Z_3$

$$
\begin{aligned}
Q_W = \; & H_{30} \wedge H_{32} \vee && /*\ Q_1\ */ \\
& H_{30} \wedge H_{33} \vee && /*\ Q_2\ */ \\
& H_{31} \wedge H_{33} && /*\ Q_3\ */
\end{aligned}
$$

Recall:
$H_3 = H_{30} \wedge \ldots \wedge H_{33}$

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \wedge Q_2) - P(Q_2 \wedge Q_3) - \sout{P(Q_1 \wedge Q_3)} \\
& + \sout{P(Q_1 \wedge Q_2 \wedge Q_3)}
\end{aligned}
$$

= H_3 (hard !)

Also = H_3

The remaining terms are inversion-free, hence PTIME

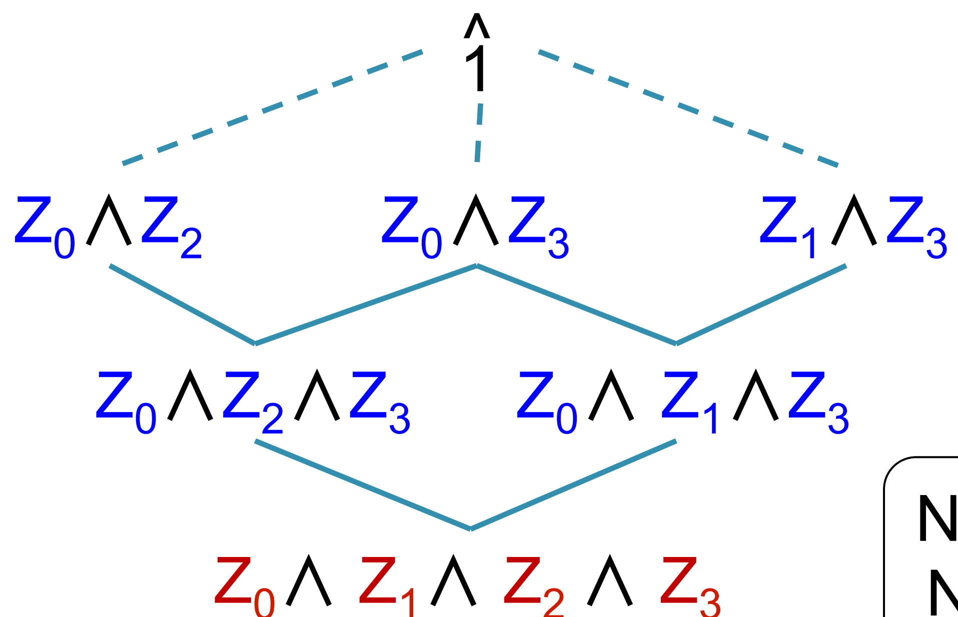
The DNF Lattice

Definition.
The DNF lattice **L** of a monotone DNF $f = t_1 \vee t_2 \vee \ldots$ is:

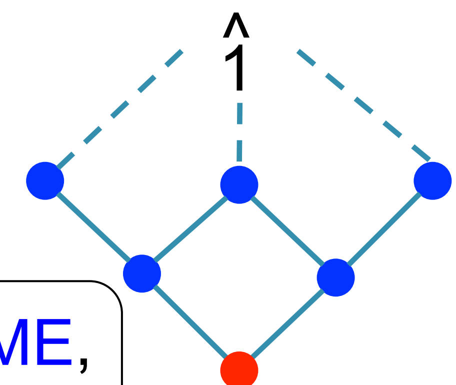
- Elements of L are terms $t_{i1} \wedge t_{i2} \wedge \ldots$;
- Order is logical implication

$f = Z_0 \wedge Z_2 \vee Z_0 \wedge Z_3 \vee Z_1 \wedge Z_3$

Fact: if μ = 0 then $P(F_n(Q))$ is in PTIME



Nodes • in PTIME,
Nodes • #P hard.



Research Question

Given an FO sentence Q

	Non-hierarchical Q (e.g. H_0)	Inversion-free Q	$Q = f(H_{k0},\ldots,H_{kk})$
Is $P(F_n(Q))$ in PTIME? Or #P-hard?	#P-hard	PTIME	PTIME or #P-hard
How large is Knowledge Compilation for $F_n(Q)$?	size $2^{\Omega(\sqrt{n})}$	Poly-size	size $2^{\Omega(\sqrt{n})}$

Outline

- Problem statement
- Review: FBDD, Decision-DNNF
- Hard Queries
- Easy Queries
- Hard/Easy Queries
- Conclusion

The View from the Database Side

High level idea:

- Boolean function F generated by small program Q

For FO sentence Q in FO($\wedge, \vee, \forall$)

- Hard/hard
- Easy/easy
- Easy/hard

Separation of grounded v.s. lifted inference:

- Limitation of DPLL-based algorithms
- Inclusion/exclusion possible only on the FO sentence

Möbius Über Alles