Pseudo-Boolean Constraints from a Knowledge Representation Perspective

Daniel Le Berre¹,², Pierre Marquis¹,²,³, Stefan Mengel¹, Romain Wallon¹,²
¹CRIL-CNRS UMR 8188, Lens, France
²Université d’Artois
³Institut 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 lan-
guage 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 clause efficiently by CARD approach to the questions above.

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. |= denotes entailment and ≡ denotes equivalence. The size of a representation ϕ, denoted |ϕ|, 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 \( \sum_{i=1}^{n} a_i l_i \leq k \), where \( n \) is some non-negative integer, \( \forall i \in \{1, \ldots, n\}, a_i \in \mathbb{Z}, \ l_i \) is a literal, \( \Delta \in \{\leq, <, =, \geq, >\} \) 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 \( \forall i \in \{1, \ldots, n\}, \ a_i \in \mathbb{N}, \ k \in \mathbb{N} \) 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 \( \kappa \) be a (normalized) pseudo-Boolean constraint \( \kappa = \sum_{i=1}^{n} a_i l_i \geq k \) where for some \( i_0 \) in \( \{1, \ldots, n\}, \ a_{i_0} > k \). Then, \( a_{i_0} \) can be replaced by \( k \) without changing the models of the constraint.

- **Division.** Let \( \kappa \) be a (normalized) pseudo-Boolean constraint \( \kappa = \sum_{i=1}^{n} \alpha l_i \geq k \) and an integer \( \alpha > 0 \). \( \kappa \) entails the constraint \( \sum_{i=1}^{n} \left\lfloor \frac{\alpha}{\kappa} \right\rfloor l_i \geq \left\lfloor \frac{k}{\alpha} \right\rfloor \), where \( \lfloor x \rfloor \) denotes the smallest integer greater or equal to \( x \). Moreover, if \( \alpha \) is a divisor of each \( a_i \), then \( \kappa \) is equivalent to the latter constraint.

Note that, in the division rule, we do not require \( k \) to be divisible by \( \alpha \). 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...
In this section, \( \text{PBC} \) and \( \text{CARD} \) are compared in terms of succinctness with many languages considered in the knowledge compilation map, namely \( \text{NNF}, \text{CNF}, \text{DNF}, \text{IP}, \text{DNF}, \text{FBDD}, \text{OBDD}, \text{OBDD}_\prec, \) and \( \text{MODS} \). Special attention is devoted to the language of (DAG-based) Negation Normal Form representations (\( \text{NNF} \)) [Darwiche, 1999; 2001], the language of Ordered Binary Decision Diagram (\( \text{OBDD}_\prec \)) [Bryant, 1986], and the language of Prime Implicants (\( \text{IP} \)) [Quine, 1952] since all the new succinctness results which have been identified are by-products of the succinctness results involving \( \text{PBC}, \text{CARD} \) 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 \( L_1 \rightarrow L_2 \) means that \( L_1 \) is strictly more succinct than \( L_2 \), i.e., \( L_1 \preceq_s L_2 \) and \( L_2 \npreceq_s L_1 \). 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.** \( \text{PBC} \preceq_s \text{CARD} \).

**Proof.** First, since \( \text{CARD} \) is a subset of \( \text{PBC} \), we get that \( \text{PBC} \preceq_s \text{CARD} \).

Now, let us prove that \( \text{CARD} \npreceq_s \text{PBC} \). Let \( \kappa \) 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 \( \text{Var}(\kappa') \subseteq \text{Var}(\kappa) \). We first show that, necessarily, \( k' = 1 \), i.e., \( \kappa' \) is a clause.

Let us suppose that \( k' > 1 \), and let us consider \( M \) the model of \( \kappa \) such that \( x \) is satisfied, and any literal which is in \( \kappa' \) 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 \( \kappa' \), then \( \sum_{i=1}^{m} l_i = 1 \). Otherwise, \( \sum_{i=1}^{m} l_i = 0 \). In both cases, since we have assumed \( k' > 1 \), \( \kappa' \) is not satisfied, so \( M \) is not a model of \( \kappa' \). This contradicts that \( \kappa \models \kappa' \), so \( k' \leq 1 \). Since \( \kappa' \) is supposed to be non-valid, \( k' \) cannot be equal to 0, so \( k' = 1 \).

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

**Proposition 3.2.** \( \text{NNF} \preceq_s \text{PBC} \).

**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 \( \text{NNF} \) representation of polynomial size (see e.g. [Vollmer, 1999]). Now, when a conjunction of pseudo-Boolean constraints is considered, an \( \text{NNF} \) representation of it consists of the conjunction of all these “elementary” \( \text{NNF} \) 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$<_{s}$.

Proposition 3.4. PBC $\not\leq_{s}$ OBDD$<_{s}$

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

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

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

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

Sketch of Proof. If there is $x \in \text{Var}(\varphi)$ such that $x \notin \text{Var}(\kappa)$, then any counter-model of $\kappa$ can be turned into a model $M$ of $\varphi$ by keeping the same assignment, except for $x$, which must be satisfied. $M$ is still a counter-model of $\kappa$, 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 $\varphi$. However, $M$ does not satisfy any literal of $\kappa$, so it is not a model of $\kappa$. 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 $i \in \{1, \ldots, m\}$, $a_i < k$ and that the variable associated with it is $x_{j_0}$. There are two possible cases: $x_{j_0} \in L^{+}$ or $x_{j_0} \in L^{-}$. 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 $\varphi$. In the remaining case, $\overline{x_{j_0}} \in L^{-}$, and then let us consider the interpretation given by $M = \{x_{j} | \overline{x_{j}} \in L^{-}\} \setminus x_{j_0}$. Since $|L^{-}|$ is even, $|M| = |L^{-}| - 1$ is odd, and $M$ is a model of $\varphi$. In both cases, $\sum_{i=1}^{m} a_{ii} = a_{ii} < k$, so $M$ is not a model of $\kappa$. This contradicts $\varphi \models \kappa$, so $\forall i \in \{1, \ldots, m\}, a_i = k$.

So, $\kappa = \sum_{i=1}^{n} k l_i \geq k$, and the division rule gives us that $\kappa \equiv \sum_{i=1}^{m} l_i \geq 1$, i.e., $\kappa$ is equivalent to clause. Then, the only way to represent $\varphi$ as a conjunction of pseudo-Boolean constraints is to use clauses, since all these constraints must be implied by $\varphi$. Since $\varphi$ 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 $\varphi = \bigwedge_{j=1}^{n} (x_j \land y_j)$ be a formula from IP [Darwiche and Marquis, 2002]. We now show that representing $\varphi$ as a conjunction of pseudo-Boolean constraints requires an exponential number of constraints.

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

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

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

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

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

Claim 3.11. For all $j \in \{1, \ldots, n\}$, if $x_j$ and $y_j$ appear positively in $\kappa$, 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 $\kappa$, respectively, and $a_{j_0} + a_{j_1} < k$, then the model of $\varphi$ satisfying these two literals but falsifying all the others of $\kappa$ is not a model of $\kappa$, which contradicts $\varphi \models \kappa$.

Claim 3.12. Without loss of generality, $\kappa$ can contain only positive literals.

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

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

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

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

Any counter-model of $\varphi$ 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 $\varphi$.

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 $\varphi$. Then, it must not satisfy at least one of the constraints
Claim 3.13. There exists a set of $2^{n-1}$ tuples of $B^n$ 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 $\varphi$ 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.

<table>
<thead>
<tr>
<th>CARD</th>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>PBC</td>
<td>√</td>
<td>o</td>
<td>?</td>
<td>√</td>
<td>√</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

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.

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 $\land BC$ and $\land C$.

**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 $\lor BC$.

**Proof.** Let $\varphi$ be the disjunction of the two cardinality constraints $\kappa = y \geq 1$ ($\kappa' = y$) and $\kappa' = \sum_{i=1}^{2n} x_i \geq n$. It is easy to see that $\varphi$ 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 \( \forall \text{BC} \).**

**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 \( \sum_{i=1}^{2n} x_i \leq n - 1 \), 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 \( \text{Var}(\kappa) \subseteq \text{Var}(\Delta) \). Let us note \( L^+ \) and \( L^- \) the sets of positive and negative literals of \( \kappa \), respectively.

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

**Sketch of Proof.** Consider \( I \) a counter-model of \( \kappa \). Since \( \Delta \models \kappa \), \( I \) satisfies \( n \) of the \( x_i \). If \( x \in \text{Var}(\Delta) \) but \( x \notin \text{Var}(\kappa) \), then the interpretation \( I' \) defined by \( I' = I \cup \{ x \} \) satisfies \( n + 1 \) of the \( x_i \), so it is a model of \( \Delta \), but not a model of \( \kappa \), 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 \( M = \{ x_i \mid \text{Var}(\kappa) \subseteq \text{Var}(\Delta) \} \) satisfies strictly more than \( n \) of the \( x_i \), so it is a model of \( \Delta \), but not a model of \( \kappa \). If \( |L^-| < |L^+| \), then by Claim 5.5, we necessarily have \( |L^+| > n \). The interpretation \( M = \{ x_i \mid \text{Var}(\kappa) \subseteq \text{Var}(\Delta) \} \) satisfies strictly less than \( n \) of the \( x_i \), so it is a model of \( \Delta \), but not a model of \( \kappa \). In both cases, \( \Delta \models \kappa \) is contradicted, so \( |L^+| = |L^-| \).

**Corollary 5.8.** **CARD and PBC satisfy neither \( \forall \text{C} \) nor \( \neg \text{C} \).**

We also have that:

Proposition 5.9. **PBC does not satisfy SFO.**

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

Let \( \varphi = \kappa \land \kappa'_s \). By construction, \( \varphi \models \kappa' \) and \( \varphi \not\models \kappa \). So, \( \exists x \varphi \models \kappa \land \kappa' \). If an algorithm existed to forget a single variable in a set of pseudo-Boolean constraints, then we could use it to compute in polynomial time the disjunction of \( \kappa \) and \( \kappa' \). 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.

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 \( \forall \text{BC} \), 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 \( \forall \text{BC} \), 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 \( \Sigma \) can be turned in linear time into a CNF representation \( \Phi \) which is not equivalent to it, but has the same logical consequences over the variables occurring in \( \Sigma \). The translations used to go from \( \Sigma \) to \( \Phi \) (especially those reported in [Plaisted and Greenbaum, 1986; Tseitin, 1968]) consist in introducing in \( \Phi \) additional variables not occurring in \( \Sigma \) (in fact, \( \Sigma \) is equivalent to the formula we get by forgetting those variables in \( \Phi \)). Since such translations can be achieved in linear time, the size of \( \Phi \) is also linear in that of \( \Sigma \). While using \( \Phi \) instead of \( \Sigma \) is a feasible alternative for answering certain queries over the variables occurring in \( \Sigma \), it must be noted that \( \Phi \) cannot be substituted to \( \Sigma \) in every setting, i.e., for answering any query or achieving any transformation. For instance, \( \Phi \) and \( \Sigma \) 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


