Pseudo-Boolean Constraints from a Knowledge Representation Perspective

Daniel Le Berre[1,2], **Pierre Marquis**[1,2,3], **Stefan Mengel**[1], **Romain Wallon**[1,2]
[1]CRIL-CNRS UMR 8188, Lens, France
[2]Université d'Artois
[3] Institut Universitaire de France
leberre@cril.fr, marquis@cril.fr, mengel@cril.fr, wallon@cril.fr

Abstract

We study *pseudo-Boolean constraints* (PBC) and their special case *cardinality constraints* (CARD) from the perspective of knowledge representation. To this end, the succinctness of PBC and CARD is compared to that of many standard propositional languages. Moreover, we determine which queries and transformations are feasible in polynomial time when knowledge is represented by PBC or CARD, and which are not (unconditionally or unless P = NP). In particular, the advantages and disadvantages compared to CNF are discussed.

1 Introduction

Many applications in AI are based on propositional information. When dealing with such applications, an important issue is thus choosing a representation from a given *language*, for the information that is suited to the way that the information is processed [Gogic *et al.*, 1995]. However there exist dozens of languages for representing propositional information.

The large number of candidate languages is explained by the fact that there is no single language that is the best for all potential tasks to be achieved on propositional information. Indeed, the "right" choice typically depends on the way the information is primarily reported and on the efficiency of the computations which must be made on it for the application at hand. Thus, one needs to consider selection criteria to make an informed choice [Darwiche and Marquis, 2002].

In many cases, propositional information represents laws or constraints which must be aggregated conjunctively. In such cases, the language CNF consisting of conjunctions of propositional clauses is often used. Thus, many benchmarks corresponding to various AI applications (and many applications outside the AI area as well) are encoded as CNF formulae, most often in the standard DIMACS format [DIMACS, 1993]. The focus on CNF is justified by the fact that clauses are basic objects, which can be interpreted as very simple if-then rules, and encoded and handled in a straightforward way (as lists of literals). Moreover, there is a rich ecosystem around CNF formulae with preprocessors, solvers and benchmark instances from many fields, see e.g. [SATLIB, 1999; Biere *et al.*, 2009]. However, CNF suffers from some weaknesses. Clauses are somewhat too simple to express some important constraints. For instance, it is shown in [Dixon, 2004] that stating that, in a given set of m variables, at least n of them must be set to true cannot be carried out using a number of clauses that is polynomial in n in general without adding any new variables, which is undesirable in some settings (see the conclusion of the paper). In order to encode such constraints in a compact way, generalizations of the clause representation have been considered, in particular the so-called *pseudo-Boolean constraints* (linear equations or inequalities over literals), strongly related to *threshold functions* [Crama and Hammer, 2011], and their special case *cardinality constraints* (pseudo-Boolean constraints where each literal has a coefficient equal to one). These two constraints regularly arise naturally in many settings. Obviously enough, any clause $\bigvee_{i=1}^{k} l_i$ can be turned in linear time into an equivalent cardinality constraint of the form $\sum_{i=1}^{k} l_i \geq 1$ so all information represented by a CNF formula can easily be transformed into a cardinality constraint representation.

Pseudo-Boolean constraints have also attracted attention due to associated proof systems, like the cutting planes system, that have been introduced for proving the inconsistency of a conjunction of such constraints [Cook *et al.*, 1987; Hooker, 1988; Nordström, 2015]. The cutting planes proof system is known to p-simulate the well-known resolution proof system, the underlying of modern SAT solvers, which means that there exists a fixed polynomial p such that every refutation of size l in the latter proof system can be turned into a refutation of size $\leq p(l)$ in the former proof system. However, the converse does not hold: there are families of inconsistent CNF formulae over n variables for which any resolution proof of a contradiction is of size exponential in n, while when the formulae are encoded in a natural way as pseudo-Boolean constraints, cutting planes refutations of size linear in n exist. This is the case for the family of inconsistent pigeonhole instances [Haken, 1985; Krishnamurthy, 1985]. This explains why several dedicated solvers that aim to take advantage of the strength of the cutting planes system have been implemented [Barth, 1995; Dixon and Ginsberg, 2002; Chai and Kuehlmann, 2003; Sheini and Sakallah, 2006; Le Berre and Parrain, 2010].

In this paper, we do not consider issues about proof systems but instead analyse the properties of the language PBC of pseudo-Boolean constraints and those of its subset, the language CARD of cardinality constraints, concerning the representation of propositional information. In this direction, several questions have to be considered concerning the spatial efficiency of the two languages (i.e., their ability to represent propositional information using little space) as well as their temporal efficiency (i.e., the ability to exhibit a number of queries and transformations which are tractable when the information used is encoded in the language). Of specific interest are the advantages and disadvantages of PBC and CARD compared to CNF. For instance, it is known that the language CNF is not more succinct than the language FBDD of free binary decision diagrams. Does this also hold for PBC or CARD? It is also known that deciding whether a consistent conjunction of literals implies a CNF formula can be done in polynomial time. Does this still hold for PBC or CARD representations? Addressing those issues in a principled way amounts to considering the criteria used in the knowledge compilation map for comparing target compilation languages [Darwiche and Marquis, 2002]. Though neither PBC nor CARD can be considered as compilation languages (since none of them offers a polynomial-time algorithm for deciding consistency), we argue that the criteria (succinctness, polynomial-time queries and transformations) used to compare compilation languages are meaningful for other propositional languages since they provide a systematic approach to the questions above.

Thus, in the following, the relative succinctness of PBC and CARD compared to many standard propositional languages (CNF, NNF, OBDD, etc.) is determined. Then, we examine whether the queries and transformations which are considered in the knowledge compilation map are supported efficiently by PBC and CARD or not (which amounts to deciding whether a polynomial-time algorithm for carrying out the query or the transformation exists or not – unconditionally or unless P = NP). Our results show that, on the one hand, despite the extra succinctness of CARD and PBC compared to CNF, their relation to standard propositional languages like DNF, OBDD, FBDD is the same: those languages which are incomparable to CNF w.r.t. succinctness are also incomparable to PBC w.r.t. succinctness. On the other hand, the improvement w.r.t. spatial efficiency which results from such a switch does not imply the loss of many tractable queries or transformations: all the queries and all the transformations, offered by CNF except SFO and $\vee$BC are preserved.

2 Preliminaries

We consider a propositional setting based on a finite set of propositional variables. A *literal* is a variable or its negation and a *clause* is a disjunction of literals. All the representations considered in the following are interpreted in the classical way, so that the notions of consistency, validity, entailment, equivalence are the standard ones. $\models$ denotes entailment and $\equiv$ denotes equivalence. The size of a representation φ, denoted $|\varphi|$, is the number of elementary symbols used in φ (integers are represented in binary notation).

2.1 Pseudo-Boolean Constraints

Among all the propositional representations considered in the following we are specifically interested in pseudo-Boolean constraints.

Definition 2.1 (Pseudo-Boolean constraint)**.** *A* pseudo-Boolean constraint *is of the form* $\sum_{i=1}^{n} a_i l_i \ \triangle\ k$*, where* n *is some non-negative integer,* $\forall i \in \{1,\dots,n\}$*,* $a_i \in \mathbb{Z}$*,* l_i *is a literal,* $\triangle \in \{<,\leq,=,\geq,>\}$ *and* $k \in \mathbb{Z}$*. Each* a_i *is called a* weight *and* k *is called the* degree *of the constraint.*

A *normalized pseudo-Boolean constraint* is a pseudo-Boolean constraint of the form $\sum_{i=1}^{n} a_i l_i \geq k$, where $\forall i \in \{1,\dots,n\}$, $a_i \in \mathbb{N}$, $k \in \mathbb{N}$ and each variable appears only once in the constraint, either positively or negatively. From now on, we assume that all the pseudo-Boolean constraints under consideration are normalized. This assumption is harmless, computationally speaking:

Proposition 2.2 ([Barth, 1995; Roussel and Manquinho, 2009])**.** *Any pseudo-Boolean constraint can be turned in polynomial time into an equivalent, normalized pseudo-Boolean constraint, or a conjunction of such constraints.*

In the following, we also focus on a specific family of pseudo-Boolean constraints, consisting of cardinality constraints. A *cardinality constraint* is a normalized pseudo-Boolean constraint of the form $\sum_{i=1}^{n} l_i \geq k$ where n is some non-negative integer.

Definition 2.3 (PBC, CARD)**.** *PBC (resp. CARD) is the language of conjunctions of normalized pseudo-Boolean constraints (resp. cardinality constraints).*

In this paper, we use the following inference rules to reason with pseudo-Boolean constraints:

- **Saturation.** Let κ be a (normalized) pseudo-Boolean constraint $\kappa = \sum_{i=1}^{n} a_i l_i \geq k$ where for some i_0 in $\{1,\dots,n\}$, $a_{i_0} > k$. Then, a_{i_0} can be replaced by k without changing the models of the constraint.
- **Division.** Let κ be a (normalized) pseudo-Boolean constraint $\kappa = \sum_{i=1}^{n} a_i l_i \geq k$ and an integer $\alpha > 0$. κ entails the constraint $\sum_{i=1}^{n} \lceil \frac{a_i}{\alpha} \rceil l_i \geq \lceil \frac{k}{\alpha} \rceil$, where $\lceil x \rceil$ denotes the smallest integer greater or equal to x. Moreover, if α is a divisor of each a_i, then κ is equivalent to the latter constraint.

Note that, in the division rule, we do not require k to be divisible by α. This property allows the proof system to be more powerful than resolution [Cook *et al.*, 1987]. For example, the constraint $2x + 2y + 2z \geq 3$ becomes $x + y + z \geq 2$, which eliminates some non-integral solutions.

2.2 Knowledge Compilation Map

The knowledge compilation map reported in [Darwiche and Marquis, 2002] is a systematic, multi-criteria comparative analysis of propositional representation languages. The choice of a language L is then guided by its spatial efficiency (its succinctness) and its temporal efficiency (the queries and transformations it offers, viewed as properties of L).

Succinctness. Succinctness captures the (relative) ability of a language to represent information using little space, and yields a pre-order $\leq_s$. A propositional language L_1 is said to be *at least as succinct as* another language L_2, denoted

$\mathsf{L}_1 \leq_s \mathsf{L}_2$, if and only if there exists a polynomial p such that for every formula $\alpha \in \mathsf{L}_2$, there exists an equivalent formula $\beta \in \mathsf{L}_1$ where $|\beta| \leq p(|\alpha|)$. When $\mathsf{L}_1 \leq_s \mathsf{L}_2$ and $\mathsf{L}_2 \not\leq_s \mathsf{L}_1$, we write $\mathsf{L}_1 <_s \mathsf{L}_2$.

Queries. A *query* takes as input one or more representations from L, and some other inputs can be also required, such as clauses, terms, etc. It typically returns a Boolean value or a number. A language L satisfies a given query when a polynomial-time algorithm exists for answering the query, i.e., returning the expected result.

For example, let us consider clausal entailment, a query which returns a Boolean value stating whether or not a given clause is entailed by a representation from L:

Definition 2.4 (CE). *L satisfies CE (Clausal Entailment) if and only if there exists a polynomial-time algorithm that decides whether $\varphi \models \gamma$ where $\varphi \in \mathsf{L}$ and γ is a clause.*

Other queries, such as CO (consistency), VA (validity), IM (implication by a term), EQ (equivalence), SE (sentential entailment), CT (model counting) and ME (model enumeration), that have been considered in the knowledge compilation map, are also taken into account in the following. For space reasons, the reader is referred to [Darwiche and Marquis, 2002] for some formal definitions.

Transformations. A *transformation* takes as input one or more representations from L, and some other inputs can be also required, such as variables, terms, etc. The result is a representation from L.

For example, let us consider the closures under $\wedge$ and $\vee$:

Definition 2.5 ($\wedge$C, $\vee$C). *L satisfies $\wedge$C (resp. $\vee$C) if and only if there exists a polynomial-time algorithm which computes a representation from L that is equivalent to $\varphi_1 \wedge \cdots \wedge \varphi_n$ (resp. $\varphi_1 \vee \cdots \vee \varphi_n$) given a set of formulae $\varphi_1, \ldots, \varphi_n$ from L as input.*

The other transformations considered in the knowledge compilation map, namely CD (conditioning), FO (forgetting), SFO (singleton forgetting), $\wedge$BC (bounded closure under $\wedge$), $\vee$BC (bounded closure under $\vee$) and $\neg$C (closure under $\neg$), are also considered in the following. Again, formal definitions can be found in [Darwiche and Marquis, 2002].

3 Succinctness of Pseudo-Boolean Constraints

In this section, PBC and CARD are compared in terms of succinctness with many languages considered in the knowledge compilation map, namely NNF, CNF, DNF, IP, DNNF, FBDD, OBDD, $\mathsf{OBDD}_<$, and MODS. Special attention is devoted to the language of (DAG-based) Negation Normal Form representations (NNF) [Darwiche, 1999; 2001], the language of *Ordered Binary Decision Diagram* ($\mathsf{OBDD}_<$) [Bryant, 1986], and the language of *Prime Implicants* (IP) [Quine, 1952] since all the new succinctness results which have been identified actually are by-products of the succinctness results involving PBC, CARD and those three languages. The results obtained are summarized by the diagram in Figure 1.

Let us now detail the claimed results:

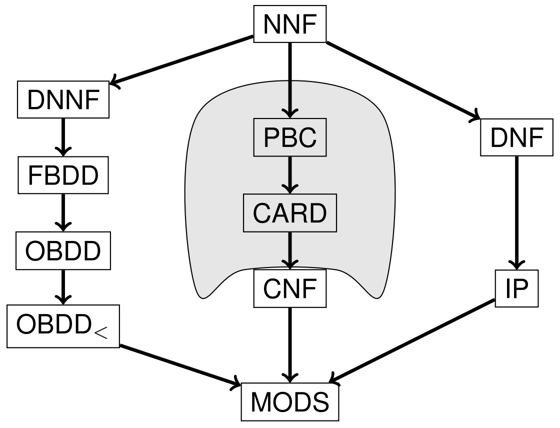

Figure 1: Succinctness of various propositional languages. In this diagram, an arrow $\mathsf{L}_1 \rightarrow \mathsf{L}_2$ means that L_1 is strictly more succinct than L_2, i.e. $\mathsf{L}_1 \leq_s \mathsf{L}_2$ and $\mathsf{L}_2 \not\leq_s \mathsf{L}_1$. No arrow between two languages means that they are incomparable. The grayed area highlights the results we claim in this paper (including incomparability).

Proposition 3.1. $\mathsf{PBC} <_s \mathsf{CARD}$.

Proof. First, since CARD is a subset of PBC, we get that $\mathsf{PBC} \leq_s \mathsf{CARD}$.

Now, let us prove that $\mathsf{CARD} \not\leq_s \mathsf{PBC}$. Let κ be the pseudo-Boolean constraint defined by $\kappa = kx + \sum_{i=1}^{2k} x_i \geq k$. Let $\kappa' = \sum_{i=1}^{m} l_i \geq k'$ be a non-valid pseudo-Boolean constraint such that $\kappa \models \kappa'$, and $\mathrm{Var}(\kappa') \subseteq \mathrm{Var}(\kappa)$. We first show that, necessarily, $k' = 1$, i.e., κ' is a clause.

Let us suppose that $k' > 1$, and let us consider M the model of κ such that x is satisfied, and any literal which is in κ' is falsified, except x. In this interpretation, we necessarily have $\sum_{i=1}^{m} l_i \leq 1$, since only x is satisfied by construction. Indeed, if x positively appears in κ', then $\sum_{i=1}^{m} l_i = 1$. Otherwise, $\sum_{i=1}^{m} l_i = 0$. In both cases, since we have assumed $k' > 1$, κ' is not satisfied, so M is not a model of κ'. This contradicts that $\kappa \models \kappa'$, so $k' \leq 1$. Since κ' is supposed to be non-valid, k' cannot be equal to 0, so $k' = 1$.

The only way to represent κ as a conjunction of cardinality constraints is then to use clauses, since all these constraints must be implied by κ. However, thanks to Proposition 4.1.4 from [Dixon, 2004], we have that an exponential number of clauses is required to represent κ, and this completes the proof. □

Proposition 3.2. $\mathsf{NNF} \leq_s \mathsf{PBC}$.

Sketch of Proof. Let $\kappa = \sum_{i=1}^{n} a_i l_i \geq k$ be a pseudo-Boolean constraint. It is well known that any pseudo-Boolean constraint $\sum_{i=1}^{n} a_i l_i \geq k$ can be represented by an equivalent NNF representation of polynomial size (see e.g. [Vollmer, 1999]). Now, when a conjunction of pseudo-Boolean constraints is considered, an NNF representation of it consists of the conjunction of all these "elementary" NNF representations – one per pseudo-Boolean constraint. □

As an immediate consequence of [Dixon, 2004] (Proposition 4.1.4), we have the following result:

Proposition 3.3. CARD $<_s$ CNF

Let us now show that PBC is not at least as succinct as OBDD$_<$.

Proposition 3.4. PBC $\not\leq_s$ OBDD$_<$

Proof. Let $\varphi = \bigoplus_{j=1}^{n} x_j$. φ is true if and only if there is an odd number of satisfied x_j.

This formula can be written as an OBDD$_<$ representation of polynomial size [Bryant, 1986]. Let us prove that every representation of φ as a conjunction of pseudo-Boolean constraints requires an exponential number of constraints.

Let $\kappa = \sum_{i=1}^{m} a_i l_i \geq k$ be a non-valid pseudo-Boolean constraint, such that $\varphi \models \kappa$, with $\mathrm{Var}(\kappa) \subseteq \mathrm{Var}(\varphi)$. Let us note L^+ and L^- the sets of positive and negative literals of κ, respectively.

Claim 3.5. *We have* $\mathrm{Var}(\kappa) = \mathrm{Var}(\varphi)$.

Sketch of Proof. If there is $x \in \mathrm{Var}(\varphi)$ such that $x \notin \mathrm{Var}(\kappa)$, then any counter-model of κ can be turned into a model M of φ by keeping the same assignment, except for x, which must be satisfied. M is still a counter-model of κ, which contradicts $\varphi \models \kappa$. □

Claim 3.6. $|L^-|$ *is even.*

Proof. Let us suppose that $|L^-|$ is odd. Let M be the interpretation given by $M = \{x_j | \overline{x_j} \in L^-\}$. M satisfies an odd number of x_j, so it is a model of φ. However, M does not satisfy any literal of κ, so it is not a model of κ. This contradicts $\varphi \models \kappa$, so $|L^-|$ is even. □

Claim 3.7. *We have* $\forall i \in \{1, \ldots, m\}, a_i = k$.

Proof. With the saturation rule, one can ensure that $a_i \leq k$ for all $i \in \{1, \ldots, m\}$. Let us assume that $\exists i_0 \in \{1, \ldots, m\}, a_{i_0} < k$ and that the variable associated with l_{i_0} is x_{j_0}. There are two possible cases: $x_{j_0} \in L^+$ or $\overline{x_{j_0}} \in L^-$. In the case when $x_{j_0} \in L^+$, let us consider the interpretation given by $M = \{x_{j_0}\} \cup \{x_j | \overline{x_j} \in L^-\}$. By Claim 3.6, since $|L^-|$ is even, $|M| = |L^-| + 1$ is odd, and M is a model of φ. In the remaining case, $\overline{x_{j_0}} \in L^-$, and then let us consider the interpretation given by $M = \{x_j | \overline{x_j} \in L^-\} \backslash \{x_{j_0}\}$. Since $|L^-|$ is even, $|M| = |L^-| - 1$ is odd, and M is a model of φ. In both cases, $\sum_{i=1}^{m} a_i l_i = a_{i_0} < k$, so M is not a model of κ. This contradicts $\varphi \models \kappa$, so $\forall i \in \{1, \ldots, m\}, a_i = k$. □

So, $\kappa = \sum_{i=1}^{m} k l_i \geq k$, and the division rule gives us that $\kappa \equiv \sum_{i=1}^{m} l_i \geq 1$, i.e., κ is equivalent to clause. Then, the only way to represent φ as a conjunction of pseudo-Boolean constraints is to use clauses, since all these constraints must be implied by φ. Since φ requires an exponential number of clauses to be represented without introducing new variables – all the clauses contain all the variables by Claim 3.5 – the claim follows. □

Proposition 3.8. PBC $\not\leq_s$ IP

Proof. Let $\varphi = \bigvee_{j=1}^{n} (x_j \wedge y_j)$ be a formula from IP [Darwiche and Marquis, 2002]. We now show that representing φ as a conjunction of pseudo-Boolean constraints requires an exponential number of constraints.

Let $\kappa = \sum_{i=1}^{m} a_i l_i \geq k$ be a pseudo-Boolean constraint such that $\varphi \models \kappa$, with $\mathrm{Var}(\kappa) \subseteq \mathrm{Var}(\varphi)$ and $k > 0$.

Claim 3.9. *For all* $j \in \{1, \ldots, n\}$, x_j *or* y_j *appears positively in* κ.

Sketch of Proof. If neither x_j nor y_j appear in $\mathrm{Var}(\kappa)$, then the model of φ satisfying x_j and y_j but falsifying all the literals of κ is not a model of κ, which contradicts $\varphi \models \kappa$. □

Claim 3.10. *For all* $j \in \{1, \ldots, n\}$, *if* x_j *appears positively in* κ, *and* y_j *does not, then the weight of* x_j *is* k. *Symmetrically, for all* $j \in \{1, \ldots, n\}$, *if* y_j *appears positively in* κ, *and* x_j *does not, then the weight of* y_j *is* k.

Sketch of Proof. If only x_j appears positively in κ, say $x_j = l_{j_0}$, and $a_{j_0} < k$, then the model of φ satisfying both x_j and y_j but falsifying all the other literals of κ is not a model of κ, which contradicts $\varphi \models \kappa$. The proof for the case when only y_j appears positively in κ is similar. □

Claim 3.11. *For all* $j \in \{1, \ldots, n\}$, *if* x_j *and* y_j *appear positively in* κ, *then the sum of their weights is at least equal to* k.

Sketch of Proof. If both x_j and y_j appear positively as l_{j_0} and l_{j_1} in κ, respectively, and $a_{j_0} + a_{j_1} < k$, then the model of φ satisfying these two literals but falsifying all the others of κ is not a model of κ, which contradicts $\varphi \models \kappa$. □

Claim 3.12. *Without loss of generality,* κ *can contain only positive literals.*

Sketch of Proof. If $\overline{y_j}$ appears in κ, let κ' be the pseudo-Boolean constraint obtained by removing the literal $\overline{y_j}$ from κ. The case where $\overline{x_j}$ appears in κ follows symmetrically.

Any model M of φ is a model of κ'. Indeed, if M satisfies y_j, then it is obviously a model of κ'. Otherwise, there exists $i \neq j$ such that M satisfies x_i and y_i, and by Claims 3.9, 3.10 and 3.11, κ' is also satisfied by M.

So, $\varphi \models \kappa'$, and since we obviously have that $\kappa' \models \kappa$, it is possible to replace κ by κ' without losing information. All negative literals can be removed by repeating the same argument. □

Let $K = \bigwedge_{j=1}^{m} \kappa_j$ be any conjunction of pseudo-Boolean constraints such that $\varphi \equiv K$. Thanks to the previous results, we have: $\forall j \in \{1, \ldots, m\}, \kappa_j = \sum_{i=1}^{n} a_{0,i,j} x_i + a_{1,i,j} y_i \geq k_j$, with, $\forall i \in \{1, \ldots, n\}, a_{0,i,j} + a_{1,i,j} \geq k_j$ by applying Claim 3.11.

Any counter-model of φ must be a counter-model of K. It remains to show, in the rest of this proof, that a subset of these counter-models requires the use of an exponential number of pseudo-Boolean constraints to get a formula logically equivalent to φ.

Any interpretation satisfying for all $i \in \{1, \ldots, n\}$ exactly one of x_i or y_i cannot be a model of K, since it is not a model of φ. Then, it must not satisfy at least one of the constraints

κ_j. Formally, this means that the following property must be satisfied.

Property ($*$). *For any function $f : \{1, \ldots, n\} \to \{0, 1\}$, there exists $j \in \{1, \ldots, m\}$ such that $\sum_{i=1}^{n} a_{f(i),i,j} < k_j$*

For each of the inequalities induced by ($*$), we define a tuple $(f(1), \ldots, f(n)) \in \mathbb{B}^n$. We claim that two inequalities associated with such tuples which have a Hamming distance at least equal to 2 cannot be satisfied for a same j. Otherwise, let $(f_1(1), \ldots, f_1(n))$ and $(f_2(1), \ldots, f_2(n))$ be two tuples with a Hamming distance at least equal to 2 for a same j. Their associated inequalities are (1) $\sum_{i=1}^{n} a_{f_1(i),i,j} < k_j$ and (2) $\sum_{i=1}^{n} a_{f_2(i),i,j} < k_j$. Since the Hamming distance between the two tuples is at least equal to 2, there exists $\alpha, \beta \in \{1, \ldots, n\}$, with $\alpha \neq \beta$, such that $f_1(\alpha) \neq f_2(\alpha)$ and $f_1(\beta) \neq f_2(\beta)$. By adding (1) and (2), since f_1 and f_2 take their values in $\{0, 1\}$, we get:

$$a_{0,\alpha,j} + a_{1,\alpha,j} + a_{0,\beta,j} + a_{1,\beta,j} \\ + \sum_{\substack{i=1 \\ i \notin \{\alpha,\beta\}}}^{n} (a_{f_1(i),i,j} + a_{f_2(i),i,j}) < 2k_j$$

However, this is impossible since Claim 3.11 gives us $a_{0,\alpha,j} + a_{1,\alpha,j} \geq k_j$ and $a_{0,\beta,j} + a_{1,\beta,j} \geq k_j$, so $a_{0,\alpha,j} + a_{1,\alpha,j} + a_{0,\beta,j} + a_{1,\beta,j} \geq 2k_j$. So, we need two distinct constraints to satisfy the inequalities of ($*$) associated with these tuples.

Claim 3.13. *There exists a set of 2^{n-1} tuples of $\mathbb{B}^n$ having pairwise Hamming distance at least 2.*

Sketch of Proof. Consider the set S of solutions over $\{0, 1\}$ of $\sum_{i=1}^{n} z_i$ is equivalent to 0 mod 2. The elements of S have pairwise Hamming distance at least 2. Finally, S contains 2^{n-1} elements since any assignment to $z_1, \ldots, z_{n-1}$ can be extended to a solution in S. □

Hence, to get satisfied, the system of inequalities associated with these tuples requires at least 2^{n-1} distinct constraints. So, representing φ as a conjunction of pseudo-Boolean constraints requires exponentially many constraints. □

The remaining succinctness results reported in Figure 1 are easy consequences of the previous one, taking advantage of the transitivity of $\leq_s$ and of the succinctness results given in [Darwiche and Marquis, 2002; Bova *et al.*, 2016].

4 Querying Pseudo-Boolean Constraints

We now present the results about the queries offered by PBC and CARD summarized in Table 1.

Proposition 4.1. *CARD does not satisfy CO, CE, EQ, SE, CT and ME, unless* P = NP.

Proof. Since the translation of any CNF formula into a conjunction of cardinality constraints can be done in polynomial time, and since CNF does not satisfy any of these properties, unless P = NP, the claim follows. □

	CO	VA	CE	IM	EQ	SE	CT	ME
CARD	○	✓	○	✓	○	○	○	○
PBC	○	✓	○	✓	○	○	○	○

Table 1: Properties of CARD and PBC about queries. A ✓ means that the query is offered by the language, and a ○ means that it is not the case, unless P = NP.

	CD	FO	SFO	∧C	∧BC	∨C	∨BC	¬C
CARD	✓	○	?	✓	✓	●	●	●
PBC	✓	●	●	✓	✓	●	●	●

Table 2: Properties of CARD and PBC about transformations. A ✓ means that the language offers the transformation, whereas a ○ means that it does not unless P = NP and a ● means that it does not unconditionally.

Since CARD is a sublanguage of PBC, we get:

Corollary 4.2. *PBC does not satisfy CO, CE, EQ, SE, CT and ME, unless* P = NP.

We also have that:

Proposition 4.3. *PBC and CARD satisfy VA.*

Proof. A conjunction of pseudo-Boolean constraints is valid if and only if all of its constraints are valid. This is the case if and only if all the constraints have a degree equal to 0, which can be checked in polynomial time. □

Since PBC and CARD satisfy VA and CD (cf. Section 5), the following corollary follows immediately by Lemma A.7 from [Darwiche and Marquis, 2002].

Corollary 4.4. *PBC and CARD satisfy IM.*

5 Transforming Pseudo-Boolean Constraints

Finally, we present the results we have obtained about the transformations offered by PBC and CARD. They are summarized in Table 2.

Proposition 5.1. *PBC and CARD satisfy CD.*

Proof. Computing the conditioning of any formula from PBC (resp. CARD) is easily done by replacing, in each constraint, every literal of the conditioning term by a Boolean constant. This produces, after a trivial normalization step made in polynomial time, a conjunction of pseudo-Boolean constraints (resp. cardinality constraints). □

Proposition 5.2. *CARD and PBC satisfy both* ∧*BC and* ∧*C.*

Proof. Given a conjunctively-interpreted set of formulae from CARD (resp. PBC), computing the conjunction of these formulae can trivially be done in polynomial time. □

Proposition 5.3. *CARD does not satisfy* ∨*BC.*

Proof. Let φ be the disjunction of the two cardinality constraints $\kappa = y \geq 1$ ($\kappa \equiv y$) and $\kappa' = \sum_{i=1}^{2n} x_i \geq n$. It is easy to see that φ is equivalent to $ny + \sum_{i=1}^{2n} x_i \geq n$. As shown in the proof of Proposition 3.1, a representation of this constraint using only cardinality constraints requires exponentially many constraints. Hence, the claim follows. □

Proposition 5.4. *PBC does not satisfy* $\vee BC$.

Proof. To prove this result, we show that any representation of the inequality $\Delta = \sum_{i=1}^{2n} x_i \neq n$, which is equivalent to the disjunction of $\sum_{i=1}^{2n} x_i \geq n+1$ and $\sum_{i=1}^{2n} \overline{x_i} \geq n+1$, requires an exponential number of pseudo-Boolean constraints when $n \geq 2$.

Let us consider a non-valid pseudo-Boolean constraint $\kappa = \sum_{i=1}^{m} a_i l_i \geq k$, such that $\Delta \models \kappa$, with $\mathrm{Var}(\kappa) \subseteq \mathrm{Var}(\Delta)$. Let us note L^+ and L^- the sets of positive and negative literals of κ, respectively.

Claim 5.5. *We have* $\mathrm{Var}(\kappa) = \mathrm{Var}(\Delta)$.

Sketch of Proof. Consider I a counter-model of κ. Since $\Delta \models \kappa$, I satisfies n of the x_i. If x in $\mathrm{Var}(\Delta)$ but $x \notin \mathrm{Var}(\kappa)$, then the interpretation I' defined by $I' = I \cup \{x\}$ satisfies $n+1$ of the x_i, so it is a model of Δ, but not a model of κ, which contradicts $\Delta \models \kappa$. □

Claim 5.6. *We have* $|L^+| = |L^-|$.

Proof. If $|L^+| < |L^-|$, then by Claim 5.5, we necessarily have $|L^-| > n$. The interpretation $M = \{x_i | \overline{x_i} \in L^-\}$ satisfies strictly more than n of the x_i, so it is a model of Δ, but not a model of κ. If $|L^-| < |L^+|$, then by Claim 5.5, we necessarily have $|L^+| > n$. The interpretation $M = \{x_i | \overline{x_i} \in L^-\}$ satisfies strictly less than n of the x_i, so it is a model of Δ, but not a model of κ. In both cases, $\Delta \models \kappa$ is contradicted, so $|L^+| = |L^-|$. □

Claim 5.7. *We have* $\forall i \in \{1, \ldots, m\}$, $a_i = k$.

Proof. Let us suppose that $\exists i_0 \in \{1, \ldots, m\}$, $a_{i_0} < k$ and that the variable associated with l_{i_0} is $x_{i'_0}$.

There are two possible cases: either $x_{i'_0} \in L^+$ or $\overline{x_{i'_0}} \in L^-$. If $x_{i'_0} \in L^+$, let us consider the interpretation $M = \{x_{i'_0}\} \cup \{x_i | \overline{x_i} \in L^-\}$. Then, by Claims 5.5 and 5.6, M satisfies $n+1$ of the x_i, so it is a model of Δ. Otherwise, $\overline{x_{i'_0}} \in L^-$ and let us consider the interpretation $M = \{x_i | \overline{x_i} \in L^-\} \backslash \{x_{i'_0}\}$. Then, M satisfies $(n-1)$ of the x_i, so it is a model of Δ. In both cases, $\sum_{i=1}^{n} a_i l_i = a_{i'_0} < k$, so M is not a model of κ. $\Delta \models \kappa$ is contradicted, so $\forall i \in \{1, \ldots, m\}$, $a_i = k$. □

Thus, the division rule gives that $\kappa = \sum_{i=1}^{m} k l_i \geq k \equiv \sum_{i=1}^{m} l_i \geq 1$, in other words, κ is equivalent to a clause. The only way to represent Δ as a conjunction of pseudo-Boolean constraints is then to use clauses, since all the constraints must be implied by Δ.

However, Δ requires an exponential number of clauses to be represented without introducing any new variable. Indeed, a clause as the ones we have produced in this proof can only eliminate a single counter-model of Δ, since all those clauses contain all the variables, and there exist $\binom{2n}{n}$ counter-models. Hence, the claim follows. □

Corollary 5.8. *CARD and PBC satisfy neither* $\vee C$ *nor* $\neg C$.

We also have that:

Proposition 5.9. *PBC does not satisfy SFO.*

Proof. Let κ and κ' be the two pseudo-Boolean constraints defined by $\sum_{i=1}^{2n} x_i \geq n+1$ and $\sum_{i=1}^{2n} \overline{x_i} \geq n+1$, respectively. Let us consider, for s a newly introduced variable, the formulae $\kappa_s = (n+1)s + \sum_{i=1}^{2n} x_i \geq n+1$ and $\kappa'_s = (n+1)\overline{s} + \sum_{i=1}^{2n} \overline{x_i} \geq n+1$, which are obtained from κ and κ' in polynomial time.

Let $\varphi = \kappa_s \wedge \kappa'_s$. By construction, $\varphi | s \equiv \kappa'$ and $\varphi | \overline{s} \equiv \kappa$. So, $\exists x\, \varphi \equiv \kappa \vee \kappa'$. If an algorithm existed to forget a single variable in a set of pseudo-Boolean constraints, then one could use it to compute in polynomial time the disjunction of κ and κ'. However, we have proven that it is not possible (cf. Proposition 5.4). Then, so it is for the forgetting of a single variable in a conjunction of pseudo-Boolean constraints. □

Corollary 5.10. *PBC does not satisfy FO.*

We also have that:

Proposition 5.11. *CARD does not satisfy FO, unless* P = NP.

Proof. As for CNF, if CARD satisfied FO, then it would be possible to compute CO on a formula from CARD in polynomial time. Indeed, a formula is consistent precisely when the representation obtained by forgetting all its variables in it evaluates to true (which can be tested easily since the set of variables of the resulting formula is empty). □

6 Conclusion

We have studied the language PBC of pseudo-Boolean constraints and the language CARD of cardinality constraints from a knowledge representation perspective. Our results show that switching from CNF to CARD and then to PBC increases the succinctness of the representations but does not lead to languages that would be at least as succinct as other propositional languages, like DNF, OBDD, or FBDD. Interestingly, such a switch does not lead to the loss of any tractable query that is supported by CNF. As to the transformations, both PBC and CARD lose $\vee$BC, and PBC loses SFO as well. No transformation is gained w.r.t. CNF.

Accordingly, when dealing with applications involving propositional representations but not requiring to perform any transformation among SFO and $\vee$BC, we now know that the languages PBC and CARD are valuable alternatives to CNF due to the succinctness increase they may lead to. Of course, every PBC representation Σ can be turned in linear time into a CNF representation Φ which is not equivalent to it, but has the same logical consequences over the variables occurring in Σ. The translations used to go from Σ to Φ (especially those reported in [Plaisted and Greenbaum, 1986; Tseitin, 1968]) consist in introducing in Φ additional variables not occurring in Σ (in fact, Σ is equivalent to the formula we get by forgetting those variables in Φ). Since such translations can be achieved in linear time, the size of Φ is also linear in that of Σ. While using Φ instead of Σ is a feasible alternative for answering certain queries over the variables occurring in Σ, it must be noted that Φ cannot be substituted to Σ in every setting, i.e., for answering any query or achieving any transformation. For instance, Φ and Σ do not have the same implicants in the general case. In addition the queries and transformations offered by the language of CNF with some existentially quantified variables (alias the existential closure of CNF) do not coincide with those offered by PBC and CARD [Fargier and Marquis, 2008]. All this explains why the extra succinctness of PBC and CARD compared to CNF can be considered as a decisive advantage for some applications.

Acknowledgements

The authors are grateful to the anonymous reviewers for their numerous comments. These comments greatly helped to improve the presentation of the paper. Part of this work was supported by the French Ministry for Higher Education and Research, the Hauts-de-France Regional Council through the "Contrat de Plan État Région (CPER) DATA" and an EC FEDER grant.

References

[Barth, 1995] Peter Barth. A Davis-Putnam based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Technical Report MPI-I-95-2, Max-Planck Institut Für Informatik, 1995.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[Bova *et al.*, 2016] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge Compilation Meets Communication Complexity. In *IJCAI'16*, pages 1008–1014, 2016.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, 1986.

[Chai and Kuehlmann, 2003] Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. In *DAC'03*, pages 830–835, 2003.

[Cook *et al.*, 1987] William Cook, Collette R. Coullard, and György Turán. On the Complexity of Cutting-plane Proofs. *Discrete Appl. Math.*, 18(1):25–38, 1987.

[Crama and Hammer, 2011] Yves Crama and Peter L. Hammer. *Boolean Functions: Theory, Algorithms, and Applications*. Cambridge University Press, 2011.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research*, 17(1):229–264, 2002.

[Darwiche, 1999] Adnan Darwiche. Compiling knowledge into decomposable negation normal form. In *IJCAI'99*, pages 284–289, 1999.

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

[DIMACS, 1993] DIMACS. Satisfiability: Suggested Format. *DIMACS Challenge. DIMACS*, 1993.

[Dixon and Ginsberg, 2002] Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-boolean satisfiability solver. In *AAAI'02*, pages 635–640, 2002.

[Dixon, 2004] Heidi Dixon. *Automating Pseudo-Boolean Inference Within a DPLL Framework*. PhD thesis, University of Oregon, 2004.

[Fargier and Marquis, 2008] Hélène Fargier and Pierre Marquis. Extending the knowledge compilation map: Closure principles. In *ECAI'08*, pages 50–54, 2008.

[Gogic *et al.*, 1995] Goran Gogic, Henry Kautz, Christos Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *IJCAI'95*, pages 862–869, 1995.

[Haken, 1985] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297 – 308, 1985.

[Hooker, 1988] John N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239, 1988.

[Krishnamurthy, 1985] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–275, 1985.

[Le Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The SAT4J library, Release 2.2, System Description. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[Nordström, 2015] Jakob Nordström. On the Interplay Between Proof Complexity and SAT Solving. *ACM SIGLOG News*, 2(3):19–44, 2015.

[Plaisted and Greenbaum, 1986] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.

[Quine, 1952] Willard V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952.

[Roussel and Manquinho, 2009] Olivier Roussel and Vasco M. Manquinho. Pseudo-Boolean and Cardinality Constraints. In *Handbook of Satisfiability*, pages 695–733. 2009.

[SATLIB, 1999] SATLIB. *The Satisfiability Library*, 1999. http://www.cs.ubc.ca/ hoos/SATLIB/index-ubc.html.

[Sheini and Sakallah, 2006] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A Hybrid Pseudo-Boolean SAT Solver. *JSAT*, 2(1-4):165–189, 2006.

[Tseitin, 1968] Gregory S. Tseitin. *On the complexity of derivation in propositional calculus*, chapter Structures in Constructive Mathematics and Mathematical Logic, pages 115–125. 1968.

[Vollmer, 1999] Heribert Vollmer. Complexity Measures and Reductions. In *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York, Inc., 1999.