A Compiler for Weak Decomposable Negation Normal Form

Petr Illner and Petr Kučera

Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University, Czech Republic

illner3@gmail.com, kucerap@ktiml.mff.cuni.cz

Abstract

This paper integrates weak decomposable negation normal form (wDNNF) circuits, introduced by Akshay et al. in 2018, into the knowledge compilation map. This circuit type generalises decomposable negation normal form (DNNF) circuits in such a way that they allow a restricted form of sharing variables among the inputs of a conjunction node. We show that wDNNF circuits have the same properties as DNNF circuits regarding the queries and transformations presented in the knowledge compilation map, whilst being strictly more succinct than DNNF circuits (that is, they can represent Boolean functions compactly). We also present and evaluate a knowledge compiler, called Bella, for converting CNF formulae into wDNNF circuits. Our experiments demonstrate that wDNNF circuits are suitable for configuration instances.

Knowledge representation

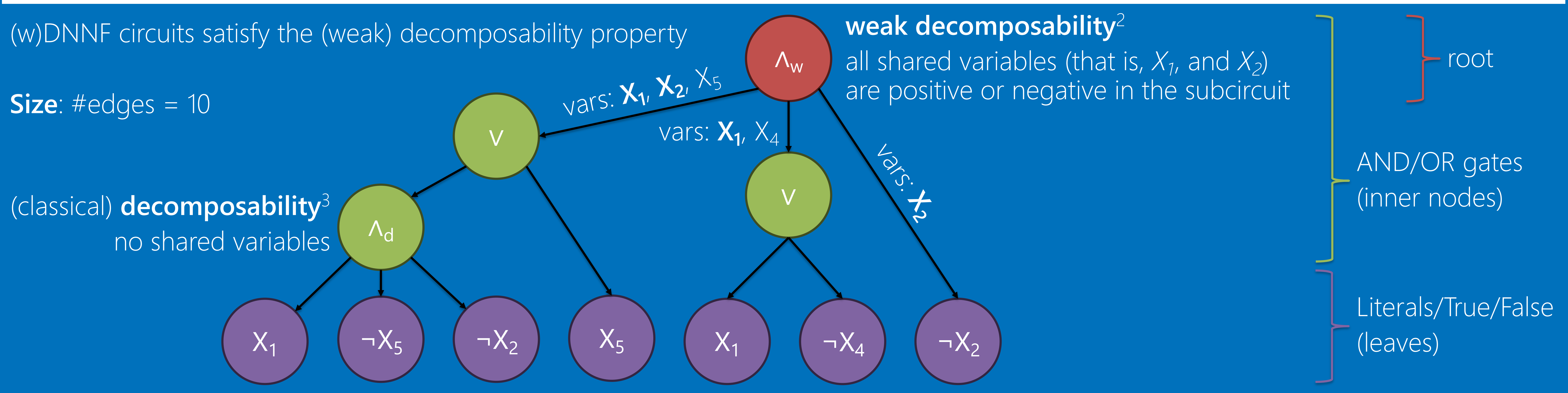
Representations (aka languages) are studied from the following perspectives:

1) Universality: Can the language represent any Boolean function?
2) Succinctness[1]: How compactly can the language represent Boolean functions with respect to other languages?
3) Tractable operations: What operations (that is, queries and transformations) does the language satisfy?
4) **Knowledge compiler**: Is there a knowledge compiler for the language?

Weak decomposable negation normal form (wDNNF) circuits[2]

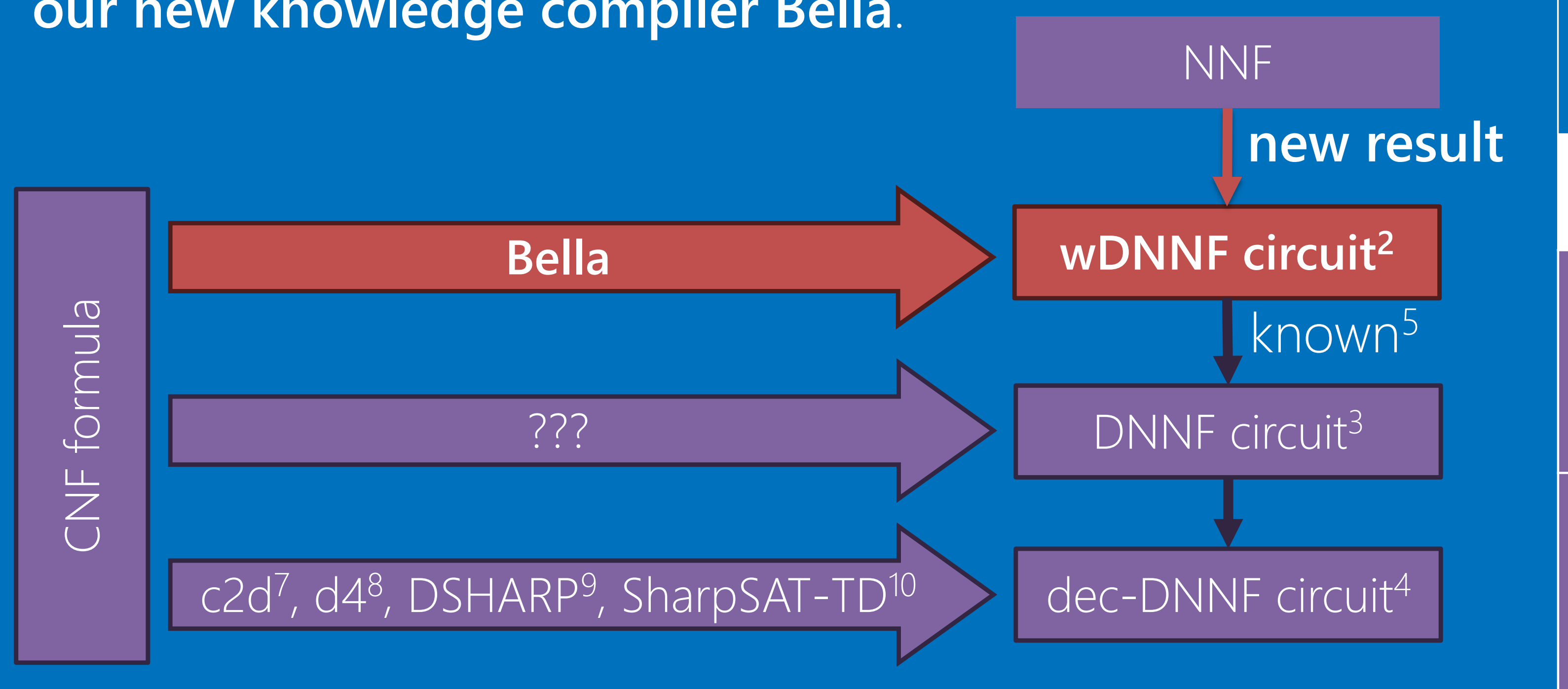
(w)DNNF circuits satisfy the (weak) decomposability property

Size: #edges = 10


weak decomposability[2]
all shared variables (that is, X_1, and X_2) are positive or negative in the subcircuit
(classical) **decomposability**[3]
no shared variables
$\wedge_w$
vars: X_1, X_2, X_5
vars: X_1, X_4
vars: X_2
$\vee$
$\vee$
$\wedge_d$
X_1
$\neg X_5$
$\neg X_2$
X_5
X_1
$\neg X_4$
$\neg X_2$
root
AND/OR gates (inner nodes)
Literals/True/False (leaves)

Knowledge compilers and succinctness[1]

Currently, if we need any of the operations satisfied by DNNF circuits, we must use decision-DNNF circuits, which also satisfy harder operations (for example, model counting). Instead of DNNF circuits, we can use wDNNF circuits and **our new knowledge compiler Bella**.



L_1 -> L_2 means that L_1 is strictly more succinct than L_2

Open question solved!

Is there a knowledge compiler for DNNF circuits?

Applications

Problem: **Enumeration of valid partial configurations**
Required operations: Forgetting (FO), and model enumeration (ME)

Experimental results

Configuration instance[11]	wDNNF circuit		decision-DNNF circuit			
	Bella		d4		SharpSAT-TD	
	time (s)	size	time (s)	size	time (s)	size
C202_FS	**66.3**	**4 245 485**	1 256.2	154 862 147	176.1	341 407 473
C202_FW	**70.7**	**7 344 961**	---	---	151.4	253 123 753
C210_FS	**33.2**	**4 300 692**	1 065.3	240 087 456	239.6	425 732 421
C210_FW	**56.7**	**6 429 027**	4 048.2	715 624 258	339.9	609 298 328

Queries[6]

Query	NNF	**wDNNF** =	DNNF	d-DNNF
Consistency (CO)	✗	✓	✓	✓
Validity	✗	✗*	✗	✓
Clausal entailment (CE)	✗	✓*	✓	✓
Implicant	✗	✗*	✗	✓
Equivalence	✗	✗*	✗	?
Sentential entailment	✗	✗*	✗	✗
Model counting	✗	✗*	✗	✓
Model enumeration (ME)	✗	✓*	✓	✓

Transformations[6]

Transformation	NNF	**wDNNF** =	DNNF	d-DNNF
Conditioning (CD)	✓	✓	✓	✓
Singleton forgetting (SFO)	✓	✓	✓	✗
Forgetting (FO)	✗	✓	✓	✗
Conjunction	✓	✗*	✗	✗
Bounded conjunction	✓	✗*	✗	✗
Disjunction	✓	✓*	✓	✗
Bounded disjunction	✓	✓*	✓	✗
Negation	✓	✗*	✗	?

✓ polytime ✓ polynomial delay ✗ not polytime unless $P = NP$ *new

[1] GOGIC, Goran, et al. The comparative linguistics of knowledge representation. In: IJCAI (1). 1995. p. 862-869.
[2] AKSHAY, S., et al. Knowledge compilation for Boolean functional synthesis. In: 2019 Formal Methods in Computer Aided Design (FMCAD). IEEE, 2019. p. 161-169.
[3] DARWICHE, Adnan. Compiling knowledge into decomposable negation normal form. In: IJCAI. 1999. p. 284-289.
[4] HUANG, Jinbo; DARWICHE, Adnan. The language of search. Journal of Artificial Intelligence Research, 2007, 29: 191-219.
[5] DE COLNET, Alexis; MENGEL, Stefan. A Compilation of Succinctness Results for Arithmetic Circuits. arXiv preprint arXiv:2110.13014, 2021.
[6] DARWICHE, Adnan; MARQUIS, Pierre. A knowledge compilation map. Journal of Artificial Intelligence Research, 2002, 17: 229-264.
[7] DARWICHE, Adnan, et al. New advances in compiling CNF to decomposable negation normal form. In: Proc. of ECAI. Citeseer, 2004. p. 328-332.
[8] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Preprocessing for propositional model counting. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2014.
[9] MUISE, Christian, et al. Fast d-DNNF Compilation with sharpSAT. In: Proceedings of the 8th AAAI Conference on Abstraction, Reformulation, and Approximation. 2010. p. 54-60.
[10] KIESEL, Rafael; EITER, Thomas. Knowledge compilation and more with SharpSAT-TD. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning. 2023.
[11] https://www.cril.univ-artois.fr/kc/benchmarks.html