Backdoor Decomposable Monotone Circuits and their Propagation Complete Encodings

Petr Kučera* Petr Savický†

We describe a compilation language of backdoor decomposable monotone circuits (BDMCs) which generalizes several concepts appearing in the literature, e.g. DNNFs and backdoor trees. A BDMC sentence is a monotone circuit which satisfies decomposability property (such as in DNNF) in which the inputs (or leaves) are associated with CNF encodings of some functions required to be propagation complete (PC) or at least unit refutation complete (URC). BDMCs are strictly more succinct than both DNNF and backdoor trees. On the other hand, we show that a representation of a boolean function with a BDMC can be compiled into a PC encoding of the same function whose size is polynomial in the size of the input BDMC sentence. As a consequence, BDMCs are equally succinct as PC encodings, however, their structure allows to incorporate parts equivalent to a DNNF. This makes BDMCs suitable for applications, where it is beneficial to combine DNNF with tractable classes of CNF formulas like 2-CNF or renamable Horn formulas.

1 Introduction

We describe a compilation language for representing boolean functions which generalizes several concepts appearing in the literature. A boolean function is represented using a structure consisting of a monotone circuit satisfying the decomposability property (such as in DNNF) and whose inputs called leaves are associated with CNF encodings from a suitable base class representing some simpler functions. We call this structure backdoor decomposable monotone circuit (BDMC), because it is a generalization of backdoor trees introduced in [26]. We mainly consider two versions, a PC-BDMC where the base class consists of propagation complete (PC) encodings, and URC-BDMC where the base class consists of unit refutation complete (URC) encodings. These base classes are the largest

*Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University, Czech Republic, kucerap@ktiml.mff.cuni.cz
†Institute of Computer Science, The Czech Academy of Sciences, Czech Republic, savicky@cs.cas.cz
ones, for which our construction can be used. The main result holds also for their subclasses, in particular for 2-CNF formulas and renamable Horn formulas which are suitable as base classes in applications, since they can be recognized in polynomial time.

A DNNF \([11]\) is a BDMC with the literals in the leaves, so it is both a PC-BDMC and a URC-BDMC. Moreover, a disjunction of URC encodings \([4]\) is a URC-BDMC. If we consider circuits with only one node, we obtain that PC-BDMC sentences generalize PC formulas introduced in \([5]\) and URC-BDMC sentences generalize URC formulas introduced in \([14]\). A general BDMC combines these structures and this makes it suitable for applications, where it is beneficial to combine DNNF with tractable classes of CNF formulas like 2-CNF or renamable Horn formulas.

The main result of this paper is that one can compile a PC-BDMC or URC-BDMC into a PC or URC encoding of size polynomial with respect to the total size of the input BDMC sentence. As a consequence, PC-BDMCs and PC encodings are equally succinct and also URC-BDMCs and URC encodings are equally succinct. Moreover, since PC encodings and URC encodings are equally succinct by Theorem 1 in \([2]\), these four models are equivalent in the sense of the knowledge compilation map \([13]\).

Combining the results of \([6]\) or \([8]\) with the fact that both DNNFs and PC encodings are special cases of PC-BDMCs, we obtain that the language of PC-BDMCs is strictly more succinct than the language of DNNFs. We also present an example of a CNF formula for which every backdoor tree with respect to the base class of renamable Horn formulas has exponential size, although the function can be represented by a DNNF sentence, so also by a renamable Horn BDMC of linear size.

A smooth DNNF can be compiled into a propagation complete encoding of linear size with respect to the size of the input by techniques described in \([19, 15]\). Our result generalizes this to a more general structure, where the leaves contain URC or PC encodings instead of single literals and smoothness is not required. On the other hand, in case of PC encodings, the method of the transformation is different from the method used in \([19, 15]\) and the size of the output is not bounded by a linear function of the size of the input, although it is still polynomial. In case of URC encodings, the transformation is linear.

The authors of \([4]\) studied properties of URC encodings and proved, in particular, that the disjunction closure can be computed in polynomial time for URC encodings. Our result generalizes this in two directions. We describe a polynomial time transformation of an arbitrary URC-BDMC sentence, which is a more general structure built on top of a collection of URC encodings than disjunction, into a single URC encoding. Moreover, our approach generalizes to PC-BDMCs and PC encodings instead of URC-BDMCs and URC encodings. Similarly as in \([4]\), our construction uses a Tseitin transformation of the BDMC in the first step and then simulates the unit propagation under conditions represented by additional literals. In particular, for a URC-BDMC sentence that is a disjunction of URC encodings, the construction is essentially the same as in \([4]\) up to the naming of the variables.

Consider a boolean function \(f(x)\) which is represented by a CNF \(\varphi\). It is known that the size of a DNNF representing \(f\) can be parameterized by incidence treewidth of \(\varphi\) \(([7, 24])\). It follows by construction described in \([15]\) that the size of a PC encoding of \(f\) can
be parameterized by incidence treewidth of \( \varphi \) as well. Since backdoor trees are a special case of BDMCs, our result implies that \( f \) has a PC encoding of size \( O(p(t)\text{poly}(|\varphi|)) \), where \( p(t) \) is a function which depends only on the minimum size \( t \) of a backdoor tree with any of the base classes of Horn formulas, renamable Horn formulas, or 2-CNF formulas. This implies that the size of a PC encoding can be parameterized also by the size of a backdoor tree with some of these base classes, although in these cases, DNNF can have an exponential size.

The paper is organized as follows. In Section 2 we give necessary definitions and recall the results we use in the rest of the paper. In Section 3 we introduce the notion of a backdoor decomposable monotone circuit and compare it to related target compilation languages with respect to succinctness. In Section 4 we describe our construction of a PC encoding given a BDMC with PC encodings in the leaves. We discuss a construction of a URC encoding in Section 5. In Section 6 we discuss consequences of our construction for the size of a PC encoding of a boolean function parameterized by the size of a backdoor tree of a CNF representing it.

### 2 Definitions and Notation

#### 2.1 CNF Encoding

A formula in conjunctive normal form (CNF formula) is a conjunction of clauses. A clause is a disjunction of a set of literals and a literal is a variable \( x \) (positive literal) or its negation \( \neg x \) (negative literal). Given a set of variables \( x \), \( \text{lit}(x) \) denotes the set of literals on variables in \( x \). A \( k \)-CNF formula consists only of clauses of length at most \( k \). We treat a clause as a set of literals and a CNF formula as a set of clauses. In particular, \(|C|\) denotes the number of literals in a clause \( C \) and \(|\varphi|\) denotes the number of clauses in a CNF formula \( \varphi \). We denote \( \|\varphi\| = \sum_{C \in \varphi} |C| \) the length of a CNF formula \( \varphi \).

Clause \( C \) is Horn if it contains at most one positive literal, it is definite Horn, if it contains exactly one positive literal. A definite Horn clause \( \neg x_1 \lor \cdots \lor \neg x_k \lor y \) represents the implication \( x_1 \land \cdots \land x_k \rightarrow y \) and we use both kinds of notation interchangeably. The set of variables \( \{x_1,\ldots,x_k\} \) in the assumption of a definite Horn clause is called its source set, variable \( y \) is called its target. A CNF formula \( \varphi \) is renamable Horn if it can be placed in Horn form by replacing some variables with their respective negation, for more details see [10).

A partial assignment \( \alpha \) of values to variables in \( z \) is a subset of \( \text{lit}(z) \) that does not contain a complementary pair of literals, so we have \(|\alpha \cap \text{lit}(x)| \leq 1 \) for each \( x \in z \). By \( \varphi(\alpha) \) we denote the formula obtained from \( \varphi \) by the partial setting of the variables defined by \( \alpha \). We identify a set of literals \( \alpha \) (in particular a partial assignment) with the conjunction of these literals if \( \alpha \) is used in a formula such as \( \varphi(x) \land \alpha \). A mapping \( a : x \rightarrow \{0,1\} \) or, equivalently, \( a \in \{0,1\}^x \) represents a full assignment of values to \( x \). A full assignment can also be considered as a special type of partial assignment and we use these representations interchangeably.

We consider encodings of boolean functions defined as follows.
Definition 2.1 (Encoding) Let \( f(x) \) be a boolean function on variables \( x = (x_1, \ldots, x_n) \). Let \( \varphi(x, y) \) be a CNF formula on \( n + m \) variables where \( y = (y_1, \ldots, y_m) \). We call \( \varphi \) a CNF encoding of \( f \) if for every \( a \in \{0, 1\}^x \) we have
\[
f(a) = 1 \iff (\exists b \in \{0, 1\}^y) \, \varphi(a, b) = 1.
\]
(1)
The variables in \( x \) and \( y \) are called input variables and auxiliary variables, respectively.

2.2 Propagation and Unit Refutation Complete Encodings

We are interested in encodings which are propagation complete or at least unit refutation complete. These notions rely on unit resolution or unit propagation which is a well known procedure in SAT solving \[3\]. The unit resolution rule allows to derive clause \( C \setminus \{l\} \) given a clause \( C \) and a unit clause \( \neg l \). We say that a clause \( C \) can be derived from \( \varphi \) by unit resolution, if \( C \) can be derived from \( \varphi \) by a series of unit resolution rules and we denote this fact with \( \varphi \vdash_1 C \). The notion of propagation complete CNF formulas was introduced in \[5\] as a generalization of unit refutation complete CNF formulas introduced in \[14\]. We use the following more general notions of PC and URC encodings.

Definition 2.2 Let \( f(x) \) be a boolean function on variables \( x = (x_1, \ldots, x_n) \). Let \( \varphi(x, y) \) be a CNF encoding of \( f(x) \) with auxiliary variables \( y \).

- We say that \( \varphi \) is a unit refutation complete encoding (URC encoding) of \( f(x) \) if the following implication holds for every partial assignment \( \alpha \subseteq \text{lit}(x) \):
\[
f(x) \land \alpha \models \bot \implies \varphi \land \alpha \vdash_1 \bot
\]
(2)

- We say that \( \varphi \) is a propagation complete encoding (PC encoding) of \( f(x) \) if for every partial assignment \( \alpha \subseteq \text{lit}(x) \) and for each \( h \in \text{lit}(x) \), such that
\[
f(x) \land \alpha \models h
\]
(3)
we have
\[
\varphi \land \alpha \vdash_1 h \quad \text{or} \quad \varphi \land \alpha \vdash_1 \bot.
\]
(4)
The definition of a PC encoding is less restrictive than requiring that formula \( \varphi \) is propagation complete as defined in \[5\]. This definition assumes that \( f \) is the function represented by \( \varphi \), so we do not distinguish input and auxiliary variables and the implication from \( \text{lit}(x) \) to \( \text{lit}(x) \) is required for the literals on all the variables.

It was shown in \[1\] that a prime 2-CNF formula is always propagation complete, thus the same holds for 2-CNF encodings. On the other hand, Horn and renamable Horn formulas are unit refutation complete \[14\].
2.3 DNNF

Let us briefly recall the notion of DNNF [11].

Definition 2.3 A sentence in NNF is a rooted, directed acyclic graph (DAG) where each leaf node is labeled with 1, 0, \( x \) or \( \neg x \), \( x \in \mathbf{x} \) where \( \mathbf{x} \) is a set of variables. Each internal node is labeled with \( \wedge \) or \( \vee \) and can have arbitrarily many inputs.

Assume \( D \) is a NNF with variables \( \mathbf{x} \) and nodes \( V = \{v_1, \ldots, v_N\} \). For technical reasons, we assume that the inputs of each gate precede it in the list of nodes. Hence, if \( v_i \) is an input to \( v_j \), then \( i < j \). For every \( v \in V \), let \( \text{var}(v) \) denote the set of variables from which the node \( v \) is reachable by a directed path. Each node \( v_i, i = 1, \ldots, N \) represents a function \( f_i(\mathbf{x}_i) \) defined on variables \( \text{var}(v_i) = \mathbf{x}_i \subseteq \mathbf{x} \). The language of DNNF sentences is defined as follows.

Definition 2.4 We say that a NNF \( D \) is decomposable (DNNF), if the inputs \( v_{j_1}, \ldots, v_{j_k} \) of each \( \wedge \)-gate in \( D \) represent functions defined on pairwise disjoint sets of variables \( \text{var}(v_{j_1}), \ldots, \text{var}(v_{j_k}) \).

2.4 Backdoor Trees

We first recall the concept of backdoor sets introduced in [28, 27]. As a base class for a backdoor set we consider a class of CNF formulas for which the satisfiability and the membership problem can be solved in polynomial time. Let \( C \) be such a base class and let \( \varphi \) be a CNF formula on variables \( \mathbf{x} \). Then \( B \subseteq \mathbf{x} \) is a strong \( C \)-backdoor set of \( \varphi \), if for every assignment \( a : B \rightarrow \{0, 1\} \), the formula \( \varphi(a) \) belongs to \( C \). Finding smallest strong Horn-backdoor sets and strong 2-CNF-backdoor sets is fixed-parameter tractable with respect to the size of the smallest backdoor set [22]. Other classes of CNF formulas were considered as base classes in literature, let us mention backdoors to a heterogeneous class of CNFs [16].

Backdoor sets were generalized in [26] to backdoor trees. Assume, \( \varphi(\mathbf{x}) \) is a CNF formula. Consider a decision tree \( T \) on a set of variables \( B \subseteq \mathbf{x} \) in which each leaf \( v \) of \( T \) represents the partial assignment \( \tau_v \) given by the values of variables on the path from the root of \( T \) to \( v \). We say that \( T \) is a \( C \)-backdoor tree of \( \varphi \) if for every leaf \( v \) the formula \( \varphi(\tau_v) \) belongs to \( C \). We denote \( |T| \) the number of leaves in \( T \). The size of \( T \) is defined as \( \log_2 |T| \) so that it is comparable with the sizes of backdoor sets. In particular, it was observed in [26] that if \( b \) is the size of a smallest \( C \)-backdoor set of a CNF \( \varphi \), then the number of leaves in a smallest \( C \)-backdoor tree of \( \varphi \) satisfies \( b + 1 \leq |T| \leq 2^b \) and the size \( s \) of \( T \) satisfies \( \log_2 b < s \leq b \). It was shown in [26] that finding a \( C \)-backdoor tree of a given size \( s \) is fixed-parameter tractable for classes of Horn and 2-CNF formulas.

3 Backdoor Decomposable Monotone Circuits

In this section we introduce a language of backdoor decomposable monotone circuits (BDMC) which consists of sentences formed by a combination of a decomposable monotone circuit with CNF formulas from a suitable base class \( C \) at the leaves. Let \( \varphi_i(\mathbf{x}_i, \mathbf{y}_i) \)
for \( i = 1, \ldots, L \) be encodings from \( C \) whose input variables \( x_i \) are subsets of a set of variables \( x \) and let us consider their combination by a monotone circuit \( D \) with \( L \) inputs. This is a DAG with \( N \geq L \) nodes \( V = \{v_1, \ldots, v_N\} \) where nodes \( v_1, \ldots, v_L \) are leaves and \( v_N \) is the root of \( D \) representing the output. Each leaf \( v_i \) is associated with the encoding \( \varphi_i(x_i, y_i) \) where \( x_i \subseteq x \) are the input variables of the encoding and \( y_i \) are its auxiliary variables. Given two different leaves \( v_i \) and \( v_j \) with associated CNF encodings \( \varphi_i(x_i, y_i) \) and \( \varphi_j(x_j, y_j) \), we assume that \( y_i \cap y_j = \emptyset \), i.e. the sets of auxiliary variables of encodings in different leaves are pairwise disjoint. Each non-leaf node is labeled with \( \lor \) or \( \land \) and its inputs precede it in the list, so if \( v_r \) is an input of \( v_i \), then \( r < i \).

For each non-leaf node \( v_i \) with inputs \( v_{i_1}, \ldots, v_{i_k} \), let \( x_i = x_{i_1} \cup \cdots \cup x_{i_k} \), where the sets \( x_r \) for \( r \in \{i_1, \ldots, i_k\} \) are defined inductively in the same way. Moreover, let \( f_i(x_i) \) be the function defined on the variables \( x_i \) as follows. If \( 1 \leq i \leq L \), \( f_i \) is the function represented by the encoding \( \varphi_i \). If \( v_i \) is an \( \land \)-node or an \( \lor \)-node, \( f_i \) is the conjunction or the disjunction, respectively, of the functions \( f_r \) represented by the inputs \( v_r \) of \( v_i \).

**Definition 3.1 (Backdoor Decomposable Monotone Circuit)** Let \( C \) be a base class of CNF encodings. A sentence in the language of backdoor decomposable monotone circuits with respect to base class \( C \) (C-BDMC) is a directed acyclic graph as described above, where each leaf node is labeled with a CNF encoding from \( C \) and each internal node is labeled with \( \land \) or \( \lor \) and can have arbitrarily many inputs. Moreover, the nodes labeled with \( \land \) satisfy the decomposability property with respect to the input variables of the encodings in the leaves, which means that if \( v_i \) is an \( \land \)-node with inputs \( v_{i_1}, \ldots, v_{i_k} \), then the functions \( f_r(x_r) \) for \( r \in \{i_1, \ldots, i_k\} \) are defined on pairwise disjoint sets of variables \( x_r \). The function represented by the sentence is the function \( f_N \) defined on the variables \( x = x_N = \bigcup_{i=1}^L x_i \).

In Section 4 we consider the cases when \( C \) is equal to the class of PC or URC encodings. These classes admit a polynomial time satisfiability test. However, the corresponding membership tests are co-NP-complete, since it is co-NP complete to check if a formula is URC [9, 17] or PC [1]. For this reason, when the complexity of algorithms searching for a BDMC for a given function is in consideration, a suitable subclass with a polynomial time membership test can be used, such as 2-CNF or (renamable) Horn formulas.

By the results of [6] and [8], there are classes \( C \) of monotone CNF formulas such that for each \( \varphi \in C \), the DNNF size of \( \varphi \) is \( 2^{O(|\varphi|)} \). In particular, [8] presents a class \( C \) with the above property consisting of monotone 3-CNF formulas and [6] presents a class \( C \) consisting of monotone 2-CNF formulas. In both cases, the proof of existence of the corresponding class is non-constructive. Every irredundant monotone CNF is in prime implicate form, which means that it is formed by all the prime implicates of the represented function. Such a formula is clearly propagation complete, see [1] for more detail. Together with the known fact that PC encodings are at least as succinct as DNNFs, the lower bounds on DNNF size from [6] and [8] imply the following.

**Corollary 3.2** The language of PC encodings is strictly more succinct than the language of DNNF sentences.
The language of PC-BDMCs and also the language of 2-CNF-BDMCs contains the language of DNNFs as a subset consisting of BDMCs with the literals in the leaves. Hence, the lower bound on DNNF size from [6] implies the following.

**Corollary 3.3** The language of PC-BDMCs and also the language of 2-CNF-BDMCs is strictly more succinct than the language of DNNF sentences.

Let us also point out that Theorem 4.7 proven below implies the following.

**Proposition 3.4** The language of PC encodings and the language of PC-BDMC sentences are equally succinct.

**Proof.** PC encodings are a special case of PC-BDMC with one node. The opposite direction follows from Theorem 4.7. □

Note that a decision node can be rewritten as a disjunction of two decomposable conjunctions. Consequently, if C is a base class which contains all single literal formulas, then C-backdoor trees form a special case of C-BDMCs. Let us consider C-BDMCs, where C is the class of renamable Horn formulas, and let us compare the succinctness of them with the backdoor trees with respect to base class C. When using a backdoor tree as a representation of a function, the whole structure consists of the backdoor tree itself and the original formula. However, since we prove a lower bound on the size of the representation, it is sufficient to formulate the bound in terms of the number of the leaves of the backdoor tree.

Theorem 3.5 below can be viewed as a stronger version of the second part of Proposition 9 in [26] reformulated for comparing renamable Horn BDMCs to renamable Horn backdoor trees. In the proof, we use the same construction as the authors of [26], however, the obtained lower bound is larger.

**Theorem 3.5** For every n divisible by 3, there is a boolean function \( f_n \) of \( n \) variables with the following properties:

- \( f_n \) is expressible by a CNF formula of size \( O(n) \),
- \( f_n \) is expressible by a renamable Horn BDMC and a DNNF of size \( O(n) \),
- for every CNF formula \( \varphi \) representing \( f_n \), every backdoor tree for \( \varphi \) with respect to the base class of renamable Horn formulas has at least \( 2^{n/3} \) leaves.

**Proof.** We use the same construction as the one which is used in the proof of Proposition 9 in [26]. Given \( m \geq 1 \), define for each \( i = 1, \ldots, m \)
\[
\psi_i = (a_i \vee b_i \vee c_i)(-a_i \vee -b_i \vee -c_i)
\]
and let us consider the function \( f_n \) on \( n = 3m \) variables \( \{a_i, b_i, c_i\}_{i=1}^m \) defined by
\[
\psi = \bigwedge_{i=1}^m \psi_i.
\]
For any $i \in \{1, \ldots, m\}$, it can be easily checked that the function represented by $\psi_i$ is not renamable Horn, however, it can be expressed by a DNF of size 3. If $\psi_i$ is replaced by this DNF for each $i$, the formula $\psi$ becomes a DNNF of size $O(n)$ for $f_n$ and it can be interpreted also as a renamable Horn BDMC for $f_n$ of size $O(n)$ with the literals in the leaves.

Let us prove that any renamable Horn backdoor tree of any CNF formula $\varphi$ equivalent to $\psi$ contains at least $2^m$ nodes. Consider a backdoor tree $T$ with respect to $\varphi$ which has renamable Horn formulas in the leaves. We prove that every leaf of $T$ is visited by at most $3^m$ satisfying assignments of $\psi$. Since $\psi$ has $6^m$ satisfying assignments, the tree has at least $2^m = 2^{n/3}$ leaves. Consider a leaf with an associated partial assignment $\alpha$. One can prove that $\alpha$ either changes $\psi_i$ to the zero function for at least one index $i$ or fixes at least one variable in $\psi_i$ for every $i = 1, \ldots, m$. In the first case, the leaf is not visited by any satisfying assignment of $\psi$. In the second case, the leaf is visited by a set $M$ of satisfying assignments each of which is a combination of satisfying assignments of $\psi_i$ for each $i$. Moreover, all the elements of $M$ can be obtained by selecting for each $i$ at most 3 different satisfying assignments of $\psi_i$ consistent with $\alpha$ and considering all of the combinations of these assignments. It follows that $|M| \leq 3^m$ as required. □

4 PC Encoding of a PC-BDMC

In this section we describe a construction of a PC encoding of a function which is represented by a PC-BDMC. The construction combines propagation of the literals in the encodings in the leaves with propagation in the Tseitin encoding of the circuit part. The propagation of a literal or the contradiction in a leaf has to be distinguished from deriving the literal or the contradiction from the whole sentence. We use a variant of the well-known dual rail encoding to achieve this.

In Section 4.1, we introduce meta-variables used in the construction. In Section 4.2, we describe the dual rail encoding in the form which we use. In Section 4.3 we describe the construction of a PC encoding of a function $f(x)$ given by a PC-BDMC $D$ and in Section 4.4 we estimate its size.

4.1 Meta-variables

The dual rail encoding uses new variables representing the literals on the variables of the original encoding. In addition to this, we associate a special variable with the contradiction. These new variables will be called meta-variables and denoted as follows. The meta-variable associated with a literal $l$ will be denoted $[l]$, the meta-variable associated with $\bot$ will be denoted $[\bot]$, and the set of the meta-variables corresponding to a vector of variables $x$ will be denoted

$$\text{meta}(x) = \{[l] \mid l \in \text{lit}(x) \cup \{\bot\}\}.$$  

For notational convenience, we extend this notation also to sets of literals that are meant as a conjunction, especially to partial assignments. If $\alpha \subseteq \text{lit}(x)$ is a set of literals, then
[α] denotes the set of meta-variables associated with the literals in α, thus

\[ [\alpha] = \{[l] \mid l \in \alpha \}. \]

If \([\alpha]\) is used in a formula such as \(\psi \land [\alpha]\), we identify this set of literals with the conjunction of them, similarly to the interpretation of \(\alpha\) in \(\varphi \land \alpha\).

In order to construct a PC encoding from a PC-BDMC in Section 4.3, we first construct a definite Horn formula \(\psi(\text{meta}(x), z)\) representing derivations of the literals from \(\text{lit}(x)\) using the meta-variables. These derivations have to be done separately in each node of \(D\) and we use indices to distinguish the meta-variables attached to different nodes. For every \(i = 1, \ldots, N\) and every \(l \in \text{lit}(x)\) we denote \([l]_i\) the meta-variable associated with \(l\) in node \(v_i\). For every leaf \(v_i, i = 1, \ldots, L\) we moreover consider meta-variables \([l]_i\) associated with literals \(l \in \text{lit}(y_i)\). The set of auxiliary variables \(z\) used in \(\psi(\text{meta}(x), z)\) is as follows:

\[ z = \{[l]_i \mid 1 \leq i \leq L, l \in \text{lit}(x_i \cup y_i) \cup \{\bot\}\} \cup \{[l]_i \mid L < i \leq N, l \in \text{lit}(x_i) \cup \{\bot\}\}. \quad (5) \]

### 4.2 Dual rail encoding

The dual rail encoding [2, 4, 18, 20] transforms an encoding of a function into a Horn formula simulating the unit propagation in the original encoding.

**Definition 4.1 (Dual rail encoding)** Let \(\varphi(x)\) be an arbitrary CNF formula. If \(\varphi\) contains the empty clause, then \(\text{DR}(\varphi) = [\bot]\). Otherwise, the dual rail encoding \(\text{DR}(\varphi)\) is the definite Horn formula on the meta-variables \(\text{meta}(x)\) defined as follows.

\[
\text{DR}(\varphi) = \bigwedge_{C \in \varphi} \bigwedge_{l \in C} \left( \bigwedge_{e \in C \setminus \{l\}} \neg e \rightarrow [l] \right) \land \bigwedge_{x \in X} ([x] \land \neg [\neg x] \rightarrow [\bot]). \quad (6)
\]

The proof of the following lemma is omitted, since it is well-known, although different authors use different notation for the variables representing the literals. Moreover, the contradiction is frequently represented by an empty set and not by a specific literal. Dual rail encoding with an explicit representation of the contradiction is used in the first part of the proof of Theorem 1 in [2] for a similar purpose as in this paper.

**Lemma 4.2** Let \(\varphi(x)\) be a CNF not containing the empty clause and let \(\alpha \subseteq \text{lit}(x)\). Then for every \(l \in \text{lit}(x) \cup \{\bot\}\) we have

\[
\varphi \land \alpha \vdash l \iff \text{DR}(\varphi) \land [\alpha] \vdash [l]. \quad (7)
\]

We use the dual rail encoding of the PC encoding \(\varphi_i(x_i, y_i)\) associated with a leaf \(v_i\) of a PC-BDMC using the meta-variables specific to the node \(v_i\). Namely, we denote \(\text{DR}(v_i, \varphi_i(x_i, y_i))\) the dual rail encoding of formula \(\varphi_i(x_i, y_i)\) which uses meta-variables \([l]_i\) in place of \([l]\) for \(l \in \text{lit}(x_i \cup y_i) \cup \{\bot\}\).
Table 1: The clauses of Horn formula $\psi(\text{meta}(x), z)$. We use shortcuts $[l]_r = [\bot]_r$ for literals $l \in \text{lit}(x_i \setminus x_r)$ for every $r \in \{r_1, \ldots, r_k\}$ in the group $\{g8\}$.

### 4.3 Constructing the Encoding

As already mentioned, we first construct a definite Horn formula $\psi(\text{meta}(x), z)$ representing derivations of the literals using meta-variables. The list of clauses of this formula is in Table 1. Note that the clauses of group $\{g8\}$ use shortcuts $[l]_r = [\bot]_r$ for literals $l \in \text{lit}(x_i \setminus x_r)$ for every $r \in \{r_1, \ldots, r_k\}$.

We use $\psi(\text{meta}(x), z)$ to derive positive literals with unit propagation when presented with only positive literals in the assumption. Such a form of unit propagation in a Horn formula is also called *forward chaining* and we use this notion when we want to express that the unit propagation is used in this sense. For the proof of Theorem 4.5 below, we need the following lemma.

**Lemma 4.3** For every $1 \leq i \leq N$, every partial assignment $\alpha \subseteq \text{lit}(x_i)$, and $l \in \text{lit}(x_i) \cup \{\bot\}$, if

$$f_i(x) \land \alpha \models l$$

then

$$\psi(\text{meta}(x), z) \land [\alpha] \vdash [l]_i$$

**Proof.** Let $\alpha \subseteq \text{lit}(x)$ and $l \in \text{lit}(x_i) \cup \{\bot\}$. We shall proceed by induction on $i$. Let us first assume that $1 \leq i \leq L$, i.e. $v_i$ is a leaf. Since $\varphi_i(x_i, y_i)$ is a PC encoding of $f_i(x_i)$, (8) and (9) imply that $\varphi_i(x_i, y_i) \land \alpha \vdash l$ or $\varphi_i(x_i, y_i) \land \alpha \vdash \bot$. The clauses

| group | clause | condition |
|-------|--------|-----------|
| (g1)  | $[l] \rightarrow [l]_i$ | $l \in \text{lit}(x_i)$ |
| (g2)  | $C$ | $C \in \text{DR}(v_i, \varphi_i(x_i, y_i))$ |
| (g3)  | $[\bot]_i \rightarrow [l]_i$ | $l \in \text{lit}(x_i)$ |
| (g4)  | $[\bot]_r \rightarrow [\bot]_i$ | $r \in \{r_1, \ldots, r_k\}$ |
| (g5)  | $[l]_r \rightarrow [l]_i$ | $r \in \{r_1, \ldots, r_k\}$, $l \in \text{lit}(x_r)$ |
| (g6)  | $[\bot]_i \rightarrow [l]_i$ | $l \in \text{lit}(x_i)$ |
| (g7)  | $[\bot]_{r_1} \land \cdots \land [\bot]_{r_k} \rightarrow [\bot]_i$ |
| (g8)  | $[l]_{r_1} \land \cdots \land [l]_{r_k} \rightarrow [l]_i$ | $l \in \text{lit}(x_i)$ |
| (g9)  | $[l]_N \rightarrow [l]$ | $l \in \text{lit}(x) \cup \{\bot\}$ |
of group \([g2]\) form the encoding \(DR(v_i, \varphi_i(x_i,y_i))\). By Lemma 4.2, this encoding and clauses in groups \([g1]\) and \([g3]\) imply \(\psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash_{l} \llbracket l \rrbracket\).

Assume \(v_i = r_{v_1} \land \cdots \land r_{v_k}\) and that the implication from \([8]\) to \([9]\) holds with \(i\) replaced with any of the indices \(r_1, \ldots, r_k\). Since \(D\) is decomposable, \([8]\) implies that \(f_r(x) \land \alpha \models l\) for some \(r \in \{r_1, \ldots, r_k\}\). If \(f_r(x) \land \alpha \models \bot\), we get by induction hypothesis \(\psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash_{l} \llbracket l \rrbracket\), and using clauses of groups \([g4]\) and \([g6]\) we get \([9]\). If \(f_r(x) \land \alpha \models l\) for \(l \in \text{lit}(x_c)\), we have \(\psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash_{l} \llbracket l \rrbracket\), and using an appropriate clause of group \([g5]\) we get \([9]\).

Assume \(v_i = v_{r_1} \lor \cdots \lor v_{r_k}\) and that the implication from \([8]\) to \([9]\) holds with \(i\) replaced with any of the indices \(r_1, \ldots, r_k\). If \([8]\) holds for \(v_i\), then \(f_r(x) \land \alpha \models l\) for all \(r \in \{r_1, \ldots, r_k\}\). By induction hypothesis and using the shortcuts \(\llbracket l \rrbracket_r = \llbracket \bot \rrbracket_r\) in cases \(l \not\in \text{lit}(x)\), we get \(\psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash_{l} \llbracket l \rrbracket\), for every \(r \in \{r_1, \ldots, r_k\}\). Using clause \([g7]\) if \(l = \bot\) and a clause from group \([g8]\) if \(l \neq \bot\), we get \([9]\). \(\square\)

Consider new variables \(v\) corresponding to the nodes of \(D\) and the formula \(\theta(x, y, v)\) described by Table 2 combining the Tseitin encoding of \(D\) and the implications \(v_i \rightarrow \varphi_i(x_i, y_i)\). Let us verify that this is an encoding of \(f(x)\).

**Lemma 4.4** The set of clauses \(\theta(x, y, v)\) described in Table 2 is an encoding of \(f(x)\) with auxiliary variables \(y \cup v\).

**Proof.** Let \(a : x \rightarrow \{0, 1\}\) and let us show that \(\theta(a, y, v)\) is satisfiable if and only if \(f(a) = 1\). Assume \(f(a) = 1\). We describe assignments \(b\) and \(c\) of values to the variables in \(y\) and \(v\) respectively such that \(\theta(a, b, c)\) is satisfied. For \(1 \leq i \leq L\), construct an evaluation of \(y_i\) and \(v_i\) as follows. If \(\varphi_i(a_i, y_i)\) is satisfiable, let \(b_i\) be an assignment of values to variables in \(y_i\) which satisfies \(\varphi_i(a_i, b_i)\) and let \(c(v_i) = 1\). Otherwise let \(b_i\) be an arbitrary assignment of values to \(y_i\) and let \(c(v_i) = 0\). We then set \(b\) to be the union of the assignments \(b_1, \ldots, b_L\). For \(L < j \leq N\), the value \(c(v_j)\) is set according to the accepting computation of \(D\) starting from \(c(v_1)\), \(1 \leq i \leq L\). One can verify that all the clauses listed in Table 2 are satisfied by \(a\), \(b\), and \(c\).

Assume, \(\theta(a, y, v)\) is satisfiable and let us denote by \(b\) a satisfying assignment of the variables in \(y \cup v\). Let us denote by \(b_i\), \(i = 1, \ldots, L\), the restriction of \(b\) to the variables in \(y_i\). Let \(c\) be the assignment of the variables in \(v\) obtained as the values of the gates in \(D\) in a computation starting from the input values in the leaves given
by \( c(v_i) = b(v_i) \) for \( i = 1, \ldots, L \). The clauses listed in Table 2 contain the part of the Tseitin encoding of \( D \) that guarantees \( b(v_j) \leq b(v_{r_1}) \land \cdots \land b(v_{r_k}) \) for conjunction nodes and \( b(v_j) \leq b(v_{r_1}) \lor \cdots \lor b(v_{r_k}) \) for disjunction nodes. Moreover, we have \( c(v_j) = c(v_{r_1}) \land \cdots \land c(v_{r_k}) \) for conjunction nodes and \( c(v_j) = c(v_{r_1}) \lor \cdots \lor c(v_{r_k}) \) for disjunction nodes. Since the nodes \( v_j \) are numbered in a topological order, we can use this to prove \( b(v_j) \leq c(v_j) \) for all \( 1 \leq j \leq N \) by induction over \( j = 1, \ldots, N \). In particular, we obtain \( b(v_N) = c(v_N) = 1 \). Moreover, for every \( i = 1, \ldots, L \), if \( b(v_i) = 1 \), then \( b_i \) is a satisfying assignment of \( \varphi_i(a_i, y_i) \). If \( b(v_i) = 0 \), the unsatisfiability of \( \varphi_i(a_i, y_i) \) is not guaranteed, however, the definition of BDMC and the fact that \( D \) is a monotone circuit imply \( f(a) = 1 \). □

Moreover, forward chaining in \( \psi(\text{meta}(x), z) \) can be connected to deduction from \( \theta(x, y, v) \), if each of the variables of \( \psi(\text{meta}(x), z) \) is associated with a suitable clause on the variables \( x \cup y \cup v \). The meta-variable \( \llbracket l \rrbracket_i \) is associated with \( l \) itself. The variable \( \llbracket l \rrbracket_i \) is associated with the clause \( \neg v_i \lor l \). For notational convenience, we introduce additional meta-variables of the form \( \llbracket \neg v_i \lor l \rrbracket \), where \( i = 1, \ldots, N \) and \( l \in \text{lit}(x_i \cup y_i) \), where we assume \( y_i = \emptyset \), if \( i > L \). If we consider \( \llbracket \neg v_i \lor l \rrbracket \) as an alternative name for \( \llbracket l \rrbracket_i \), every clause of \( \psi(\text{meta}(x), z) \) has the form \( \llbracket C_1 \rrbracket \land \cdots \land \llbracket C_k \rrbracket \rightarrow \llbracket c \rrbracket \) for some clauses \( C_1, \ldots, C_k, C \) on the variables \( x \cup y \cup v \). Moreover, for each clause of \( \psi(\text{meta}(x), z) \), we have

\[
\theta(x, y, v) \land C_1 \land \cdots \land C_k \models C
\]

since unit propagation derives a contradiction from \( \neg C \), the clauses \( C_1, \ldots, C_k \), and at most one of the clauses of \( \theta \). For example, consider the clause of group (g8) corresponding to a node \( v_i = v_{r_1} \lor v_{r_2} \) and a given literal \( l \in \text{lit}(x_i) \), such that \( l \in \text{lit}(x_{r_1}) \setminus \text{lit}(x_{r_2}) \). In this case, \( \neg C = v_i \land \neg l, C_1 = \neg v_{r_1} \lor l, C_2 = \neg v_{r_2}, \) and the formula \( \theta(x, y, v) \) contains the clause \( \neg v_{r_1} \lor v_{r_1} \lor v_{r_2} \). One can verify that unit propagation derives the contradiction from these assumptions.

**Theorem 4.5** For every \( \alpha \subseteq \text{lit}(x) \) and \( l \in \text{lit}(x) \cup \{ \bot \} \), we have

\[
f(x) \land \alpha \models l \iff \psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash \llbracket l \rrbracket
\]

**Proof.** Assume \( f(x) \land \alpha \models l \). Since \( f(x) = f_N(x) \), we get by Lemma 4.3 that \( \psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash \llbracket l \rrbracket_N \). Using an appropriate clause from group (g9f) we get \( \psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash \llbracket l \rrbracket \).

Assume \( \psi(\text{meta}(x), z) \land \llbracket \alpha \rrbracket \vdash \llbracket l \rrbracket \). Each step of this derivation uses a clause of \( \psi(\text{meta}(x), z) \). Using (11) for each of these clauses, we obtain by induction on the number of steps of the derivation that \( \theta(x, y, v) \land \alpha \models l \). Since \( \theta(x, y, v) \) is an encoding of \( f(x) \), we obtain \( f(x) \land \alpha \models l \) which finishes the proof of (11). □

Given a Horn formula \( \psi(\text{meta}(x), z) \) satisfying the equivalence (11), we can form a PC encoding of \( f(x) \) by simply substituting meta-variables in meta\((x)\) with the respective literals or \( \bot \) based on the following proposition. 

12
Theorem 4.6 Let $\varphi(x, z)$ be obtained from $\psi$ by substituting meta-variable $[l]$ for all $l \in \text{lit}(x) \cup \{\bot\}$. Then $\varphi(x, z)$ is a PC encoding of $f(x)$.

Proof. By Theorem 4.5, $\psi$ satisfies the equivalence (11). First, assume a full assignment $a : x \rightarrow \{0, 1\}$, such that $f(a) = 1$, and let us prove that the formula $\varphi(a, z)$ is satisfiable. Let $\alpha$ be the set of literals on variables from $x$ satisfied by $a$. Since $f(x) \wedge \alpha \neq \bot$, we have by (11) that

$$\psi(\alpha) \wedge [\alpha] \not\models [l].$$

It follows that $\psi(\alpha) \wedge [\alpha] \not\models [-l]$ for any $l \in \alpha$. Indeed, assume $\psi(\alpha) \wedge [\alpha] \models [-l]$ for some $l \in \alpha$. By (11) we get $f(x) \wedge \alpha \models \neg l$. Since $l \in \alpha$, clearly $f(x) \wedge \alpha \models l$ and thus together we have $f(x) \wedge \alpha \models \bot$ which is a contradiction.

Consider the assignment $b$ of values to variables in $\text{meta}(x) \cup z$ obtained by setting to 1 all the variables derived by forward chaining in the formula $\psi(\alpha)$ and setting to 0 all the remaining variables. Clearly, we have $b([\bot]) = 0$ and for every $l \in \text{lit}(x)$, we have $b([l]) = a(l)$. Let $b'$ be the assignment of the variables $x \cup z$ defined as $b'(z) = b(z)$ for all $z \in z$ and $b'(x) = b([x]) = a(x)$ for all $x \in x$. We prove that $b'$ satisfies each clause $C' \in \varphi$ using the fact that $b$ satisfies the corresponding clause $C \in \psi$. Let us consider the following cases.

- Assume that $C$ is satisfied by a literal $l \in \text{lit}(z)$, i.e. $l \in C$ and $b(l) = 1$. Since $l$ is unchanged by the substitution, we have $l \in C'$. Since $b'(l) = b(l)$, $C'$ is satisfied by $b'$.

- If $C$ is satisfied by the literal $\neg[l]$, the clause is removed from $\varphi$ by the substitution, so $C$ does not correspond to any clause $C' \in \varphi$.

- If $C$ is satisfied by a literal $[l]$, where $l \in \text{lit}(x)$, then $l \in \alpha$ and $C'$ contains $l$ satisfied by $b'$.

- If $C$ is satisfied by a literal $\neg[l]$, where $l \in \text{lit}(x)$, then $l \notin \alpha$ and $C'$ contains $\neg l$ satisfied by $b'$.

It follows that $\varphi(a, z)$ is satisfiable. In order to prove that $\varphi(x, z)$ is an encoding of $f(x)$, it remains to prove that it is unsatisfiable, if $f(a) = 0$. This is a consequence of propagation completeness proven below, if we consider $l = \bot$.

Let $\alpha \subseteq \text{lit}(x)$ and $l \in \text{lit}(x) \cup \{\bot\}$, such that $f(x) \wedge \alpha \models l$. By (11) we have

$$\psi(\alpha) \wedge [\alpha] \not\models [l] \quad \text{(12)}$$

We prove that either

$$\varphi(x, z) \wedge \alpha \not\models l \quad \text{(13)}$$

or

$$\varphi(x, z) \wedge \alpha \not\models \bot \quad \text{(14)}$$

\[13\]
by the following argument. Let us fix a minimal forward chaining derivation of either \([\bot]\) or \([\bot]\) from \(\psi(\text{meta}(x), z) \land [\alpha]\). Let \(g_i\), \(i = 1, \ldots, m\), be the sequence of positive literals on the meta-variables \(\text{meta}(x) \cup z\) in the order given by the fixed derivation. In particular \(g_m \in \{[\bot], [\bot]\}\) and \(g_i \notin \{[\bot], [\bot]\}\) for \(i = 1, \ldots, m - 1\). Let \(g'_i\), \(i = 1, \ldots, m\), be obtained from \(g_i\) by the substitution given by the assumption, i.e. if \(g_i = [e]\) for some \(e \in \text{lit}(x) \cup \{\bot\}\), then \(g'_i = e\), otherwise \(g'_i = g_i\). In particular \(g'_m\) is either \(l\) or \(\bot\). Let us prove by induction over \(i = 1, \ldots, m\) that either for all \(i = 1, \ldots, m\)

\[\varphi(x, z) \land \alpha \vdash_{1} g'_i\]  \hspace{2cm} (15)

implying (13), or we obtain (14) without completing all the induction steps.

Let \(i \in \{1, \ldots, m\}\) and assume that (15) holds with \(i\) replaced by any \(j < i\). Let \(C\) be the Horn clause of \(\psi(\text{meta}(x), z) \land [\alpha]\) used to derive \(g_i\). Note that by the choice of the derivation, \(C\) does not contain the literal \(\neg[\bot]\), so we do not obtain \(\top\) in \(C\) by the substitution from the assumption. Let \(C'\) be the set of the literals obtained from \(C\) by the substitution and possibly removing \(\bot\), if it is obtained. If \(C'\) contains complementary literals \(e\) and \(\neg e\), then one can verify that \(e \in \text{lit}(x)\), \(C\) contains negative literals \(\neg[e]\) and \(\neg[\neg e]\), and both the literals \([e]\) and \([\neg e]\) occur in the sequence \(g_j\), \(j = 1, \ldots, i-1\). In this case, the corresponding literals \(g'_j\) are \(e\) and \(\neg e\) and using (15) for them, we obtain (14) by one additional unit propagation step. If \(C'\) does not contain complementary literals, it is a clause of \(\varphi(x, z) \land \alpha\). The clause \(C\) is a definite Horn clause with the target \(g_i\). If \(C\) contains negative literals on meta-variables, they are negations of some literals \(g_j\) with indices \(j < i\). Consequently, the corresponding literals in \(C'\) are negations of literals \(g'_j\) with the same indices \(j < i\). By induction hypothesis, the literals \(g'_j\) can be derived from the formula \(\varphi(x, z) \land \alpha\) before \(g'_j\). Hence, unit propagation using \(C'\) derives \(g'_j\), where \(g'_j\) is either a literal included in \(C'\) or \(g'_j = g'_m = \bot\). Altogether, we obtain (15) or (14) implying (13) or (14). It follows that \(\varphi(x, z)\) is a PC encoding of \(f(x)\). □

4.4 Size Estimate

The main result of this section is contained in the following theorem.

**Theorem 4.7** Let \(D\) be a PC-BDMC sentence representing function \(f(x)\) of variables \(x = (x_1, \ldots, x_n)\). Assume that \(D\) has \(N\) nodes \(V = \{v_1, \ldots, v_N\}\) with leaves \(v_1, \ldots, v_L\) and \(E\) edges. Let us denote \(\varphi_i(x_i, y_i)\) the PC encoding of function \(f_i(x_i)\) associated with a leaf \(v_i, i = 1, \ldots, L\). Let us further denote \(m = \sum_{i=1}^{L} |y_i|\) the total number of auxiliary variables, \(S = \sum_{i=1}^{L} ||\varphi_i||\) the total length of all PC encodings associated with the leaves of \(D\), and \(\ell\) the maximum length of a clause in any of the encodings associated with the leaves of \(D\). Then \(f\) has PC encodings \(\varphi(x, z)\) and \(\varphi'(x, z')\) satisfying

\[
|z| = O(m + nN), \quad ||\varphi|| = O(S + nE), \quad ||\varphi'|| = O(\ell S + nE), \quad (16a-c)
\]

\[
|z'| = O(S + nN), \quad ||\varphi'|| = O(S + nE), \quad ||\varphi'|| = O(S + nE). \quad (17a-c)
\]

**Proof.** If \(D\) consists of a single node \(v_1\), then \(v_1\) is the root and the only leaf of \(D\). In this case \(\varphi(x, z) = \varphi'(x, z') = \varphi_1(x_1, y_1)\) is a PC encoding of \(f(x) = f_1(x_1)\). The size
and length of this encoding are both upper bounded by $S$ and the number of auxiliary variables is $m \leq S$, thus we get (16a-c) and (17a-c).

Assume, $D$ contains more than one node, in particular $N \leq E + 1 = O(E)$. Consider the formula $\psi(\text{meta}(x), z)$ described by Table 1. By Theorem 4.5 it satisfies (11) and by Theorem 4.6 we obtain a PC encoding $\varphi(x, z)$ of $f(x)$ by substituting meta-variables $\text{meta}(x)$ with the literals on the variables $x$ and ⊥. By (5), the number of the auxiliary variables $z$ satisfies (16a). The number of clauses in $\varphi$ is bounded by the number of clauses in $\psi$. By counting the clauses in Table 1 and considering $N = O(E)$, we get (16b).

Let us now estimate the length $\|\varphi\|$ of the constructed encoding. By definition of dual rail encoding, the total length of clauses in group (g2) is upper bounded by $\ell S + 3 \sum_{i=1}^{E} |x_i \cup y_i| \leq (\ell + 3)S = O(\ell S)$. The clauses in groups (g1) to (g6) and (g9) consist of two literals, so their total length is twice their number which is $O(nE)$. The length of a clause in group (g7) or (g8) for a node $v_i = v_{r_1} \lor \cdots \lor v_{r_k}$ is $k + 1$. If we sum $k + 1$ over all $\lor$-nodes in $D$, we get that the total length of clauses in groups (g7) and (g8) is $O(n(E + N)) = O(nE)$ thus showing (16c).

In order to obtain encoding $\varphi'(x, z')$ it is enough to first use standard transformation of PC encodings associated with leaves of $D$ to 3-CNF formulas. It is not hard to observe that this step preserves propagation completeness and it limits the length of the clauses in the encodings to at most 3. This way we get that the total number of auxiliary variables used in the PC encodings in leaves is $O(S)$ and the total length of these encodings is $O(S)$ as well. If we use (16a-c) with $m$ replaced with $O(S)$ and with $\ell = 3$, we obtain (17a-c). Note that the length of the formula $\varphi'$ is smaller by factor $\ell$ at the price of having more auxiliary variables in $z'$ than in $z$. □

5 URC encoding of a URC-BDMC

In this section we describe the construction of URC encoding from a URC-BDMC by pointing out the differences from the construction in Section 4.3. The resulting construction generalizes the disjunction closure of URC encodings from [4], where URC encodings are denoted by $\exists \text{URC-C}$.

If we start with a URC-BDMC sentence $D$ representing function $f(x)$, the construction in Section 4.3 can be simplified to obtain a URC encoding of $f$ as follows. We do not have to propagate literals on the input variables in the Tseitin encoding of $D$, it is sufficient to propagate the contradiction. It follows that for each inner node $v_i$ of $D$ we only need the meta-variable $[\bot]_i$. The set of clauses forming a URC encoding is a subset of the set of clauses forming a PC encoding and is presented in Table 3. For simplicity of the notation, we denote this formula and its variables in the same way as the corresponding formula used in the construction of PC encoding in Section 4. If not stated otherwise, in this section, the formula $\psi(\text{meta}(x), z)$ refers to Table 3 and the auxiliary variables $z$ are defined as follows.

$$z = \{[\bot]_i | 1 \leq i \leq L, l \in \text{lit}(x_i \cup y_i) \cup \{\bot\}\}$$

$$\cup \{[\bot]_i | L < i \leq N\}$$  (18)
Clauses for a leaf node $v_i$, $1 \leq i \leq L$

| Group | Clause | Condition |
|-------|--------|-----------|
| (g1') | $[l] \rightarrow [l]_i$ | $l \in \text{lit}(x_i)$ |
| (g2') | $C$ | $C \in \text{DR}(v_i, \varphi_i(x_i, y_i))$ |

Clauses for node $v_j = v_{r_1} \land \cdots \land v_{r_k}$

| Group | Clause | Condition |
|-------|--------|-----------|
| (g4') | $[\bot]_{r} \rightarrow [\bot]_j$ | $r \in \{r_1, \ldots, r_k\}$ |

Clauses for node $v_j = v_{r_1} \lor \cdots \lor v_{r_k}$

| Group | Clause | Condition |
|-------|--------|-----------|
| (g7') | $[\bot]_{r_1} \land \cdots \land [\bot]_{r_k} \rightarrow [\bot]_j$ | |

Additional clauses for the root node $v_N$

| Group | Clause | Condition |
|-------|--------|-----------|
| (g9') | $[\bot]_N \rightarrow [\bot]$ | |

Table 3: The set of clauses of the formula $\psi(\text{meta}(x), z)$ for the construction of URC encoding.

In order to simplify the comparison of the construction of PC and URC encodings, the numbering of the groups of clauses in Table 3 follows the numbering in Table 1 except of the additional prime in the labels. For URC encoding, we do not need clauses of group (g3) because we do not need to propagate literals. For the same reason we do not need clauses in groups (g5), (g6), and (g8). On the other hand, clauses in groups (g4) and (g7) are used to propagate the contradiction in gates of $D$ and they are present in the construction of a URC encoding. In group (g9), we use only the clause with $l = \bot$.

As in Theorem 4.6 we can turn $\psi(\text{meta}(x), z)$ to a URC encoding by substituting meta-variables in $\text{meta}(x)$ with the respective literals or $\bot$. The obtained URC encoding is smaller than a PC encoding described in Theorem 4.7. Following the same notation we get the following bounds.

**Theorem 5.1** Let $D$ be a URC-BDMC sentence representing function $f(x)$ of variables $x = (x_1, \ldots, x_n)$. Assume that $D$ has $N$ nodes $V = \{v_1, \ldots, v_N\}$ with leaves $v_1, \ldots, v_L$ and $E$ edges. Let us denote $\varphi_i(x_i, y_i)$ the URC encoding of function $f_i(x_i)$ associated with a leaf $v_i$, $i = 1, \ldots, L$. Let us further denote $m = \sum_{i=1}^{L} |y_i|$ the total number of auxiliary variables and $S = \sum_{i=1}^{L} \|\varphi_i\|$ the total length of all URC encodings associated with leaves of $D$, and $\ell$ the maximum length of a clause in any of the encodings associated with the leaves of $D$. Then $f$ has URC encodings $\varphi(x, z)$ and $\varphi'(x, z')$ satisfying

$$
|z| = O(n + m + N), \quad |\varphi| = O(S + E), \quad \|\varphi\| = O(\ell S + E), \quad (19a-c)
|z'| = O(S + N), \quad |\varphi'| = O(S + E), \quad \|\varphi'\| = O(S + E). \quad (20a-c)
$$

**Proof.** If $D$ consists of a single node, we proceed as in the proof of Theorem 4.7. In the rest of the proof, we assume $2 \leq N \leq E + 1 = O(E)$. Consider the formula
ψ(meta(x), z) described by Table 5. Analogously to Theorem 4.5 it satisfies (11) with \( l = \bot \) and similarly to Theorem 4.6 we obtain a URC encoding \( \varphi(x, z) \) of \( f(x) \) by substituting meta-variables meta(x) with the literals on the variables x and \( \bot \). By (18), the number of the auxiliary variables z satisfies (19a). The number of clauses in \( \varphi \) is bounded by the number of clauses in \( \psi \). By counting the clauses in Table 5 we get (19b).

Let us now estimate the length \( \| \varphi \| \) of the constructed encoding. By definition of dual rail encoding, the total length of clauses in group (g2') is upper bounded by \( \ell S + 3 \sum_{i=1}^{L} |x_i \cup y_i| \leq (\ell + 3)S = O(\ell S) \). The clauses in groups (g1'), (g4'), and (g9') consist of two literals, so their total length is twice their number which is \( O(E) \). The length of a clause in group (g7') for a node \( v_j = v_{r_1} \lor \cdots \lor v_{r_k} \) is \( k + 1 \). If we sum \( k + 1 \) over all \( \lor \)-nodes in \( D \), we get that the total length of clauses in group (g7') is \( O(E + N) = O(E) \) thus showing (19c).

In order to obtain encoding \( \varphi'(x, z') \) we can proceed in the same way as in the proof of Theorem 4.7 by using standard transformation of URC encodings associated with leaves of \( D \) to 3-CNF formulas. It is not hard to observe that this step preserves unit refutation completeness and it limits the length of the clauses in the encodings to at most 3. This way we get that the total number of auxiliary variables used in the URC encodings in leaves is \( O(S) \) and the total length of these encodings is \( O(S) \) as well. If we use (19a-c) with \( m \) replaced with \( O(S) \) and with \( \ell = 3 \), we obtain (20a-c). Note that the bound on the length of the formula \( \varphi' \) is smaller by factor \( \ell \) than the bound for the formula \( \varphi \) at the price of having more auxiliary variables in \( z' \) than in \( z \). □

6 Parameterized Size of a PC Encoding

As already mentioned in the introduction, the known results imply that the size of a PC encoding of a function \( f(x) \) given by a CNF \( \varphi \) can be parameterized with the primal or incidence treewidth of \( \varphi \). This means that given a CNF \( \varphi \), the size of a smallest PC encoding of \( f(x) \) can be upper bounded by \( g(k) \cdot p(\| \varphi \|) \) for some polynomial \( p \) and a computable function \( g(k) \) which depends only on the parameter \( k \) (the treewidth of primal or incidence graph associated with \( \varphi \) in our case). Moreover, if a tree decomposition of a suitable type is known, the PC encoding can be computed efficiently. For more background on fixed parameter tractability, see e.g. [21].

The above bounds on the size of a PC encoding can be obtained by compiling \( \varphi \) into a smooth DNNF and using the construction described in [15]. Parameterized algorithms for constructing a DNNF for a given CNF are described in literature for different types of parameters. A construction parameterized by primal treewidth of \( \varphi \) was described in [12], a construction parameterized by dual treewidth was described in [24], a construction parameterized by incidence treewidth was considered in [23] where the authors mainly considered a parameter called CV-width which dominates the incidence treewidth. Finally, the construction parameterized by cliquewidth of \( \varphi \) was described in [25].

Using Theorem 4.7 we can extend the list of possible parameterizations by the size of a smallest \( C \)-backdoor tree for a class \( C \) which consists of PC or URC formulas or
encodings. This can be seen as follows.

As explained in Section 3, a $C$-backdoor tree for a class $C$ which consists of PC formulas or encodings (such as 2-CNF) is a special case of PC-BDMC and using Theorem 4.7 we obtain the required PC encoding. If we start from URC backdoor tree, we include one more step. It follows by Theorem 1 in [2] that given a URC encoding $\varphi(x, y)$ of a boolean function $f(x)$, we can efficiently construct a PC encoding $\psi(x, z)$ of $f(x)$.

**Lemma 6.1 ([2])** Let $\varphi(x, y)$ be a URC encoding of function $f(x)$ with input variables $x = (x_1, \ldots, x_n)$ and auxiliary variables $y = (y_1, \ldots, y_\ell)$. Then one can construct efficiently a PC encoding $\psi(x, z)$ of $f(x)$ with $|\psi| = O(n\|\varphi\|)$ and $|z| = O(n(n + \ell))$.

**Proof.** (sketch) We can construct $\psi$ using the failed literal rule as follows. For each literal $l \in \text{lit}(x)$ we construct a dual rail encoding of $\varphi \land l$ in addition to a dual rail encoding of $\varphi$. That is $2^n + 1$ copies of $\varphi$, each with $O(\|\varphi\|)$ clauses and $n + \ell$ auxiliary variables. Then we add clauses passing the values of input variables, and clauses getting the results. The number of these additional clauses is smaller than the main part of $\psi$ consisting of $2n + 1$ copies of $\varphi$. □

Lemma 6.1 allows us to turn a URC-BDMC to a PC-BDMC efficiently. Together with constructions described in sections 4 and 5 we can show the following bounds on the sizes of PC and URC encodings parameterized by the size of a backdoor tree with respect to a suitable base class.

**Theorem 6.2** Let $f(x)$ be a function represented with a CNF $\varphi$ on variables $x = (x_1, \ldots, x_n)$. Let $b_q$ be the size of a minimum backdoor tree of $\varphi$ with respect to 2-CNF formulas, $b_h$ the size of a minimum backdoor tree of $\varphi$ with respect to Horn formulas, $b_{rh}$ the size of a minimum backdoor tree of $\varphi$ with respect to renamable Horn formulas.

(i) There is a PC encoding $\psi(x, y)$ of size $|\psi| = O(2^{b_q} \cdot \|\varphi\|)$ and with $|y| = O(2^{b_q} n)$ auxiliary variables.

(ii) There is a PC encoding $\psi(x, y)$ of size $|\psi| = O(2^b \cdot n \cdot \|\varphi\|)$ and with $|y| = O(2^b n^2)$ auxiliary variables where $b = \min\{b_h, b_{rh}\}$.

(iii) There is a URC encoding $\psi(x, y)$ of size $|\psi| = O(2^b \cdot \|\varphi\|)$ and with $|y| = O(2^b + n)$ auxiliary variables where $b = \min\{b_q, b_h, b_{rh}\}$.

**Proof.** If $b$ denotes the size of a backdoor tree $T$, then the number of leaves in $T$ is at most $2^b$ and the number of edges is at most $2^b$ as well. Thus if we consider $T$ as a BDMC sentence with formulas $\varphi_i$, $i = 1, \ldots, L$ in leaves, we get that $S = \sum_{i=1}^L \|\varphi_i\| \leq 2^b \|\varphi\|$.

The first proposition follows by Theorem 4.7 from the fact that a prime 2-CNF is always propagation complete (see e.g. [1]) and assuming $n \leq \|\varphi\|$. Using the fact that Horn and renamable Horn formulas are URC we get by Theorem 4.7 and Lemma 6.1 the second proposition. The third proposition follows from Theorem 5.1. □
Acknowledgements. Both authors gratefully acknowledge the support by Grant Agency of the Czech Republic (grant No. GA19–19463S). Petr Kučera was partially supported by Czech-French Mobility program (Czech Ministry of Education, grant No. 7AMB17FR027). The authors would like to thank Pierre Marquis, Jean-Marie Lagniez, Gilles Audemard, and Stefan Mengel (from CRIL, U. Artois, Lens, France) for discussion on the compilation of CNF into more general structures than DNNF.

References

[1] Martin Babka, Tomáš Balyo, Ondřej Čepek, Štefan Gurský, Petr Kučera, and Václav Vlček. Complexity issues related to propagation completeness. *Artificial Intelligence*, 203(0):19 – 34, 2013.

[2] Christian Bessiere, George Katsirelos, Nina Narodytska, and Toby Walsh. Circuit complexity and decompositions of global constraints. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 412–418, 2009.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, 2009.

[4] Lucas Bordeaux, Mikoláš Janota, Joao Marques-Silva, and Pierre Marquis. On unit-refutation complete formulae with existentially quantified variables. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR’12, pages 75–84. AAAI Press, 2012.

[5] Lucas Bordeaux and Joao Marques-Silva. Knowledge compilation with empowerment. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 612–624. Springer Berlin / Heidelberg, 2012.

[6] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovský. Expander CNFs have exponential DNNF size. *CoRR*, abs/1411.1995, 2014.

[7] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovský. On compiling CNFs into structured deterministic DNNFs. In Marijn Heule and Sean Weaver, editors, *Theory and Applications of Satisfiability Testing – SAT 2015*, pages 199–214, Cham, 2015. Springer International Publishing.

[8] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovský. Knowledge compilation meets communication complexity. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI’16, pages 1008–1014. AAAI Press, 2016.
[9] Ondřej Čepek, Petr Kučera, and Václav Vlček. Properties of slur formulae. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *LNCS*, pages 177–189. Springer Berlin Heidelberg, 2012.

[10] Y. Crama and P.L. Hammer. *Boolean Functions: Theory, Algorithms, and Applications*. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, 2011.

[11] Adnan Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI’99, pages 284–289, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[12] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, July 2001.

[13] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[14] Alvaro del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Knowledge Representation and Reasoning*, pages 551–561, 1994.

[15] Graeme Gange and Peter J. Stuckey. Explaining propagators for s-dnnf circuits. In Nicolas Beldiceanu, Narendra Jussien, and Éric Pinson, editors, *Integration of AI and OR Techniques in Constrain Programming for Combinatorial Optimization Problems*, pages 195–210. Springer Berlin Heidelberg, 2012.

[16] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Živný. Backdoors into heterogeneous classes of SAT and CSP. *Journal of Computer and System Sciences*, 85:38 – 56, 2017.

[17] Matthew Gwynne and Oliver Kullmann. Generalising and unifying slur and unit-refutation completeness. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, pages 220–232, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[18] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. On tackling the limits of resolution in sat solving. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 164–183, Cham, 2017. Springer International Publishing.

[19] Jean C. Jung, Pedro Barahona, George Katsirelos, and Toby Walsh. Two Encodings of DNNF Theories. In *ECAI’08 Workshop on Inference methods based on Graphical Structures of Knowledge*, 2008.
[20] V. M. Manquinho, P. F. Flores, J. P. M. Silva, and A. L. Oliveira. Prime implicant computation using satisfiability algorithms. In Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence, pages 232–239, Nov 1997.

[21] Rolf Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.

[22] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to horn and binary clauses. SAT, 4:96–103, 2004.

[23] Umut Oztok and Adnan Darwiche. CV-width: A new complexity parameter for cnfs. In Proceedings of the Twenty-first European Conference on Artificial Intelligence, ECAI’14, pages 675–680, Amsterdam, The Netherlands, The Netherlands, 2014. IOS Press.

[24] Knot Pipatsrisawat and Adnan Darwiche. Top-down algorithms for constructing structured DNNF: Theoretical and practical implications. In Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pages 3–8, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[25] Igor Razgon and Justyna Petke. Cliquewidth and knowledge compilation. In Matti Järvisalo and Allen Van Gelder, editors, Theory and Applications of Satisfiability Testing – SAT 2013, pages 335–350, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[26] Marko Samer and Stefan Szeider. Backdoor trees. In Dieter Fox and Carla P. Gomes, editors, Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, pages 363–368. AAAI Press, 2008.

[27] Richard Williams, Carla Gomes, and Bart Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search (extended abstract). In Proceedings of SAT 2003, 2003.

[28] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In Georg Gottlob and Toby Walsh, editors, IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003, pages 1173–1178. Morgan Kaufmann, 2003.