Representing Boolean Functions as Linear Pseudo-Boolean Constraints

Jan-Georg Smaus

Institut für Informatik, Universität Freiburg, Georges-Köhler-Allee 52, 79110 Freiburg im Breisgau, Germany, smaus@informatik.uni-freiburg.de

August 16, 2006

Technical Report No. 227

Abstract. A *linear pseudo-Boolean constraint* (LPB) is an expression of the form $a_1 \cdot l_1 + \ldots + a_m \cdot l_m \geq d$, where each l_i is a *literal* (it assumes the value 1 or 0 depending on whether a propositional variable x_i is true or false) and the $a_1, \ldots, a_m, d$ are natural numbers. The formalism can be viewed as a generalisation of a propositional clause. It has been said that LPBs can be used to represent Boolean functions more compactly than the well-known *conjunctive* or *disjunctive* normal forms. In this paper, we address the question: *how much* more compactly? We compare the expressiveness of a single LPB to that of related formalisms, and we give a statement that outlines how the problem of computing an LPB representation of a given CNF or DNF might be solved recursively. However, there is currently still a missing link for this to be a full algorithm.
Note: This report is indentical to [11] except for several proofs that it contains all the proofs missing there.

1 Introduction

A *linear pseudo-Boolean constraint* (LPB) [1, 2, 4–7] is an expression of the form $a_1l_1 + \ldots + a_ml_m \geq d$. Here each l_i is a *literal* of the form x_i or $\bar{x}_i \equiv 1 - x_i$, i.e. x_i becomes 0 if x_i is false and 1 if x_i is true, and vice versa for $\bar{x}_i$. Moreover, the $a_1, \ldots, a_m, d$ are natural numbers.

An LPB can be used to represent a Boolean[1] function; e.g. $x_1 + \bar{x}_2 + x_3 \geq 3$ represents the same function as the propositional formula $x_1 \wedge \neg x_2 \wedge x_3$ (in the following we identify propositional formulae with functions). It has been observed that a function can often be represented more compactly as a set of LPBs than as a *conjunctive* or *disjunctive* normal form (CNF or DNF) [4, 6, 7]. In fact, one can easily find an LPB which expresses a function that needs more than one clause to be expressed as DNF, say: $2x_1 + \bar{x}_2 + x_3 + x_4 \geq 2$ corresponds to the DNF $x_1 \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_4) \vee (x_3 \wedge x_4)$.

[1] We will omit the qualification "Boolean" when it is clear from the context.

The interest in Boolean functions, or propositional logic, comes from countless applications in verification and design automation concerning finite state systems [1–9, 14].

Previous works on LPBs [1, 4–7] have focused on generalising techniques applied in CNF-based propositional satisfiability solving [8, 9, 14] to LPBs, emphasising that this is beneficial because of the compactness of LPB representations: solving a problem based on such a compact representation can often be more efficient than based on a CNF or DNF encoding.

But where do the LPBs come from? One possibility is that for some application domain, one gives a direct representation of the problems as LPBs and in addition argues that the alternative representation as CNF would be less compact [1, 5, 7]. Another possibility is that one considers problem representations given as propositional formulae and transforms these into compact LPB representations. We are not aware that the latter has ever been proposed. In addition, almost all the arguments that we found in favour of LPBs were not *strictly* about LPBs but about *cardinality constraints*, which are a special case of LPBs.

This raises the question: how can a propositional formula be transformed into an LPB representation that is as compact as possible? In this paper, we approach this question from two extreme ends: on the one hand, we analyse, under various aspects, how much can be expressed by *a single* LPB. On the other hand, we consider propositional formulae for which the LPB representation saves nothing.

In fact, one cannot expect that a single LPB could model any useful problem—in particular, satisfiability of a single LPB is trivial. Our work is, we hope, a step towards the ultimate aim of representing an arbitrary Boolean function as a *set* (usually interpreted as conjunction but also a disjunction is thinkable) of LPBs.

In Sec. 3 we show that there is a strict inclusion chain from clauses to cardinality constraints to LPBs to the *monotone* Boolean functions, provided the dimension m is not too small. Monotone functions are those which can be represented by a formula where each variable occurs only in one polarity.

In Sec. 4, we show that in some cases an LPB representation can be exponentially more compact than a CNF or DNF representation, while in other cases one saves nothing at all.

In Sec. 5 we give a first result addressing the question of how a propositional formula can be converted to an LPB: if a DNF can be expressed by an LPB, then the dual CNF can be expressed by a very similar LPB, and vice versa.

In Sec. 6 we propose important components of what could become an algorithm for converting a DNF Φ to an LPB if possible. First of all, it is possible to determine the relative order of the coefficients a_i. Then, put simply, one has to split Φ into two subsets and remove the variable with the largest coefficient. This gives two subformulae for which LPBs can now be computed recursively. If two LPBs I^0, I^1 can be found which are in a certain sense very similar, then they can be composed into an LPB for Φ. The reason why this is not yet an algorithm is the following: if I^0, I^1 are not immediately "very similar", one has to see if there are I'^0, I'^1, equivalent to I^0, I^1, respectively, such that I'^0, I'^1 are "very similar". It is currently unclear how this can be done systematically.

2 Preliminaries

We assume the reader to be familiar with the basic notions of propositional logic. We write $\oplus$ to denote exclusive-or and $\leftrightarrow$ for equivalence.

An m**-dimensional Boolean function** f is a function $Bool^m \rightarrow Bool$. We say that f **properly depends** on the ith argument if there exists $\beta \in Bool^{i-1}$, $\beta' \in Bool^{m-i}$ with $f(\beta, 0, \beta') \neq f(\beta, 1, \beta')$.

We follow [4]. A **0-1 ILP constraint** is an inequality of the form

$$a_1x_1 + \ldots + a_mx_m \geq d \qquad a_i, d \in \mathbb{R}, x_i \in Bool \ (Bool \equiv \{0, 1\}). \tag{1}$$

We identify 0 with *false* and 1 with *true*. We call the a_i **coefficients** and d the **threshold**.

Using the relation $\bar{x}_i \equiv 1 - x_i$ and noting that it is sufficient to consider integer coefficients, one can rewrite a 0-1 ILP constraint as a **linear pseudo-Boolean constraint** (LPB)

$$a_1l_1 + \ldots + a_ml_m \geq d \qquad a_i \in \mathbb{N}, d \in \mathbb{N}, l_i \in \{x_i, \bar{x}_i\}. \tag{2}$$

For example, $x_1 - 0.5x_2 - 0.5x_3 \geq 0$ can be written as $2x_1 + \bar{x}_2 + \bar{x}_3 \geq 2$. An occurrence of a **literal** x_i (resp., $\bar{x}_i$) is called an occurrence of x_i in **positive** (resp., **negative**) polarity.

An LPB where $a_i = 1$ or $a_i = 0$ for all $i \in [1..m]$ is called **cardinality constraint** (e.g. for $m = 4$: $1x_1 + 0x_2 + 1x_3 + 0x_4 \geq 1$, short $x_1 + x_3 \geq 1$).

Note that $l_1 + \ldots + l_m \geq 1$ corresponds to disjunction and $l_1 + \ldots + l_m \geq m$ to conjunction.

A **CNF** is a formula of the form $c_1 \wedge \ldots \wedge c_n$ where each **clause** c_j is a disjunction of literals. A **DNF** is defined dually; a conjunction of literals is called **(dual) clause**. Formally, CNFs and DNFs are sets of sets of literals, i.e. the order of clauses or the order of literals within a clause are insignificant. For DNFs and CNFs, we assume without loss of generality that no clause is a subset of another clause (the latter clause would be redundant since it is *subsumed*). Given a CNF, the **dual** DNF is obtained by swapping $\wedge$ and $\vee$. Any Boolean function can be represented by a DNF or CNF [13].

An **assignment** σ is a function $\{x_1, \ldots, x_m\} \rightarrow Bool$. The notion σ **satisfies** an LPB I is defined as expected [6].

3 Inclusion Results

The results of this section are not difficult but provide some useful insights into the expressiveness of an LPB or cardinality constraint.

Following [12], we define **monotone** functions as follows.

Definition 3.1. A Boolean function is **monotone** (*unate* [4]) if it can be written as $\vee, \wedge$-combination of literals, where each variable occurs in only one polarity. A monotone function is **isotone** if all variables appear with positive polarity.

In this section, we assume that each variable has positive polarity. This is no loss of generality since the polarity of a particular variable is an issue that is orthogonal to the inclusion results of this section: each monotone function has 2^m variants obtained by modifying the polarity of each variable.

We say that assignment σ **minimally** satisfies the LPB I if σ satisfies I and any assignment obtained from σ by changing some variable of I from *true* to *false* does not satisfy I. We say that a dual clause **corresponds** to an assignment if it consists of the variables assigned *true* by σ.

Proposition 3.2. An LPB I represents the DNF that consists of exactly those clauses that correspond to assignments that minimally fulfill I.

We now give the inclusion results.

Lemma 3.3. Each LPB represents a monotone function. For $m \geq 4$, there is at least one monotone function not represented by any LPB. For $m \leq 3$, each monotone function can be represented as LPB.

Proof. The first statement is a consequence of Prop. 3.2.

The second statement is witnessed by the example $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$: Suppose $I \equiv a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \geq d$ is equivalent to $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Then some straightforward considerations about when I must be true imply that $a_1 + a_3 < d$, $a_2 + a_4 < d$, $a_1 + a_2 \geq d$, $a_3 + a_4 \geq d$, which is a contradiction.

Concerning the third statement: We show that there are exactly 9 isotone functions *properly* depending on 3 variables. The generalisation to *monotone* functions, as well as taking care of functions that do not *properly* depend on 3 variables, is straightforward. The following enumeration shows all CNFs representing isotone functions, and the respective LPB representations:

$$
\begin{array}{rl}
x_1 \vee x_2 \vee x_3 & x_1 + x_2 + x_3 \geq 1 \\
x_1 \wedge (x_2 \vee x_3) & 2x_1 + x_2 + x_3 \geq 3 \\
x_2 \wedge (x_1 \vee x_3) & 2x_2 + x_1 + x_3 \geq 3 \\
x_3 \wedge (x_1 \vee x_2) & 2x_3 + x_1 + x_2 \geq 3 \\
(x_1 \vee x_2) \wedge (x_1 \vee x_3) & 2x_1 + x_2 + x_3 \geq 2 \\
(x_1 \vee x_2) \wedge (x_2 \vee x_3) & 2x_2 + x_1 + x_3 \geq 2 \\
(x_2 \vee x_3) \wedge (x_1 \vee x_3) & 2x_3 + x_1 + x_2 \geq 2 \\
(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3) & x_1 + x_2 + x_3 \geq 2 \\
x_1 \wedge x_2 \wedge x_3 & x_1 + x_2 + x_3 \geq 3
\end{array}
$$

□

Lemma 3.4. Every cardinality constraint is an LPB. For $m \geq 3$, there is at least one LPB not expressible as cardinality constraint. For $m \leq 2$, each monotone function can be represented as a cardinality constraint.

Proof. The first statement holds by definition.

The second is witnessed by the example $2x_1 + x_2 + x_3 \geq 2$.

Concerning the third statement: By inspection one sees that there are 14 monotone functions (6 of which *iso*tone) of which 10 can be represented as usual clause, and another 4 can be represented as cardinality constraint.[2] □

4 Compactness of LPB Representations

4.1 Maximally Compact LPB Representations

Here, we consider the case that an LPB representation is exponentially more compact than a CNF or DNF representation. More precisely, encoding one cardinality constraint as CNF can entail an exponential blowup in formula size (not considering encodings involving auxiliary variables, encodings which are not equivalence preserving). Namely, encoding $x_1 + \ldots + x_m \geq k$ requires $\binom{m}{(m-k)+1} = \binom{m}{k-1}$ clauses of length $m-k+1$ as CNF [2] and $\binom{m}{k}$ clauses of length k as DNF (in [6], this is said for CNF but in fact it should be DNF). Note that $\binom{m}{\lfloor m/2 \rfloor} \geq 2^{m/2}$.

In addition, one can show that the blowup when encoding an *LPB* as CNF or DNF is not worse. That is to say, while there are obviously DNFs that can be expressed by an LPB but not by a cardinality constraint, an LPB cannot, in general, express *bigger* DNFs than a cardinality constraint can. This result is omitted here for space reasons.

4.2 Non-compact LPB Representations

While the focus so far has been to see which Boolean functions can be represented as a single LPB, the ultimate goal must be to represent *any* Boolean function as a *set* of LPBs that is as small as possible.

Trivially, the LPB representation is at least as compact as a DNF or CNF representation since $l_1 \vee \ldots \vee l_m$ can be written as $l_1 + \ldots + l_m \geq 1$ and $l_1 \wedge \ldots \wedge l_m$ as $x_1 + \ldots + x_m \geq m$ (if one neglects the space taken by "$\geq m$" or "≥ 1").

Here we show that there are Boolean functions for which the LPB representation is not more compact than the CNF or DNF representation. We believe that to be interesting, this statement must be strengthened: we show that there is a class of Boolean functions for which the minimal CNF or DNF representation has an exponential (in m) number of clauses and for which the representation as LPB set has the same number of LPBs.

Since one LPB can only represent a monotone function, the most straightforward idea is to use here the most "un-monotone" Boolean functions that exist, i.e., the functions for which any flipping of an argument will always flip the result. These are the functions denoted using $\oplus$ and $\leftrightarrow$: the formula $\bigoplus_{i\in[1..m]} x_i$ is true iff an odd number of variables from $x_1, \ldots, x_m$ is true, and $\leftrightarrow_{i\in[1..m]} x_i$ is

[2] Observe that the remaining two (non-monotone) functions are $x_1 \leftrightarrow x_2$ and $x_1 \oplus x_2$ [10].

true iff an even number of variables from $x_1, \ldots, x_m$ is true. It is rather straightforward to show that for these functions, an LPB representation saves nothing. We omit the formal statement here.

Much less obvious, we now show the desired result for a class of *monotone* functions.

To simplify the exposition, we assume that m is an even number, and we divide the variables into two equally sized subsets, which can be denoted most easily by distinguishing the variables with even index from the ones with odd index.

We take as starting point a class of Boolean functions for which the gain of compactness in using an LPB instead of a DNF representation is maximal: the cardinality constraints $x_1 + \ldots + x_m \geq m/2$, see Subsec. 4.1. The DNF representation has $\binom{m}{m/2}$ clauses of length $m/2$. Now we remove half of the dual clauses of this DNF. This is of course still an exponential number in m, and it turns out that the corresponding function cannot be represented any more compactly as disjunction of LPBs.

Lemma 4.1. Consider variables $x_1, \ldots, x_m$, and let Φ be the disjunction of all dual clauses of the form $x_{i_1} \wedge \ldots \wedge x_{i_{m/2}}$ such that $\sum_{j=1}^{m/2} i_j = 0 \mod 2$, i.e., Φ has $\binom{m}{m/2}/2$ clauses of length $m/2$. Then Φ requires $\binom{m}{m/2}/2$ LPBs of $m/2$ variables to be represented as disjunction of LPBs.

Proof. The DNF Φ corresponds to a disjunction of LPBs in the obvious way. This disjunction contains exactly one LPB for each assignment making true exactly a set of variables $x_{i_1}, \ldots, x_{i_{m/2}}$ with $\sum_{j=1}^{m/2} i_j = 0 \mod 2$.

We now show that there cannot be an LPB representation with fewer LPBs. Suppose an LPB I is true for two different assignments making true exactly all x_i with $i \in K$ and all x_j with $j \in J$, respectively, where $K, J \subseteq [1..m]$ are index sets of $m/2$ elements each such that $\sum K = 0 \mod 2$ and $\sum J = 0 \mod 2$. Since there are $m/2$ variables with even and the same number with odd index, it cannot be the case that both K, J contain only even indices or that both K, J contain only odd indices. Thus there exist $i \in K, j \in J$ with different parity. W.l.o.g. assume that $a_i \geq a_j$ (recall that a denotes the coefficients of LPB I). Then the assignment making true exactly all x_l with $l \in J \cup \{i\} \setminus \{j\}$ is a further assignment that fulfils I. But $\sum J \cup \{i\} \setminus \{j\} = 1 \mod 2$ and hence I cannot be part of a disjunction of LPBs representing Φ. ◻

5 Duality

We now return to the question of what functions can be represented by a single LPB. We show that if a DNF can be represented as an LPB, then the dual CNF can also be represented as an LPB, and the two LPBs are closely related.

As in Sec. 3, we assume that each variable has positive polarity.

Theorem 5.1. If a DNF can be represented by an LPB $I \equiv a_1 x_1 + \ldots + a_m x_m \geq d$, then the dual CNF is represented by the LPB $a_1 x_1 + \ldots + a_m x_m \geq \sum_{i=1}^{m} a_i + 1 - d$, and vice versa.

Proof. For a CNF $\Phi \equiv c_1 \wedge \ldots \wedge c_n$ or a DNF $\Phi \equiv c_1 \vee \ldots \vee c_n$, for any c_j, we call the set of variable indices occurring in c_j a *horizontal index set* of Φ. Moreover, we call any set $V \subseteq [1..m]$ such that $\forall j \in [1..n].\ \exists x_i \in c_j.\ i \in V$ holds a *vertical index set* of Φ.

Assume now the DNF $\Phi \equiv c_1 \vee \ldots \vee c_n$ is represented by $I \equiv a_1x_1 + \ldots + a_mx_m \geq d$. If we make all the variables in one c_j true, then Φ must be true. If we make all the variables in a vertical index set false, then Φ must be false. Hence for all horizontal index sets H and all vertical index sets V, it must hold that:

$$\sum_{i \in H} a_i \geq d \qquad \text{and} \qquad \sum_{i \notin V} a_i < d \tag{3}$$

Let Φ' be the CNF dual to Φ. Note that Φ, Φ' have the same horizontal and vertical index sets. If we make all the variables in one c_j false, then Φ' must be false. If we make all the variables in a vertical index set true, then Φ' must be true. So if $I' \equiv a'_1x_1 + \ldots a'_mx_m \geq d'$ is an LPB representing Φ', then for all horizontal index sets H and all vertical index sets V, it must hold that:

$$\sum_{i \notin H} a'_i < d' \qquad \text{and} \qquad \sum_{i \in V} a'_i \geq d' \tag{4}$$

We show that by setting $a'_i = a_i$ for $i \in [1..m]$ and $d' = \sum_{i=1}^{m} a_i + 1 - d$, (4) is fulfilled and thus I' is indeed an LPB representing Φ'.

Let H be an arbitrary horizontal index set of Φ. Then we have

$$\sum_{i \in H} a_i \geq d \Rightarrow \sum_{i \in H} a_i > d - 1 \Rightarrow 0 < \sum_{i \in H} a_i + 1 - d \Rightarrow \sum_{i \notin H} a_i < \sum_{i=1}^{m} a_i + 1 - d,$$

thus the first inequality of (4) holds. Now let V be an arbitrary vertical index set of Φ. Then we have

$$\sum_{i \notin V} a_i < d \Rightarrow \sum_{i \notin V} a_i \leq d - 1 \Rightarrow 0 \geq \sum_{i \notin V} a_i + 1 - d \Rightarrow \sum_{i \in V} a_i \geq \sum_{i=1}^{m} a_i + 1 - d,$$

thus the second inequality of (4) holds.

The proof of the converse is analogous. □

Note the border cases: $a_1x_1 + \ldots + a_mx_m \geq \sum a_i$ represents a conjunction (of literals), $a_1x_1 + \ldots + a_mx_m \geq 1$ represents a disjunction.

Example 5.2. Consider $5x_1 + 2x_2 + 2x_3 + 2x_4 \geq i$ for $i \in [1..11]$. Note first that for $i = 1, 2$ the represented function is the same, and the dual of that function is represented by setting $i = 11, 10$. Similarly one has $i = 3, 4$ vs. $i = 9, 8$. For $i = 5$, the DNF is $x_1 \vee (x_2 \wedge x_3 \wedge x_4)$, and the dual CNF $x_1 \wedge (x_2 \vee x_3 \vee x_4)$ is represented setting $i = 7$. For $i = 6$, the LPB represents $(x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$. According to Thm. 5.1, since $12 - 6 = 6$, the dual CNF is represented by the same LPB, which means that the CNF is equivalent to its dual. This can easily be confirmed.

6 Representing a DNF as LPB

A reader coming from a SAT solving background might object to assuming a DNF as input, since satisfiability solving of DNFs is trivial. However, any results of this section can be applied to CNFs rather than DNFs using Sec. 5. As before, we assume that each variable has positive polarity.

By Prop. 3.2, there is a naïve semi-decision procedure for the problem of converting a DNF to the LPB it represents, involving enumeration of all LPBs. In this section we want to see if one can do better.

6.1 Determining the Order of Coefficients

Given a DNF Φ, one can determine a size order of the potential coefficients of an LPB representing Φ. That is to say, if Φ can be represented as LPB at all, then the coefficients must respect this order.

The following notion is useful for reasoning about the structure of a formula.

Definition 6.1. Variables x and y are **symmetric** in Φ if Φ is equivalent to the formula obtained by exchanging x and y. A set of variables X is **symmetric** in Φ if each pair is symmetric in Φ.

Since the clause order and the order within a clause of a DNF or CNF is insignificant, symmetry is a straightforward syntactic property.

The following lemma relates symmetric variables with identical coefficients.

Lemma 6.2. If LPB $I \equiv a_1x_1 + \ldots + a_mx_m \geq d$ represents a DNF Φ, then $a_i = a_k$ implies that x_i, x_k are symmetric in Φ; moreover, there exists an LPB $I' \equiv a'_1x_1 + \ldots + a'_mx_m \geq d'$ representing Φ such that symmetry of x_i, x_k in Φ implies $a'_i = a'_k$.

Proof. For the first statement, for any clause that corresponds to an assignment that minimally satisfies I, the clause obtained by swapping x_i and x_k also corresponds to an assignment that minimally satisfies I.

For the second statement, assume that Φ can be represented by I and consider symmetric variables x_i, x_k. For any assignment that minimally satisfies I, the symmetry of x_i, x_k in Φ implies that the assignment obtained by swapping the truth values of x_i, x_k also minimally satisfies I. But then one can see that by defining $a'_i = a'_k = a_i + a_k$, $a'_l = 2a_l$ for all $l \in [1..m] \setminus \{i, k\}$, and $d' = 2d$, the LPB I' represents Φ and meets the requirement for x_i, x_k (intuitively, one replaces a_i and a_k by their average). The argument must be repeated for any pair of symmetric variables. □

For example, $x_1 \vee x_2$ can be represented by $2x_1 + x_2 \geq 1$ or $x_1 + x_2 \geq 1$.

We want to measure how often a variable occurs in a DNF, taking the length of the clauses into account. For this it is useful to consider multisets of natural numbers. We write multisets as $\{\!\{\ldots\}\!\}$. We refrain from an exact formalisation, which is a technicality.

Definition 6.3. Let A, B be two multisets of numbers. We write $B \preceq A$ if B is obtained from A as follows: for each occurrence of a number n in A, either leave this occurrence in B, or replace it by an arbitrary (possibly 0) number of occurrences of numbers $> n$. We write $B \prec A$ if $B \preceq A$ and $A \not\preceq B$.

Example 6.4. We have $\{\![2,2,2,2]\!\} \succ \{\![2,2,2]\!\} \succ \{\![2,2,3]\!\} \succ \{\![2,3]\!\}$.

Note that $\{\![2,2,3]\!\} \succ \{\![2,3]\!\}$ can be established in two ways: removing one occurrence of 2 from $\{\![2,2,3]\!\}$, or removing the occurrence of 3 from $\{\![2,2,3]\!\}$ and replacing one occurrence of 2 from $\{\![2,2,3]\!\}$ by 3.

Definition 6.5. For a DNF Φ, the multiset $OP(\Phi, x)$ is the **occurrence pattern** of x if $OP(\Phi, x)$ has an occurrence of n for each clause of length n in Φ that contains x.

Example 6.6. Consider $\Phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_1 \wedge x_5) \vee (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_3 \wedge x_4 \wedge x_5)$. The occurrence patterns are $OP(\Phi, x_1) = \{\![2,2,2,2]\!\}$, $OP(\Phi, x_2) = \{\![2,2,2]\!\}$, $OP(\Phi, x_3) = OP(\Phi, x_4) = \{\![2,2,3]\!\}$, and $OP(\Phi, x_5) = \{\![2,3]\!\}$. Φ can be represented by $4x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 5$.

The next lemma says that the coefficients of an LPB representing a DNF must correspond to the order given by the occurrence patterns.

Lemma 6.7. Let Φ be a DNF represented by the LPB $I \equiv a_1x_1 + \ldots + a_mx_m \geq d$. Then $a_i \geq a_k$ implies $OP(\Phi, x_i) \succeq OP(\Phi, x_k)$; moreover, there exists an LPB $I' \equiv a'_1x_1 + \ldots + a'_mx_m \geq d'$ representing Φ such that $OP(\Phi, x_i) = OP(\Phi, x_k)$ implies $a'_i = a'_k$.

Proof. Without loss of generality assume $a_1 \geq \ldots \geq a_m$, and consider some x_i, x_k with $i < k$, i.e. $a_i \geq a_k$. We compare $OP(\Phi, x_i)$ and $OP(\Phi, x_k)$.

For each clause of length n in Φ that contains x_i and x_k, we have an occurrence of n in $OP(\Phi, x_i)$ and a uniquely matching occurrence of n in $OP(\Phi, x_k)$.

For each clause of length n in Φ that contains x_i, and for which replacing x_i with x_k gives another clause in Φ, we have an occurrence of n in $OP(\Phi, x_i)$ and a uniquely matching occurrence of n in $OP(\Phi, x_k)$.

Now consider a clause of length n in Φ that contains x_i, and for which replacing x_i with x_k does not give another clause in Φ (the reason for this must be that the corresponding assignment does not fulfill I). For such a clause, replacing x_i with x_k plus additional variables may give a clause in Φ, in which case the occurrence of n in $OP(\Phi, x_i)$ is mapped to one or more occurrences of numbers $> n$ in $OP(\Phi, x_k)$ (note however that conversely, an occurrence of such a number in $OP(\Phi, x_k)$ is not necessarily *uniquely* mapped to the occurrence of n in $OP(\Phi, x_i)$). Or it may be the case that no way of replacing x_i with x_k plus additional variables gives a clause in Φ, in which case the occurrence of n in $OP(\Phi, x_i)$ is mapped to 0 occurrences of numbers $> n$ in $OP(\Phi, x_k)$.

Summarising, we have $OP(\Phi, x_i) \succeq OP(\Phi, x_k)$, showing the first statement.

In particular, if there exists a clause of length n in Φ that contains x_i, and for which replacing x_i with x_k does not give another clause in Φ, then $OP(\Phi, x_i) \succ$

$OP(\Phi, x_k)$. Therefore $OP(\Phi, x_i) = OP(\Phi, x_k)$ implies that for each clause of length n in Φ that contains x_i, swapping x_i with x_k also gives a clause in Φ. I.e., x_i and x_k are symmetric. Hence an LPB I' as required exists by Lemma 6.2. $\square$

Example 6.8. Consider $(x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_1 \wedge x_5) \vee (x_1 \wedge x_6) \vee (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_2 \wedge x_5) \vee (x_2 \wedge x_6) \vee (x_3 \wedge x_4 \wedge x_5) \vee (x_3 \wedge x_4 \wedge x_6)$. The following picture illustrates the occurrence patterns of x_1 and x_5, and thereby all four cases of the proof of Lemma 6.7:

$$\begin{array}{llllll}
x_1: & x_1 \wedge x_5 & x_1 \wedge x_2 & x_1 \wedge x_3 & x_1 \wedge x_4 & x_1 \wedge x_6 \\
 & \quad | & \quad | & \quad \diagdown & \quad \diagup & \quad \perp \\
x_5: & x_1 \wedge x_5 & x_2 \wedge x_5 & \multicolumn{2}{c}{x_3 \wedge x_4 \wedge x_5} &
\end{array}$$

The crucial point is the following: it is not possible that a clause containing x_1 is mapped to a shorter clause containing x_5, since $a_1 \geq a_5$.

Occurrence patterns capture in which ways replacing one variable in a clause of Φ yields or does not yield another clause of Φ. In Ex. 6.6, $OP(\Phi, x_1) \succ OP(\Phi, x_2)$ reflects that replacing x_1 by x_2 does not always yield a clause of Φ.

The following is a corollary of Lemmas 6.2 and 6.7.

Corollary 6.9. If the DNF Φ is represented by an LPB I, then Φ is also representable by an LPB such that x_i, x_k are symmetric in Φ iff x_i, x_k have identical occurrence patterns iff x_i, x_k have identical coefficients.

The results so far can be used to make statements about which DNFs can definitely not be represented as a single LPB. For example, it has been said that a single LPB can express an implication [6]. In [7], implications of the form $y \to (x_1 \wedge x_2)$ are expressed as LPB. However, a simple implication like $(x_1 \vee x_2) \to (y_1 \wedge y_2)$ is beyond what can be expressed by a single LPB. We omit the general statement of this for space reasons.

6.2 Computing an LPB Recursively

We aim for an algorithm that given a DNF Φ finds an LPB representing Φ if possible. By Lemma 6.7 we can use the occurrence patterns to establish the relative order of the coefficients. Assume the numbering of the variables is such that we have $OP(\Phi, x_1) \succeq \ldots \succeq OP(\Phi, x_m)$. Consider now the set $X = \{x_1, \ldots, x_l\}$ such that $OP(\Phi, x_1) = \ldots = OP(\Phi, x_l)$ $(=: OP(\Phi, X))$. It might be that $l = 1$, which corresponds to the intuitive explanation in the introduction. If X is not symmetric in Φ, we know by Cor. 6.9 that Φ cannot be represented as LPB and we can stop. Otherwise, we partition Φ according to how many variables from X each clause contains. We then remove the variables from X from each clause and solve the problem of finding an LPB for each partitition recursively. It turns out that the LPB for Φ can be obtained by combining the LPBs for the subproblems, although it is not obvious that this combination is an effective operation.

Definition 6.10. Given a DNF Φ and a subset X of its variables, we denote by $Cut(\Phi, X, k)$ the disjunction of clauses from Φ containing exactly k variables from X, with those variables removed.

Example 6.11. Let $\Phi \equiv (x_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_5) \vee (x_1 \wedge x_2) \vee (x_1 \wedge x_3 \wedge x_4)$ and $X = \{x_1\}$. Then $Cut(\Phi, X, 0) = (x_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_5)$ and $Cut(\Phi, X, 1) = x_2 \vee (x_3 \wedge x_4)$.

We illustrate our way of proceeding by giving examples.

Example 6.12. Consider the DNF $\Phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$ and $X = \{x_1\}$. Then $Cut(\Phi, X, 0) = x_2 \wedge x_3 \wedge x_4$, represented by $x_2 + x_3 + x_4 \geq 3$. Moreover, $Cut(\Phi, X, 1) = x_2 \vee x_3 \vee x_4$, represented by $x_2 + x_3 + x_4 \geq 1$.

Since the coefficients of the two LPBs agree, it turns out that Φ can be represented by $2x_1 + x_2 + x_3 + x_4 \geq 3$. The coefficient of x_1 is given by the difference of the two thresholds, i.e. $3 - 1$.

In the following example, X contains more than one variable.

Example 6.13. Consider $\Phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$ and $X = \{x_1, x_2\}$. We have $Cut(\Phi, X, 0) = \mathit{false}$, represented by $x_3 + x_4 \geq 4$, $Cut(\Phi, X, 1) = x_3 \wedge x_4$, represented by $x_3 + x_4 \geq 2$, and $Cut(\Phi, X, 2) = \mathit{true}$, represented by $x_3 + x_4 \geq 0$. The DNF Φ is represented by $2x_1 + 2x_2 + x_3 + x_4 \geq 4$. The coefficient of x_1, x_2 is given by $4 - 2 = 2 - 0 = 2$ (we say that the thresholds are *equidistant*).

The previous example shows where the problem for an effective procedure of combining the LPBs lies: for each subproblem, the LPBs are not always the simplest ones; one has to be "smart" to find LPBs that agree in their coefficients.

We now give the theorem.

Theorem 6.14. Let Φ be a DNF in variables $x_1, \ldots, x_m$ and suppose $X = \{x_1, \ldots, x_l\}$ are symmetric variables such that $OP(\Phi, X)$ is maximal w.r.t. $\preceq$ in Φ. Then Φ is represented by an LPB $a_1x_1 + \ldots + a_mx_m \geq d$, where $a_1 = \ldots = a_l$, iff for all $k \in [0..l]$, the DNF $Cut(\Phi, X, k)$ is represented by $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$.

Proof. For an assignment σ and a set of variables V, we denote by $\sigma \setminus V$ the assignment that is undefined on V and else equal to σ. We denote by $\sigma \cup \{V \mapsto \mathit{true}\}$ the assignment that maps all variables of V to *true* and is else equal to σ. For a clause c, we denote by $c \setminus V$ the clause obtained from c by deleting the variables in V.

Since the variables in X are symmetric, the following holds:

$$\text{If } c \in Cut(\Phi, X, k)\text{, then for every } V \subseteq X \text{ with } \#V = k,\ c \cup V \in \Phi. \qquad (5)$$

Throughout, we use Prop. 3.2.

"$\Leftarrow$": We assume that each $Cut(\Phi, X, k)$ is represented by $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$, and show that the assignments minimally fulfilling $a_1x_1 + \ldots + a_mx_m \geq d$ correspond exactly to the clauses of Φ.

a) Consider an assignment σ that minimally fulfils $a_1x_1 + \ldots + a_mx_m \geq d$ and makes exactly k variables from X true, say, the set $V \subseteq X$. Let c be the clause corresponding to σ. Then $\sigma \setminus V$ minimally fulfils $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$, and hence $c \setminus V \in \mathit{Cut}(\Phi, X, k)$. By (5) this implies $c \in \Phi$.

b) Conversely, consider a clause c in Φ that contains exactly k variables from X, say the set $V \subseteq X$. Let σ be the assignment that corresponds to c. Then $c \setminus V \in \mathit{Cut}(\Phi, X, k)$ and thus $\sigma \setminus V$ minimally fulfils $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$, and thus σ minimally fulfils $a_1x_1 + \ldots + a_mx_m \geq d$.

"$\Rightarrow$": We assume that Φ is represented by $a_1x_1 + \ldots + a_mx_m \geq d$ and show that for each k the assignments minimally fulfilling $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$ correspond exactly to the clauses of $\mathit{Cut}(\Phi, X, k)$.

c) For arbitrary $k \in [1..l]$, let σ be an assignment that minimally fulfils $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$ and c the clause corresponding to σ. Then for any $V \subseteq X$ with $\#V = k$, it holds that $\sigma \cup \{V \mapsto \mathit{true}\}$ fulfils $a_1x_1 + \ldots + a_mx_m \geq d$, and since $a_1 = \ldots = a_l > a_{l+1} \geq \ldots \geq a_m$, $\sigma \cup \{V \mapsto \mathit{true}\}$ fulfils $a_1x_1 + \ldots + a_mx_m \geq d$ *minimally* and hence $c \cup V \in \Phi$ and hence $c \in \mathit{Cut}(\Phi, X, k)$.

d) Conversely, consider a clause $c \in \mathit{Cut}(\Phi, X, k)$ and let σ be the assignment corresponding to c. Then by (5), for any $V \subseteq X$ with $\#V = k$, we have that $c \cup V \in \Phi$ and thus $\sigma \cup \{V \mapsto \mathit{true}\}$ is an assignment that minimally fulfils $a_1x_1 + \ldots + a_mx_m \geq d$, and thus σ minimally fulfils $a_{l+1}x_{l+1} + \ldots + a_mx_m \geq d - k \cdot a_1$. □

Unfortunately, Thm. 6.14 does not immediately dictate an algorithm for deciding whether a DNF can be represented as LPB and if yes, computing the LPB. The reason is that a DNF might be represented by alternative LPBs, and so even if the LPBs computed recursively do not have agreeing coefficients and equidistant thresholds, it might be possible to find alternative LPBs for which this is the case, a process we call *unification* of LPBs (see Ex. 6.13).

The unification problem can be divided into the problems of making the coefficients agree and making the thresholds equidistant. Concerning the second problem, the idea is to record, for each LPB, to what extent its threshold can be shifted without changing the meaning. We then try to shift the thresholds of the LPBs to be unified to make the thresholds equidistant. We have some formal statements solving this problem, but in this workshop paper, we prefer to focus on the first problem, for which we have no complete solution yet.

6.3 Making Coefficients Agree

Some adaptations of LPBs are straightforward: disjunctions and conjunctions, *true*, and *false*, can be represented using arbitrary coefficients and choosing d as $\min a_i$, $\sum a_i$, some number ≤ 0, some number $> \sum a_i$, respectively.

We have the following thesis for how unification of several LPBs works, which we have confirmed on a dozen examples: if the coefficients of several LPBs can be made to agree, then the coefficients of the resulting LPB can be obtained by a linear combination of the individual LPBs. Formally, given LPBs $I^k \equiv a_1^k x_1 + \ldots + a_m^k x_m \geq d^k$, for $k \in [0..l]$, if $a_1, \ldots, a_m$ exist such that each I^k is equivalent to $I'^k \equiv a_1 x_1 + \ldots + a_m x_m \geq d'^k$, then $a_1, \ldots, a_m, d'^0, \ldots, d'^l$ can be chosen so that there are numbers $\mu^0, \ldots, \mu^l$ such that for all $j \in [1..m]$, we have $a_j = \mu^0 a_j^0 + \ldots + \mu^l a_j^l$. This will be the main focus of our future work.

Example 6.15. Consider Φ represented by $7x_1+6x_2+4x_3+4x_4+3x_5+3x_6+3x_7 \geq 15$. It turns out that the minimal representation of $Cut(\Phi, x_1, 0)$ is $I^0 \equiv 2x_2+x_3+x_4+x_5+x_6+x_7 \geq 5$ and that of $Cut(\Phi, x_1, 1)$ is $I^1 \equiv 4x_2+3x_3+3x_4+2x_5+2x_6+2x_7 \geq 6$. There are alternative LPBs $I'^0 \equiv 6x_2+4x_3+4x_4+3x_5+3x_6+3x_7 \geq 15$, $I'^1 \equiv 6x_2 + 4x_3 + 4x_4 + 3x_5 + 3x_6 + 3x_7 \geq 8$, and so Thm. 6.14 is applicable. Note that $lhs(I'^0) = lhs(I'^1) = 1 \cdot lhs(I^0) + 1 \cdot lhs(I^1)$.

In Ex. 6.15, for I^0 we have $a_3 = \ldots = a_7$, whereas for I^1 we have $a_3 = a_4 \neq a_5 = a_6 = a_7$. Generally, given two subproblems $Cut(\Phi, X, k)$, $Cut(\Phi, X, k')$, it can happen that we have two variable sets Y, Z with $OP(Cut(\Phi, X, k), Y) \neq OP(Cut(\Phi, X, k), Z)$ while $OP(Cut(\Phi, X, k'), Y) = OP(Cut(\Phi, X, k'), Z)$. That is to say, one subproblem says: Y, Z should have different coefficients, the other says: Y, Z should have the same coefficient.

This raises the question: how different or how similar must coefficients be? The following lemma is one key to the answer.

Lemma 6.16. Let $\Phi \equiv (\bigwedge_{i \in J_1} x_i \wedge c) \vee (\bigwedge_{i \in J_2} x_i \wedge c) \vee \ldots$. If Φ is representable as LPB $I \equiv a_1 x_1 + \ldots + a_m x_m \geq d$, then the following hold:

$$\sum_{i \in J_1} a_i > \sum_{i \in J_2} a_i - \min_{i \in J_2} a_i \quad \text{and} \quad \sum_{i \in J_2} a_i > \sum_{i \in J_1} a_i - \min_{i \in J_1} a_i. \tag{6}$$

Proof. We write J_c for the set of variable indices in c. Then, since $\bigwedge_{i \in J_1} x_i \wedge c$ and $\bigwedge_{i \in J_2} x_i \wedge c$ are clauses of Φ, the following hold

$$\begin{array}{l} \sum_{i \in J_c} a_i + \sum_{J_1} a_i \geq d > \sum_{i \in J_c} a_i + \sum_{J_1} a_i - \min_{i \in J_1} a_i \\ \sum_{i \in J_c} a_i + \sum_{J_2} a_i \geq d > \sum_{i \in J_c} a_i + \sum_{J_2} a_i - \min_{i \in J_2} a_i. \end{array}$$

These imply the result. □

The following two lemmas are consequences. Both can be useful to show how coefficients must be modified, or show that such a modification is impossible.

Lemma 6.17. Let Φ be a DNF and Y and X be sets of symmetric variables of it. Suppose that Φ contains clauses $c \wedge c_X$, $c \wedge c_Y$ with $c_X \subseteq X$, $c_Y \subseteq Y$, and that Φ can be represented as an LPB $I = a_1 x_1 + \ldots a_m x_m \geq d$, where all variables in X (resp., Y) have the same coefficient a_X (resp., a_Y). Then (assuming for convenience that division by 0 gives ∞)

$$\frac{|c_Y| - 1}{|c_X|} < \frac{a_X}{a_Y} < \frac{|c_Y|}{|c_X| - 1} \tag{7}$$

Proof. Linking the notation of Lemma 6.17 to Lemma 6.16 by setting $X = \{x_i \mid i \in J_1\}$ and $Y = \{x_i \mid i \in J_2\}$, we have $a_i = a_X$ for all $i \in J_1$ and $a_i = a_Y$ for all $i \in J_2$. Thus by Lemma 6.16, we have $|c_X| \cdot a_X > |c_Y| \cdot a_Y - a_Y$ and $|c_Y| \cdot a_Y > |c_X| \cdot a_X - a_X$. The result follows by straightforward arithmetic. □

Example 6.18. Let Φ be the DNF such that $Cut(\Phi, x_1, 0)$ is represented by $I^0 \equiv 3x_2 + 3x_3 + 3x_4 + 2x_5 + 2x_6 + 2x_7 + 2x_8 \geq 9$ and $Cut(\Phi, x_1, 1)$ by $I^1 \equiv 4x_2 + 4x_3 + 4x_4 + x_5 + x_6 + x_7 + x_8 \geq 4$. Inspecting those DNFs shows by Lemma 6.17 that $Cut(\Phi, x_1, 0)$ requires $\frac{a_2}{a_5} < \frac{3}{2-1}$ and $Cut(\Phi, x_1, 1)$ requires $\frac{4-1}{1} < \frac{a_2}{a_5}$ which is a contradiction, showing that I^0 and I^1 cannot be unified in the sense of Thm. 6.14 and so Φ cannot be represented by a single LPB.

Lemma 6.19. Let Φ be a DNF and $X = \{x_1, \ldots, x_l\}$ a set of symmetric variables of it. Suppose Φ is represented by an LPB I with $a_1 \geq \ldots \geq a_l$. Then for each k such that Φ contains a clause with exactly k variables from X, we have that $\sum_{i=l-k+1}^{l} a_i > \sum_{i=1}^{k-1} a_i$ (in words, the sum of the k smallest coefficients must be greater than the sum of the $k-1$ greatest coefficients).

Proof. If Φ contains one clause with exactly k variables from X, then due to the symmetry, Φ must contain *every* clause with exactly k variables from X, in particular the clauses $c \wedge x_{l-k+1} \wedge \ldots \wedge x_l$ and $c \wedge x_1 \wedge \ldots x_k$. The result follows from Lemma 6.16. □

Note that the cases $k = 1$ and $k = l$ are genuine special cases: $\sum_{i=l}^{l} a_i > \sum_{i=1}^{0} a_i$ and $\sum_{i=1}^{l} a_i > \sum_{i=1}^{l-1} a_i$ are tautologies and thus nothing interesting can be derived from them. However, if Φ contains some clause of at least 2 and at most $l-1$ variables from X, we can conclude $a_{l-1} + a_l > a_1$ which is quite a strong statement.

Consider Ex. 6.15. For $Cut(\Phi, \{x_1\}, 0)$, one must find a_3, a_5 such that $3a_5 > 2a_3$. The coefficients of I^1, $a_5 = 2$ and $a_3 = 3$, differ too much. But $a_5 = 3$ and $a_3 = 4$ of I'^0, I'^1 are close enough.

So we know some points about how unification of LPBs works, and we can use these to compute LPBs in a more or less ad-hoc way, but we have not solved the problem of making coefficients agree, in general.

7 Conclusion

Linear pseudo-Boolean constraints have attracted interest because they can be used to represent Boolean functions more compactly than CNFs or DNFs, and because techniques applied in CNF-based propositional satisfiability solving can be generalised to LPBs, which can be more efficient than solving a problem based on a CNF representation [1, 4–7]. What is implicit is that there is a problem from some application domain, which has a natural encoding as LPB, and for which the CNF encoding would be larger. There are mainly three points that gave us the impression that the theoretical understanding of LPBs could be improved.

Firstly, several authors have emphasised that an LPB representation of a function can be exponentially more concise than a CNF (or DNF) representation [1, 4–7]. However, it is shown in fact that *cardinality constraints* already can be exponentially more concise than a CNF. Also the examples are mostly about cardinality constraints rather than full LPBs. Thus it is not argued convincingly that the additional expressive power that LPBs have compared to cardinality constraints is useful.

Secondly, it has been noted *en passant* that a single LPB can be used to express an implication [6], but it remains unclear what kind of implications can or cannot be expressed. In fact, an implication like $(x_1 \vee x_2) \rightarrow (y_1 \wedge y_2)$ cannot be expressed.

But most importantly, since an LPB representation can be more compact than a CNF representation, could one not apply the approach of [1, 4–7] to problems that are naturally encoded as CNF, i.e. could one not convert a CNF to an LPB and then apply [1, 4–7]? And if a CNF cannot be represented as a single LPB, can one find a relatively small *set* of LPBs that represents the CNF? As became apparent in Subsec. 6.3, our answer is not conclusive, but it is here that we see the potential for practical application of our work.

We add some more evidence to the first point above. Barth [2] mentions that LPBs arise in artificial intelligence applications [3]. Since he used a solver that could only deal with cardinality constraints, he proposes a transformation of LPBs to cardinality constraints.

In [7], LPBs are used for bounded model checking. At one point, an LPB of the form $x_1 + x_2 + 2\bar{y} \geq 2$ (which is not a cardinality constraint) is used.

Apart from that, the above works say little about where the problem instances come from, and if anything, then these are in fact cardinality constraints rather than LPBs. In [1], problems Min-Cover, Max-SAT and MAX-ONEs are mentioned. E.g., Max-SAT is the problem of finding a variable assignment that maximises the number of satisfied clauses of an unsatisfiable SAT instance. Furthermore, applications from design automation [4], the pigeonhole problem [5] and gate level netlists [6] are mentioned as applications.

However, we are not suggesting that our approach of converting a CNF or DNF to an LPB is the only way to go. If for a problem domain, there is a natural direct translation to an LPB not going via CNF or DNF, then this should definitely be considered.

We summarise our contributions to the understanding of LPBs. We demonstrated that the functions expressible as one LPB constraint are a strict subset of the monotone functions. We showed that in some cases an LPB representation can be exponentially more compact than a CNF or DNF representation, while in other cases the LPB representation has exponential size just like the CNF or DNF representation. We showed that the problems of encoding a DNF or a CNF as LPB have a very simple duality. Finally, we outlined a recursive procedure for computing an LPB representation for a DNF, where the link currently still missing is the problem of making coefficients agree.

Acknowledgements I thank Markus Behle, Martin Fränzle, Marc Herbstritt, Christian Herde, Bernhard Nebel, and the other colleagues from the AVACS project, for useful discussions.

References

1. Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In Lawrence T. Pileggi and Andreas Kuehlmann, editors, *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design*, pages 450–457. ACM, 2002.
2. Peter Barth. Linear 0-1 inequalities and extended clauses. In Andrei Voronkov, editor, *Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning*, volume 698 of *LNCS*, pages 40–51. Springer-Verlag, 1993.
3. Peter Barth and Alexander Bockmayr. Solving 0-1 problems in CLP(PB). In *Proceedings of the 9th Conference on Artficial Intelligence for Applications*. IEEE, 1993.
4. Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. In *Proceedings of the 40th Design Automation Conference*, pages 830–835. ACM, 2003.
5. Heidi E. Dixon and Matthew L. Ginsberg. Combining satisfiability techniques from AI and OR. *The Knowledge Engineering Review*, 15:31–45, 2000.
6. Martin Fränzle and Christian Herde. Efficient SAT engines for concise logics: Accelerating proof search for zero-one linear constraint systems. In Moshe Y. Vardi and Andrei Voronkov, editors, *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2850 of *LNCS*, pages 302–316. Springer-Verlag, 2003.
7. Martin Fränzle and Christian Herde. Hysat: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*, 2006. To appear.
8. João P. Marques Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.
9. Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference*, pages 530–535. ACM, 2001.
10. Raúl Rojas. *Neural Networks. A Systematic Introduction*. Springer-Verlag, 1996.
11. Jan-Georg Smaus. Representing Boolean functions as linear pseudo-Boolean constraints. In Youssef Hamadi, editor, *Proceedings of the CP 2006 Workshop on the Integration of SAT and CP techniques*, 2006.
12. Vetle Ingvald Torvik and E. Trintaphyllou. Inference of monotone Boolean functions. In Chris A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 472–480. Kluwer Academic Publishers, 2001.
13. Ingo Wegener. *The Complexity of Boolean Functions*. Wiley & Sons, `http://eccc.uni-trier.de/eccc-local/ECCC-Books/wegener_book_readme.html`, 1987.
14. Hantao Zhang. SATO: An efficient propositional prover. In William McCune, editor, *Proceedings of the 14th International Conference on Automated Deduction*, volume 1249 of *LNCS*, pages 272–275. Springer-Verlag, 1997.