Journal of Artificial Intelligence Research 69 (2020) 1395-1420 Submitted 03/2020; published 12/2020

Bounds on the size of PC and URC formulas

Petr Kučera KUCERAP@KTIML.MFF.CUNI.CZ
Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Petr Savický SAVICKY@CS.CAS.CZ
Institute of Computer Science of the Czech Academy of Sciences
Pod Vodárenskou Věží 2, 182 07 Praha 8, Czech Republic

Abstract

In this paper, we investigate CNF encodings, for which unit propagation is strong enough to derive a contradiction if the encoding is not consistent with a partial assignment of the variables (unit refutation complete or URC encoding) or additionally to derive all implied literals if the encoding is consistent with the partial assignment (propagation complete or PC encoding). We prove an exponential separation between the sizes of PC and URC encodings without auxiliary variables and strengthen the known results on their relationship to the PC and URC encodings that can use auxiliary variables. Besides of this, we prove that the sizes of any two irredundant PC formulas representing the same function differ at most by a factor polynomial in the number of the variables and present an example of a function demonstrating that a similar statement is not true for URC formulas. One of the separations above implies that a q-Horn formula may require an exponential number of additional clauses to become a URC formula. On the other hand, for every q-Horn formula, we present a polynomial size URC encoding of the same function using auxiliary variables. This encoding is not q-Horn in general.

1. Introduction

Since unit propagation is a basic procedure used in DPLL based SAT solvers including CDCL solvers, it has become a common practice to require that unit propagation maintains at least some level of local consistency in the constraints being encoded into a conjunctive normal form (CNF) formula. Close connection between unit propagation in SAT solvers and maintaining generalized arc consistency was investigated for example by Bacchus (2007). CNF encodings do not support efficient consistency testing in general. In this paper, we investigate CNF encodings with a high level of propagation strength — they are unit refutation complete or propagation complete formulas. Both these classes of formulas were introduced as target languages of knowledge compilation. The class of unit refutation complete (URC) formulas was introduced by del Val (1994) and it consists of formulas whose consistency with any given partial assignment can be tested by unit propagation. Independently, Schlipf et al. (1995) introduced the class of SLUR (single lookahead unit resolution) formulas which was later shown to coincide with the class of unit refutation complete formulas by Gwynne and Kullmann (2013). The class of propagation complete formulas (PC) was introduced by Bordeaux and Marques-Silva (2012). PC formulas are URC and, moreover, unit propagation derives all implied literals provided the formula is consistent with a given partial assignment of its variables. The notions of empowerment

and absorption which serve as a basis for the definition of propagation complete formulas were introduced earlier when studying learning in CDCL solvers (Pipatsrisawat & Darwiche, 2011; Atserias et al., 2011).

A CNF encoding of a boolean function $f(\mathbf{x})$ is a CNF formula $\varphi(\mathbf{x}, \mathbf{y})$, such that $f(\mathbf{x}) = (\exists \mathbf{y})\varphi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is the set of the main variables and $\mathbf{y}$ is a (possibly empty) set of auxiliary variables. Note that an encoding of $f(\mathbf{x})$ is typically a formula representing a function different from f, since it depends also on $\mathbf{y}$. On the other hand, a CNF formula representing $f(\mathbf{x})$ is an encoding of f without auxiliary variables. CNF encodings with auxiliary variables are strictly more succinct in the sense of knowledge compilation map (Darwiche & Marquis, 2002) than encodings without auxiliary variables. For example, Tseitin encoding of a boolean circuit is a CNF encoding of the function represented by the circuit.

Every boolean function has a URC and a PC encoding. In particular, a canonical CNF on the main variables which consists of all prime implicates is always PC and, hence, URC. On the other hand, it is co-NP complete to check if a formula is URC (Čepek et al., 2012) or PC (Babka et al., 2013).

PC encodings were already used in knowledge compilation. In particular, Abío et al. (2016) describe a construction of a PC encoding of a function represented with an ordered decision diagram. Bordeaux et al. (2012) considered URC encodings as the existential closure of URC formulas. They included URC encodings and their disjunctions into the knowledge compilation map which was introduced by Darwiche and Marquis (2002) with the goal of a systematic study of target languages for knowledge compilation.

Let us note that when encoding constraints into CNF formulas a weaker concept than propagation completeness, namely generalized arc consistency (GAC) also called domain consistency is often considered (see e.g., Bacchus, 2007; Bessiere, Katsirelos, Narodytska, & Walsh, 2009; Abío et al., 2016, and Section 3 for further discussion on related work). PC or URC encodings treat all variables in the same way: the propagation properties of PC encoding with respect to the auxiliary variables are the same as with respect to the main variables. In the case of GAC encodings, the propagation properties are required only for the main variables.

In this paper, we prove an exponential separation between the size of URC and PC encodings without auxiliary variables and compare their succinctness with URC and PC encodings that can use auxiliary variables. In particular, we prove new exponential lower bounds for PC and URC encodings without auxiliary variables for functions that have a polynomial size PC encoding with auxiliary variables. Our lower bounds are stronger than previously known lower bounds of this type in that they apply to functions defined by polynomial size CNF formulas which, moreover, belong to well-known tractable classes of Horn and q-Horn formulas (introduced by Boros et al., 1990).

We also show that the sizes of irredundant PC formulas representing the same function are polynomially related which, as we show, is not the case of irredundant URC formulas. This result suggests that irredundancy is a helpful criterion for the search of small PC formulas, however, it can be completely misleading for URC ones, although the minimum size of a URC formula is always at most the minimum size of a PC formula.

Section 2 summarizes known notions used in this paper. Section 3 contains a discussion on related work. A complete summary of the results is presented in Section 4. The proofs of

the results are postponed to the remaining sections. In Section 5, we prove the exponential separations for PC and URC encodings described above. In Section 6, we prove the results on irredundant PC and URC formulas. In Section 7, we use a variant of the dual rail encoding (which was already used many times in literature, see e.g., Bessiere et al., 2009; Bonet et al., 2018; Bordeaux et al., 2012; Bryant et al., 1987; Ignatiev et al., 2017; Manquinho et al., 1997; Morgado et al., 2019) in order to interpret a PC formula as a representation of a specific Horn function. In Section 8, we present a construction of a URC encoding for a general q-Horn formula of size polynomial in the size of the input formula. The paper is closed by formulating several directions for further research in Section 9.

2. Basic Concepts

In this section, we recall the known concepts we use and introduce the necessary notation.

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 $\mathbf{x}$, $\mathrm{lit}(\mathbf{x})$ denotes the set of literals on variables in $\mathbf{x}$. We treat a clause as a set of literals on distinct variables 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 φ. We denote $\|\varphi\| = \sum_{C\in\varphi} |C|$ the *length* of a CNF formula φ. For a clause C, let $\mathrm{var}(C)$ denote the set of variables used in C.

We say that a clause C is an *implicate* of a CNF φ if every model of φ is also a model of C, i.e. $\varphi \models C$. We say that C is a *prime implicate* if there is no proper subclause C' of C which is an implicate of φ. CNF φ is *prime* if it consists only of prime implicates of φ.

A clause C is *Horn* if it contains at most one positive literal and it is *definite Horn*, if it contains exactly one positive literal. A definite Horn clause $\neg x_1 \vee \cdots \vee \neg x_k \vee y$ represents the implication $x_1 \wedge \cdots \wedge x_k \to y$ and we use both kinds of notation interchangeably.

We treat a *partial assignment* α of values to variables in $\mathbf{z}$ as a subset of $\mathrm{lit}(\mathbf{z})$ that does not contain a complementary pair of literals, so we have $|\alpha \cap \mathrm{lit}(x)| \leq 1$ for each $x \in \mathbf{z}$. By $\varphi(\alpha)$ we denote the formula obtained from φ by applying the partial setting of the variables defined by α. In particular, the clauses satisfied by α are removed from φ and negations of literals in α are removed from the remaining clauses. A set of literals α (in particular a partial assignment) used in a formula such as $\varphi(\mathbf{x}) \wedge \alpha$ is interpreted as the conjunction of the literals in α.

2.1 URC and PC Formulas

We are interested in formulas which have good properties with respect to unit propagation which is a well-known procedure in SAT solving (Biere, Heule, van Maaren, & Walsh, 2009). For technical reasons, we represent unit propagation using unit resolution. The *unit resolution* rule allows to derive the clause $C \setminus \{l\}$ given a clause C containing l and a unit clause $\neg l$. A clause C can be derived from φ by *unit resolution*, if C is contained in φ or can be derived from φ by a series of applications of the unit resolution rule. We denote this fact with $\varphi \vdash_1 C$. The notion of a propagation complete CNF formula was introduced by Bordeaux and Marques-Silva (2012) as a generalization of a unit refutation complete CNF formula introduced by del Val (1994).

Definition 2.1. *Let $\varphi(\mathbf{x})$ be a CNF formula.*

- *We say that φ is* unit refutation complete (URC) *if the following implication holds for every partial assignment $\alpha \subseteq \mathrm{lit}(\mathbf{x})$*

$$\varphi(\mathbf{x}) \wedge \alpha \models \bot \quad \Longrightarrow \quad \varphi \wedge \alpha \vdash_1 \bot\,. \tag{1}$$

- *We say that φ is* propagation complete (PC) *if for every partial assignment $\alpha \subseteq \mathrm{lit}(\mathbf{x})$ and for every $l \in \mathrm{lit}(\mathbf{x})$, such that*

$$\varphi(\mathbf{x}) \wedge \alpha \models l \tag{2}$$

we have

$$\varphi \wedge \alpha \vdash_1 l \quad \textit{or} \quad \varphi \wedge \alpha \vdash_1 \bot\,. \tag{3}$$

One can verify that a formula $\varphi(\mathbf{x})$ is PC if and only if for every partial assignment $\alpha \subseteq \mathrm{lit}(\mathbf{x})$ and for every $l \in \mathrm{lit}(\mathbf{x})$ such that

$$\varphi(\mathbf{x}) \wedge \alpha \nvdash_1 \bot \quad \text{and} \quad \varphi(\mathbf{x}) \wedge \alpha \nvdash_1 \neg l$$

the formula $\varphi(\mathbf{x}) \wedge \alpha \wedge l$ is satisfiable.

It was shown by Bordeaux and Marques-Silva (2012) that propagation complete formulas can be characterized using the notions of empowerment (Pipatsrisawat & Darwiche, 2011) and absorption (Atserias et al., 2011). We extend these notions to non-implicates, since this is useful for algorithmic purposes. A non-empty clause C is *empowering* with respect to a CNF formula φ if for some literal $l \in C$ we have that

$$\varphi \wedge \bigwedge_{e \in C \setminus \{l\}} \neg e \;\nvdash_1 l \quad \text{and} \quad \varphi \wedge \bigwedge_{e \in C \setminus \{l\}} \neg e \;\nvdash_1 \bot$$

and C is *absorbed* by φ otherwise. By the results of Bordeaux and Marques-Silva (2012), we have the following. A CNF formula φ is propagation complete if and only if it does not admit any empowering implicate or, equivalently, every non-empty implicate of φ is absorbed by φ. Moreover, if φ is a PC formula and $C \in \varphi$ is such that C is absorbed by $\varphi \setminus \{C\}$, then $\varphi \setminus \{C\}$ is a PC formula equivalent to φ. Since testing absorption can be done in polynomial time, this can be used to obtain in polynomial time an inclusion minimal subformula of a given PC formula that is a PC formula representing the same function.

Following del Val (1994), a resolution step which resolves a clause $A \vee x$ with a clause $B \vee \neg x$ is called *non-merge* if A is disjoint from B. It was discussed by Bordeaux and Marques-Silva (2012) (Section 5) that if a clause C is derived by a series of non-merge resolutions from a formula φ, then C is absorbed by φ. In particular, if every prime implicate of φ is either in φ or can be derived by non-merge resolution from φ, then φ is a PC formula. The same proposition for URC formulas was earlier shown by del Val (1994).

We define the CNF encodings as follows.

Definition 2.2 (Encoding)**.** *Let $f(\mathbf{x})$ be a boolean function on variables $\mathbf{x} = (x_1, \ldots, x_n)$. Let $\varphi(\mathbf{x}, \mathbf{y})$ be a CNF formula on $n+m$ variables where $\mathbf{y} = (y_1, \ldots, y_m)$. We call φ a* CNF encoding *of f if for every $\mathbf{a} \in \{0,1\}^{\mathbf{x}}$ we have*

$$f(\mathbf{a}) = (\exists \mathbf{b} \in \{0,1\}^{\mathbf{y}})\, \varphi(\mathbf{a}, \mathbf{b})\,, \tag{4}$$

where we identify 1 *and* 0 *with logical values true and false. The variables in* $\mathbf{x}$ *and* $\mathbf{y}$ *are called* main variables *and* auxiliary variables*, respectively.*

The notions of PC and URC encodings are defined as follows.

Definition 2.3. PC encoding *and* URC encoding *is an encoding that is a PC formula and a URC formula, respectively.*

2.2 q-Horn Formulas

The class of q-Horn formulas was introduced by Boros et al. (1990) as a generalization of both renamable Horn formulas and 2-CNF formulas. It is a tractable class which means that satisfiability of q-Horn formulas can be tested in polynomial time, the class of q-Horn formulas is closed under partial assignment, and we can check if a formula is q-Horn in linear time (Boros et al., 1994). Although Horn formulas are URC, q-Horn formulas are not URC in general. For example, an unsatisfiable 2-CNF consisting only of binary clauses (i.e., clauses of size 2) is q-Horn, but it is not URC.

Following the characterization of q-Horn formulas described by Boros et al. (1994), let us define these formulas as follows.

Definition 2.4 (Boros et al., 1990, 1994)**.** *Let* φ *be a CNF formula. We say that* $\gamma : \mathrm{lit}(\varphi) \to \{0, \frac{1}{2}, 1\}$ *is a valuation of literals in* φ *if* $\gamma(u) + \gamma(\neg u) = 1$ *for every literal* $u \in \mathrm{lit}(\varphi)$*. Formula* φ *is* q-Horn *if there is a valuation* γ *of literals in* φ *which satisfies*

$$\sum_{u \in C} \gamma(u) \leq 1 \tag{5}$$

for every clause $C \in \varphi$*. The value* $\gamma(u)$ *is called the* weight *of literal* u*. If* x *is a variable, then* $\gamma(x)$ *is called the* weight *of variable* x*.*

It is easy to observe that any renamable Horn formula is q-Horn, only values 0 and 1 are sufficient in a valuation. Also, any 2-CNF is q-Horn, just assign $\frac{1}{2}$ to every literal.

2.3 Succinctness

Let us recall the notion of succinctness introduced by Gogic et al. (1995) and used later extensively by Darwiche and Marquis (2002).

Definition 2.5 (Succinctness)**.** *Let* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *be two representation languages. We say that* $\mathbf{L}_1$ *is* at least as succinct as $\mathbf{L}_2$*, iff there exists a polynomial* p *such that for every sentence* $\varphi \in \mathbf{L}_2$*, there exists an equivalent sentence* $\psi \in \mathbf{L}_1$ *where* $|\psi| \leq p(|\varphi|)$*. We say that* $\mathbf{L}_1$ *is* strictly more succinct *than* $\mathbf{L}_2$ *if* $\mathbf{L}_1$ *is at least as succinct as* $\mathbf{L}_2$ *but* $\mathbf{L}_2$ *is not at least as succinct as* $\mathbf{L}_1$*.*

3. Related Work

When encoding constraints into CNF formulas a weaker concept than propagation completeness, namely generalized arc consistency (GAC) also called domain consistency, is often considered. For a presentation of GAC in the context of other local consistencies in

constraint programming see Bessiere (2006). CNF encodings which maintain generalized arc consistency via unit propagation (*GAC encodings*) were described first by Bacchus (2007). Two such notions were considered by Bessiere et al. (2009) in the form of a CNF decomposition of a consistency checker (*CC encoding*) and a CNF decomposition of a domain consistency propagator. The latter is equivalent to an encoding which maintains generalized arc consistency (GAC encoding). CC encoding differs from a URC encoding in that the implication (1) is required only for partial assignments of the main variables. Similarly, GAC encoding differs from a PC encoding in that the implication from (2) to (3) is required only if α and l consist only of literals on the main variables.

It was shown by Bessiere et al. (2009) that CC encodings and GAC encodings are polynomially equivalent. In particular, any CC encoding can be translated into a GAC encoding in polynomial time. This is in contrast to URC and PC encodings where it is open whether every URC encoding can be translated in polynomial time into a PC encoding of the same function.

Bessiere et al. (2009, Corollary 4) showed that there is no polynomial size CNF decomposition of the AllDifferent consistency checker. We can rephrase their result as that there is no polynomial size CC encoding for the AllDifferent constraint which implies that there is no GAC, URC or PC encoding for this constraint of polynomial size either.

Jung, Barahona, Katsirelos, and Walsh (2008) describe a polynomial time construction of a CNF encoding of a function represented with a decomposable negation normal form (DNNF, introduced by Darwiche, 1999). Later, Gange and Stuckey (2012) and Abío et al. (2016) argued that after a minor modification, this construction leads to a GAC encoding. A polynomial time translation of a significant subset of DNNFs into a PC encoding was first obtained for ordered decision diagrams (OBDD and MDD) by Abío et al. (2016). This construction was generalized to DNNFs by Kučera and Savický (2019). Bova, Capelli, Mengel, and Slivovsky (2014, 2016) show examples of monotone CNF formulas which have only exponentially larger DNNF. Given the fact that every monotone CNF formula is PC, we get that PC encodings are strictly more succinct than DNNFs.

It is well-known that there are PC formulas of polynomial size which have exponentially many prime implicates. Bordeaux and Marques-Silva (2012) show this in Proposition 5 for a linear size PC encoding of the parity (XOR) function considered as a function of all its variables. An even simpler example is the PC formula $\bigwedge_{i=1}^{n}(x_i \vee \neg y_i) \wedge (y_1 \vee \ldots \vee y_n)$. Together with the fact that a formula consisting of all prime implicates is PC (Babka et al., 2013) it follows that PC encodings without auxiliary variables are strictly more succinct than the list of all prime implicates denoted as PI by Darwiche and Marquis (2002).

4. Results

In this section, we state the results proven in the paper.

4.1 Separations

In our results, we consider four representation languages — URC and PC encodings with and without auxiliary variables. By definition, a PC encoding is also a URC encoding and the classes of these encodings contain encodings without auxiliary variables. This implies all the relations "at least as succinct" in Figure 1. It summarizes the known relationships

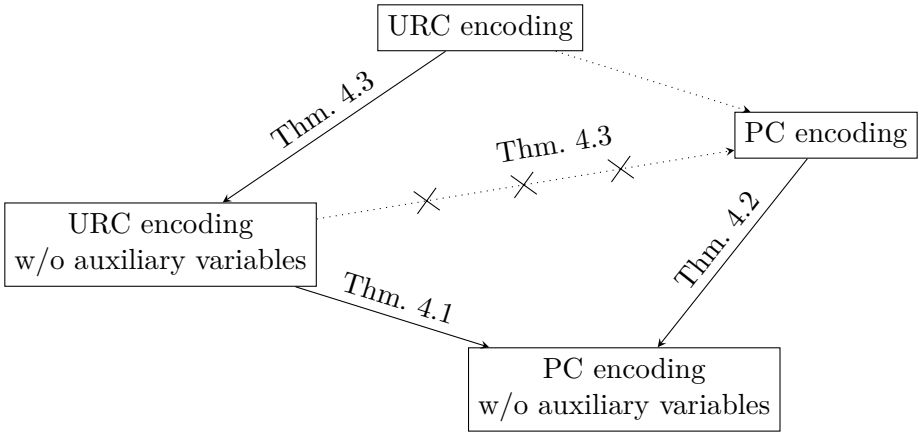

Figure 1: Diagram showing the relations between the classes of encodings we discuss in the paper. Solid arrow from $\mathbf{L}_1$ to $\mathbf{L}_2$ represents the fact that $\mathbf{L}_1$ is strictly more succinct than $\mathbf{L}_2$ and the separation uses functions represented by CNF formulas of polynomial size. Dotted arrow from $\mathbf{L}_1$ to $\mathbf{L}_2$ represents the fact that $\mathbf{L}_1$ is at least as succinct as $\mathbf{L}_2$. The crossed over dotted arrow means that the relation represented by the dotted arrow is not satisfied.

between the considered classes and the separations proven in our paper. Some of the relations in the diagram follow by transitivity from the others and are omitted.

Our emphasis is on the influence of URC and PC property on the succinctness of encodings. In particular, we prove in Section 5.1 as a consequence of Theorem 4.2 the following separation which is the first result on separation of the corresponding classes of encodings.

Theorem 4.1. *The language of URC encodings without auxiliary variables is strictly more succinct than the language of PC encodings without auxiliary variables.*

Encodings which can use auxiliary variables are known to be strictly more succinct than encodings without auxiliary variables and this remains true even if we require URC or PC property. A known example is the XOR function of several variables, which requires a CNF encoding without auxiliary variables of exponential size, however, it has a PC encoding with auxiliary variables which has a linear size. In this case, the large size of a CNF encoding without auxiliary variables follows already from the assumption that the formula represents the XOR function of several variables. As one of our results we show stronger separations which are based on functions represented by CNF formulas of polynomial size. This means that the lower bound parts of the separations follow solely from the requirements on the propagation strength. The strongest of these separations follows from Theorem 4.3 below implying that URC encodings without auxiliary variables are not at least as succinct as PC encodings that can use auxiliary variables. By transitivity, we obtain that each of the classes of URC and PC encodings without auxiliary variables is not at least as succinct as each of the classes of PC and URC encodings that can use auxiliary variables, although some of these separations follow more directly from Theorem 4.2 below.

The separations discussed above are based on the following. Lower bounds on the size of PC encodings without auxiliary variables follow from the next theorem proven in Section 5.1.

Theorem 4.2. *For every $n \geq 1$, there is a Horn and, hence, a URC formula φ_n of size $O(n)$, such that*

(a) every PC formula equivalent to φ_n has size $2^{\Omega(n)}$,

(b) there is a PC encoding of φ_n of size $O(n)$.

Lower bounds on the size of URC encodings without auxiliary variables follow from the next theorem proven in Section 5.2.

Theorem 4.3. *For every $n \geq 1$, there is a q-Horn formula φ_n of size $O(n)$, such that*

(a) every URC formula equivalent to φ_n has size $2^{\Omega(n)}$,

(b) there is a PC and, hence, a URC encoding of φ_n of size $O(n)$.

4.2 Irredundant PC and URC Formulas

We prove a remarkable difference in the properties of the size of irredundant PC and irredundant URC formulas representing a given function. A PC formula is called a *PC-irredundant* formula, if removing any clause either changes the represented function or yields a formula that is not a PC formula (such formulas were also called minimal propagation complete formulas by Bordeaux and Marques-Silva (2012)). A *URC-irredundant* formula is defined in a similar way. Namely, we prove:

- The sizes of any two PC-irredundant formulas for the same function differ at most by a factor polynomial in the number of the variables.
- There are URC-irredundant formulas for the same function such that one has size polynomial and the other has size exponential in the number of the variables.

A PC-irredundant formula can be obtained from any PC formula in polynomial time by repeated removal of absorbed clauses (see Bordeaux & Marques-Silva, 2012). The first statement means that this is also a guarantee of a small size of a PC formula, if we disregard factors polynomial in the number of the variables. On the other hand, the second statement means that irredundancy cannot be used in the same way for URC formulas.

The statement above concerning PC formulas follows from the next theorem proven in Section 6.1.

Theorem 4.4. *If φ_1 and φ_2 are PC-irredundant representations of the same function of n variables, then $|\varphi_2| \leq n^2|\varphi_1|$.*

The statement above concerning URC formulas follows from the next theorem proven in Section 6.2.

Theorem 4.5. *For every $n \geq 1$, there is a PC formula $\varphi_{n,1}$ of size $O(n)$ and an equivalent URC-irredundant formula $\varphi_{n,2}$ of size $2^{\Omega(n)}$.*

The proof of Theorem 4.4 is similar to the proof of the known result that the sizes of two equivalent irredundant Horn formulas differ at most by a factor at most $n-1$ (see Hammer & Kogan, 1993). For the PC formulas, the factor is larger, since we have to work with absorbed clauses instead of implicates provable by unit propagation. However, the relationship to Horn formulas can be made precise, since a PC formula can be understood as a representation of the closure operator defined by semantic consequence on the literals assuming a given formula. This representation is described in Section 7 using the correspondence between closure operators and Horn functions and can be understood as an explanation of the difference between the properties of irredundant PC and URC formulas. In particular, the above mentioned property of irredundant Horn formulas applies to implicational dual rail encoding of PC formulas (see Section 7) and implies only a slightly weaker upper bound than the bound in Theorem 4.4, namely $|\varphi_2| \leq 2n^2(|\varphi_1|+1)$ using the same notation. According to Theorem 4.5, there is no similar description of URC property using a closure operator.

4.3 An Encoding for q-Horn Formulas

We prove that although q-Horn formulas can be strongly non-URC in the sense formulated in Theorem 4.3, every q-Horn formula has a URC encoding of size polynomial in the size of the original formula.

Theorem 4.6. *Assume that $\varphi(\mathbf{x})$ is a q-Horn formula with a valuation γ representing a q-Horn function $f(\mathbf{x})$. Moreover, assume a partition of the variables $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2$, such that $\gamma(x) \in \{0,1\}$ for every $x \in \mathbf{x}_1$ and $\gamma(x) = 1/2$ for every $x \in \mathbf{x}_2$. Then there is a URC encoding $\psi(\mathbf{x},\mathbf{y})$ of $f(\mathbf{x})$ satisfying $|\mathbf{y}| = O(|\mathbf{x}_2|^2)$, $|\psi| = O(|\varphi| + |\mathbf{x}_2|^3)$, and $\|\psi\| = O(\|\varphi\| + |\mathbf{x}_2|^3)$.*

The encoding constructed in the proof of Theorem 4.6 is not q-Horn in general, although it represents a q-Horn function on the main variables. Note also that for the q-Horn function used in Theorem 4.3 we obtain a polynomial size PC encoding which has higher propagation strength than the encoding guaranteed by Theorem 4.6.

5. Lower Bounds on the Size of Specific Formulas

In this section, we present two examples of functions suitable for proving Theorem 4.2 and Theorem 4.3.

5.1 Lower Bound on the Size of a PC Formula

Assume that φ is a CNF formula not containing the variable x and let

$$\psi = \bigwedge_{C \in \varphi} (\neg x \vee C)\,. \tag{6}$$

A formula constructed in a similar way using a positive common literal was used for instance by Selman and Kautz (1996) to construct a small CNF formula such that every formula representing its Horn least upper bound has exponential size. One can verify that the set

of prime implicates of ψ is the set of all clauses of the form $\neg x \vee C$, where C is a prime implicate of φ (see also Bordeaux & Marques-Silva, 2012).

We claim that the only prime PC representation of ψ (i.e. a prime CNF formula that is equivalent to ψ and is PC) is the set of all its prime implicates. Assume that ψ' is a prime formula equivalent to ψ. If a prime implicate $\neg x \vee D$ of ψ is not in ψ', let α be the partial assignment $\neg D$. Clearly, $\psi' \wedge \alpha$ implies $\neg x$, however, every clause of ψ' has the form $\neg x \vee C$, where C is a prime implicate of φ different from D. It follows that C contains at least one literal not falsified by α and, hence, unit propagation does not derive any additional literal from α together with $\neg x \vee C$. This implies that φ' is not a PC formula, hence the only prime PC formula equivalent to (6) consists of all its prime implicates.

Each clause of the formula (6) contains the literal $\neg x$. In order to obtain a formula with bounded number of occurrences of each variable, we demonstrate that a similar lower bound on the size of a PC formula can be obtained, if the occurrences of $\neg x$ are replaced by different literals whose equivalence is guaranteed by additional clauses.

Lemma 5.1. *Assume that $\varphi = \bigwedge_{i=1}^{m} C_m$ is a satisfiable CNF formula with p prime implicates. Let $x_1, \ldots, x_m$ be new variables and let*

$$\psi = \bigwedge_{i=1}^{m-1} (\neg x_i \vee x_{i+1}) \wedge (\neg x_m \vee x_1) \wedge \bigwedge_{i=1}^{m} (\neg x_i \vee C_i) \,.$$

Then, the number of the prime implicates of ψ is $mp + m(m-1)$ and the size of a smallest PC representation of ψ is $p + m$.

Proof. Let Π be the set of all prime implicates of φ. Let Π' be the set of all clauses $\neg x_i \vee x_j$ where $i \neq j$, $1 \leq i, j \leq m$ and the clauses $\neg x_i \vee D$ where $1 \leq i \leq m$ and $D \in \Pi$. Each clause in Π' is either in ψ or can be derived from ψ by resolution. On the other hand, each resolvent of two clauses in Π' is either in Π' or subsumed by a clause in Π'. It follows that Π' is equal to the set of all prime implicates of ψ. In particular, the number of the prime implicates of ψ is $mp + m(m-1)$.

Consider the formula

$$\psi' = \bigwedge_{i=1}^{m-1} (\neg x_i \vee x_{i+1}) \wedge (\neg x_m \vee x_1) \wedge \bigwedge_{C \in \Pi} (\neg x_1 \vee C) \,.$$

Any prime implicate of ψ that is not in ψ' can be derived by non-merge resolution from ψ'. Following the discussion in Section 2 (see also Bordeaux & Marques-Silva, 2012, Section 5) we can conclude that ψ' is a PC representation of ψ. It follows that ψ has a PC representation of size $p + m$.

Assume, ψ'' is any prime PC representation of ψ. By applying a satisfying assignment of φ to ψ'', the prime implicates $\neg x_i \vee x_j$ remain unchanged and the prime implicates $\neg x_i \vee D$ become satisfied. Since the restricted formula represents the equivalence of the variables x_i for $1 \leq i \leq m$, the formula ψ'' contains at least m clauses of the form $\neg x_i \vee x_j$.

Assume for a contradiction that there is a prime implicate $D \in \Pi$, such that ψ'' does not contain any of the clauses $\neg x_i \vee D$, $i = 1, \ldots, m$. Let α be the partial assignment $\neg D$. Since each of the clauses $\neg x_i \vee D$ is an implicate of ψ, $\psi \wedge \alpha$ implies $\neg x_i$, for every $i = 1, .., m$.

On the other hand, for each clause $\neg x_j \vee C$ in ψ'' at least one literal in C is not falsified by α, since C and D are different prime implicates of φ. Since the clause also contains the literal $\neg x_j$, unit propagation does not derive any additional literal from $\neg x_j \vee C$ and α. In particular, it does not derive any of the literals $\neg x_i$. This is a contradiction. It follows that φ'' contains at least p clauses of the second type and, hence, has size at least $p + m$. □

For every $m \geq 3$, let ψ_m be the Horn formula of $3m - 2$ variables consisting of the clauses

$$\begin{aligned}
&\neg x_i \vee \neg y_i \vee z_i && i = 1, \ldots, m-1 \\
&\neg x_m \vee \neg z_1 \vee \cdots \vee \neg z_{m-1} && \\
&\neg x_i \vee x_{i+1} && i = 1, \ldots, m-1 \\
&\neg x_m \vee x_1\ . &&
\end{aligned}$$

Note that each variable has at most three occurrences in ψ_m which makes this formula more interesting than (6). Note also that the binary clauses on the variables x_i in ψ_m are important for the next proposition, since removing them changes ψ_m to a PC formula.

Proposition 5.2. *Formula ψ_m has $m2^{m-1} + O(m^2)$ prime implicates and its smallest PC representation has size $2^{m-1} + O(m)$.*

Proof. The formula ψ_m has the form from Lemma 5.1, if φ is

$$\begin{aligned}
&\neg y_i \vee z_i && i = 1, \ldots, m-1 \\
&\neg z_1 \vee \cdots \vee \neg z_{m-1} &&
\end{aligned}$$

Assuming $m \geq 3$, formula φ has 2^{m-1} prime implicates of the form

$$C_I = \bigvee_{i \in I} \neg y_i \vee \bigvee_{i \in \{1,\ldots,m-1\} \setminus I} \neg z_i$$

where $I \subseteq \{1, \ldots, m-1\}$. For any given set I, C_I originates from φ by resolving $\neg z_1 \vee \cdots \vee \neg z_{m-1}$ with $\neg y_i \vee z_i$, $i \in I$. In addition, φ has $m - 1$ implicates $\neg y_i \vee z_i$. No further clauses can be derived by resolution and thus φ has $p = 2^{m-1} + m - 1$ prime implicates. It follows by Lemma 5.1 that the number of the prime implicates of ψ_m is $mp + m(m-1) = m2^{m-1} + 2m(m-1)$ and the smallest size of a PC formula equivalent to ψ_m is $p + m = 2^{m-1} + 2m - 1$. □

Lemma 5.3. *There is a PC encoding of the function represented by ψ_m of size linear in the number of the variables.*

Proof. Consider the order of the variables given as $x_1, y_1, z_1, \ldots, x_{m-1}, y_{m-1}, z_{m-1}, x_m$. One can verify that the standard construction of a Decision-DNNF using this order of the variables in all paths leads to an OBDD of constant width and size $s = \Theta(m)$. The encoding CompletePath introduced by Abío et al. (2016) can be used to form a CNF encoding Ψ_m which encodes the function represented by ψ_m and which is propagation complete (Abío et al., 2016, Theorem 8). The CompletePath encoding is defined for every multi-valued decision diagram (MDD). In case of an OBDD we get that the domain size of variables used for decision nodes (i.e. variables $x_1, y_1, z_1, \ldots, x_{m-1}, y_{m-1}, z_{m-1}, x_m$) is $d = 2$. Abío et al. (2016) show that the number of variables of the encoding Ψ_m is upper bounded by $s(d+2)$

and the number of clauses is upper bounded by $s(4d+5)+nd$ where n is the number of variables in ψ_m, in our case $n = 3m-2$. It follows that Ψ_m uses $4s = \Theta(m)$ variables and $13s + 2(3m-2) = \Theta(m)$ clauses. Let us note that since the OBDD has constant width, the encoding has size linear in the number of the variables even if the exactly-one constraints used in CompletePath are represented by a formula of quadratic size without auxiliary variables. □

We are now ready to prove Theorem 4.2.

Theorem (Theorem 4.2)**.** *For every $n \geq 1$, there is a Horn and, hence, a URC formula φ_n of size $O(n)$, such that*

(a) every PC formula equivalent to φ_n has size $2^{\Omega(n)}$,

(b) there is a PC encoding of φ_n of size $O(n)$.

Proof. Let $m = \max(n, 3)$. By construction, ψ_m is a Horn formula. Condition (a) follows from Proposition 5.2 and condition (b) follows from Lemma 5.3. □

Theorem (Theorem 4.1)**.** *The language of URC encodings without auxiliary variables is strictly more succinct than the language of PC encodings without auxiliary variables.*

Proof. Clearly, the language of URC encodings without auxiliary variables is at least as succinct as the language of PC encodings without auxiliary variables. Consequently, Theorem 4.1 follows from (a) in Theorem 4.2. □

5.2 Lower Bound on the Size of a URC Formula

Given a natural number n, define a formula $\psi_n(\mathbf{x}, \mathbf{a}, \mathbf{b})$ where $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{a} = (a_1, \ldots, a_n)$, and $\mathbf{b} = (b_1, \ldots, b_n)$ as follows

$$\psi_n = \bigwedge_{i=1}^{n-1} (\neg a_i \vee \neg x_i \vee x_{i+1})(\neg a_i \vee x_i \vee \neg x_{i+1})(\neg b_i \vee \neg x_i \vee x_{i+1})(\neg b_i \vee x_i \vee \neg x_{i+1})$$
$$\wedge (\neg a_n \vee \neg x_1 \vee \neg x_n)(\neg a_n \vee x_1 \vee x_n)(\neg b_n \vee \neg x_1 \vee \neg x_n)(\neg b_n \vee x_1 \vee x_n).$$

We can also write ψ_n more concisely as

$$\psi_n \equiv \bigwedge_{i=1}^{n-1} [(a_i \vee b_i) \Rightarrow (x_i \Leftrightarrow x_{i+1})] \wedge [(a_n \vee b_n) \Rightarrow (x_1 \Leftrightarrow \neg x_n)] \,. \tag{7}$$

Observe that ψ_n is not URC, because $\psi_n \wedge \bigwedge_{i=1}^{n} a_i \models \bot$, however, $\psi_n \wedge \bigwedge_{i=1}^{n} a_i \not\models_1 \bot$. On the other hand, one can verify that ψ_n is q-Horn using the valuation $\gamma(x_i) = \gamma(\neg x_i) = \frac{1}{2}$, $\gamma(a_i) = \gamma(b_i) = 1$, and $\gamma(\neg a_i) = \gamma(\neg b_i) = 0$ for all $i = 1, \ldots, n$. The main part of Theorem 4.3 is the following lower bound.

Proposition 5.4. *If φ is a URC formula equivalent to ψ_n, then $|\varphi| \geq 2^n$.*

Proof. Without loss of generality, we can assume that φ is a prime formula. Let us start by showing that there is no prime implicate of ψ_n which would contain exactly one literal from $\mathrm{lit}(\mathbf{x})$. Let us assume for a contradiction that there is a prime implicate C of ψ_n which contains precisely one literal $l \in \mathrm{lit}(\mathbf{x})$. Since C is prime, there is a model $\mathbf{a}$ of ψ_n, such that l is the only satisfied literal in C (otherwise $C \setminus \{l\}$ would still be an implicate of ψ_n contradicting the primality of C). Let $\mathbf{a}'$ be an assignment that is produced from $\mathbf{a}$ by negating the values of all variables in $\mathbf{x}$. Since all the occurrences of the variables $\mathbf{x}$ in (7) are involved in equivalences and non-equivalences, the set of satisfying assignments of ψ_n is invariant under taking the negation of all the variables $\mathbf{x}$ simultaneously. In particular, $\mathbf{a}'$ is also a model of ψ_n. However, C is falsified by $\mathbf{a}'$ which is in contradiction with the fact that C is an implicate of ψ_n. It follows that every prime implicate of ψ_n containing a literal from $\mathrm{lit}(\mathbf{x})$ contains at least two such literals. In particular, this is true for every clause of φ, since φ is prime.

Let U denote the set of partial assignments $\alpha \cup \beta$ where $\alpha \subseteq \mathbf{a}$, $\beta \subseteq \mathbf{b}$ and for every index $i = 1, \ldots, n$ exactly one of the variables a_i and b_i belongs to $\alpha \cup \beta$. In particular $|\alpha \cup \beta| = n$ and $|U| = 2^n$. Clearly, for every $\alpha \cup \beta \in U$, the formula $\psi_n \wedge \alpha \wedge \beta$ is inconsistent. It follows that the clauses in the set $\overline{U} = \{\neg(\alpha \wedge \beta) \mid \alpha \cup \beta \in U\}$ are implicates of ψ_n. Moreover, one can verify that $\overline{U}$ is precisely the set of all prime implicates of ψ_n containing only the literals from $\mathrm{lit}(\mathbf{a} \cup \mathbf{b})$. Let us prove that φ contains all of the implicates in $\overline{U}$.

Assume a partial assignment $\alpha \cup \beta \in U$ and the clause $C = \neg(\alpha \wedge \beta)$. Assume for a contradiction that C is not in φ. Since $\varphi \wedge \alpha \wedge \beta$ is inconsistent and φ is URC, we have $\varphi \wedge \alpha \wedge \beta \vdash_1 \bot$. Using the argument from the first paragraph of the proof, this derivation does not use any of the prime implicates containing a literal from $\mathrm{lit}(\mathbf{x})$. However, unit propagation using implicates from $\overline{U} \setminus \{C\}$ derives only negative literals on the variables $\mathbf{a} \cup \mathbf{b}$ and these literals cannot be used to continue unit propagation using clauses from $\overline{U} \setminus \{C\}$. This contradicts the assumption $\varphi \wedge \alpha \wedge \beta \vdash_1 \bot$. It follows that C is in φ as required. □

Lemma 5.5. *There is a PC encoding of the function represented by ψ_n of size linear in the number of the variables.*

Proof. Use the same approach as in the proof of Lemma 5.3 with the order of the variables given as $x_1, a_1, b_1, x_2, a_2, b_2, \ldots, x_n, a_n, b_n$. □

Theorem (Theorem 4.3)**.** *For every $n \geq 1$, there is a q-Horn formula φ_n of size $O(n)$, such that*

(a) every URC formula equivalent to φ_n has size $2^{\Omega(n)}$,

(b) there is a PC and, hence, a URC encoding of φ_n of size $O(n)$.

Proof. A consequence of Proposition 5.4 and Lemma 5.5. □

The size of the encoding guaranteed by Lemma 5.5 is $cn + O(1)$ with a relatively large constant c. Let us note that a PC encoding of the function represented by ψ_n of size $4n + O(1)$ can be described as follows. We use additional auxiliary variables $\mathbf{c} = (c_1, \ldots, c_n)$

and define

$$\begin{aligned}\psi'_n = &\bigwedge_{i=1}^{n}(\neg a_i \vee c_i)(\neg b_i \vee c_i)\\ &\wedge (\neg c_1 \vee \cdots \vee \neg c_n)\\ &\wedge \bigwedge_{i=1}^{n-1}(\neg c_i \vee \neg x_i \vee x_{i+1})(\neg c_i \vee x_i \vee \neg x_{i+1})\\ &\wedge (\neg c_n \vee \neg x_1 \vee \neg x_n)(\neg c_n \vee x_1 \vee x_n).\end{aligned}$$

One can check that $\psi'_n(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{c})$ is a PC encoding of the function represented by $\psi_n(\mathbf{x}, \mathbf{a}, \mathbf{b})$. The proof can be done by case inspection and is omitted since Lemma 5.5 is sufficient for the proof of Theorem 4.3 above.

6. Irredundant PC and URC Formulas

In this section, we prove the results formulated in Section 4.2.

6.1 Irredundant PC Formulas

Recall that a formula is PC-irredundant, if removing any clause either changes the represented function or leads to a formula that is not PC. Let us now prove Theorem 4.4.

Theorem (Theorem 4.4)**.** *If φ_1 and φ_2 are PC-irredundant representations of the same function of n variables, then $|\varphi_2| \leq n^2|\varphi_1|$.*

Proof. If $C \in \varphi_1$ and $l \in C$, then $\varphi_2 \wedge \neg(C \setminus \{l\}) \vdash_1 l$ or $\varphi_2 \wedge \neg(C \setminus \{l\}) \vdash_1 \bot$, since φ_2 is a PC formula and C is its implicate. Let $\theta_{C,l}$ be a set of clauses of φ_2 used in one of these derivations. Since the derivation is unit resolution and each clause is used to derive a literal on a different variable, we have $|\theta_{C,l}| \leq n$. Moreover, we have $\theta_{C,l} \models C$. Let φ'_2 be the union of $\theta_{C,l}$ for all $C \in \varphi_1$ and $l \in C$. Clearly, φ'_2 is a subset of φ_2 and $|\varphi'_2| \leq n^2|\varphi_1|$. It remains to verify that φ'_2 is a PC formula equivalent to φ_2. This implies that $\varphi'_2 = \varphi_2$, since φ_2 is PC-irredundant, and the statement follows.

Formula φ'_2 is implied by φ_2 and implies φ_1. Together, this implies that φ'_2 is equivalent to both φ_1 and φ_2. Consider the formula $\varphi'_2 \wedge \varphi_1$. Since it contains φ_1, it is a PC formula. By construction of φ'_2, each clause of φ_1 is absorbed by φ'_2. This implies that we can successively remove all clauses of φ_1 from $\varphi'_2 \wedge \varphi_1$ while keeping its PC property. □

6.2 A Large Irredundant URC Formula

Recall that a formula is URC-irredundant, if removing any clause either changes the represented function or leads to a formula that is not URC. In this section, we present an example of a formula that can be extended by additional clauses to a PC formula of size linear in the number of the variables and has also a URC-irredundant extension of size exponential in the number of the variables.

Let $m \geq 2$. Consider the variables a_i, b_i, c_i, d_i, the definite Horn formulas

$$\delta_i = (\neg a_i \vee b_i) \wedge (\neg a_i \vee c_i) \wedge (\neg b_i \vee \neg c_i \vee d_i)$$

for $i = 1, \ldots, m$, and the formulas

$$\gamma_m = \left(\bigvee_{i=1}^{m} a_i\right) \wedge \bigwedge_{i=1}^{m} \delta_i$$
$$\gamma'_m = \gamma_m \wedge \bigwedge_{i=1}^{m} (\neg a_i \vee d_i)$$
$$\gamma''_m = \gamma_m \wedge \bigwedge_{I \in E_m} \left(\bigvee_{i \notin I} a_i \vee \bigvee_{i \in I} d_i\right)$$

where E_m is the system of all non-empty subsets $I \subseteq \{1, \ldots, m\}$ such that $|I|$ is even. Clearly, we have

$$\begin{aligned} |\gamma_m| &= 3m + 1 \\ |\gamma'_m| &= 4m + 1 \\ |\gamma''_m| &= 3m + 2^{m-1} \,. \end{aligned}$$

Lemma 6.1. *γ'_m and γ''_m are equivalent to γ_m.*

Proof. For every $i = 1, \ldots, m$, the clause $\neg a_i \vee d_i$ can be obtained by resolution from the clause $\neg b_i \vee \neg c_i \vee d_i$ and the clauses $\neg a_i \vee b_i$, $\neg a_i \vee c_i$. This implies that γ'_m is equivalent to γ_m.

For every $I \subseteq \{1, \ldots, m\}$, not necessarily in E_m, the clause

$$\bigvee_{i \notin I} a_i \vee \bigvee_{i \in I} d_i$$

can be obtained by resolution from the clause $\bigvee_{i=1}^{m} a_i$ and the clauses $\neg a_i \vee d_i$ for $i \in I$. This implies that γ''_m is equivalent to γ_m. □

Lemma 6.2. *γ'_m is a PC formula.*

Proof. The disjunction $\bigvee_{i=1}^{m} a_i$ is a PC formula and also $\delta'_i = \delta_i \wedge (\neg a_i \vee d_i)$ are PC formulas. The formula γ'_m is obtained from $\bigvee_{i=1}^{m} a_i$ by combining it sequentially with δ'_i for $i = 1, \ldots, m$. In each step, we combine formulas which have only one variable in common. It follows that the obtained formula is a PC formula. □

Lemma 6.3. *γ''_m is a URC formula.*

Proof. Assume a partial assignment ρ, such that $\gamma''_m \wedge \rho$ is unsatisfiable and, hence, also $\gamma_m \wedge \rho$ is unsatisfiable. Let us prove that unit propagation from $\gamma''_m \wedge \rho$ derives $\bot$.

If there is an index $i \in \{1, \ldots, m\}$, such that $\delta_i \wedge \rho$ is unsatisfiable, then $\bot$ can be derived by unit propagation, since δ_i is a Horn and thus also a URC formula. For the rest of the proof, assume that each of the formulas $\delta_i \wedge \rho$ is satisfiable. Assume for a contradiction that there is an index j, such that $\rho \wedge \delta_j \not\models \neg a_j$. The setting $a_j = 1$ satisfies the clause $\bigvee_{i=1}^{m} a_i$ and it can be extended into a satisfying assignment of $\gamma_m \wedge \rho$, since the formulas δ_i are pairwise independent and each of them is satisfiable together with $\rho \wedge a_j$. Since this

contradicts the assumption, it follows that for all $i = 1, \ldots, m$, we have $\rho \wedge \delta_i \models \neg a_i$. By case inspection, this is equivalent to the assumption that for every $i = 1, \ldots, m$, we have $\{\neg a_i, \neg b_i, \neg c_i, \neg d_i\} \cap \rho \neq \emptyset$. Let J be the set of indices i, such that $\delta_i \wedge \rho \not\vdash_1 \neg a_i$. One can verify that for every $i \in J$, we have $\neg d_i \in \rho$. This implies that the clause

$$\bigvee_{i \notin J} a_i \vee \bigvee_{i \in J} d_i$$

is falsified by the literals derived by unit propagation in formulas $\delta_i \wedge \rho$. If $|J|$ is even, then $J \in E_m$ or $J = \emptyset$ and the falsified clause is contained in γ''_m, so unit propagation derives $\bot$. If $|J|$ is odd, choose an arbitrary $j \in J$ and consider $I = J \setminus \{j\}$. Since $I \in E_m$ or $I = \emptyset$, the clause

$$\bigvee_{i \notin I} a_i \vee \bigvee_{i \in I} d_i$$

is in γ''_m and the only literal in this clause that is not falsified by unit propagation using ρ and the formulas δ_i is a_j. Hence, unit propagation from $\gamma''_m \wedge \rho$ derives a_j. Together with δ_j, we further derive d_j and, finally, we derive $\bot$, since $\neg d_j \in \rho$. □

Lemma 6.4. *Formula γ''_m is URC-irredundant.*

Proof. By the previous lemma, γ''_m is URC. If any of the clauses in δ_i for $i \in \{1, \ldots, m\}$ is removed from γ''_m, then the restriction of the function obtained by setting all the variables a_j, b_j, c_j, d_j for $j \neq i$ to 1 changes, since δ_i is an irredundant formula. If the clause $\bigvee_{i=1}^{m} a_i$ is removed, then the represented function changes on the assignment setting all a_i variables to 0 and all the remaining variables to 1.

Let us verify that removing any of the clauses

$$C = \bigvee_{i \notin I} a_i \vee \bigvee_{i \in I} d_i$$

where $I \in E_m$ from γ''_m leads to a formula that is not a URC formula for γ_m. Consider the partial assignment $\neg C$ and let us prove that this assignment does not make any of the clauses of $\gamma''_m \setminus \{C\}$ unit or empty. This can be verified by case inspection for the clauses of δ_j for every $j = 1, \ldots, m$. Since $|I| \geq 2$, the clause

$$\bigvee_{i=1}^{m} a_i \in \gamma_m$$

also does not become unit or empty. If $J \in E_m \setminus \{I\}$, then the number of the literals in the clause

$$\bigvee_{i \notin J} a_i \vee \bigvee_{i \in J} d_i \in \gamma''_m$$

that are not falsified by $\neg C$ is equal to the size of the symmetric difference of the sets I and J. Since these sets are different and both have even size, their symmetric difference has size at least 2. Altogether, the assignment $\neg C$ is closed under unit propagation in the formula $\gamma''_m \setminus \{C\}$ and, hence, unit propagation does not derive a contradiction from $\neg C \wedge (\gamma''_m \setminus \{C\})$. Since C is an implicate of γ_m, this implies that $\gamma''_m \setminus \{C\}$ is not a URC formula for γ_m. □

We are now ready to prove Theorem 4.5.

Theorem (Theorem 4.5)**.** *For every $n \geq 1$, there is a PC formula $\varphi_{n,1}$ of size $O(n)$ and an equivalent URC-irredundant formula $\varphi_{n,2}$ of size $2^{\Omega(n)}$.*

Proof. For a given n, let $m = \max(n, 2)$, $\varphi_{n,1} = \gamma'_m$, and $\varphi_{n,2} = \gamma''_m$. Since $|\gamma'_m| = O(n)$ and $|\gamma''_m| = 2^{\Omega(n)}$, the theorem follows from lemmas 6.2 and 6.4. □

7. Implicational Dual Rail Encoding of PC Formulas

Unit propagation in a formula on the variables $\mathbf{x}$ is a closure operator on the sets of literals $\mathrm{lit}(\mathbf{x})$. It follows that it can be characterized by a suitable Horn function whose variables correspond to the literals $\mathrm{lit}(\mathbf{x})$. In this section, we demonstrate that this correspondence can be used to derive a slightly weaker version of Theorem 4.4, namely, the bound

$$|\varphi_2| \leq 2n^2(|\varphi_1| + 1) \tag{8}$$

where the same notation as in Theorem 4.4 is used. Although this bound is weaker than Theorem 4.4, its proof demonstrates that this property of irredundant PC formulas is a consequence of the known properties of closure operators and their representation using Horn formulas. See Guigues and Duquenne (1986), Arias, Balcázar, and Tîrnăucă (2017) for more information on the relationship between closure operators (or closure systems) and Horn formulas. In particular, a formula is a PC formula, if it represents in the sense specified below the semantic closure operator on the sets of literals which are subsets of a set of polynomial size. By Theorem 4.5, there is no similar connection between URC property and a suitable closure operator on subsets of a small set.

We use the well-known dual rail encoding of partial assignments (Bonet et al., 2018; Bryant et al., 1987; Ignatiev et al., 2017; Manquinho et al., 1997; Morgado et al., 2019) to simulate unit propagation in a general CNF formula in the same way as Bordeaux et al. (2012), Bessiere et al. (2009). Let us describe a variant of such a simulation suitable for our purposes.

Assume, φ is a CNF formula on the variables $\mathbf{x}$. Let us introduce for every $l \in \mathrm{lit}(\mathbf{x})$ a meta-variable $[\![l]\!]$ and let us denote $\mathrm{meta}(\mathbf{x})$ the set of all these variables. For each pair (C, l), such that $l \in C \in \varphi$, we express the step of unit propagation deriving l from $\neg(C \setminus \{l\})$ as an implication

$$\left(\bigwedge_{e \in C \setminus \{l\}} [\![\neg e]\!] \right) \to [\![l]\!] \tag{9}$$

or an equivalent Horn clause

$$[\![l]\!] \vee \bigvee_{e \in C \setminus \{l\}} \neg[\![\neg e]\!]\,. \tag{10}$$

The *implicational dual rail encoding* of φ is the formula $\mathrm{DR}(\varphi)$ on the variables $\mathrm{meta}(\mathbf{x})$ described as follows. If φ contains an empty clause, then $\mathrm{DR}(\varphi) = \bot$. Otherwise, $\mathrm{DR}(\varphi)$ consists of all the clauses (10) and additional consistency clauses such that

$$\mathrm{DR}(\varphi) = \bigwedge_{l \in C \in \varphi} \left([\![l]\!] \vee \bigvee_{e \in C \setminus \{l\}} \neg[\![\neg e]\!] \right) \wedge \bigwedge_{x \in \mathbf{x}} (\neg[\![x]\!] \vee \neg[\![\neg x]\!])\,.$$

Unit propagation in a CNF formula $\varphi(\mathbf{x})$ with a partial assignment $\alpha \subset \mathrm{lit}(\mathbf{x})$ can be simulated by forward chaining in the definite Horn part of $\mathrm{DR}(\varphi)$ which is equivalent to the implications (9). Sets of literals on the variables $\mathbf{x}$ are represented by total assignments of the variables $\mathrm{meta}(\mathbf{x})$ in a natural way. Using this, the satisfying assignments of the definite Horn part of $\mathrm{DR}(\varphi)$ represent precisely the sets of literals on the variables $\mathbf{x}$ closed under unit propagation in φ including those that contain complementary pairs of literals. Since a closure operator is fully characterized by its closed sets, the definite Horn part of $\mathrm{DR}(\varphi)$ fully characterizes unit propagation in φ including propagation in contradictory cases.

When considering unit propagation, it is not desirable to distinguish different contradictory cases. For this reason, $\mathrm{DR}(\varphi)$ is a Horn formula containing also negative clauses which imply a contradiction whenever the simulated unit propagation reaches a complementary pair of literals. As a consequence, we obtain that the models of $\mathrm{DR}(\varphi)$ correspond precisely to the partial assignments (consistent sets of literals) closed under unit propagation in φ. Let us denote this set of partial assignments as $S(\varphi)$. One can verify that $S(\varphi)$ characterizes unit propagation in φ as follows. If $\alpha \subset \mathrm{lit}(\mathbf{x})$ is a partial assignment, then unit propagation from $\varphi \wedge \alpha$ derives a partial assignment β, if β is the smallest partial assignment, such that $\alpha \subseteq \beta \in S(\varphi)$, or a contradiction, if such a partial assignment β does not exist. Altogether, two formulas provide the same derivations using unit propagation, if and only if their implicational dual rail encodings are equivalent. In particular, we have

Proposition 7.1. *Let φ and φ' be equivalent CNF formulas and assume that φ' is a PC formula. Then φ is a PC formula if and only if* $\mathrm{DR}(\varphi)$ *and* $\mathrm{DR}(\varphi')$ *are equivalent.*

Proposition 7.1 implies that φ is a PC formula if and only if $\mathrm{DR}(\varphi)$ represents a fixed function. Since a formula φ is PC if and only if unit propagation in it coincides with the semantic consequence on the literals assuming φ, it is the characteristic function of the partial assignments closed under semantic consequence on the literals assuming φ. This function can be obtained, for example, as $\mathrm{DR}(\varphi')$, where φ' is the set of all prime implicates of φ.

Let us use this to prove (8). Assume that φ_1 and φ_2 are irredundant PC formulas for the same function of n variables. By Proposition 7.1, $\mathrm{DR}(\varphi_1)$ and $\mathrm{DR}(\varphi_2)$ are equivalent Horn formulas of $2n$ variables. Let ψ_1 and ψ_2 be irredundant subformulas of $\mathrm{DR}(\varphi_1)$ and $\mathrm{DR}(\varphi_2)$, respectively. By the results on Horn formulas proven by Hammer and Kogan (1993), the sizes of ψ_1 and ψ_2 differ at most by a factor $2n-1$, so we have

$$|\psi_2| \leq 2n|\psi_1| \leq 2n(n + n|\varphi_1|) = 2n^2(|\varphi_1| + 1)\ . \tag{11}$$

Formula $\mathrm{DR}(\varphi_2)$ contains a group of $|C|$ definite Horn clauses for each clause $C \in \varphi_2$. If there is a clause $C \in \varphi_2$, such that none of the corresponding clauses of $\mathrm{DR}(\varphi_2)$ is in ψ_2, then Proposition 7.1 implies a contradiction with irredundancy of φ_2. Hence, we have $|\varphi_2| \leq |\psi_2|$ and together with (11), we obtain (8).

8. URC Encoding of a q-Horn Formula

In this section, we show that every q-Horn function represented by a q-Horn formula $\varphi(\mathbf{x})$ has a URC encoding of size polynomial in the size of φ. The encoding is based on simulating the satisfiability test for q-Horn formulas presented by Boros et al. (1990).

Let us fix a valuation γ which shows that φ is q-Horn. For simplicity, we assume $\gamma(x) \geq \gamma(\neg x)$ for every variable $x \in \mathbf{x}$. If this assumption is not satisfied for a variable x, we can replace all occurrences of x with $\neg x$ and vice versa and set $\gamma(x) = 1 - \gamma(x)$. As a consequence, all variables in φ have weight 1 or $\frac{1}{2}$. The set of variables of weight 1 and $\frac{1}{2}$ will be denoted as $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Unless stated otherwise, we assume that the sets of variables $\mathbf{x}$, $\mathbf{x}_1$, $\mathbf{x}_2$, and the corresponding valuation γ is fixed.

Recall that given a CNF formula $\varphi(\mathbf{x})$ and a partial assignment $\beta \subseteq \mathrm{lit}(\mathbf{x})$ we denote $\varphi(\beta)$ the formula which originates from φ by applying partial assignment β. A partial assignment $\beta \subseteq \mathrm{lit}(\mathbf{x})$ is an *autark assignment* (Kullmann, 2000) for a CNF φ if for every clause $C \in \varphi$ such that $C(\beta)$ is different from C, we have that C is satisfied by β. Since satisfied clauses are removed when applying a partial assignment, this implies $\varphi(\beta) \subseteq \varphi$. It follows that φ is satisfiable if and only if $\varphi(\beta)$ is satisfiable, since using β, one can obtain a satisfying assignment of φ from a satisfying assignment of $\varphi(\beta)$ and vice versa.

Lemma 8.1. *Assume, φ is a q-Horn formula with the valuation γ and let δ be the partial assignment which sets all $\mathbf{x}_1$ variables that appear in φ to 0. If φ does not contain a unit clause on a variable from $\mathbf{x}_1$, then δ is an autark assignment for φ.*

Proof. If C contains a positive literal on a variable from $\mathbf{x}_1$, then C contains only variables from $\mathbf{x}_1$ and, hence, also a negative literal on some other variable from $\mathbf{x}_1$. It follows that all clauses of φ affected by δ are satisfied by δ. □

If unit propagation on a q-Horn formula φ does not lead to a contradiction, then the resulting formula satisfies the assumption of Lemma 8.1 and the formula $\varphi(\delta)$ is a 2-CNF formula. This allows to reduce the test of satisfiability of a q-Horn formula to a test of satisfiability of a 2-CNF formula and, hence, implies a polynomial time satisfiability test for q-Horn formulas. Moreover, in order to satisfy the assumption of Lemma 8.1, it is sufficient to perform unit propagation restricted to the clauses containing only the variables $\mathbf{x}_1$. We split the formula $\varphi(\mathbf{x}_1, \mathbf{x}_2)$ into two subformulas $\varphi_1(\mathbf{x}_1)$ and $\varphi_2(\mathbf{x}_1, \mathbf{x}_2)$. A clause $C \in \varphi$ belongs to φ_1 if $\mathrm{var}(C) \subseteq \mathbf{x}_1$ and it belongs to φ_2 otherwise. By construction, φ_1 is a Horn formula. Clauses in φ_2 contain one or two variables of weight $\frac{1}{2}$. Other literals in these clauses have weight 0 and they are negations of variables of weight 1. Using this notation, the satisfiability test for a q-Horn formula φ described by Boros et al. (1990) is formulated as Algorithm 1. Note that by Lemma 8.1, δ is an autark assignment for $\varphi(\beta)$ and satisfies all clauses of $\varphi_1(\beta)$.

We shall construct a URC encoding $\psi(\mathbf{x}, \mathbf{y})$ for a given q-Horn formula $\varphi(\mathbf{x})$ with the valuation γ. Unit propagation in the encoding allows to simulate Algorithm 1 with a formula $\varphi \wedge \alpha$ as an input, where $\alpha \subseteq \mathrm{lit}(\mathbf{x})$. Formula θ created in Step 5 is a 2-CNF formula on variables $\mathbf{x}_2$ and we can thus check its satisfiability in linear time in Step 6 (see for example Aspvall et al., 1979). In the URC encoding, Step 6 is implemented by simulating a resolution derivation of a contradiction from θ in which only resolvents of size at most 2 are needed. The number of such resolvents is at most quadratic in the number of the variables in $\mathbf{x}_2$. We can thus encode the resolution rules into a polynomial number of clauses of the encoding.

Let us first introduce a necessary notation. Given a q-Horn formula $\varphi = \varphi_1 \wedge \varphi_2$, we denote $\varphi_q = \{C \cap \mathrm{lit}(\mathbf{x}_2) \mid C \in \varphi_2\}$ which is a 2-CNF formula on variables of weight $\frac{1}{2}$. Moreover, let φ_q^+ be the set of all clauses of size 2 which can be derived by resolution from

Input: q-Horn formula $\varphi(\mathbf{x})$, valuation γ satisfying $\gamma(x) \in \{\frac{1}{2}, 1\}$ for all $x \in \mathbf{x}$
Output: *SAT* if φ is satisfiable, *UNSAT* otherwise
1 let φ_1, φ_2, $\mathbf{x}_1$, and $\mathbf{x}_2$ be as described in the text
2 **if** $\varphi_1 \vdash_1 \bot$ **then return** *UNSAT*
3 $\beta \leftarrow \{u \in \mathrm{lit}(\mathbf{x}_1) \mid \varphi_1 \vdash_1 u\}$
4 $\delta \leftarrow \{\neg x \mid x \in \mathbf{x}_1, x \text{ not fixed by } \beta\}$
5 $\theta \leftarrow \varphi_2(\beta \wedge \delta)$
6 **if** θ *is satisfiable* **then return** *SAT* **else return** *UNSAT*

Algorithm 1: Satisfiability checking of a q-Horn formula (Boros et al., 1990)

φ_q and we associate a meta-variable $[\![C]\!]$ with every clause $C \in \varphi_q^+$. Note that if we test satisfiability of $\varphi \wedge \alpha$ instead of φ in Algorithm 1, the formula θ can contain additional unit clauses from $\alpha \cap \mathrm{lit}(\mathbf{x}_2)$, however, all its binary clauses belong to φ_q. The encoding $\psi(\mathbf{x}, \mathbf{y})$ for $\varphi(\mathbf{x})$ uses variables

$$\mathbf{y} = \{[\![C]\!] \mid C \in \varphi_q^+\}$$

as auxiliary variables and consists of the clauses defined by Table 1.

Let us look more closely at clauses of the encoding. Group (q1) consists of all clauses of φ_1 and some of the clauses of φ_2. The clauses of (q1) that belong to φ_1 allow unit propagation on the clauses of φ_1, thus implementing the first part of Algorithm 1. Clauses of group (q1) that belong to φ_2 and clauses of group (q2) allow to derive by unit propagation a representation of all clauses of θ. Unit clauses are represented directly and each binary clause in θ is represented by the corresponding meta-variable. Clauses of groups (q3) to (q5) allow to simulate resolution on unit clauses and at least one binary clause from φ_q^+ represented by meta-variables by unit propagation in $\psi(\mathbf{x}, \mathbf{y})$. In particular, clause $\neg[\![u \vee v]\!] \vee \neg[\![\neg v \vee w]\!] \vee [\![u \vee w]\!]$ in group (q3) allows to derive literal $[\![u \vee w]\!]$ representing resolvent assuming literals $[\![u \vee v]\!]$ and $[\![\neg v \vee w]\!]$ representing the original clauses. Similarly, clause $\neg[\![u \vee v]\!] \vee \neg[\![u \vee \neg v]\!] \vee u$ in group (q4) allows to derive resolvent u assuming $[\![u \vee v]\!]$ and $[\![u \vee \neg v]\!]$. Finally, clause $\neg[\![u \vee v]\!] \vee v \vee u$ in group (q5) allows to derive resolvent u assuming $[\![u \vee v]\!]$ and $\neg v$. Clauses

group	clause	condition
(q1)	C	$C \in \varphi$ and $\lvert\mathrm{var}(C) \cap \mathbf{x}_2\rvert \leq 1$
(q2)	$\neg x_{i_1} \vee \cdots \vee \neg x_{i_k} \vee [\![u \vee v]\!]$	$\neg x_{i_1} \vee \cdots \vee \neg x_{i_k} \vee u \vee v \in \varphi_2$
(q3)	$\neg[\![u \vee v]\!] \vee \neg[\![\neg v \vee w]\!] \vee [\![u \vee w]\!]$	$u \vee v, \neg v \vee w, u \vee w \in \varphi_q^+$
(q4)	$\neg[\![u \vee v]\!] \vee \neg[\![u \vee \neg v]\!] \vee u$	$u \vee v, u \vee \neg v \in \varphi_q^+$
(q5)	$\neg[\![u \vee v]\!] \vee u \vee v$	$u \vee v \in \varphi_q^+$
(q6)	$\neg u \vee [\![u \vee v]\!]$	$u \vee v \in \varphi_q^+$

Table 1: The clauses of encoding $\psi(\mathbf{x}, \mathbf{y})$ for a q-Horn formula $\varphi(\mathbf{x})$, where u, v, and w denote arbitrary literals from $\mathrm{lit}(\mathbf{x}_2)$.

of groups (q5) and (q6) together represent the equivalences $u \vee v \Leftrightarrow [\![u \vee v]\!]$ for all clauses in φ_q^+. They thus define the semantics of the meta-variables $[\![u \vee v]\!]$ in $\mathbf{y}$.

Lemma 8.2. *Formula $\psi(\mathbf{x}, \mathbf{y})$ is an encoding of $\varphi(\mathbf{x})$.*

Proof. The clauses of groups (q5) and (q6) imply the clauses of groups (q3) and (q4). Moreover, they also imply that the clauses of groups (q1) and (q2) are equivalent to $\varphi(\mathbf{x})$. It follows that $\psi(\mathbf{x}, \mathbf{y})$ is equivalent to the conjunction of $\varphi(\mathbf{x})$ and the clauses of groups (q5) and (q6). This conjunction is clearly an encoding of $\varphi(\mathbf{x})$ obtained by adding definitions of the new variables $\mathbf{y}$. □

In order to prove that ψ is a URC encoding, we distinguish three cases for α in the implication (1). The first of them is the following lemma which states that ψ implements consistency checker.

Lemma 8.3. *Let $\alpha \subseteq \mathrm{lit}(\mathbf{x})$ be a partial assignment. If $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \models \bot$, then we have $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 \bot$.*

Proof. Assume, $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \models \bot$. By Lemma 8.2, we have $\varphi(\mathbf{x}) \wedge \alpha \models \bot$. If γ satisfies (5) for the formula φ, it satisfies (5) also for the formula $\varphi \wedge \alpha$. It follows that Algorithm 1 detects unsatisfiability of $\varphi \wedge \alpha$ using the valuation γ assumed for φ. Let $\alpha_1 = \alpha \cap \mathrm{lit}(\mathbf{x}_1)$ and $\alpha_2 = \alpha \cap \mathrm{lit}(\mathbf{x}_2)$.

If $\varphi_1 \wedge \alpha_1 \vdash_1 \bot$, then also $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 \bot$ since $\varphi_1 \wedge \alpha_1 \subseteq \psi \wedge \alpha$ due to clauses in group (q1). Now assume $\varphi_1 \wedge \alpha_1 \nvdash_1 \bot$. Consider the partial assignments β and δ used to obtain θ in Algorithm 1. By construction, $\beta \supseteq \alpha_1$. Since δ is an autark assignment for the formula $(\varphi \wedge \alpha)(\beta) = \varphi(\beta) \wedge \alpha_2$ and satisfies all clauses of $\varphi_1(\beta)$, we have $\theta = \varphi_2(\beta \wedge \delta) \wedge \alpha_2$ and $\theta \models \bot$. We claim that for any clause $u \vee v \in \theta$ we have $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 [\![u \vee v]\!]$. Since $u \vee v \in \theta$, there must be a clause $C = \neg x_{i_1} \vee \cdots \vee \neg x_{i_k} \vee u \vee v$ in φ_2 and $\varphi_1 \wedge \alpha_1 \vdash_1 x_{i_j}$ for every $j = 1, \ldots, k$. Using clauses of group (q1) we derive that also $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 x_{i_j}$ for every $j = 1, \ldots, k$ and then using a clause of group (q2) corresponding to C we get $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 [\![u \vee v]\!]$. Similarly, using clauses of group (q1) in φ_2, we obtain $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 u$ for any unit clause $u \in \theta$. Unit clauses in α_2 are contained in $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha$ directly. Since θ is an unsatisfiable 2-CNF formula not containing the empty clause, there is a literal $u \in \mathrm{lit}(\mathbf{x}_2)$, such that unit clauses u and $\neg u$ can be derived by resolution from θ. As explained above, clauses of groups (q3) to (q5) allow to simulate resolution on unit and binary clauses using unit propagation. Hence, we have $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 u$ and $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 \neg u$ which implies $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 \bot$. □

The next case shows that we can allow positive occurrences of variables from $\mathbf{y}$ in the partial assignment α.

Lemma 8.4. *Let $\alpha \subseteq \mathrm{lit}(\mathbf{x}) \cup \mathbf{y}$ be a partial assignment. If $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \models \bot$, then we have $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \vdash_1 \bot$.*

Proof. Assume $\psi(\mathbf{x}, \mathbf{y}) \wedge \alpha \models \bot$. Let us split α into two partial assignments $\alpha_x = \alpha \cap \mathrm{lit}(\mathbf{x})$ and $\alpha_y = \alpha \cap \mathbf{y}$, where α_y represents a set of binary clauses. The formula

$$\varphi'(\mathbf{x}) = \varphi(\mathbf{x}) \wedge \bigwedge_{[\![C]\!] \in \alpha_y} C$$

is q-Horn using the valuation γ, since we add to φ binary clauses on the variables $\mathbf{x}_2$. The encoding of $\varphi'(\mathbf{x})$ constructed according to Table 1 is $\psi'(\mathbf{x},\mathbf{y}) = \psi(\mathbf{x},\mathbf{y}) \wedge \alpha_y$. Since $\alpha = \alpha_x \wedge \alpha_y$, we have $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha = \psi'(\mathbf{x},\mathbf{y}) \wedge \alpha_x$ and thus $\psi'(\mathbf{x},\mathbf{y}) \wedge \alpha_x \models \bot$. By Lemma 8.3 we obtain $\psi'(\mathbf{x},\mathbf{y}) \wedge \alpha_x \vdash_1 \bot$. This is equivalent to $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha \vdash_1 \bot$. □

In the last case, we consider α without restrictions.

Lemma 8.5. *Let $\alpha \subseteq \mathrm{lit}(\mathbf{x} \cup \mathbf{y})$ be a partial assignment. If $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha \models \bot$, then we have $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha \vdash_1 \bot$.*

Proof. Assume $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha \models \bot$. Let us split α into three partial assignments $\alpha_x = \alpha \cap \mathrm{lit}(\mathbf{x})$, $\alpha_y = \alpha \cap \mathbf{y}$, and $\alpha_{\overline{y}} = \alpha \cap \{\neg y \mid y \in \mathbf{y}\}$. Moreover, let $\alpha'_{\overline{y}} = \{\neg u, \neg v \mid \neg[\![u \vee v]\!] \in \alpha_{\overline{y}}\}$. Since $\psi(\mathbf{x},\mathbf{y}) \models u \vee v \Leftrightarrow [\![u \vee v]\!]$, we have $\psi(\mathbf{x},\mathbf{y}) \models \alpha_{\overline{y}} \Leftrightarrow \alpha'_{\overline{y}}$. It follows that

$$\psi(\mathbf{x},\mathbf{y}) \wedge \alpha_x \wedge \alpha_y \wedge \alpha'_{\overline{y}} \models \bot\,.$$

Since the left-hand side satisfies the assumption of Lemma 8.4, we get $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha_x \wedge \alpha_y \wedge \alpha'_{\overline{y}} \vdash_1 \bot$. Using clauses of group (q6), this implies $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha_x \wedge \alpha_y \wedge \alpha_{\overline{y}} \vdash_1 \bot$ and thus $\psi(\mathbf{x},\mathbf{y}) \wedge \alpha \vdash_1 \bot$ as required. □

The main result of this section is the following statement, where the valuation γ is an arbitrary valuation with values $\{0, 1/2, 1\}$.

Theorem (Theorem 4.6)**.** *Assume that $\varphi(\mathbf{x})$ is a q-Horn formula with a valuation γ representing a q-Horn function $f(\mathbf{x})$. Moreover, assume a partition of the variables $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2$, such that $\gamma(x) \in \{0,1\}$ for every $x \in \mathbf{x}_1$ and $\gamma(x) = 1/2$ for every $x \in \mathbf{x}_2$. Then there is a URC encoding $\psi(\mathbf{x},\mathbf{y})$ of $f(\mathbf{x})$ satisfying $|\mathbf{y}| = O(|\mathbf{x}_2|^2)$, $|\psi| = O(|\varphi| + |\mathbf{x}_2|^3)$, and $\|\psi\| = O(\|\varphi\| + |\mathbf{x}_2|^3)$.*

Proof. Transform the variables of φ as described at the beginning of this section to obtain a q-Horn formula φ' with valuation γ' with values $\{1/2, 1\}$ and construct its encoding according to Table 1. Using lemmas 8.2 and 8.5, this is a URC encoding of φ'. By reverting the transformation of the variables, we obtain a URC encoding of φ. The size estimates follow directly from the construction described in Table 1. □

Let us point out that the encoding described in Table 1 is not q-Horn in general. If the groups (q5) and (q6) of clauses are present in the encoding for some literals u, v, they contain clauses $\neg[\![u \vee v]\!] \vee v \vee u$, $\neg u \vee [\![u \vee v]\!]$, $\neg v \vee [\![u \vee v]\!]$. One can verify that a valuation γ satisfying (5) for these clauses has to satisfy $\gamma(u) = \gamma(v) = \gamma([\![u \vee v]\!]) = 0$. The variable $[\![u \vee v]\!]$ is included in the encoding, if the original q-Horn formula requires $\gamma(u) = \gamma(v) = 1/2$. In this case, the system of inequalitites (5) for the CNF encoding in Table 1 is inconsistent.

9. Conclusion and Further Research

We presented an exponential separation between the sizes of PC and URC encodings without auxiliary variables and strengthened the known results on their relationship to the PC and URC encodings that can use auxiliary variables. Moreover, we demonstrated a difference in the properties of irredundant URC and PC formulas which can have applications to their use in knowledge compilation. The methods for compilation not introducing

new auxiliary variables are investigated also in a related ongoing research using a program pccompile (Kučera, 2020). One of the algorithms implemented in this program uses the correspondence between dual rail encoding of a PC formula and a specific Horn function presented in Section 7 in this paper.

By the results of Section 5, there is no guarantee for the existence of a reasonably sized PC formula equivalent to a given CNF, even if it belongs to a tractable class of Horn or q-Horn formulas. On the other hand, preliminary experiments (Kučera, 2019) with the program pccompile demonstrate that a compilation into a PC formula not introducing new auxiliary variables is frequently tractable for encodings of a few hundreds of clauses that appear as benchmarks in knowledge compilation. Further research is needed to clarify the conditions under which such a compilation is tractable.

Let us close the paper with several questions left open for further research. In Theorem 4.4 we have shown that if $\varphi_1(\mathbf{x})$ and $\varphi_2(\mathbf{x})$ are two PC-irredundant formulas representing the same function $f(\mathbf{x})$ on $n = |\mathbf{x}|$ variables, then $|\varphi_2| \leq n^2|\varphi_1|$. We are not aware of a function for which the factor n^2 is needed in the upper bound, however, it is open, whether the bound can be strengthened in the following sense.

Question 1. *Is it possible to strengthen the bound from Theorem 4.4 to $|\varphi_2| = O(n|\varphi_1|)$?*

One of the relationships depicted in Figure 1 is that the language of URC encodings is at least as succinct as the language of PC encodings. The following related question is open

Question 2. *Is the language of PC encodings at least as succinct as the language of URC encodings?*

This question asks for a construction related to the polynomial construction of a propagator from a consistency checker described by Bessiere et al. (2009). Question 2 is different in that both the input and output encodings are required to satisfy a specified propagation property on all variables, not only on the main ones.

By Theorem 4.6 there is a polynomial size URC encoding for an arbitrary q-Horn formula. It is natural to ask whether we can in fact construct a PC encoding of polynomial size. This question is open already for the class of Horn formulas contained in the class of q-Horn formulas and we can thus pose the following question.

Question 3. *Let $\varphi(\mathbf{x})$ be a Horn formula or, more generally, a q-Horn formula. Is there a PC encoding $\psi(\mathbf{x}, \mathbf{y})$ of φ of size polynomial in the size of φ?*

Using the notation from Theorem 4.6, the size of the URC encoding constructed for a q-Horn formula $\varphi(\mathbf{x}_1, \mathbf{x}_2)$ is $O(|\varphi| + |\mathbf{x}_2|^3)$.

Question 4. *Is there a URC encoding for every q-Horn formula φ of size $O(|\varphi| + |\mathbf{x}_2|^c)$, where $c < 3$?*

Acknowledgments

Both authors gratefully acknowledge the support by Grant Agency of the Czech Republic (grant No. GA19–19463S).

References

Abío, I., Gange, G., Mayer-Eichberger, V., & Stuckey, P. J. (2016). On CNF encodings of decision diagrams. In Quimper, C.-G. (Ed.), *Integration of AI and OR Techniques in Constraint Programming*, pp. 1–17, Cham. Springer International Publishing.

Arias, M., Balcázar, J. L., & Tîrnăucă, C. (2017). Learning definite Horn formulas from closure queries. *Theoretical Computer Science*, *658*, 346–356. Horn formulas, directed hypergraphs, lattices and closure systems: related formalism and application.

Aspvall, B., Plass, M. F., & Tarjan, R. E. (1979). A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, *8*(3), 121–123.

Atserias, A., Fichte, J. K., & Thurley, M. (2011). Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Int. Res.*, *40*(1), 353–373.

Babka, M., Balyo, T., Čepek, O., Gurský, Š., Kučera, P., & Vlček, V. (2013). Complexity issues related to propagation completeness. *Artificial Intelligence*, *203*, 19–34.

Bacchus, F. (2007). GAC via unit propagation. In Bessière, C. (Ed.), *Principles and Practice of Constraint Programming – CP 2007*, Vol. 4741 of *Lecture Notes in Computer Science*, pp. 133–147. Springer Berlin Heidelberg.

Bessiere, C. (2006). Constraint propagation. In Rossi, F., van Beek, P., & Walsh, T. (Eds.), *Handbook of Constraint Programming*, chap. 3. Elsevier Science Inc., USA.

Bessiere, C., Katsirelos, G., Narodytska, N., & Walsh, T. (2009). Circuit complexity and decompositions of global constraints. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 412–418.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (2009). *Handbook of Satisfiability*, Vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands.

Bonet, M. L., Buss, S., Ignatiev, A., Marques-Silva, J., & Morgado, A. (2018). MaxSAT resolution with the dual rail encoding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bordeaux, L., Janota, M., Marques-Silva, J., & Marquis, P. (2012). 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, pp. 75–84. AAAI Press.

Bordeaux, L., & Marques-Silva, J. (2012). Knowledge compilation with empowerment. In Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., & Turán, G. (Eds.), *SOFSEM 2012: Theory and Practice of Computer Science*, Vol. 7147 of *Lecture Notes in Computer Science*, pp. 612–624. Springer Berlin / Heidelberg.

Boros, E., Crama, Y., & Hammer, P. L. (1990). Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, *1*(1), 21–32.

Boros, E., Hammer, P. L., & Sun, X. (1994). Recognition of q-Horn formulae in linear time. *Discrete Applied Mathematics*, *55*(1), 1–13.

Bova, S., Capelli, F., Mengel, S., & Slivovsky, F. (2014). A strongly exponential separation of DNNFs from CNF formulas. arXiv: 1411.1995 [cs.CC]. `https://arxiv.org/abs/1411.1995`.

Bova, S., Capelli, F., Mengel, S., & Slivovsky, F. (2016). Knowledge compilation meets communication complexity. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 1008–1014. AAAI Press.

Bryant, R. E., Beatty, D., Brace, K., Cho, K., & Sheffler, T. (1987). COSMOS: A compiled simulator for MOS circuits. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, DAC '87, p. 9–16, New York, NY, USA. Association for Computing Machinery.

Čepek, O., Kučera, P., & Vlček, V. (2012). Properties of SLUR formulae. In Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., & Turán, G. (Eds.), *SOFSEM 2012: Theory and Practice of Computer Science*, Vol. 7147 of *LNCS*, pp. 177–189. Springer Berlin Heidelberg.

Darwiche, A. (1999). Compiling knowledge into decomposable negation normal form. In *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'99, pp. 284–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

del Val, A. (1994). Tractable databases: How to make propositional unit resolution complete through compilation. In *Knowledge Representation and Reasoning*, pp. 551–561.

Gange, G., & Stuckey, P. J. (2012). Explaining propagators for s-DNNF circuits. In Beldiceanu, N., Jussien, N., & Pinson, É. (Eds.), *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, pp. 195–210. Springer Berlin Heidelberg.

Gogic, G., Kautz, H., Papadimitriou, C., & Selman, B. (1995). The comparative linguistics of knowledge representation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pp. 862–869, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guigues, J.-L., & Duquenne, V. (1986). Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences humaines*, *95*, 5–18.

Gwynne, M., & Kullmann, O. (2013). Generalising and unifying SLUR and unit-refutation completeness. In van Emde Boas, P., Groen, F. C. A., Italiano, G. F., Nawrocki, J., & Sack, H. (Eds.), *SOFSEM 2013: Theory and Practice of Computer Science*, pp. 220–232, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hammer, P., & Kogan, A. (1993). Optimal compression of propositional Horn knowledge bases: Complexity and approximation. *Artificial Intelligence*, *64*, 131–145.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2017). On tackling the limits of resolution in SAT solving. In Gaspers, S., & Walsh, T. (Eds.), *Theory and Applications of Satisfiability Testing – SAT 2017*, pp. 164–183, Cham. Springer International Publishing.

Jung, J. C., Barahona, P., Katsirelos, G., & Walsh, T. (2008). Two Encodings of DNNF Theories. In *ECAI'08 Workshop on Inference methods based on Graphical Structures of Knowledge*.

Kučera, P. (2019). Tool presentation: pccompile. Presented at the KOCOON Workshop in Arras. `http://kocoon.gforge.inria.fr/slides/pccompile.pdf`, Accessed: 2020-12-21.

Kučera, P. (2020). Program pccompile. `http://ktiml.mff.cuni.cz/~kucerap/pccompile`. Accessed: 2020-12-21.

Kučera, P., & Savický, P. (2019). Propagation complete encodings of smooth DNNF theories. arXiv: 1909.06673 [cs.AI]. `https://arxiv.org/abs/1909.06673`.

Kullmann, O. (2000). Investigations on autark assignments. *Discrete Applied Mathematics*, *107*(1–3), 99–137.

Manquinho, V. M., Flores, P. F., Silva, J. P. M., & Oliveira, A. L. (1997). Prime implicant computation using satisfiability algorithms. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, pp. 232–239.

Morgado, A., Ignatiev, A., Bonet, M. L., Marques-Silva, J., & Buss, S. (2019). DRMaxSAT with MaxHS: First contact. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 239–249. Springer.

Pipatsrisawat, K., & Darwiche, A. (2011). On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, *175*(2), 512–525.

Schlipf, J. S., Annexstein, F. S., Franco, J. V., & Swaminathan, R. P. (1995). On finding solutions for extended Horn formulas. *Inf. Process. Lett.*, *54*(3), 133–137.

Selman, B., & Kautz, H. (1996). Knowledge compilation and theory approximation. *Journal of the ACM (JACM)*, *43*(2), 193–224.