Knowlegde Compilation for Model Counting: Affine Decision Trees

CRIL, Université d'Artois & CNRS, Lens, France
{koriche,lagniez,marquis,thomas}@cril.fr

Introduction

The Knowledge compilation is a family of approaches for addressing the **intractability of a number of AI problems**.

This amounts to a **translation** issue:

- **Off-line phase:** Turn some pieces of information Σ into a **compiled form** $comp(\Sigma)$
- **On-line phase:** Exploit the compiled form $comp(\Sigma)$ (and the remaining pieces of information α) to achieve the task(s) under consideration

Problem

CT (model counting) and **CD** (conditioning) are a key issue in a number of AI problems (inference in Bayesian networks, contengency planning, ...).

According to the KC map, few languages satisfy **CT**: DT, d-DNNF and its subsets $\text{OBDD}_<$, and FBDD

The Affine Family

AFF satisfies **CT** (triangulation):

$$\begin{pmatrix} x & y & z & & & | & 1 \\ x & & & u & w & | & 0 \\ & & & u & & | & 0 \end{pmatrix} \rightarrow \begin{pmatrix} x & y & z & & & | & 1 \\ & y & z & u & w & | & 1 \\ & & & u & & | & 1 \end{pmatrix}$$

Unfortunatly AFF is not a *complete* language, some propositional formulae cannot be represented into conjunctions of affine clauses (e.g. the clause $x \vee y$).

AFF[$\vee$] is a *complete* language, but loses many queries, particulary **CT**.

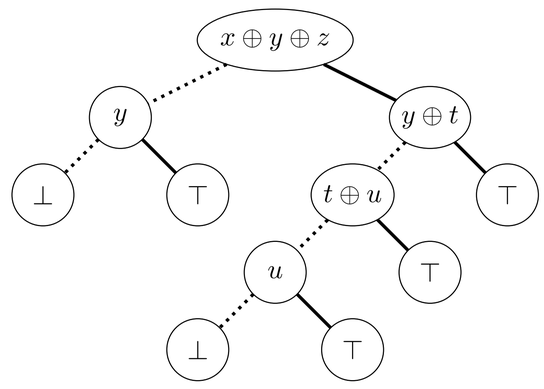
Affine Decision Tree

ADT is based on a generalized form of Shannon expansion:

$\Delta \equiv ((x \Leftrightarrow \neg\gamma) \wedge \Delta_{|x \leftarrow \neg\gamma}) \vee ((x \Leftrightarrow \gamma) \wedge \Delta_{|x \leftarrow \gamma})$

Where Δ is an ECNF and γ is an affine clause

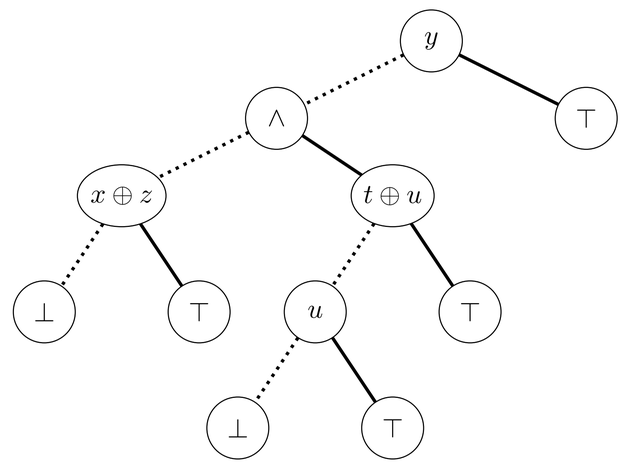
Ex. For $\mathcal{F} = (\overline{x} \vee y \vee \overline{z}) \wedge (x \vee y \vee z) \wedge (y \vee t \vee u)$:



Extended ADT

EADT is an ADT where the afiňĄne decomposable node is added.

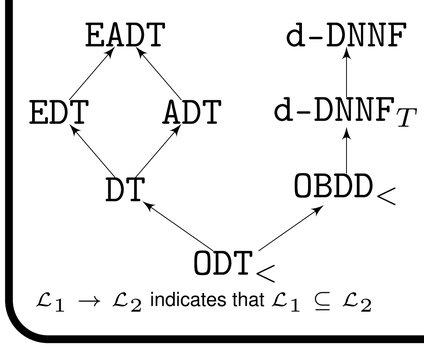
Ex. For $\mathcal{F} = (\overline{x} \vee y \vee \overline{z}) \wedge (x \vee y \vee z) \wedge (y \vee t \vee u)$:



Queries and Transformations

	Queries:									Transformations:						
$\mathcal{L}$	CO	VA	CE	IM	EQ	SE	CT	ME	CD	FO	SFO	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
EADT	√	√	√	√	?	○	√	√	√	○	○	○	○	○	○	√
EDT	√	√	√	√	?	○	√	√	√	○	○	○	○	○	○	√
ADT	√	√	√	√	√	√	√	√	√	○	√	○	√	○	√	√
DT	√	√	√	√	√	√	√	√	√	○	√	○	√	○	√	√
$\text{ODT}_<$	√	√	√	√	√	√	√	√	√	○	√	○	√	○	√	√
d-DNNF	√	√	√	√	?	○	√	√	√	○	○	○	○	○	○	?
$\text{OBDD}_<$	√	√	√	√	√	√	√	√	√	○	√	○	√	○	√	√

Succinctness



$\mathcal{L}_1 \rightarrow \mathcal{L}_2$ indicates that $\mathcal{L}_1 \subseteq \mathcal{L}_2$

- CNF $\not\leq_s$ ADT;
- DNF $\not\leq_s$ ADT;
- $\text{OBDD}_< \not\leq_s$ ADT;
- d-DNNF$_T \not\leq_s$ ADT;
- EADT $<_s$ ADT.

A CNF-to-EADT Compiler

Based on the state-of-the-art CDCL SAT solver MiniSAT

Extended MiniSAT to manage the ECNF

Heuristic used: VSIDS (w)

- Extend the VSIDS weight to the clauses, $\text{VSIDS}(\alpha) = \sum_{x \in \alpha} w(x)$
- Select a clause α which maximizes $\text{VSIDS}(\alpha)$
- replace $x = max(\alpha)$ which maximizes the VSIDS weight by the XOR-disjonction of the $k-1$ literals of $\alpha \setminus \{x\}$
- For our experimentations $k = 2$

Experimental Results

Instance			Cachet		c2d				eadt			
name	$\#var$	$\#cla$	$\#F$	$\overline{Q}$	C	$\overline{Q}$	α	β	C	$\overline{Q}$	α	β
ais6	61	581	1000	0.531	1.23	4 E−5	8 E−7	2	**0.01**	< 1 **E**−7	< 2 **E**−7	**1**
ais8	113	1520	866	0.540	3.04	2 E−4	3 E−4	5	**0.24**	1 **E**−5	2 **E**−5	**1**
ais10	181	3151	325	0.578	12.3	1 E−3	2 E−3	21	**7.69**	1 **E**−4	2 **E**−4	**13**
ais12	265	5666	80	0.573	-	-	-	-	**410**	1 **E**−3	2 **E**−3	**717**
bmc-ibm-2	2810	11683	1000	0.569	-	-	-	-	**0.37**	2 **E**−5	4 **E**−5	**1**
bmc-ibm-3	14930	72106	999	13.04	412	0.93	7 E−2	34	**180**	6 **E**−3	5 **E**−4	**13**
bmc-ibm-4	28161	139716	**1000**	**5.412**	1128	9.09	1.679	$+\infty$	-	-	-	-
bw_large.a	459	4675	1000	0.537	15.05	2 E−5	4 E−5	28	< 1 **E**−4	< 1 **E**−7	< 2 **E**−7	**1**
bw_large.b	1087	13772	1000	0.612	48.88	5 E−5	8 E−5	79	**0.01**	< 1 **E**−7	< 2 **E**−7	**1**
bw_large.c	3016	50457	996	2.452	283.3	1 E−4	6 E−5	115	**0.16**	1 **E**−5	4 **E**−6	**1**
bw_large.d	6325	131973	896	31.44	-	-	-	-	**1.9**	2 **E**−5	6 **E**−7	**1**
(bw) medium	116	953	1000	0.526	2.39	2 E−5	4 E−5	4	< 1 **E**−4	< 1 **E**−7	< 2 **E**−7	**1**
(bw) huge	459	7054	1000	0.543	15.11	2 E−5	4 E−5	27	< 1 **E**−4	< 1 **E**−7	< 2 **E**−7	**1**
hanoi4	718	4934	505	0.557	559.6	3 E−5	5 E−5	1004	**0.13**	< 1 **E**−7	< 2 **E**−7	**1**
hanoi5	1931	14468	440	0.619	2240	8 E−5	1 E−4	3621	**1.1**	1 **E**−5	2 **E**−5	**1**
logistics.a	828	6718	**993**	**1.266**	-	-	-	-	6757	2.12	1.676	$+\infty$
ssa7552-038	1501	3575	1000	0.634	**20.99**	**0.042**	**0.065**	**35**	-	-	-	-

Conclusion and Perspectives

- New languages introduced and studied, EADT appears as quite appealing when the **CT** is a key query
- EADT and its subset EDT satisfy all queries offered by d-DNNF and more transformations
- ADT and its subset DT satisfy all queries and transformations ofered by $\text{OBDD}_<$
- We want to complete the KC map (Do EADT and EDT satisfy **EQ** ? Obtain more results about the succinctness)
- Take advantage of preprocessing techniques and modify the solver to efficiently manage the XOR-clauses
- Considering other heuristics for selecting the branching affine clauses (e.g. privileging the decomposability)