Introduction to Knowledge Compilation

Florent Capelli

CRIStAL, Université de Lille & Inria & CNRS France



- 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

- 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

- 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).

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

Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

1										1		LOGARITHUS									
	LOGARITHMS																				
		1.	1.		2						Mean (Meaning)										
	to		000	0000			01Z 90	50 005	4 0004	0374	4 . 0 12 17 21 25 29 33 37	8 100									
											4 8 11 15 19 23 25 30 34 3 7 50 54 17 21 24 28 31										
	12																				
	14	1481	1402			1584 1	014 56 903 19		3 1703 9 1987		3 6 9 12 15 18 21 24 27 3 6 8 11 14 17 20 22 25	20 JUNE HARD MARK MARK MARK MARK MARK 1000 7000 7000 1 1 2 1 4 4 5 5 7									
	15			-		140 0	126 200			10000		41 2924 7935 2958 2960 2952 2959 2956 7973 2960 7971 1 2 2 4 4 5 5 5									
	17										2 5 7 10 12 15 17 20 22 2 5 7 9 12 14 16 19 21										
	19										2 4 7 9 11 13 16 18 20 2 4 6 8 11 13 15 17 19	3 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1									
	21				294 3	304 33	24 334	5 330	3385	3404	2 4 5 5 10 10 14 14 14	4 8195 8202 8208 8218 8228 8228 8228 8228 8228									
	22 23	8177 2	444 3 636 3	855 3	674 3	662 37	22 354 11 072				2 4 6 6 10 12 14 15 17 2 4 6 7 9 11 13 16 17	48 AND AND AND 8407 8419 800 841 800 841 810 11 12 13 4 4 5 6									
	24 3										2 4 5 7 9 11 12 14 16 2 3 5 7 9 10 12 14 15										
	m le	150 4		43 4	200 43	16 42	32 424	9 4200	4261	4208	2 3 5 7 8 10 11 13 15	11 8513 8619 8525 8531 8537 853 853 853 853 853 853 853 854 855 854 855 854 855 854 855 855 855									
			17 47	02 41	14 45	33 45	68 450		4534	4009	2 3 5 6 8 9 11 13 14	73 66133 66160 8756 8750 8756 8722 8727 8731 8756 8745 1 8 8 8 8 8 8 8									
1 1	9 40	24 45	22 40	14 44	100 45	83 461	45 4713	8 4721	4742	4757	1 3 4 6 7 9 10 12 13 1 3 4 6 7 9 10 11 13	76 8751 8756 8762 8768 8774 8779 8765 8791 8797 8662 1 1 2 2 3 3 4 5 5									
	200	- E									1 3 4 6 7 8 10 11 12	MICH MICH <th< td=""></th<>									
32		51 305	15 . 50	7 50	62 55	05 511	9 5135	5145	5159		5 3 4 5 7 8 9 11 12 1 3 4 5 6 8 9 10 12	1 1027 8027 8027 8032 8038 8943 8949 8054 8960 8965 8971 1 1 2 2 3 3 4 4 5									
33			\$ 534	10 535	53 534	07 525 16 537	B 5391	5400	5416	6428	1 3 4 5 6 8 9 10 11	77 0010 0000 0042 0047 0053 0056 0043 0080 0074 0079 1 1 2 2 3 3 4 4 5 F									
35			3 540		78 541						1 2 4 5 6 7 9 10 11 1 2 4 5 6 7 8 10 11	11 2025 2030 2026 2026 2027 2026 2027 2022 2028 2020 2020 2020 2020 2020									
36			5 558 4 570	6 578		1 562 9 574	0 5752		5775	5785	1 2 3 5 6 7 8 9 10										
38	5794		582			3 585					1 2 3 5 6 7 8 9 10 1 2 3 4 5 7 8 9 10	44 9243 9248 9253 9258 9263 9269 9274 9279 9264 9271 12 2 2 3 3 4 4 5 ·									
40	6021	6031	004	000	3 606	4 607	5 6085	6096	6107	6117	1 2 3 4 5 6 8 9 10										
41	6128		614	616	0 637	0 618	6191	6201	6212	6222	1 2 3 4 5 8 7 8 9 1 2 3 4 5 6 7 8 9										
42			6253			4 628- 5 6385						M 9445 9450 9455 9469 9604 9509 9518 9518 9518 9529 953 9538 0 1 1 2 2 3 3 4 4									
44								6503	6513	6613	1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9	90 9542 9547 9552 9557 9562 9567 9562 9566 9517 9510 44									
45	6532		6646				6684	6693	6702	6712	123456778										
47		8730	6739	624)	6758	6767	6776	6785			123455678										
	6812	6011	6830		6937	6946	6955	0564	6972	6081	1 2 3 4 4 5 6 7 8 1 2 3 3 4 5 6 7 8										
	1990	6296	7007	7016	7924	7033	7042				1 2 3 3 4 5 6 7 8	M REAL REAL REAL REAL REAL REAL REAL REAL									
		7084			7110		7120	7135				96 9623 9827 9832 9835 9865 9809 9831 9930 9930 9930 9930 9330 9330 9330 9330 9330 9330 9331 9330 9331 9									
		7168			7193							MT MAX MT Seed									
54 2	324	7332	7340	7346	7356	7364	7372	7380	7368	7396	22345667	the state and a state of the st									
	10.		-						No		kog	# 1 2 3 17172 21715 2000 5540 1000 1000 1000									
203			0.49	15	in x-1	1-1,00	M - Kg	hox	(1/60-2	30259	0.36222	NER" GADAS GADAS TANDA 12942 AND TANDAS TANDAS									
0=23			0.454	29	tog r -	log x	Mioga		M-0.4	9429	1.83778	bg * 1585 1394 2887 Cold 47 NAVNET COLLEN CO									

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
$$\approx 10^{0.61826}$$

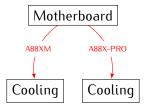
$$\approx 4.1520$$
by looking it in an antilog table.

Processor config demo.

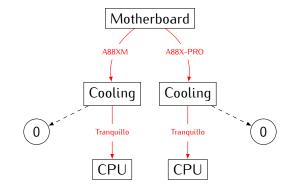
How would you represent the constraints of the previous demo?

- 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?

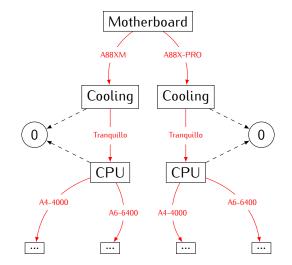
Decision trees for the processor configuration



Decision trees for the processor configuration

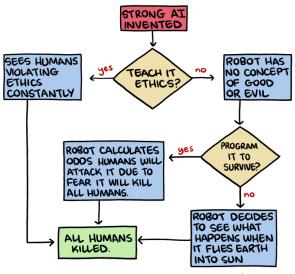


Decision trees for the processor configuration



Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

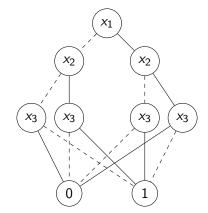
Flowcharts.



Smbc-comics.com

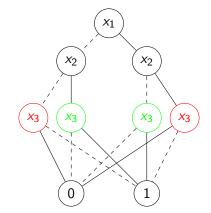
To simplify, this week, we will mostly deal with **Boolean functions**.

Decision trees and branching programs



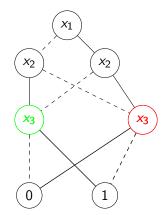
Problem: as many leaves as there are solutions.

Decision trees and branching programs

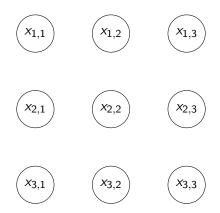


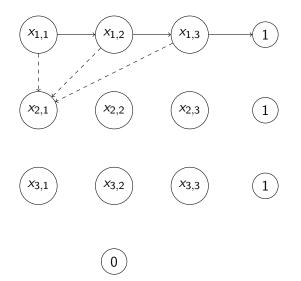
Problem: as many leaves as there are solutions.

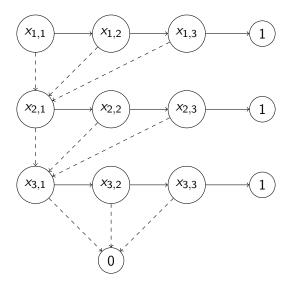
Decision trees and branching programs

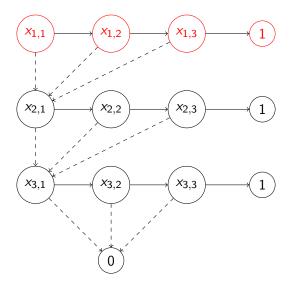


Problem: as many leaves as there are solutions.









Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

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

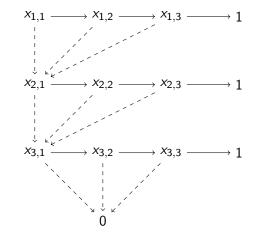
- it has one vertex with indegree 0 called the <u>source</u>
- vertices of outdegree 0 are call <u>sinks</u> and are labelled with constant 0 or 1
- other vertices, called <u>decision nodes</u>, are labelled by a variable x and have two outgoing edges: one labelled with 1 and the other with 0.

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:

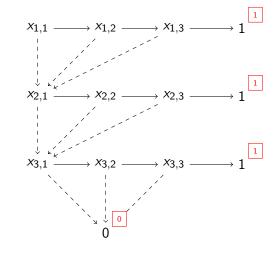
- start from the source
- ▶ when you are on a decision node testing x, take the edge labelled with \(\tau(x))\)
- repeat until you reach a sink.
- τ is accepted by C if and only if P_{τ} reaches a 1-sink.

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 <u>functional</u> (FBDD) if on every source-sink path each variable is tested at most once.

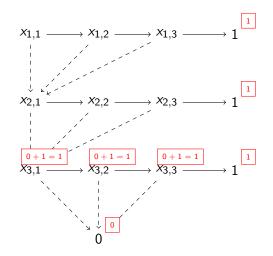


Start from the leaves

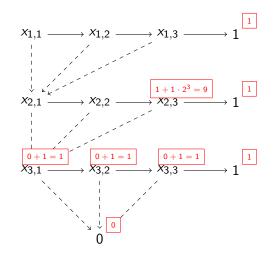


Start from the leaves

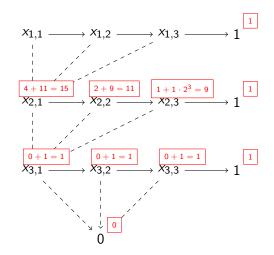
- Start from the leaves
- Recursively count the number of solutions of the branching program starting from each node.



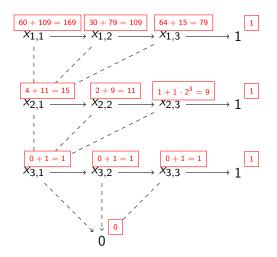
- Start from the leaves
- Recursively count the number of solutions of the branching program starting from each node.
- Beware of the missing variables



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



- 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



No.

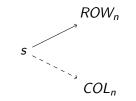
- ROW_n : is there a 1-row in a $n \times n$ matrices?
- COL_n : is there a 1-column in a $n \times n$ matrices?

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

Proof on Friday morning! Don't miss it.

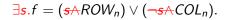
Things we cannot do on FBDD

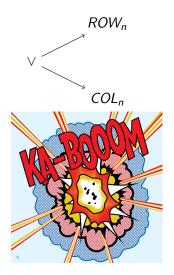
$$f = (s \land ROW_n) \lor (\neg s \land COL_n).$$

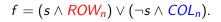


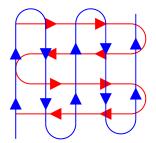
Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

Things we cannot do on FBDD









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

Proof on Friday morning! Don't miss it.

Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

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 $f(C_u^1, C_v^2)$ where C_v denotes the sub-OBDD of *C* starting from *v*.

Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

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.

How can we study the "compilability" of a query?

Research School on Knowledge Compilation, ENS Lyon, December 4th-8th, 2017

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 $X \in \{0,1\}^*$, $|c(X)| \le poly(|X|)$
- ▶ 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

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).

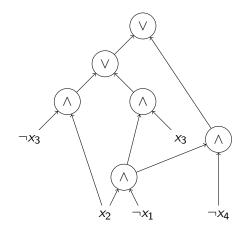
DNNF are a restricted form of boolean circuits:

- input are literals
- \lor and \land gates (no internal negation!)
- A-gate are <u>decomposable</u>: input subcircuits have disjoint variables

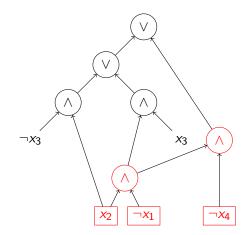
More restrictive conditions:

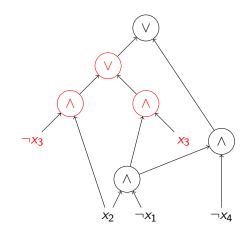
- ► deterministic DNNF (d-DNNF): \lor -gates verify $\alpha \lor \beta$ such that $\alpha \land \beta \equiv \bot$
- ► decision DNNF (dec-DNNF): \lor -gates are of the form $(x \land \alpha) \lor (\neg x \land \beta)$. They are also deterministic.

Example



Example





Notation	Query	Explanation
CO	Consistency check	Is <i>D</i> 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 <i>D</i> .

Notation	Query	Explanation
CO	Consistency check	Is <i>D</i> 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	\checkmark	×	\checkmark	×	×	\checkmark
d-DNNF	\checkmark	\checkmark	\checkmark	×	\checkmark	\checkmark
dec-DNNF	\checkmark	\checkmark	\checkmark	×	\checkmark	\checkmark
FBDD	\checkmark	\checkmark	\checkmark	×	\checkmark	\checkmark
OBDD	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark

Knowledge Compilation Map

Transformations

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

Knowledge Compilation Map

Transformations

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

	$[\tau]$	Ξ	\wedge	\vee	-
DNNF	\checkmark	\checkmark	×	\checkmark	×
d-DNNF	\checkmark	×	×	×	?
dec-DNNF	\checkmark	×	×	×	?
FBDD	\checkmark	×	×	×	\checkmark
OBDD	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark

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

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

- ▶ OBDD \subseteq FBDD: $(s \land ROW_n) \lor (\neg s \land COL_n)$
- ► dec-DNNF \subsetneq d-DNNF: (*EVEN* \land *ROW*) \lor (*ODD* \land *COL*)

Open question: DNF vs d-DNNF?