A Compilation of Succinctness Results for Arithmetic Circuits

Alexis de Colnet, **Stefan Mengel**
CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), Lens, F-62300, France
Univ. Artois, UMR 8188, Lens, F-62300, France
{decolnet, mengel}@cril.fr

Abstract

Arithmetic circuits (AC) are circuits over the real numbers with $0/1$-valued input variables whose gates compute the sum or the product of their inputs. Positive AC – that is, AC representing non-negative functions – subsume many interesting probabilistic models such as probabilistic sentential decision diagram (PSDD) or sum-product network (SPN) on indicator variables. Efficient algorithms for many operations useful in probabilistic reasoning on these models critically depend on imposing structural restrictions to the underlying AC. Generally, adding structural restrictions yields new tractable operations but increases the size of the AC. In this paper we study the relative succinctness of classes of AC with different combinations of common restrictions. Building on existing results for Boolean circuits, we derive an unconditional succinctness map for classes of *monotone* AC – that is, AC whose constant labels are non-negative reals – respecting relevant combinations of the restrictions we consider. We extend a small part of the map to classes of *positive* AC. Those are known to generally be exponentially more succinct than their monotone counterparts, but we observe here that for so-called deterministic circuits there is no difference between the monotone and the positive setting which allows us to lift some of our results. We end the paper with some insights on the relative succinctness of positive AC by showing exponential lower bounds on the representations of certain functions in positive AC respecting structured decomposability.

1 Introduction

Arithmetic circuits (AC) are a circuit model for representing polynomials by giving the order in which their inputs have to be combined by sums and multiplications. Thus, AC are not only very natural representations for real-valued polynomials, but also give programs for computing them; this can e.g. be traced back to (Valiant 1980) who called them $(+,\times)$-programs. Today AC play an important role in artificial intelligence because they encompass several classes of circuits with practical applications in probabilistic reasoning, for instance probabilistic sentential decision diagrams (PSDD) (Kisa et al. 2014) or sum product networks (SPN) with indicator variables (Poon and Domingos 2011). AC are also strongly related to concepts such as AND/OR-circuits (Dechter and Mateescu 2007) and Cutset Networks (Rahman, Kothalkar, and Gogate 2014). When used in probabilistic reasoning, AC always represent non-negative functions and are therefore called (somewhat misleadingly perhaps) *positive AC*. Positive AC constitute a subclass of what in the probabilistic graphical models community is called *probabilistic circuits* (Choi, Vergari, and Van den Broeck 2020). In the literature, positivity is is often syntactically enforced by assuming that all constants in the computation are non-negative, see e.g. (Darwiche 2003; Poon and Domingos 2011), in which case the AC are called *monotone*. Essentially, compared to their monotone counterparts, positive AC encode programs which allow subtraction as an additional operation. This has no impact on the tractability of most operations performed on the AC (Dennis 2016) and it is known already since (Valiant 1980) that it can decrease the size of AC exponentially.

While research on arithmetic circuits in complexity theory focuses almost exclusively on trying to show lower bounds on the size of AC representing notoriously challenging polynomials like the permanent, see e.g. (Jerrum and Snir 1982; Shpilka and Yehudayoff 2010; Raz 2009), the goals pursued in artificial intelligence are often different: on the one hand, algorithms for generating AC from other models like Bayesian networks (Chavira and Darwiche 2008; Choi, Kisa, and Darwiche 2013; Kisa et al. 2014), or by learning from data (Lowd and Domingos 2008; Rooshenas and Lowd 2016), are a major focus. On the other hand, it is studied how imposing constraints on the structure of AC can render operations like computation of marginals or of maximum a posteriori hypotheses (MAP) or more complex queries tractable on them (Huang, Chavira, and Darwiche 2006; Vergari et al. 2021; Khosravi et al. 2019). In this latter line of work, the earliest and most well-studied properties are decomposability (also called syntactic multilinearity), smoothness (also called completeness), and determinism. There is an ongoing effort to find new properties: on the one hand, more restrictive properties to allow new operations, for example *structured decomposability* (Kisa et al. 2014; Dang, Vergari, and Van den Broeck 2020), on the other hand, more general properties that are sufficient to ensure tractability of important operations. For instance *weak* decomposability (also called consistency) is a relaxation of decomposability which, if combined with smoothness, allows efficient marginals computation (Peharz et al. 2015).

While the analysis of more restrictive properties is driven by the prospect of AC to support more operations efficiently and therefore be more useful in practice, the quest for more generic properties is motivated by the succinctness of resulting AC: while generally all classes of AC commonly considered can represent all functions, more general classes should intuitively allow smaller representations.

The trade-off between usefulness and succinctness has also been observed for Boolean circuits in negation normal form (NNF) and attracted a lot of attention there (Darwiche and Marquis 2002; Pipatsrisawat and Darwiche 2008; Bova et al. 2016; Amarilli et al. 2020). Indeed, all structural restrictions on AC mentioned above are also defined for NNF, and classes of NNF respecting combinations of restrictions have been studied almost exhaustively. In particular, for NNF, succinctness maps have been drawn that intuitively describe the relative succinctness for the classes of NNF one gets by applying different combinations of restrictions. When it comes to AC, research on lower bounds in complexity theory focused on classes with properties such as bounded-depth, tree-like structure, or multilinearity (Grigoriev and Karpinski 1998; Raz 2009; Raz 2010; Shpilka and Yehudayoff 2010) that have deep implications in theory but are not particularly desirable in practice – with the exception of *syntactic* multilinearity which is in fact decomposability. In comparison to Boolean circuits, the succinctness analysis for classes of arithmetic circuits of practical interest is fairly young and far from complete (Martens and Medabalimi 2014; Choi and Darwiche 2017).

In this paper we initiate a systematic succinctness map for AC modeled after that proposed in (Darwiche and Marquis 2002) for NNF. We focus on classes of AC with $0/1$-variables that respect decomposability or weak decomposability and possibly determinism and/or smoothness. Most of our results deal with classes of *monotone* AC and are obtained by lifting results from the existing succinctness map for NNF. To this end, we observe that understanding the succinctness relations between different classes of monotone AC reduces to understanding that between classes of NNF with analogous restrictions. However, several classes of NNF obtained with the reduction, namely those respecting weak decomposability, have only recently been introduced for NNF (Akshay et al. 2019) and thus their position in the maps has not been studied. To analyze monotone AC, we thus prove the missing succinctness relations for these classes. From the map for NNF and the lifting technique, we obtain the complete map linking the eight classes of *monotone* AC one gets combining the different restrictions. In a modest contribution to the understanding of *positive* AC, we show that under particular restrictions, all including determinism, the expressive power of classes of positive AC coincide with that of their monotone counterparts. Thus some succinctness relations in the monotone map easily extend to the positive map. However, for positive AC, several relations between classes remain open.

Finally, in an effort to motivate further research on the succinctness relations left to prove, we describe a technique to show lower bounds on the size of positive AC. We apply it to prove lower bounds for positive AC with *structured* decomposability, which is the case for e.g. PSDD (Kisa et al. 2014). We stress that all separations between classes that we prove are unconditional (so no "unless P = NP" or similar assumptions) and exponential.

2 Preliminaries

2.1 AC and NNF

An arithmetic circuit (short AC) is defined to be a directed acyclic graph with a single source whose sinks are each labeled with a real number, a $0/1$-variable, or by complemented variables $\overline{x}$, and whose internal nodes each have two successors and are labeled by $+$ or $\times$. A Boolean circuit in negation normal form (short NNF) is defined completely analogously to an AC, but the internal nodes are labeled with $\vee$ and $\wedge$ and the only constants that can appear as sink-labels are 0 and 1. The following definitions are the same for AC and NNF, so we do not differentiate the two settings here.

The sinks of a circuit C are called its inputs. We say that variable x appears with negative (resp. positive) polarity in C if $\overline{x}$ (resp. x) labels a sink of C. If g is an internal node then we denote by g_l and g_r its left and right successors. We define the size $|C|$ of the circuit as the number of nodes in the underlying graph, which, since the operations are binary, is at most twice the number of edges.

Let X be the variables appearing in C. An assignment a to X is a mapping from X to $\{0, 1\}$. The weight of a, denoted by $w(a)$, is the number of variables it maps to 1. A partial assignment is defined as an assignment to a subset $Y \subseteq X$. In the particular case when $Y = \emptyset$, we have the unique empty assignment denoted $a_\emptyset$. Given a partial assignment a', the circuit obtained by *conditioning* C on a', denoted by $C|a'$, is obtained by replacing in C for all $y \in Y$ all inputs labeled y by $a'(y)$ and all inputs labeled $\overline{y}$ by $1 - a'(y)$. Given two assignments a and a' to X and X' such that a and a' are consistent on $X \cap X'$, we let $a \cup a'$ denote the assignment to $X \cup X'$ whose restrictions to X and X' are a and a', respectively. For convenience, a literal $\ell_x \in \{x, \overline{x}\}$ will sometimes be seen as an assignment of x to the value satisfying the literal, so we may write $C|\ell_x$ or $a \cup \ell_x$.

Given an assignment a to X, C computes a value $C(a)$ on a in the obvious way by first conditioning C on a and computing in a bottom-up fashion in $C|a$ the results of the internal nodes by computing the result of the operation they are labeled with for the values computed by their successors. $C(a)$ is the value computed by the source node. The function $f : X \to \mathbb{R}$, resp. $f : X \to \{0, 1\}$, computed by C is defined as the function defined by $f(a) = C(a)$ for all assignments a.

For an NNF or AC C over variables X and a node g in C, let $\mathit{var}(g)$ denote the subset of X such that $x \in \mathit{var}(g)$ if and only if x or $\overline{x}$ labels a sink reachable from g. Note that if g is a sink labeled with x or $\overline{x}$ then $\mathit{var}(g) = \{x\}$ and that if g is labeled with a constant then $\mathit{var}(g) = \emptyset$. By extension $\mathit{var}(C)$, sometimes called the *scope* of C, denotes the set $\mathit{var}(s)$ where s is the source of C.

The set of assignments to $\mathit{var}(C)$ for which an AC C computes a non-zero value is called the *support* of C denoted by $\mathit{supp}(C)$. For an NNF, these assignments are called *models*, or satisfying assignments, and we use the more common notation *sat*(C) for that case instead of *supp*(C). For a node g in C we let C_g be the sub-circuit of C consisting of nodes reachable from g. We write *supp*(g) for *supp*(C_g). Note that when g is an input labeled with a literal ℓ_x, *supp*$(g) = \{\ell_x\}$ and that for constant inputs there is *supp*$(0) = \emptyset$ and *supp*$(\alpha) = \{a_\emptyset\}$ for any constant $\alpha \neq 0$.

2.2 Subclasses of AC and NNF

In applications, in particular probabilistic reasoning, the possible outputs of AC are restricted to be non-negative. Thus we define *positive* AC to be the AC that compute non-negative functions, i.e., for all assignments a of its inputs, a positive AC must return a value greater or equal to 0. We denote the class of all positive AC by AC$_p$. A proper subclass of AC$_p$ is that of *monotone* AC, denoted AC$_m$, which are the AC whose constant inputs are all non-negative.

The classes studied in this paper correspond to circuits whose nodes enforce one or more of the properties defined below: smoothness (or completeness), determinism, decomposability and weak decomposability.

Definition 1. *An internal node g in a circuit C is called* smooth *when var(g_l) = var(g_r) holds.*

An AC is called smooth *(or complete) when all its $+$-nodes are smooth. We denote by* s-AC *the class of smooth AC.*

Definition 2. *An internal node g in a circuit C is called* deterministic *when there is no assignment a such that $a_l \in$ supp(g_l) and $a_r \in$ supp(g_r), where a_l and a_r are the restrictions of a to $var(g_l)$ and $var(g_r)$, respectively.*

An AC is called deterministic *when all its $+$-nodes are deterministic. We denote by* d-AC *the class of deterministic AC.*

Definition 3. *An internal node g in a circuit C is called* decomposable *when var$(g_l) \cap$ var$(g_r) = \emptyset$ holds.*

An AC is called decomposable *when all its $\times$-nodes are decomposable. We denote by* D-AC *the class of decomposable AC.*

We remark that in the complexity theory literature decomposable AC are often called *syntactically multilinear* AC.

Definition 4. *An internal node g in a circuit C is called* weakly decomposable *when, for all $x \in$ var$(g_l) \cap$ var(g_r), the variable x appears with a unique polarity under g, i.e., either x appears under g or $\overline{x}$ appears under g, but not both.*

An AC is weakly decomposable *when all its $\times$-nodes are weakly decomposable. We denote by* wD-AC *the class of weakly decomposable AC.*

Weak decomposability is sometimes referred to as *consistency*, but we avoid using this term here since for Boolean circuits it is often used to mean satisfiability.

The classes s-NNF, d-NNF, D-NNF and wD-NNF are defined analogously as subclasses of NNF by replacing $+$-nodes by $\vee$-nodes, $\times$-nodes by $\wedge$-nodes. However, we will use the more common notations DNNF and wDNNF instead of D-NNF and wD-NNF.

We will consider intersections of the classes just introduced. The names for the intersection classes combine the prefixes s-, d-, D- and wD- accordingly. For instance the class of deterministic decomposable NNF is denoted d-DNNF, that of smooth weakly decomposable AC is denoted by swD-AC, and so on. Observe that weak decomposability is a generalisation of decomposability, so the intersection of D-AC with wD-AC (resp. DNNF with wDNNF) is just D-AC (resp. DNNF).

Imposing specific combinations of structural restrictions above often makes operations that are intractable on unconstrained AC tractable. For instance, when AC are used in probabilistic reasoning, queries such as the computation of marginals, maximum a posteriori (MAP) or marginal MAP are tractable for different combinations of the four aforementioned constraints (Peharz et al. 2015; Shen, Choi, and Darwiche 2016; Khosravi et al. 2019; Choi, Vergari, and Van den Broeck 2020; Vergari et al. 2021). It turns out that, for all problems studied so far, interesting combinations all include either decomposability or weak decomposability. So, the classes studied in this paper are summarized as followed:

$$\{\emptyset, \text{s}\}\{\emptyset, \text{d}\}\{\text{D}, \text{wD}\}\text{-}\{\text{AC}_m, \text{AC}_p, \text{NNF}\}.$$

which is interpreted as: the subclasses of positive AC, monotone AC and NNF ($\{\text{AC}_m, \text{AC}_p, \text{NNF}\}$) that implement decomposability or weak decomposability ($\{\text{D}, \text{wD}\}$), and possibly smoothness ($\{\emptyset, \text{s}\}$) , and possibly determinism ($\{\emptyset, \text{d}\}$). So eight classes of NNF and sixteen classes of AC.

Let X be a finite set of $\{0, 1\}$ valued variables, every function $f : X \to \mathbb{R}_+$ has a representation in all these classes of AC$_m$ and AC$_p$. To see this, one can just write f as $f(X) = \sum_{a \in supp(f)} f(a)1_a(X)$, where $1_a(X)$ is the function returning 1 on assignment a and 0 otherwise. The terms $f(a)1_a(X)$ are easily encoded in positive AC with only decomposable $\times$-nodes, then building a positive AC computing f and implementing smoothness, determinism and decomposability upon those terms AC is straightforward. Analogously, every Boolean functions on X has a representation in all eight classes of NNF studied. However, in both cases there are often more compact circuits than those just described.

2.3 Succinctness

As just discussed, to compare the different classes of circuits considered here, expressivity is not an issue since all classes are fully expressive in the sense that they can represent all functions. However, as we will see, the size of representations in different classes may differ greatly. Since the classes allow polynomial time algorithms for different problems, it is meaningful to compare the minimum size of a circuit computing the same function in different classes. This naturally leads to the introduction of *succinctness* as a means to compare the classes. For a class $\mathcal{C}$, the size of the minimum circuit computing f in $\mathcal{C}$ is called the $\mathcal{C}$-size of f.

Definition 5. *For two classes of circuits $\mathcal{C}_1$ and $\mathcal{C}_2$, we say that $\mathcal{C}_1$ is* at least as succinct as $\mathcal{C}_2$, *written $\mathcal{C}_1 \leq \mathcal{C}_2$, if there is a polynomial p such that for all $C_2 \in \mathcal{C}_2$, there exists $C_1 \in \mathcal{C}_1$ computing the same function with $|C_2| \leq p(|C_1|)$.*

Equivalently, $\mathcal{C}_1 \leq \mathcal{C}_2$ if, for all functions f, the $\mathcal{C}_2$-size is polynomially bounded by the $\mathcal{C}_1$-size. We write $\mathcal{C}_1 < \mathcal{C}_2$

when $\mathcal{C}_1 \leq \mathcal{C}_2$ but $\mathcal{C}_2 \not\leq \mathcal{C}_1$, and $\mathcal{C}_1 \simeq \mathcal{C}_2$ when both $\mathcal{C}_1 \leq \mathcal{C}_2$ and $\mathcal{C}_2 \leq \mathcal{C}_1$ hold; in this case we say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *equally succinct*. Succinctness is a transitive relation.

2.4 Term Subcircuits

For a (w)D-AC (resp. a (w)DNNF) C on variables X, we define *term subcircuits* of C iteratively. Starting from the source, whenever a $\times$-node (resp. $\wedge$-node) is encountered, its two successors are added to the subcircuit, and whenever a $+$-node (resp. $\vee$-node) is encountered, exactly one arbitrary successor is added to the subcircuit. As indicated by the name, each term subcircuit encodes a function which is a single term $\alpha \times \ell_{x_1}\ell_{x_2}\cdots\ell_{x_k}$ where α is a constant and $\ell_{x_i} \in \{x_i, \overline{x_i}\}$, $(x_i)_{i\in[k]} \subseteq X$. By distributivity, the sum (resp. the disjunction) of all term subcircuits of C is equivalent to C. For DNNF, term subcircuits are more often called *certificates* or *proof trees*, but term subcircuits of wDNNF are generally not shaped like trees. The following easy lemma is shown in the appendix.

Lemma 1. *Let C be a (weakly) decomposable* AC *(resp.* NNF*) then:*

- *if C is smooth, all variables appear in all term subcircuits*
- *if C is deterministic, then any two distinct term subcircuits T and T' verify $T \times T' = 0$ (resp. $T \wedge T' \equiv 0$).*

3 Succinctness Map for Monotone AC

3.1 From Monotone AC to NNF

One attractive approach towards understanding the succinctness relations between classes of AC is lifting the corresponding map for classes of NNF to classes of AC. This is because the map for NNF is quite substantial and well understood by now, so building the map for AC upon it would save us the trouble of many proofs. Here we will show that we can apply this approach for classes of *monotone* AC. The idea is that separating the classes of Boolean functions corresponding to the support of monotone AC is enough to separate these classes of AC.

Given a monotone AC C, we define a Boolean circuit $\phi(C)$ that has the same underlying graph as C and is obtained by just modifying the labels on the nodes of C. Sinks labeled by x or $\overline{x}$ or the constant 0 are unchanged, but sinks labeled by constants different from zero are now labeled by the constant 1. For internal nodes, all $\times$-nodes become $\wedge$-nodes and all $+$-nodes become $\vee$-nodes. Clearly $var(C) = var(\phi(C))$ and, since C and $\phi(C)$ have the same graph, we have $|C| = |\phi(C)|$. The following lemmas are easy to derive. Proofs are deferred to the appendix.

Lemma 2. *When C is a monotone* AC*, $\phi(C)$ is an* NNF *whose models are $supp(C)$. Moreover if C is (weakly) decomposable, deterministic, or smooth, then $\phi(C)$ is as well.*

Lemma 3. *For every* NNF *D, there exists an* AC *C of size $|D|$ whose support are the models of D. Moreover if D is (weakly) decomposable, deterministic, or smooth, then so is C.*

For a class $\mathcal{C}$ of AC, we define the class of NNF $\phi(\mathcal{C}) := \{\phi(C) \mid C \in \mathcal{C}\}$. Lemma 2 and Lemma 3 directly yield the following:

$$\begin{tikzcd}
& \text{sdwD-AC}_m \arrow[rr, equal] & & \text{sdD-AC}_m \arrow[dd, equal] \\
\text{swD-AC}_m \arrow[ur] \arrow[rr, equal] & & \text{sD-AC}_m \arrow[ur] \arrow[dd, equal] & \\
& \text{dwD-AC}_m \arrow[uu] \arrow[rr] & & \text{dD-AC}_m \\
\text{wD-AC}_m \arrow[uu] \arrow[ur] \arrow[rr] & & \text{D-AC}_m \arrow[ur] &
\end{tikzcd}$$

Figure 1: Succinctness map for monotone AC. An arrow $C_1 \rightarrow C_2$ means that $\mathcal{C}_1 < \mathcal{C}_2$. A double line $C_1 = C_2$ means that $\mathcal{C}_1 \simeq \mathcal{C}_2$. The absence of connector between two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ means either that the succinctness relation is derived from transitivity or that the two classes are incomparable, i.e., $\mathcal{C}_1 \not\leq \mathcal{C}_2$ and $\mathcal{C}_2 \not\leq \mathcal{C}_1$.

Proposition 1. *Let γ be any combination of properties from $\{$s,d,D,wD$\}$, then $\phi(\gamma$-AC$_m) = \gamma$-NNF.*

For instance $\phi(\text{AC}_m) = \text{NNF}$, $\phi(\text{D-AC}_m) = \text{DNNF}$, $\phi(\text{dD-AC}_m) = \text{d-DNNF}$, etc. Moreover, since the circuit size is preserved by ϕ, the following holds:

Proposition 2. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be classes of monotone AC, then $\mathcal{C}_1 \leq \mathcal{C}_2$ if and only if $\phi(\mathcal{C}_1) \leq \phi(\mathcal{C}_2)$.*

Since it is known already that s-DNNF $\simeq$ DNNF $<$ d-DNNF $\simeq$ sd-DNNF (Darwiche and Marquis 2002), it follows that sD-AC$_m$ $\simeq$ D-AC$_m$ $<$ dD-AC$_m$ $\simeq$ sdD-AC$_m$, and these relations are *unconditional* (so no “unless P = NP” or other complexity theoretic assumptions are needed). Weak decomposability has not been studied as widely as decomposability for NNF, so we here draw the map with the additional classes wDNNF, s-wDNNF, d-wDNNF and sd-wDNNF. Then, using Proposition 2, we will obtain the succinctness map for monotone AC shown in Figure 1.

Theorem 1. *The results of Figure 1 hold.*

Section 3.3 is dedicated to the proof of Theorem 1. But first we will show a useful auxiliary result.

3.2 Smoothing Restricted wDNNF

It was shown by Peharz et al. (2015) that transforming general wD-AC$_m$ into swD-AC$_m$ leads to an unavoidable exponential blow-up. By Proposition 2, the same is true for wDNNF and s-wDNNF. Here we show that this is not the case when all term subcircuits have the same variables.

Proposition 3. *Let D be a* wDNNF *over n variables such that for any two term subcircuits T and T', $var(T) = var(T')$ holds. Then there is a smooth* wDNNF *D^* equivalent to D of size $|D^*| = O(n|D|)$. Furthermore, if D is deterministic, then so is D^*.*

Proposition 3 will be used in the next section to prove the succinctness map for NNF. Before we prove it, we give some more definitions.

Definition 6. *Let $\ell_x \in \{x, \overline{x}\}$. An $(\wedge\ell_x)$-link is a $\wedge$-node whose successors include a leaf labeled by ℓ_x. Let g be a node and let p be a predecessor of g, then inserting an*

$(\wedge \ell_x)$-link between g and p means replacing the connection

$$\begin{tikzcd}[row sep=small, column sep=small] & p \\ g \arrow[ur, dash] & \end{tikzcd} \quad \text{by} \quad \begin{tikzcd}[row sep=small, column sep=small] & p & \\ & \wedge \arrow[u, dash] & \\ g \arrow[ur, dash] & & \ell_x \arrow[ul, dash] \end{tikzcd} .$$

We call the intermediate $\wedge$-node in the construction above the *link node*. A succession of link nodes is a *chain of links*. We remark that links have already been used by Peharz et al. (2015) to analyze the impact of smoothness on wD-AC$_m$, but we use them here in a different way.

For a term subcircuit T containing a node g, let T_g denote the sub-circuit of T under g, and let $T_{\overline{g}}$ be the subcircuit of T corresponding to all nodes accessible from the source without passing through g. Observe that because of *weak* decomposability, some nodes accessible from g may be reached by paths in T not passing through g, so T_g and $T_{\overline{g}}$ are not necessarily disjoint.

Proof (of Proposition 3). Let g be an $\vee$-node such that $var(g_l) \neq var(g_r)$. Let $x \in var(g_l)$ and $x \notin var(g_r)$. There exists an $\wedge$-node that is an ancestor of g in D, otherwise not all term subcircuits of D would have the same variables. So, for all term subcircuits T of D containing g, $T_{\overline{g}}$ is not empty. Moreover x must be contained in $var(T_{\overline{g}})$ for otherwise we can construct a term subcircuit that does not contain x by extending $T_{\overline{g}}$ to a term subcircuit choosing g_r as the child of g.

We claim that x appears with unique polarity under g_l. To see this, assume first that x appears positively in $T_{\overline{g}}$. Now if $\overline{x}$ appeared below g_l as the label of a node g^*. Then, we could extend $T_{\overline{g}}$ to a term subcircuit T^* containing g^* and thus the variable $\overline{x}$. But then T^* would contain both x and $\overline{x}$ which is impossible because C is weakly decomposable. If x appears negatively in $T_{\overline{g}}$, we reason analogously. So in any case, x appears with unique polarity under g_l; we assume in the remainder that it appears positively, the other case is completely analogous.

Analogously to above, one sees that for all term subcircuits T' containing g, we have that $T'_{\overline{g}}$ contains x. So, for all T' passing through g, we have $T \equiv T \wedge x$. Now insert an $(\wedge x)$-link between g and g_r and let D' be the resulting wDNNF. We write $(g, g_r) \in T$ when the wire from g to g_r is in the term subcircuit T of D. There is a bijection φ between the term subcircuits of D and those of D': for a term subcircuit T of D, set $\varphi(T) = T$ if $(g, g_r) \notin T$, and let $\varphi(T)$ be the term subcircuit of D' we get from T by inserting the $(\wedge x)$-link between g and g_r otherwise. Clearly, when $(g, g_r) \in T$, then $\varphi(T) \equiv T \wedge x$, and we have already seen that $T \wedge x \equiv T$ in that case. So

$$\begin{aligned} D' &\equiv \bigvee_{T:(g,g_r)\in T} \varphi(T) \vee \bigvee_{T:(g,g_r)\notin T} \varphi(T) \\ &\equiv \bigvee_{T:(g,g_r)\in T} T \vee \bigvee_{T:(g,g_r)\notin T} T \equiv D \end{aligned}$$

Observe that $var(g)$ is identical in D and D' since x was already in $var(g_l)$ in D. Observe also that the $\wedge$-link node is decomposable. So D' is a wDNNF. Now in D' the variable x appears under both successors of g. We repeat that process until the successors of g have the same set of variables, so until g is smooth. Doing this for all non-smooth $\vee$-nodes yields a wDNNF D^* that is smooth. The construction only adds chains of links between nodes that were originally in D, and since there are n variables, at most n links are inserted between any two connected nodes of D, hence $|D^*| = O(n|D|)$.

Finally we argue that if D is deterministic, then so is D^*. We just need to prove this for D', i.e., one single addition of an $(\wedge x)$-node. Assume that g is deterministic in D. Let g'_r be the $\wedge$-node inserted between g and g_r in D'. The successors of g'_r are x and g_r. Assume there is an assignment a' to $var(g)$ whose restrictions a'_l and a'_r to $var(g_l)$ and $var(g'_r)$ are in $sat(g_l)$ and $sat(g'_r)$ respectively. Then a'_r satisfies g_r so g is not deterministic in D. This is a contradiction, so g remains deterministic in D' and D' is deterministic. ☐

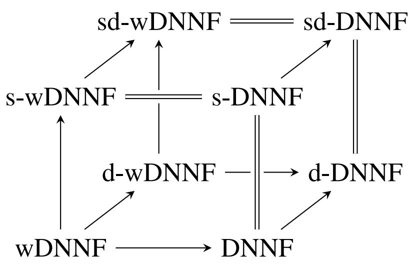
$$\begin{tikzcd} & \text{sd-wDNNF} \arrow[rr, equal] & & \text{sd-DNNF} \arrow[dd, equal] \\ \text{s-wDNNF} \arrow[ur] \arrow[rr, equal] & & \text{s-DNNF} \arrow[ur] \arrow[dd, equal] & \\ & \text{d-wDNNF} \arrow[uu] \arrow[rr] & & \text{d-DNNF} \\ \text{wDNNF} \arrow[uu] \arrow[ur] \arrow[rr] & & \text{DNNF} \arrow[ur] & \end{tikzcd}$$

Figure 2: Succinctness map for different subclasses of NNF.

3.3 Proof of Theorem 1

By Proposition 2, Theorem 1 is equivalent to proving the correctness of the corresponding map for subclasses of wDNNF that for the convenience of the reader is given in Figure 2. So we will exclusively work on that map here and Theorem 1 follows directly.

It was shown by Darwiche and Marquis (2002) that s-DNNF and DNNF are equally succinct, and that sd-DNNF and d-DNNF are equally succinct, the paper also contains the statement DNNF $<$ d-DNNF conditioned on standard complexity theoretic assumptions. The result was made unconditional in (Bova et al. 2016). So we already have the right face of the cube-like succinctness map of Figure 2.

Lemma 4. wDNNF $<$ DNNF.

Proof. Since DNNF $\subseteq$ wDNNF there only is DNNF $\not\leq$ wDNNF to prove. It is readily verified that monotone NNF, that is, NNF with non-negative literal inputs, are wDNNF. In (Bova et al. 2014), see also (Capelli 2016), the separation DNNF $\not\leq$ CNF is shown finding an infinite class of monotone 2-CNF that have polynomial size but whose equivalent DNNF all have exponential size. Monotone CNF are wDNNF so this proves DNNF $\not\leq$ wDNNF. ☐

Peharz et al. (2015) give an algorithm to transform any smooth *weakly* decomposable monotone AC into an equivalent smooth decomposable monotone AC in polynomial time[1]. Careful examination of the algorithm shows that it

[1]Peharz et al. work on sum product networks (SPN) with indicator variables inputs. Their SPN differ from our monotone AC in that the non-negative constants are not inputs of the circuit but

can be adapted to turn any s-wDNNF into an equivalent s-DNNF in polynomial time (the existence of the transformation actually derives from Lemmas 2 and 3). Examining the algorithm even further, one sees that it preserves determinism, so the adapted variant for NNF also gives a polynomial time transformation from sd-wDNNF to sd-DNNF.

Lemma 5. s-wDNNF $\simeq$ s-DNNF *and* sd-wDNNF $\simeq$ sd-DNNF. *But* wDNNF $<$ s-wDNNF.

Proof sketch. The proof that s-DNNF $\leq$ s-wDNNF and sd-DNNF $\leq$ sd-wDNNF is an adaptation of the techniques in (Peharz et al. 2015) to the case of Boolean circuits. The reverse succinctness relations holds since decomposability is a particular kind of weak decomposability.

As for wDNNF $<$ s-wDNNF, wDNNF $\leq$ s-wDNNF comes from s-wDNNF being a subclass of wDNNF, and s-wDNNF $\not\leq$ wDNNF holds for otherwise DNNF $\not\leq$ wDNNF would be violated by transitivity. $\square$

Lemma 6. d-wDNNF $\not\leq$ DNNF.

Proof. Consider the class $\mathcal{F}$ of functions introduced by Sauerhoff (2003) and used in (Bova et al. 2016). All $f \in \mathcal{F}$ on n variables have DNNF-size polynomial in n but d-DNNF-size at least $2^{\Omega(\sqrt{n})}$. For an integer k, let $D_k(f)$ be the smallest d-DNNF representing $f \wedge [w(\cdot) = k]$, i.e., the function whose models are exactly the models of f of weight k. Since the circuit $\bigvee_{k=0}^{n} D_k(f)$ is d-DNNF representing f, there must be a function $\kappa : \mathcal{F} \to \mathbb{N}$ such that $|D_{\kappa(f)}(f)| = 2^{\Omega(\sqrt{|var(f)|})}$ holds for all f. Define the class $\mathcal{F}^* = \{f \wedge [w(\cdot) = \kappa(f)] \mid f \in \mathcal{F}\}$.

We claim that in any wDNNF representing a satisfiable function in $\mathcal{F}^*$, all term subcircuits have the same variables. Consider a wDNNF representing $f \wedge [w(\cdot) = k]$. Let T be one of its term subcircuit and assume $var(f) \setminus var(T) \neq \emptyset$. Let $x \in var(f) \setminus var(T)$, T has a model a with x set to 0 and another model a' identical to a but with x set to 1. But $w(a) \neq w(a')$ so a and a' cannot both satisfy $f \wedge [w(\cdot) = k]$, a contradiction. So all term subcircuits contain all variables.

Combining the above and Proposition 3 for $\mathcal{F}^*$, we get that there is a polynomial relating the d-DNNF- and d-wDNNF-sizes of functions of $\mathcal{F}^*$. So all functions of $\mathcal{F}^*$ have exponential d-wDNNF-size. Since DNNF support polynomial time restriction to models of fixed-weight – see for instance the proof of (Amarilli et al. 2017, Proposition 4.1) which can easily be adapted to DNNF – the functions in $\mathcal{F}^*$ also have polynomial DNNF-size. So the class $\mathcal{F}^*$ gives us d-wDNNF $\not\leq$ DNNF. $\square$

Lemma 7. DNNF $\not\leq$ d-wDNNF.

Proof. We consider the class $\mathcal{F}$ of monotone 2-CNF used in (Bova et al. 2014) to prove DNNF $\not\leq$ CNF. Let F be a monotone 2-CNF from $\mathcal{F}$ on n variables $x_1, \ldots, x_n$, $F = \bigwedge_{k=1}^{m}(x_{k_0} \vee x_{k_1})$. The size of F is polynomial in n while its equivalent DNNF have size exponential in n. Now consider m fresh variables $Z = \{z_1, \ldots, z_m\}$ and define $F' = \bigwedge_{k=1}^{m}((\neg z_k \wedge x_{k_0}) \vee (z_k \wedge x_{k_1}))$. F' is a d-wDNNF, and $\exists Z.F' \equiv F$ (F equals F' after forgetting variables Z, see (Darwiche and Marquis 2002) if needed). Since DNNF support polynomial time variables forgetting (Darwiche and Marquis 2002), DNNF circuits equivalent to F' have exponential size. Thus the class of the circuits $\{F' | F \in \mathcal{F}\}$ proves the separation DNNF $\not\leq$ d-wDNNF. $\square$

weights on the connectors of the $+$-nodes. Such SPN are converted into our monotone AC in polynomial time by replacing each weighted edge by a $\times$-node whose successors include the weight.

Lemma 8. wDNNF $<$ d-wDNNF.

Proof. d-wDNNF $\subset$ wDNNF implies wDNNF $\leq$ d-wDNNF. For d-wDNNF $\not\leq$ wDNNF observe that otherwise we would have d-wDNNF $\simeq$ wDNNF, which would imply d-wDNNF $\leq$ DNNF, thus contradicting Lemma 6. $\square$

Lemma 9. d-wDNNF $<$ d-DNNF *and* d-wDNNF $<$ sd-wDNNF.

Proof. d-DNNF is a subclass of d-wDNNF so d-wDNNF $\leq$ d-DNNF. And d-DNNF $\not\leq$ d-wDNNF holds for otherwise DNNF $\not\leq$ d-wDNNF would be violated by transitivity.

sd-wDNNF is a subclass of d-wDNNF so d-wDNNF $\leq$ sd-wDNNF. Since sd-wDNNF, sd-DNNF and d-DNNF are equally succinct, there must be sd-wDNNF $\not\leq$ d-wDNNF otherwise d-DNNF $\not\leq$ d-wDNNF would be violated by transitivity. $\square$

This last lemma finishes the proof of Theorem 1.

4 Beginning the Map for Positive AC

In this section, we will start drawing a succinctness map for positive AC. Recall that positive AC compute non-negative functions but allow for negative constant inputs or equivalently subtraction. It is known that adding subtraction to arithmetic circuits can decrease their size exponentially (Valiant 1980), so AC$_p$ $<$ AC$_m$.

Since there is no apparent mapping between positive AC and a class of Boolean circuits similar to the mapping ϕ introduced in Section 3.1, we do not obtain a succinctness map for positive AC in the same way we did for monotone AC. We here solve some of the relations on the corresponding map, leaving its completion for future work.

Lemma 10. *Let C be a (smooth) (weakly) decomposable deterministic positive* AC. *Switching the signs of all negative constants in C yields an equivalent (smooth) (weakly) decomposable deterministic monotone* AC. *Therefore the relation* d-γ-AC$_p$ $\simeq$ d-γ-AC$_m$ *holds for any* $\gamma \in$ {D,wD,sD,swD}.

Proof. Smoothness and (weak) decomposability are clearly preserved by the transformation. Recall that no two term subcircuits of C can compute an non-zero value on the same assignment, and that the sum of the functions they compute is that computed by C. So each term subcircuit T computes a positive function, therefore the negative constants in T must be in even number. But then switching the signs of negative constants in C does not change the function computed by any term subcircuit. Thus the monotone AC we get

$$\begin{tikzcd}
& \text{sdwD-AC}_p \arrow[rr, equal] & & \text{sdD-AC}_p \arrow[dd, equal] \\
\text{swD-AC}_p \arrow[ur] \arrow[rr, equal] & & \text{sD-AC}_p \arrow[ur] \arrow[dd, equal] & \\
& \text{dwD-AC}_p \arrow[uu] \arrow[rr] \arrow[dr, dotted, "?"] & & \text{dD-AC}_p \\
\text{wD-AC}_p \arrow[ur] \arrow[uu, dotted, dash, "?"] \arrow[rr, dotted, dash, "?"] & & \text{D-AC}_p \arrow[ur] &
\end{tikzcd}$$

Figure 3: Partial succinctness map for subclasses of positive AC

is equivalent to C and, since its term subcircuits still have pairwise disjoint support, it is deterministic. □

Lemma 11. *γ-AC$_p$ $<$ d-γ-AC$_p$ for any $\gamma \in$ {D,w,sD,swD}.*

Proof. Monotone AC are positive AC so γ-AC$_p \leq \gamma$-AC$_m$. Using Lemma 10 and Theorem 1, we get γ-AC$_p \leq \gamma$-AC$_m <$ d-γ-AC$_m \simeq$ d-γ-AC$_p$ and hence the result. □

Lemma 12. *D-AC$_p \simeq$ sD-AC$_p \simeq$ swD-AC$_p$.*

Proof sketch. The algorithm of (Peharz et al. 2015) works on AC with non-negative constants but remains sound (with no change) when negative constants are allowed. So smooth weakly decomposable positive AC can be made smooth and decomposable in polynomial time, hence sD-AC$_p \leq$ swD-AC$_p$. Since sD-AC$_p$ is also a subclass of swD-AC$_p$, the second succinctness equivalence holds.

D-AC$_p \simeq$ sD-AC$_p$: there only is sD-AC$_p \leq$ D-AC$_p$ to prove. Let $C \in$ D-AC$_p$, if $g \in C$ is a $+$-node such that $x \in var(g_r)$ and $x \notin var(g_l)$, then add a $\times$-node between g_l and g whose successors are g_l and $\overset{+}{\diagup\,\diagdown}$ with children x and $\overline{x}$. This does not impact decomposability. Inserting $\times$-nodes this way for each non smooth $+$-nodes yields a smooth decomposable AC equivalent to C of size at most $O(|var(C)| \times |C|)$. □

The above lemmas are summarized in Figure 3. Three relations, indicated by question marks in the figure are open. Note that the known relations between classes of positive AC coincide with the corresponding relations between classes of monotone AC, which motivates the following question:

Open Question 1. *Do all succinctness relations between classes of monotone* AC *shown Figure 1 hold for the corresponding classes of positive* AC *as well?*

Note that completing the map for positive AC might be very hard: in fact, it would in particular require showing strong lower bounds for D-AC$_p$, a well-known open problem in complexity theory for which the best current result is a recent nearly quadratic lower bound (Alon, Kumar, and Volk 2020). Another question is the relations between the map of monotone AC and that of positive AC: when imposing determinism, the expressive power of positive AC is exactly that of monotone AC, while for unrestricted circuits it is known that positive AC are more succinct than monotone AC (Valiant 1980).

Open Question 2. *For which $\gamma \in$ {D,wD,sD,swD} do we have γ-AC$_p < \gamma$-AC$_m$?*

5 Lower Bounds for Positive AC

5.1 Sum of Decomposable Products

In this section we describe a technique to show lower bounds on the size of *structured*-decomposable positive AC. For NNF, structured decomposability is defined with help of a v-tree (variable tree) (Pipatsrisawat and Darwiche 2008) but the definition usually assumes that constant inputs have been propagated away in the circuit. This is impossible in our model, so we use the v-tree-free definition from (Vergari et al. 2021). The definition assumes smoothness for simplicity.

Definition 7. *An AC C is called smooth* structured-decomposable *when it is smooth and decomposable and, for all $Y \subseteq var(C)$ there is a partition $Y = Y_0 \cup Y_1$ such that, the successors of all $\times$-nodes g in C with $var(g) = Y$ verify $var(g_l) = Y_i$ and $var(g_r) = Y_{1-i}$ for some $i \in \{0,1\}$.*

Definition 8. *Let Z be a set of variables. A* decomposable product *over Z is a function from Z to $\mathbb{R}$ that can be written as a product $f(X) \times h(Y)$ where (X, Y) is a partition of Z and f and h are functions to $\mathbb{R}$. The decomposable product is called* balanced *when $\frac{|Z|}{3} \leq |X|, |Y| \leq \frac{2|Z|}{3}$.*

A common approach to proving lower bounds for decomposable AC analyzes representations of the function it computes in terms of sums of balanced decomposable products. Roughly put, the idea is that when more summands are needed in such a representation, AC for it need to be larger. This technique has been used in recent and not so recent articles, see e.g. (Valiant 1980; Raz and Yehudayoff 2011; Martens and Medabalimi 2014). Translated to Boolean circuits, decomposable products correspond to *combinatorial rectangles*, a tool from communication complexity used in the context of DNNF (Bova et al. 2016).

Variations of the next theorem have been shown several times independently in the literature, see for instance (Martens and Medabalimi 2014, Theorem 38). The structured case comes from an easy modification of that proof, the rough idea is that each decomposable product is built from a different node of the circuit and, thanks to structuredness, all these nodes have the same set of variables, which eventually yields the same partition for the decomposable products.

Theorem 2. *Let F be a non-negative function on $0/1$-variables computed by a decomposable smooth* AC *C. Then F can be written as a sum of N balanced decomposable products over $var(F)$, with $N \leq |C|$ in the form*[2]*.*

$$F = \sum_{i=1}^{N} f_i(X_i) \times h_i(Y_i).$$

If C is structured, the N partitions (X_i, Y_i) are all identical.

[2]Note that (Martens and Medabalimi 2014, Theorem 38) is stated with $N \leq |C|^2$ because the internal nodes in their AC (or SPN) do not have exactly two successors, as do ours. However, they reduce to AC with that property and the square comes from the quadratic size increase in this reduction.

5.2 Lower Bounds for Structured Decomposable Positive AC

In this section we prove the following lower bound.

Proposition 4. *There is a class of positive functions $\mathcal{F}$ such that, for all $F \in \mathcal{F}$, the smallest AC computing F has size polynomial in $|var(F)|$ but the smallest smooth structured decomposable AC computing F has size $2^{\Omega(|var(F)|)}$.*

By Theorem 2, the smallest N for which one can write F as $F = \sum_{i=1}^{N} f_i(X) \times h_i(Y)$ where $f_i(X) \times h_i(Y)$ are balanced decomposable products for the unique partition (X, Y) of *var*(F), is a lower bound on the size of all smooth structured decomposable AC computing F. Thus, proving Proposition 4 boils down to finding non-negative functions where the smallest such N depends exponentially on the number of variables.

Let us fix a function F and a partition (X, Y). The *value matrix* of F with respect to (X, Y) is a $2^{|X|} \times 2^{|Y|}$ matrix M_F whose rows (resp. columns) are uniquely indexed by assignments to X (resp. Y) and such that, for each pair of indices (a_X, a_Y), the entry of M_F at the a_X row and a_Y column is $F(a_X \cup a_Y)$.

Lemma 13. *Let $F = \sum_{k=1}^{N} f_k(X) \times h_k(Y)$ where for all k we have $f_k \times h_k \neq 0$. Let M_F be the value matrix for F and let M_i denote the the value matrix for $f_i \times h_i$ with respect to partition (X, Y). Then*

$$rk(M_F) \leq \sum_{k=1}^{N} rk(M_k) = N.$$

Proof. By construction, $M_F = \sum_{k=1}^{N} M_k$, so $rk(M_F) \leq \sum_{k=1}^{N} rk(M_k)$ holds by sub-additivity of the rank. We now show that $rk(M_k) = 1$ holds for each k. Since $f_k \times h_k \neq 0$, there is a row in M_k which is not a 0-row. Say it is the row indexed by a_X. Then the entries in that row are $f_k(a_X) \times h_k(a_Y)$ for varying a_Y. In any other rows indexed by a'_X, the entries are $f_k(a'_X) \times h_k(a_Y) = (f_k(a'_X)/f_k(a_X)) \times f_k(a_X) \times h_k(a_Y)$ for varying a_Y. Consequently, all rows are multiples of the a_X-row, in other words, all rows of M_k are linearly dependent, hence $rk(M_k) = 1$. □

Using Lemma 13, one sees that proving Proposition 4 boils down to finding functions whose value matrices with respect to *any* balanced partition (X, Y) have rank exponential in the number of variables.

The functions we construct are based on graphs. Let $G = (V, E)$ be a graph, denote $n = |V|$ and, for each vertex v_i in V, create a Boolean variable x_i. We consider the function

$$F_G(x_1, \ldots, x_n) = \prod_{(v_i, v_j) \in E} (1 + \max(x_i, x_j)) \qquad (1)$$

Essentially, for each edge of G, if at least one of its endpoints is assigned 1 in the assignment, then the edge contributes a factor 2 to the product, otherwise it contributes a factor 1. Regardless of the choice of G, the function F_G has a small positive AC: one just has to write $\max(x_i, x_j) = x_i + x_j - x_i x_j$ and see that the number of $\times$ and $+$ operations needed to compute F_G is polynomial in n.

Recall that an *induced matching* is a set $E' \subseteq E$ of edges with pairwise disjoint endpoints, whose set we denote V', such that all edges of G connecting vertices in V' are in E'.

Lemma 14. *Let F_G be as described by (1), let (X, Y) be a partition of $var(F_G)$ and (V_X, V_Y) be the corresponding partition of V. If there is an induced matching m in G between vertices V_l and V_r such that $V_l \subseteq V_X$ and $V_r \subseteq V_Y$, then*

$$rk(M_{F_G}) \geq 2^{|m|}$$

where M_{F_G} is the value matrix of F_G for the partition (X, Y) and $|m|$ is the number of edges in m.

Proof. Rename $M := M_{F_G}$. Identify each vertex with its variable in *var*(F_G) and let $(x_i, y_i)_{i \in [|m|]}$ be the edges of m, with $x_i \in X$ and $y_i \in Y$. Order the variables in X as $X = (x_1, \ldots, x_{|X|})$ and the variables in Y as $Y = (y_1, \ldots, y_{|Y|})$, so that the $|m|$ first variables in each set correspond to the nodes in the matching. Permutations of rows or columns do not change the rank of a matrix so we assume that the assignments indexing the rows and the columns are ordered so that, when seeing the assignments as tuples of 0 and 1, the integers encoded in binary by the tuples are ordered. More formally a_X is before a'_X if and only if $\sum_k a(x_k) 2^{k-1} < \sum_k a'(x_k) 2^{k-1}$. Now consider all $2^{2|m|}$ truth assignments to $var(F_G)$ where variables corresponding to vertices not in $V_l \cup V_r$ are set to 0. Let M^* be the $2^{|m|} \times 2^{|m|}$ sub-matrix of M obtained by keeping only rows and columns indexed by these assignments. The rank of a sub-matrix is always at most that of the matrix, so $rk(M^*) \leq rk(M)$. To prove the lemma, it is enough to show that $rk(M^*) = 2^{|m|}$, which holds if and only if $\det(M^*) \neq 0$. For $0 \leq i \leq |m|$, let M_i^* be the matrix containing the first 2^i rows and first 2^i columns of M^*. We prove by induction on i that all M_i^* have non-zero determinant, which will prove that M^* (which is $M_{|m|}^*$) has non-zero determinant, and therefore full rank. For the base case, $M_0^* = (1)$ has determinant 1. For the general case, assume that $\det(M_i^*) \neq 0$ and observe that $M_{i+1}^* = \left(\begin{array}{c|c} M_i^* & 2M_i^* \\ \hline 2M_i^* & 2M_i^* \end{array}\right)$. The determinant of M_{i+1}^* is

$$\det\left(\begin{array}{c|c} M_i^* & 2M_i^* \\ \hline 2M_i^* & 2M_i^* \end{array}\right) = \det\left(\begin{array}{c|c} -M_i^* & 2M_i^* \\ \hline 0 & 2M_i^* \end{array}\right)$$
$$= \det(-M_i^*) \det(2M_i^*) = (-2)^{2^i} \det(M_i^*)^2 \neq 0.$$

□

So if, for *every* balanced partition of V, we have a large enough induced matching M between the two sides, then the rank of the value matrix for F_G for any balanced partition is large, thus many balanced decomposable products are needed in a sum representing F_G. The only thing left is to find graphs G with this "large enough matching" property, which turn out to be *expander graphs*. A d-regular graph is a graph whose vertices all have degree d. A (c, d)-expander graph on vertices V is a d-regular graph such that for any $S \subseteq V$ of size $|S| \leq |V|/2$, it holds that $|N(S)| \geq c|S|$, where $N(S) = \{v \in V \setminus S \mid (u, v) \in E, u \in S\}$.

Theorem 3. *(Alon and Spencer 2000, Section 9.2) There is, for some $c > 0$, an infinite sequence of $(c, 3)$-expander graphs $(G_i)_{i \in \mathbb{N}}$.*

We use these expander graphs for our lower bound.

Lemma 15. *Let $G = (V, E)$ be a $(c, 3)$-expander graph with $n = |V|$, and let $V = V_1 \uplus V_2$ be a balanced partition of V, then there exists an induced matching m of size $\Omega(n)$ between V_1 and V_2.*

Proof. V_1 or V_2 has size at most $n/2$, say $|V_1| \le n/2$. There is $N(V_1) \subseteq V_2$ and $|N(V_1)| \ge c|V_1| \ge cn/3$ where the last inequality comes from the partition being balanced. So at least $cn/3$ edges connect V_1 to V_2. Since G is 3-regular, at least a third of these edges form an matching in G, and a third of these matching edge share no endpoint in V_1, and finally a third of these edges share no endpoint in V_2 either. So we obtain a induced matching between V_1 and V_2 of size at least $cn/81$. ☐

Combining Theorems 2 and 3 with Lemmas 13, 14 and 15 yields Proposition 4.

6 Conclusion

We have started drawing succinctness maps for arithmetic circuits modeled after that proposed for NNF in (Darwiche and Marquis 2002). Due to great amount of recent work on practical applications of AC with specific structural restrictions, we have studied classes of AC for combinations of four key restrictions. Using a mapping between monotone AC and NNF, we have drawn the full succinctness map for monotone AC by lifting the existing map for NNF and extending it to incorporate new classes defined with weak decomposability. In certain cases we could show that positive and monotone AC have the same expressive power, which gave us some succinctness results between classes of positive AC for free. We leave the challenging task of determining the remaining relations between classes of positive AC as an open question. Finally, we have also introduced techniques to prove lower bounds on structured positive AC and applied them to the case of smooth structured-decomposable AC.

Appendix

Lemma 1. *Let C be a (weakly) decomposable* AC *(resp.* NNF*) then:*

- *if C is smooth, all variables appear in all term subcircuits*
- *if C is deterministic, then any two distinct term subcircuits T and T' verify $T \times T' = 0$ (resp. $T \wedge T' \equiv 0$).*

Proof. We only show the lemma for AC, as the proof for NNF is completely analogous. The two points are shown by induction on the depth of C, i.e., the number of nodes in a longest directed path in C. AC of depth 1 are single variable nodes or constant nodes, and thus the statement of the lemma is straightforward. Now assume the lemma holds for all (w)D-AC of depth at most k and consider an (w)D-AC C of depth $k + 1$. Let g be the source node of C. Let C_l and C_r be the (w)D-AC under g_l and g_r.

If g is a $\times$-node then the term subcircuits of C are products $T = T_l \times T_r$ where T_l and T_r are term subcircuits of C_l and C_r. If C is smooth, then $\mathit{var}(T) = \mathit{var}(T_l) \cup \mathit{var}(T_r) = \mathit{var}(C_l) \cup \mathit{var}(C_r) = \mathit{var}(C)$ holds by induction. If C is deterministic, then let $T = T_l \times T_r$ and $T' = T'_l \times T'_r$ be distinct term subcircuits of C. We have $T'_l \neq T_l$ or $T'_r \neq T_r$ and thus by induction, for every assignment a, we have $T_l(a) \times T'_l(a) = 0$ or $T_r(a) \times T'_r(a) = 0$, so $T' \times T = 0$.

If g is a $+$-node, then every term subcircuit T of C is either equivalent to a term subcircuit T_l of C_l or to a term subcircuit T_r of C_r. Assume C is smooth, then $\mathit{var}(C) = \mathit{var}(C_l) = \mathit{var}(C_r)$, but then $\mathit{var}(T)$ is either $\mathit{var}(T_l)$ or $\mathit{var}(T_r)$, which by induction equals $\mathit{var}(C)$. Now when C is deterministic there is $C_l \times C_r = 0$, so any term subcircuits $T = T_l$ and $T' = T'_r$ verify $T \times T' = 0$. And by induction any two distinct subcircuits $T = T_l$ and $T' = T'_l$ verify $T \times T' = T_l \times T'_l = 0$ (likewise for T_r and T'_r). ☐

Lemma 2. *When C is a monotone* AC*, $\phi(C)$ is an* NNF *whose models are $\mathit{supp}(C)$. Moreover if C is (weakly) decomposable, deterministic, or smooth, then $\phi(C)$ is as well.*

Proof. The graph of $\phi(C)$ is that of C and ϕ contains only $\wedge$- and $\vee$-nodes, thus $\phi(C)$ is an NNF. It is easy to see that for each node g in C we have $\mathit{var}(g) = \mathit{var}(\phi(g))$, so smoothness and (weak) decomposability are preserved.

We prove that by induction on the depth of C that (1) $\mathit{sat}(\phi(C)) = \mathit{supp}(C)$ and (2) if C is deterministic, then so is $\phi(C)$. If C has depth 1, then it is either a constant input or a literal input. In the case $C = \alpha$, if $\alpha > 0$ then $\mathit{supp}(C) = \{a_\emptyset\} = \mathit{sat}(1) = \mathit{sat}(\phi(C))$. If $\alpha = 0$ then $\mathit{supp}(C) = \emptyset = \mathit{sat}(0) = \mathit{sat}(\phi(C))$. In the case $C = \ell_x$ there is $\phi(C) = C$ so we are done. Now assume (1) and (2) hold for all AC of depth at most k and suppose C has depth $k + 1$. Let g be its source node.

If g is a $\times$-node, then $C(a) = 0$ if and only if $g_l(a_l) = 0$ or $g_r(a_r) = 0$, where a_l and a_r denote the restrictions of a to $\mathit{var}(g_l)$ and $\mathit{var}(g_r)$ respectively. So $a \notin \mathit{supp}(C)$ if and only if $a_l \notin \mathit{supp}(g_l)$ or $a_r \notin \mathit{supp}(g_r)$. By induction $\mathit{supp}(g_{l/r}) = \mathit{sat}(\phi(g_{l/r}))$, so $a \notin \mathit{supp}(C)$ if and only if $a \notin \mathit{sat}(\phi(g_l) \wedge \phi(g_r)) = \mathit{sat}(\phi(C))$. So (1) holds.

If g is a $+$-node, then $C(a) = 0$ iff $g_l(a_l) = 0$ and $g_r(a_r) = 0$. So $a \notin \mathit{supp}(C)$ iff $a_l \notin \mathit{supp}(g_l)$ and $a_r \notin \mathit{supp}(g_r)$. By induction $\mathit{supp}(g_{l/r}) = \mathit{sat}(\phi(g_{l/r})$, so $a \notin \mathit{supp}(C)$ iff $a \notin \mathit{sat}(\phi(g_l) \vee \phi(g_r)) = \mathit{sat}(\phi(C))$. So (1) holds. As for (2), if $a_r \in \mathit{supp}(g_r)$ implies $a_l \notin \mathit{supp}(g_l)$ and vice-versa, then $\mathit{supp}(g_{l/r}) = \mathit{sat}(\phi(g_{l/r})$ yields that the source $\vee$-node of $\phi(C)$ is deterministic. ☐

Lemma 3. *For every* NNF *D, there exists an* AC *C of size $|D|$ with $\mathit{supp}(C) = \mathit{sat}(D)$. Moreover if D is (weakly) decomposable, deterministic, or smooth, then so is C.*

Proof. It suffices to replace each $\wedge$-node in D by a $\times$-node and each $\vee$-node by $+$-node. Let $\psi(D)$ be that AC. Clearly $|D| = |\psi(D)|$, and it is easy to see that for each node g in D, $\mathit{var}(g) = \mathit{var}(\psi(g))$, so smoothness and (weak) decomposability are preserved by ψ. Moreover $\phi(\psi(D)) = D$, so $\mathit{sat}(D) = \mathit{supp}(\psi(D))$. Determinism is preserved since $\mathit{sat}(g) = \mathit{supp}(\psi(g))$ holds for all nodes g in D. ☐

Acknowledgments

The authors were supported by the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

References

Akshay, S.; Arora, J.; Chakraborty, S.; Krishna, S. N.; Raghunathan, D.; and Shah, S. 2019. Knowledge compilation for boolean functional synthesis. In Barrett, C. W., and Yang, J., eds., *2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019*, 161–169. IEEE.

Alon, N., and Spencer, J. H. 2000. *The Probabilistic Method, Second Edition*. John Wiley.

Alon, N.; Kumar, M.; and Volk, B. L. 2020. Unbalancing sets and an almost quadratic lower bound for syntactically multilinear arithmetic circuits. *Comb.* 40(2):149–178.

Amarilli, A.; Bourhis, P.; Jachiet, L.; and Mengel, S. 2017. A circuit-based approach to efficient enumeration. In Chatzigiannakis, I.; Indyk, P.; Kuhn, F.; and Muscholl, A., eds., *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, 111:1–111:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Amarilli, A.; Capelli, F.; Monet, M.; and Senellart, P. 2020. Connecting knowledge compilation classes and width parameters. *Theory Comput. Syst.* 64(5):861–914.

Bova, S.; Capelli, F.; Mengel, S.; and Slivovsky, F. 2014. Expander cnfs have exponential DNNF size. *CoRR* abs/1411.1995.

Bova, S.; Capelli, F.; Mengel, S.; and Slivovsky, F. 2016. Knowledge compilation meets communication complexity. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 1008–1014. IJCAI/AAAI Press.

Capelli, F. 2016. *Structural restriction of CNF-formulas: application to model counting and knowledge compilation*. Ph.D. Dissertation, Université Paris Diderot (Paris 7), Sorbonne Paris Cité.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artif. Intell.* 172(6-7):772–799.

Choi, A., and Darwiche, A. 2017. On relaxing determinism in arithmetic circuits. In Precup, D., and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 825–833. PMLR.

Choi, A.; Kisa, D.; and Darwiche, A. 2013. Compiling probabilistic graphical models using sentential decision diagrams. In van der Gaag, L. C., ed., *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings*, volume 7958 of *Lecture Notes in Computer Science*, 121–132. Springer.

Choi, Y.; Vergari, A.; and Van den Broeck, G. 2020. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report.

Dang, M.; Vergari, A.; and Van den Broeck, G. 2020. Strudel: Learning structured-decomposable probabilistic circuits. In Jaeger, M., and Nielsen, T. D., eds., *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, 137–148. PMLR.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res.* 17:229–264.

Darwiche, A. 2003. A differential approach to inference in bayesian networks. *J. ACM* 50(3):280–305.

Dechter, R., and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artif. Intell.* 171(2-3):73–106.

Dennis, A. W. 2016. *Algorithms for Learning the Structure of Monotone and Nonmonotone Sum-Product Networks*. Ph.D. Dissertation, Brigham Young University.

Grigoriev, D., and Karpinski, M. 1998. An exponential lower bound for depth 3 arithmetic circuits. In Vitter, J. S., ed., *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, 577–582. ACM.

Huang, J.; Chavira, M.; and Darwiche, A. 2006. Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, 1143–1148. AAAI Press.

Jerrum, M., and Snir, M. 1982. Some exact complexity results for straight-line computations over semirings. *J. ACM* 29(3):874–897.

Khosravi, P.; Choi, Y.; Liang, Y.; Vergari, A.; and den Broeck, G. V. 2019. On tractable computation of expected predictions. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 11167–11178.

Kisa, D.; den Broeck, G. V.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Lowd, D., and Domingos, P. M. 2008. Learning arithmetic circuits. In McAllester, D. A., and Myllymäki, P., eds., *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, 383–392. AUAI Press.

Martens, J., and Medabalimi, V. 2014. On the expressive efficiency of sum product networks. *CoRR* abs/1411.7717.

Peharz, R.; Tschiatschek, S.; Pernkopf, F.; and Domingos, P. M. 2015. On theoretical properties of sum-product networks. In Lebanon, G., and Vishwanathan, S. V. N., eds., *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org.

Pipatsrisawat, K., and Darwiche, A. 2008. New compilation languages based on structured decomposability. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 517–522. AAAI Press.

Poon, H., and Domingos, P. M. 2011. Sum-product networks: A new deep architecture. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, 689–690. IEEE Computer Society.

Rahman, T.; Kothalkar, P.; and Gogate, V. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In Calders, T.; Esposito, F.; Hüllermeier, E.; and Meo, R., eds., *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, volume 8725 of *Lecture Notes in Computer Science*, 630–645. Springer.

Raz, R., and Yehudayoff, A. 2011. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *J. Comput. Syst. Sci.* 77(1):167–190.

Raz, R. 2009. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM* 56(2):8:1–8:17.

Raz, R. 2010. Elusive functions and lower bounds for arithmetic circuits. *Theory Comput.* 6(1):135–177.

Rooshenas, A., and Lowd, D. 2016. Discriminative structure learning of arithmetic circuits. In Schuurmans, D., and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 4258–4259. AAAI Press.

Sauerhoff, M. 2003. Approximation of boolean functions by combinatorial rectangles. *Theor. Comput. Sci.* 301(1-3):45–78.

Shen, Y.; Choi, A.; and Darwiche, A. 2016. Tractable operations for arithmetic circuits of probabilistic models. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 3936–3944.

Shpilka, A., and Yehudayoff, A. 2010. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.* 5(3-4):207–388.

Valiant, L. G. 1980. Negation can be exponentially powerful. *Theor. Comput. Sci.* 12:303–314.

Vergari, A.; Choi, Y.; Liu, A.; Teso, S.; and den Broeck, G. V. 2021. A compositional atlas of tractable circuit operations: From simple transformations to complex information-theoretic queries. *CoRR* abs/2102.06137.