Introduction to Knowledge Compilation

Florent Capelli

CRIStAL, Université de Lille & Inria & CNRS France

Research School on Knowledge Compilation, ENS Lyon, December 4th-8th

This research school

- **Florent Capelli**, Université Lille 3, CRIStAL, LINKS, florent.capelli@univ-lille3.fr
- **Jean-Marie Lagniez**, Université d'Artois, CRIL, lagniez@cril.fr
- **Pierre Marquis**, Université d'Artois, CRIL, marquis@cril.univ-artois.fr

Schedule

- Monday, Tuesday, Wednesday (Amphi B): 8:30-12:00, 13:30-15:45
- Thursday (Amphi B): 8:30-12:00, **Free afternoon**
- Friday: 8:30-12:00 (**Amphi Schrödinger**), 13:30-14:30 (Amphi B)

All info and material: `http://researchers.lille.inria.fr/~fcapelli/research_school.html`

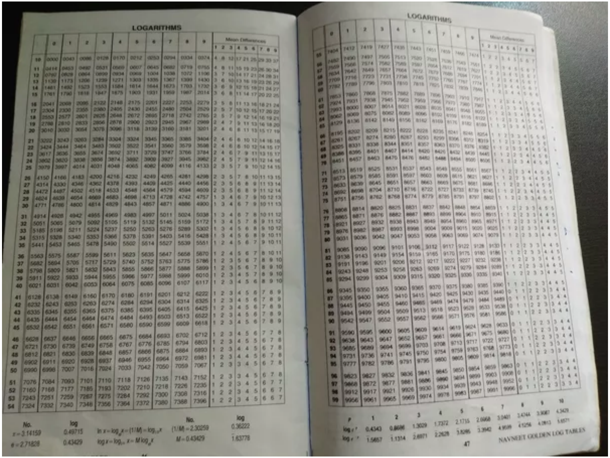
Content

- Today: introduction to the main concepts in knowledge compilation (FC).
- Tuesday: from SAT-solvers to compilers (JML+PM).
- Wednesday: preprocessing for model counting and compilation (JML+PM).
- Thursday: theoretical algorithms to compile efficiently (F.C).
- Friday: lower bounds (FC), Bayesian Networks (JML).

What is knowledge compilation?

A preprocessing to change the representation of the data to make it easier to analyse.

Log Tables



Log Tables

LOGARITHMIC TABLE

	0	1	2	3	4	5	6	7	8	9	MEAN DIFFERENCE								
											1	2	3	4	5	6	7	8	9
1.0	0.0000	0.0043	0.0086	0.0128	0.0170	0.0212	0.0253	0.0294	0.0334	0.0374	4	8	12	17	21	25	29	33	37
1.1	0.0414	0.0453	0.0492	0.0531	0.0569	0.0607	0.0645	0.0682	0.0719	0.0755	4	8	11	15	19	23	27	30	34
1.2	0.0792	0.0828	0.0864	0.0899	0.0934	0.0969	0.1004	0.1038	0.1072	0.1106	3	7	10	14	17	21	24	28	31
1.3	0.1139	0.1173	0.1206	0.1239	0.1271	0.1303	0.1335	0.1367	0.1399	0.1430	3	6	10	13	16	19	23	26	29
1.4	0.1461	0.1492	0.1523	0.1553	0.1584	0.1614	0.1644	0.1673	0.1703	0.1732	3	6	9	12	15	18	21	24	27
1.5	0.1761	0.1790	0.1818	0.1847	0.1875	0.1903	0.1931	0.1959	0.1987	0.2014	3	6	8	11	14	17	20	22	25
1.6	0.2041	0.2068	0.2095	0.2122	0.2148	0.2175	0.2201	0.2227	0.2253	0.2279	3	5	8	11	13	16	18	21	24
1.7	0.2304	0.2330	0.2355	0.2380	0.2405	0.2430	0.2455	0.2480	0.2504	0.2529	2	5	7	10	12	15	17	20	22
1.8	0.2553	0.2577	0.2601	0.2625	0.2648	0.2672	0.2695	0.2718	0.2742	0.2765	2	5	7	9	12	14	16	19	21
1.9	0.2788	0.281	0.2833	0.2856	0.2878	0.2900	0.2923	0.2945	0.2967	0.2989	2	4	7	9	11	13	16	18	20

$$\sqrt[5]{1234} = 10^{\frac{1}{5}(\log_{10}(1.234)+3)}$$

$$\approx 10^{\frac{3.0913}{5}}$$ by looking it in the table

$$\approx 10^{0.61826}$$

$$\approx 4.1520$$ by looking it in an antilog table.

Demo

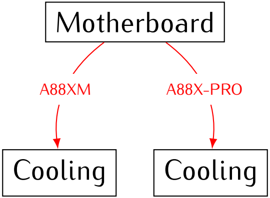
Processor config demo.

Constraints

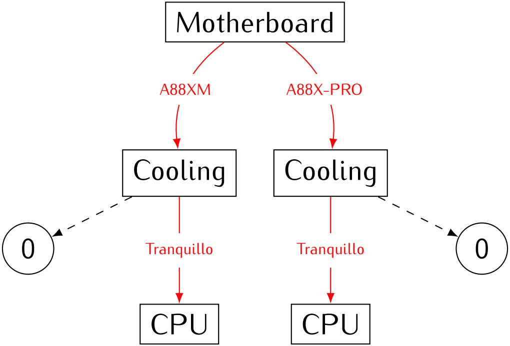
How would you represent the constraints of the previous demo?

- List of constraints: natural but finding a good configuration is NP-hard.
- A better datastructure?
  - Can you find the forced values quickly?
  - The best price?
- Is your datastructure small enough?

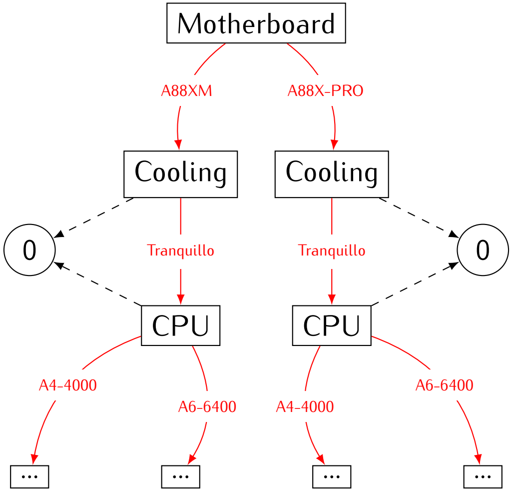
Decision trees for the processor configuration



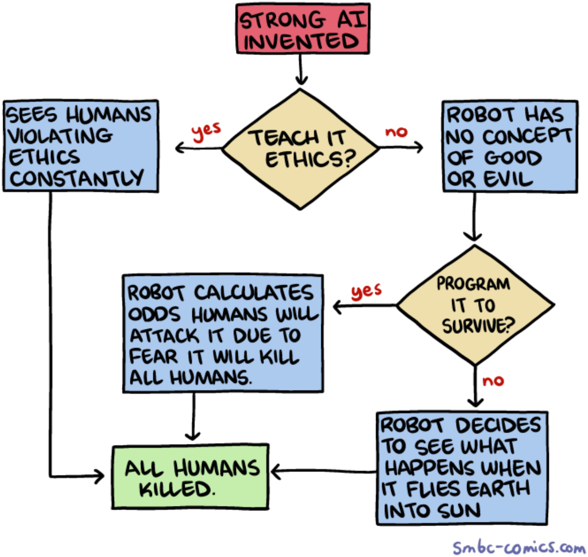
Decision trees for the processor configuration



Decision trees for the processor configuration



Flowcharts.

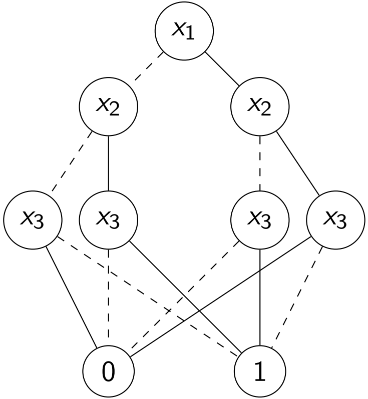


STRONG AI INVENTED
TEACH IT ETHICS?
yes
no
SEES HUMANS VIOLATING ETHICS CONSTANTLY
ROBOT HAS NO CONCEPT OF GOOD OR EVIL
PROGRAM IT TO SURVIVE?
yes
no
ROBOT CALCULATES ODDS HUMANS WILL ATTACK IT DUE TO FEAR IT WILL KILL ALL HUMANS.
ROBOT DECIDES TO SEE WHAT HAPPENS WHEN IT FLIES EARTH INTO SUN
ALL HUMANS KILLED.
smbc-comics.com

This week

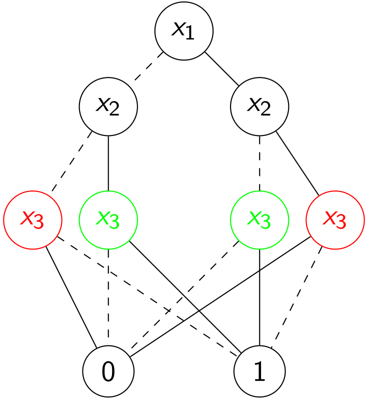
To simplify, this week, we will mostly deal with **Boolean functions**.

Decision trees and branching programs



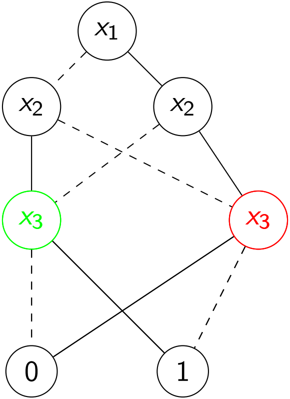
Problem: as many leaves as there are solutions.

Decision trees and branching programs



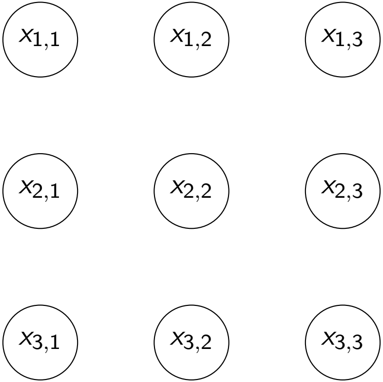
Problem: as many leaves as there are solutions.

Decision trees and branching programs

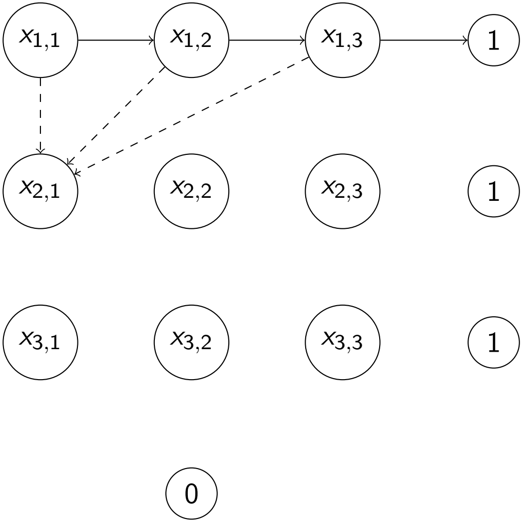


Problem: as many leaves as there are solutions.

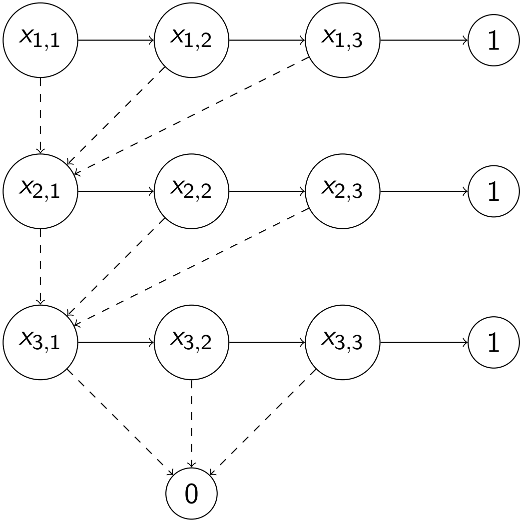
FBDD finding a row of ones in a matrix



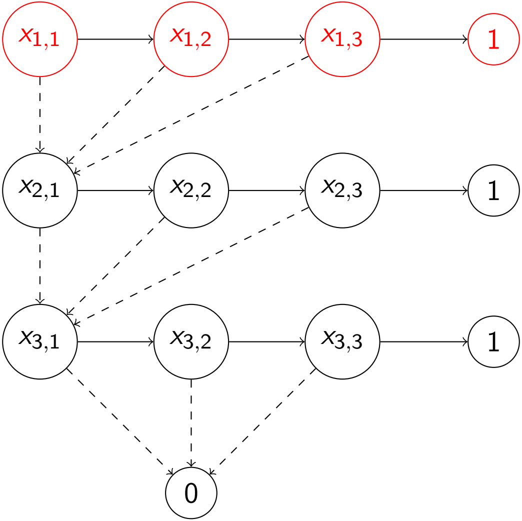
FBDD finding a row of ones in a matrix



FBDD finding a row of ones in a matrix



FBDD finding a row of ones in a matrix



Formal definition: branching programs

A branching program or binary decision diagram (BDD) C is a DAG (directed acyclic graph) such that:

- it has one vertex with indegree 0 called the source
- vertices of outdegree 0 are call sinks and are labelled with constant 0 or 1
- other vertices, called decision nodes, are labelled by a variable x and have two outgoing edges: one labelled with 1 and the other with 0.

Formal definition: accepted assignments

Let C be a BDD on variables X and $\tau : X \to \{0, 1\}$. Let P_τ be the path in C defined as follows:

- start from the source
- when you are on a decision node testing x, take the edge labelled with $\tau(x)$
- repeat until you reach a sink.

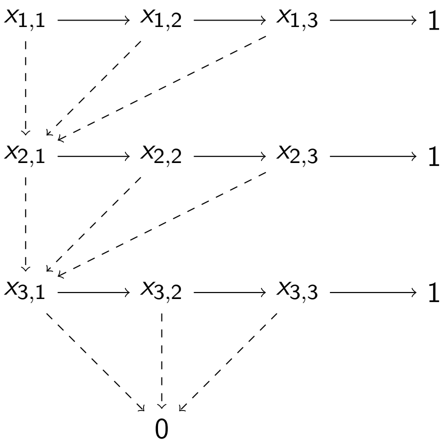
τ is accepted by C if and only if P_τ reaches a 1-sink.

Functional BDD

As we have defined BDD, it is not easy to decide if a given BDD can be satisfied as the same variable x may be tested twice on the same path.
A BDD C is functional (FBDD) if on every source-sink path each variable is tested at most once.

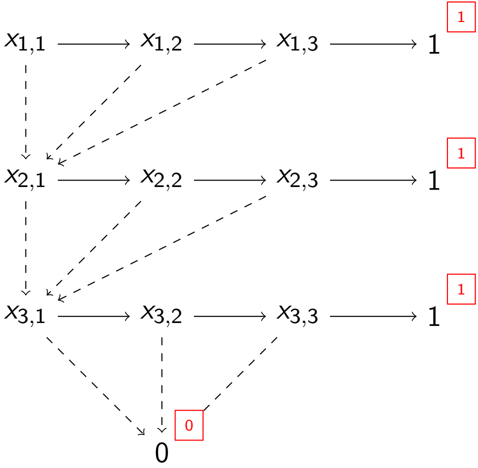
Counting with FBDD.

- Start from the leaves



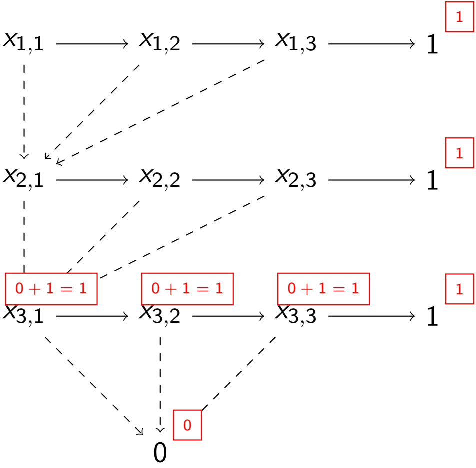
Counting with FBDD.

- Start from the leaves



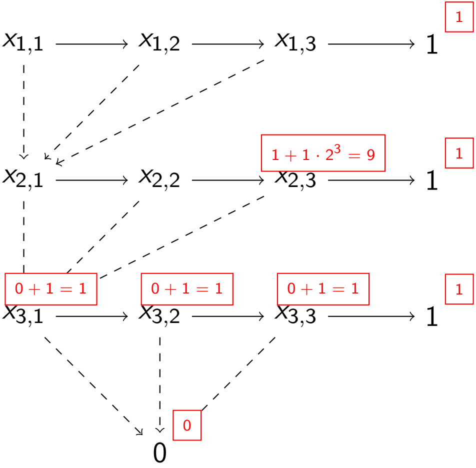
Counting with FBDD.

- Start from the leaves
- **Recursively count the number of solutions** of the branching program starting from each node.



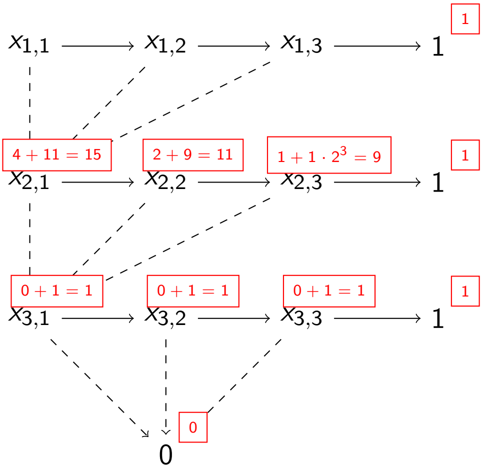
Counting with FBDD.

- Start from the leaves
- **Recursively count the number of solutions** of the branching program starting from each node.
- Beware of the missing variables



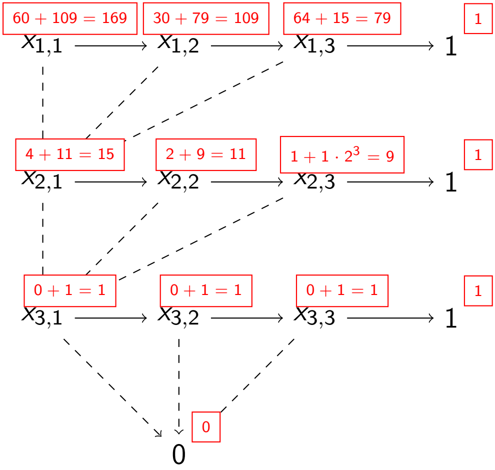
Counting with FBDD.

- Start from the leaves
- **Recursively count the number of solutions** of the branching program starting from each node.
- Beware of the missing variables



Counting with FBDD.

- Start from the leaves
- **Recursively count the number of solutions** of the branching program starting from each node.
- Beware of the missing variables
- 169 solutions



Can we represent everything with a small FBDD?

No.

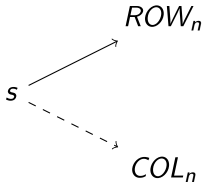
- ROW_n: is there a 1-row in a $n \times n$ matrices?
- COL_n: is there a 1-column in a $n \times n$ matrices?

FBDD representing $f_n = (ROW_n \vee COL_n)$ of size $2^{\Omega(n)}$.

Proof on Friday morning! Don't miss it.

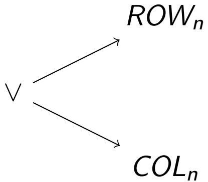
Things we cannot do on FBDD

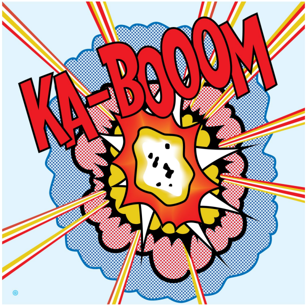
$$f = (s \wedge ROW_n) \vee (\neg s \wedge COL_n).$$



Things we cannot do on FBDD

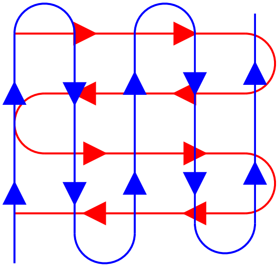
$$\exists s. f = (\cancel{s \wedge} ROW_n) \vee (\cancel{\neg s \wedge} COL_n).$$





Order matters.

$$f = (s \wedge ROW_n) \vee (\neg s \wedge COL_n).$$



Ordered FBDD (OBDD) computing f are of size $2^{\Omega(n)}$.

Proof on Friday morning! Don't miss it.

Composing OBDD

Let C^1, C^2 be two OBDD using the same underlying order and $f : \{0,1\}^2 \to \{0,1\}$. There exists an OBDD C of size $|C^1| \cdot |C^2|$ computing $f(C^1, C^2)$.

Proof idea: construct inductively an OBDD C having gates $\alpha(u, v)$ for every gate u of C_1 and v of C_2 such that $C_{\alpha(u,v)}$ computes $f(C^1_u, C^2_v)$ where C_v denotes the sub-OBDD of C starting from v.

Wrap it up!

We have seen **languages** to represent Boolean functions:

- with **tractable queries**: deciding, counting...
- with **tractable transformations**: negation, conditioning...
- some Boolean functions cannot be **succinctly represented**.

Theoretical framework

How can we study the “compilability” of a query?

P-compilability

A function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is P-compilable if there exists:

- $c : \{0,1\}^* \to \{0,1\}^*$
- g computable in P

such that

- for all $X \in \{0,1\}^*$, $|c(X)| \leq poly(|X|)$
- for all $X, Y \in \{0,1\}^*$, $f(X,Y) \Leftrightarrow g(c(X), Y)$.

- The computation of $c(X)$ is called the offline phase and can be arbitrarely long.
- Solving $y \mapsto g(c(X), y)$ is called the online phase

Limits of P-compilability

Example:

- $f(F, \tau) = 1$ iff there exists a satisfying assignment τ' of the CNF F such that $\tau' \simeq \tau$
- If f is P-compilable then $NP \subseteq P/poly$ (very unlikely).

DNNF

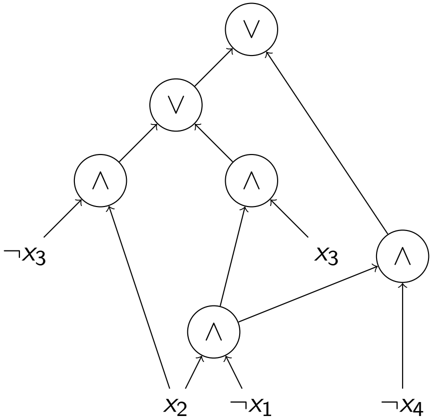
DNNF are a restricted form of boolean circuits:

- input are literals
- $\vee$ and $\wedge$ gates (no internal negation!)
- $\wedge$-gate are decomposable: input subcircuits have **disjoint variables**

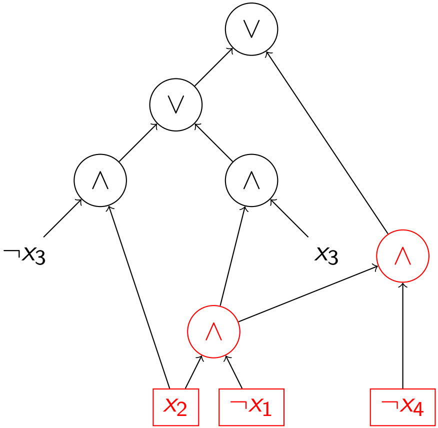
More restrictive conditions:

- **deterministic DNNF (d-DNNF)**: $\vee$-gates verify $\alpha \vee \beta$ such that $\alpha \wedge \beta \equiv \bot$
- **decision DNNF (dec-DNNF)**: $\vee$-gates are of the form $(x \wedge \alpha) \vee (\neg x \wedge \beta)$. They are also deterministic.

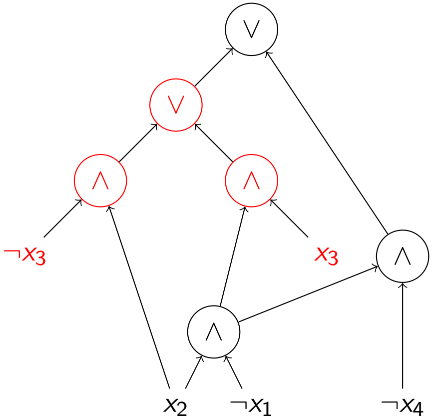
Example



Example



Example



Notation	Query	Explanation
CO	Consistency check	Is D satisfiable?
VA	Validity check	Is D a tautology?
CE	Clause entailment	does $D \Rightarrow C$ for a clause C?
SE	Sentential entailment	does $D_1 \Rightarrow D_2$?
CT	Model counting	how many solutions has D?
ME	Model enumeration	Enumerate the solutions of D.

Knowledge Compilation Map

Queries

Notation	Query	Explanation
CO	Consistency check	Is D satisfiable?
VA	Validity check	Is D a tautology?
CE	Clause entailment	does $D \Rightarrow C$ for a clause C?
SE	Sentential entailment	does $D_1 \Rightarrow D_2$?
CT	Model counting	how many solutions has D?
ME	Model enumeration	Enumerate the solutions of D.

	CO	VA	CE	SE	CT	ME
DNNF	✓	×	✓	×	×	✓
d-DNNF	✓	✓	✓	×	✓	✓
dec-DNNF	✓	✓	✓	×	✓	✓
FBDD	✓	✓	✓	×	✓	✓
OBDD	✓	✓	✓	✓	✓	✓

Knowledge Compilation Map

Transformations

Notation	Transformation	Explanation
$[\tau]$	Conditionning	$D[\tau]$ for τ a partial assignment
$\exists$	Forgetting	$\exists x.D$
$\wedge$	Conjunction	$D_1 \wedge D_2$
$\vee$	Disjunction	$D_1 \vee D_2$
$\neg$	Negation	$\neg D$

Notation	Transformation	Explanation
$[\tau]$	Conditionning	$D[\tau]$ for τ a partial assignment
$\exists$	Forgetting	$\exists x.D$
$\wedge$	Conjunction	$D_1 \wedge D_2$
$\vee$	Disjunction	$D_1 \vee D_2$
$\neg$	Negation	$\neg D$

	$[\tau]$	$\exists$	$\wedge$	$\vee$	$\neg$
DNNF	✓	✓	×	✓	×
d-DNNF	✓	×	×	×	?
dec-DNNF	✓	×	×	×	?
FBDD	✓	×	×	×	✓
OBDD	✓	✓	✓	✓	✓

How does different languages compare? We write $L \subseteq L'$ if every $C \in L$ can be simulated by $C' \in L'$ with $|C'| \leq poly(|C|)$:

$$\text{OBDD} \subsetneq \text{FBDD} \subsetneq \text{dec-DNNF} \subsetneq \text{d-DNNF} \subsetneq \text{DNNF}$$

- OBDD $\subsetneq$ FBDD: $(s \wedge ROW_n) \vee (\neg s \wedge COL_n)$
- dec-DNNF $\subsetneq$ d-DNNF: $(EVEN \wedge ROW) \vee (ODD \wedge COL)$

Open question: DNF vs d-DNNF?