Completeness of Cutting Planes Revisited

Marc Bezem[1,2] and Robert Nieuwenhuis[2]

[1] Department of Informatics, University of Bergen, Norway
[2] Barcelogic.com and Technical University of Catalonia (UPC), Barcelona, Spain

Abstract. Most of the theoretical background behind integer linear programming (ILP) and 0-1 ILP methods has historically been based on general (rational) Linear Programming.
Motivated by recent practical interest in SAT-related techniques for ILP, here we re-visit cutting planes in a more self-contained way.
We give short and –we think– insight-providing completeness proofs for general bounded ILPs as well as 0-1 ones, analyzing stronger and weaker rule systems, and giving (counter) examples.
Among the insights gained are new completeness results for less prolific and hence more easily automatizable systems including ordered ones.

1 Introduction

The theory of Integer Linear Programming, in part based on the theory of Linear Programming (LP) in the rationals, is well-developed and all results are rigorously proved. It takes, however, considerable effort to understand the sometimes deep proofs. Examples include the proof of completeness of the cutting planes method in [13, Theorem 2.14] and [19, Corollary 23.2b]. This is of course not meant as criticism, but we believe that shorter and more self-contained proofs, if they can be found, may be useful to the community.

Indeed, there has been a considerable recent practical interest in techniques for ILP that are not based on LP-relaxations. On the one hand, there is a large body of work on Pseudo-Boolean (aka 0-1 ILP) solvers based on conflict-driven constraint-learning in a similar way to CDCL clause-learning solvers for SAT (see [18] and its numerous references). On 0-1 ILP problems, such solvers frequently outperform the best commercial LP-relaxation-based tools such as Cplex or Gurobi. On the other hand, also new CDCL-like techniques for full ILP over $\mathbb{Z}$ have recently appeared: Cutsat [12] and IntSat [14]. The latter one also appears to be competitive with these commercial solvers at least on certain full ILP problem classes.

Our motivation for this work is to gain new insights in the cutting-plane inference systems underlying these new CDCL solvers. Since these solvers optimize by solving a sequence of feasibility problems, we here focus on *refutation* completeness. We give short and –we think– insight-providing completeness proofs for general bounded ILPs as well as for 0-1 ones, analyzing stronger and weaker rule systems, and giving (counter) examples. Among the insights gained are new (to the best of our knowledge) completeness results for less prolific and hence more easily automatizable systems including ordered ones.

We address general bounded ILPs in Section 3, and 0-1 ones in Section 4. Several (counter) examples are given in Section 5. Sections 6 and 7 discuss related and future work and conclude.

2 Basic definitions and notation

Let $X = \{x_n \dots x_1\}$ be a set of variables over $\mathbb{Z}$. Throughout this paper we will use the following definitions. An *(integer linear) monomial* is an expression of the form $a_i x_i$ where $a_i \in \mathbb{Z}$ $(1 \leq i \leq n)$ is called the *coefficient* of x_i. A *(homogeneous integer linear) polynomial*, denoted by $p, q, \dots$ is a sum $a_n x_n + \dots + a_1 x_1$ of n monomials. We sometimes use standard shorthands, omitting, e.g., unit coefficients, or monomials with coefficient zero. If $a_i = 0$ for all i with $1 \leq i \leq n$ then the polynomial is called *empty* and denoted by 0. If p is $a_n x_n + \dots + a_1 x_1$ and q is $b_n x_n + \dots + b_1 x_1$, then $p + q$ denotes the polynomial $c_1 x_1 + \dots + c_n x_n$ where $c_i = a_i + b_i$ $(1 \leq i \leq n)$, and if $c \in \mathbb{Z}$ then cp denotes the polynomial $(ca_n)x_n + \dots + (ca_1)x_1$. Given $d \in \mathbb{N}^+$, we write $d|p$ to denote that d divides all a_i for $1 \leq i \leq n$. If $d|p$ then p/d denotes $(a_n/d)x_n + \dots + (a_1/d)x_1$.

An *(integer linear) constraint*, denoted by $C, D, \dots$ is an inequality $p \geq c$ with $c \in \mathbb{Z}$. A *contradiction* is a constraint of the form $0 \geq c$ with $c > 0$. We denote sets of constraints by the letter S. An *integer assignment* is a function $M\colon \{x_n, \dots, x_1\} \to \mathbb{Z}$. If p is a polynomial $a_n x_n + \dots + a_1 x_1$ then $M(p)$ is the integer $a_n M(x_n) + \dots + a_1 M(x_1)$. M *satisfies* a constraint $p \geq c$ if $M(p) \geq_{\mathbb{Z}} c$, denoted $M \models C$. We write $M \models S$ if M satisfies all constraints in S. Then M is a *solution* or a *model* of S and S is called *feasible*.

A *lower bound for* x is a constraint of the form $x \geq l$ and an *upper* bound for x is a constraint of the form $-x \geq -u$. Then the integers l and u are also called *lower (resp. upper) bounds for* x. We say that a set S of constraints is *bounded* if for each variable there is both a lower and an upper bound in S.

We will use the ordering on variables induced by their subindices, as follows. We always write polynomials with variables in descending order. When we write a constraint $ax_i + p \geq c$ this means that $a \neq 0$ and moreover that all monomials bx_j in p with $b \neq 0$ have $j < i$, and we call x_i *maximal* in this constraint. Note that any constraint with non-empty polynomial has a unique maximal variable.

If S is a set of constraints, we denote by S_i the subset of S of constraints having no variable larger than x_i, i.e., in S_i all variables of $\{x_{i+1} \dots x_n\}$ have zero coefficients. Formally, given an S, for each $i = 0, \dots, n$ we define $S_i = \{\ (a_n x_n + \dots + a_1 x_1 \geq c) \in S \mid a_j = 0 \text{ for all } j > i\ \}$. We clearly have that $S_n = S$, that $S_i \subset S_{i+1}$ $(0 \leq i < n)$ and that S_0 includes all contradictions in S (if any).

The following inference rules for constraints are well-known ($\lceil\ .\ \rceil$ denotes the standard rounding-up ceiling function and gcd the greatest common divisor).

$$\textsf{Combine}: \quad \frac{p \geq c \qquad q \geq d}{ap + bq \geq ac + bd} \qquad \text{where} \quad a, b \in \mathbb{N}$$

$$\textsf{Divide}: \quad \frac{a_n x_n + \dots + a_1 x_1 \geq c}{(a_n/d)x_n + \dots + (a_1/d)x_1 \geq \lceil c/d \rceil} \qquad \text{where} \quad d = \gcd(a_n, \dots, a_1)$$

These rules are also well-known to be correct, that is, any set of constraints S has the same solutions in $\mathbb{Z}$ as $S \cup \{C\}$ if C is a conclusion obtained by Combine-ing or Divide-ing with premise(s) in S.

3 Completeness for ILP

The following result is not new, see, e.g., [13, Cor. 2.14]. In fact, it is known to be true even if S is not bounded [19, Corollary 23.2b]. But the proof we provide is new, concise and self-contained. Moreover, the proof allows us to analyse which instances of Combine and Divide are actually used. This will allow us to obtain stronger results.

Theorem 1. *If a bounded set of constraints is infeasible then its closure under* Combine *and* Divide *yields a contradiction.*

Proof. Consider any bounded set of constraints, let S be its closure under Combine and Divide, and assume there is no contradiction in S. We prove that S is feasible, by actually building a solution M of S. Recall that S_i is the subset of all constraints in S where the variables in $x_n, \ldots, x_{i+1}$ have coefficient 0. We build a solution M_i for each S_i by induction on i. The base case $i = 0$ is trivial: the empty M_0 is a solution of S_0 since S has no contradictions. For the induction step $i > 0$, assume M_{i-1} is a solution for S_{i-1} (I.H.). We extend M_{i-i} to M_i by defining $M_i(x_i) = \max\{\, c - M_{i-1}(p) \mid x_i + p \geq c \text{ in } S_i \,\}$. This is well-defined: we take the maximum of a non-empty set (since there is a lower bound $x_i \geq l$ in S) that is upper bounded by u, the upper bound for x_i in S. The latter is true since S is closed under Combine and each $x_i + p \geq c$ in S_i can be combined with $-x_i \geq -u$ to give the constraint $p \geq c - u$ in S_{i-1} and, by I.H., $c - M_{i-1}(p) \leq u$.

We now prove $M_i \models S_i$. Constraints C in $S_i \setminus S_{i-1}$ can have the following form:

A) C is of the form $x_i + p \geq c$. Then by construction of M_i we have $M_i \models C$.

B) C is of the form $-ax_i + p \geq c$ with $a > 0$. By construction, $M_i(x_i) = d - M_i(q)$ for some $x_i + q \geq d$ in S_i. By a Combine of a times $x_i + q \geq d$ with C we get $aq + p \geq c + ad$, which is in S_{i-1}, hence by I.H. $aM_i(q) + M_i(p) \geq c + ad$, so $M_i \models -ax_i + p \geq c$.

C) C is of the form $ax_i + p \geq c$ with $a > 1$.

C1) If $a|p$ then, by Divide, $x_i + p/a \geq \lceil c/a \rceil$ is in S_i and by construction $M_i(x_i) \geq \lceil c/a \rceil - M_i(p/a)$ and hence $M_i(ax_i + p) \geq c$, so $M_i \models C$.

C2) Otherwise, let x_j be the maximal variable in p with coefficient b not divisible by a. Let $x_j + q \geq d$ in S_j be the constraint causing $M_i(x_j)$ to be $d - M_i(q)$.

C2a) If $b < 0$, do a Combine of $-b$ times $x_j + q \geq d$ and C (eliminating x_j).

C2b) If $b > 0$, let r be the smallest natural number such that $b + r$ is divisible by a and do a Combine of r times $x_j + q \geq d$ and C.

In both cases we get a constraint D in S_i, where x_j's coefficient is divisible by a, and such that $M_i \models D$ iff $M_i \models C$. By repeating this at most $n - 1$ times (where each time the maximal variable with coefficient not divisible by a becomes smaller), finally we get a constraint as in case C1), proving $M_i \models C$. □

One inconvenient aspect of the previous result is that Combine is difficult to control (and hence of little practical value): in fact *any* pair of constraints $p \geq c$ and $q \geq d$ can be combined with *any* pair of *factors* $a, b \in \mathbb{N}$ to obtain $ap + bq \geq ac + bd$. Also, while Divide is fully deterministic, it is not clear when it is useful to apply this rule.

However, a closer look at our proof reveals that in fact only very particular cases of Combine and Divide are used. The Divide rule is applied only in case C1), where

the premise is of the form $ax_i + p \geq c$ (i.e., by our notation, x_i is its maximal variable) where $a > 1$, and where *all* coefficients in p are divisible by a, and where indeed the divisor d of Divide is a; this is achieved by the Ordered Exact Divide rule given below. The Combine rule has three kinds of uses in the proof:

- In case B) and before A), Combine is used to *eliminate* x_i, the maximal variable in *both* premises, and moreover in one of the premises x_i has coefficient 1. Hence in the other premise x_i's coefficient $-a$ is negative and the factors are fully determined: they must be a and 1, respectively. This is done by the Ordered VE Combine rule given below (where VE stands for Variable-Eliminating).
- In cases C2a) and C2b) premise C must be of the form $ax_i + p + bx_j + p' \geq c$ where $a > 1$, $a|p$ and $a\nmid b$ (note that by our notation, x_i is its maximal variable and x_j is its largest variable whose coefficient b is not divisible by a). The other premise C' is of the form $x_j + q \geq d$.
  - In case C2a), we have $b < 0$ and Combine is again used only to *eliminate* x_j, by adding $-b$ times C' to C, i.e., the factors must be $-b$ and 1. This is the Ordered VE Preparing Combine rule given below (called like this as it prepares for the Ordered Exact Divide step of case C1).
  - In case C2b), we have $b > 0$ and Combine must be used to add r times C' to C (i.e., factors r and 1), where r is the smallest natural number such that a divides $b + r$. This is the Ordered Preparing Combine rule given below.

Altogether, this proves that the following strongly restricted versions of Combine and Divide suffice.

Ordered VE Combine :

$$\frac{-ax_i + p \geq c \qquad x_i + q \geq d}{p + aq \geq c + ad} \quad \text{where } a > 0$$

Ordered VE Preparing Combine :

$$\frac{ax_i + p + bx_j + p' \geq c \qquad x_j + q \geq d}{ax_i + p + p' - bq \geq c - bd} \quad \text{where } a > 1,\ b < 0,\ a|p \text{ and } a\nmid b$$

Ordered Preparing Combine :

$$\frac{ax_i + p + bx_j + p' \geq c \qquad x_j + q \geq d}{ax_i + p + (b + r)x_j + p' + rq \geq c + rd} \quad \text{where } b > 0,\ a|p,\ a|(b+r),\ 0 < r < a$$

Ordered Exact Divide :

$$\frac{ax_i + p \geq c}{x_i + p/a \geq \lceil c/a \rceil} \quad \text{where } a > 1 \text{ and } a|p$$

Theorem 2. *If a bounded set of constraints is infeasible then its closure under Ordered VE Combine, Ordered VE Preparing Combine, Ordered Preparing Combine, and Ordered Exact Divide yields a contradiction.*

Some remarks related to the proof:

1. Bounds are crucial to make the definition of the model correct, ensuring that we take the maximum of a non-empty, bounded set of integers. Without bounds, finiteness of the initial constraint set is needed: the closure of $\{\ x \geq c \mid c \in \mathbb{N}\ \}$ under Combine and Divide has no contradiction but is not feasible. When the set of constraints is finite, bounds are not needed (see, e.g., [19, Corollary 23.2b]; [19, Corollary 17.1c] ensures that bounds of size polynomial in the size of the input data always exist).
2. The ordering of the variables is arbitrary.
3. The model in the above proof is built by taking for each x_i its smallest possible value, given the model built so far for $x_{i-1} \ldots x_1$. There is a dual model based on largest possible values. One can even choose per variable which of the two. The duality should not come as a surprise: if $l \leq x \leq u$, then $l \leq l + u - x \leq u$.
4. The constraints $x_i + p \geq c$ used in the model construction are called *tight* in [12].
5. Finiteness of the set of variables is not essential. The proof above can easily be generalized to a well-ordered set of variables.
6. Theorem 2 is a stronger completeness result than Theorem 1 since the rules are restricted. These weaker rules are easier to automate. On the other hand, weaker rules could make some proofs longer, for example, proofs of pigeon hole principles. This is a delicate balance: a stronger proof system is only useful for automated theorem proving if shorter proofs can indeed be found. So far this is not the case.

4 0-1 Solutions

This section is on Pseudo-Boolean constraint solving, also known as 0-1 integer linear programming. We consider *0-1 bounded* sets of constraints, i.e., containing *Boolean* bounds $x \geq 0$ and $-x \geq -1$ for every variable. We define the following two new rules that are special cases of Combine in which one of the premises is such a Boolean bound, and that replace the two Preparing rules of the previous section:

Upper 1 Combine :

$$\frac{ax_i + p + bx_j + p' \geq c \qquad -x_j \geq -1}{ax_i + p + (b-r)x_j + p' \geq c - r} \quad \text{where } a|p,\ a|(b-r),\ 0<r<a$$

Lower 0 Combine :

$$\frac{ax_i + p + bx_j + p' \geq c \qquad x_j \geq 0}{ax_i + p + (b+r)x_j + p' \geq c} \quad \text{where } a|p,\ a|(b+r),\ 0<r<a$$

Note that these rules do not include inferences with the stronger bounds $x \geq 1$ and $-x \geq 0$ stating that x is 1 or 0 (initial ones nor derived ones). These are used only by Ordered VE Combine.

Theorem 3. *If a 0-1 bounded set of constraints is infeasible then its closure under Ordered VE Combine, Upper 1 Combine, Lower 0 Combine and Ordered Exact Divide yields a contradiction.*

Proof. Consider any 0-1 bounded set of constraints, let S be its closure under the four rules and assume there is no contradiction in S. We prove that S is feasible, by actually

building a solution M_i for each S_i by induction on i. The base case $i = 0$ is trivial: the empty M_0 is a solution of S_0 since S has no contradictions. For the induction step $i > 0$, assume M_{i-1} is a solution for S_{i-1} (I.H.). We extend M_{i-i} to M_i by defining

- $M_i(x_i) = 1$ if there is a constraint of the form $x_i + p \geq c$ in S_i with $1 + M_{i-1}(p) = c$;
- $M_i(x_i) = 0$ otherwise.

Note that in both cases $M_i(x_i) = c - M_{i-1}(p)$ for some constraint of the form $x_i + p \geq c$ in S_i (in the second case the lower bound $x_i \geq 0$ is of this form). We now prove that $M_i \models C$ for all constraints C in $S_i \setminus S_{i-1}$. There are three cases, depending on C:

A) C is of the form $x_i + p \geq c$. Then by Ordered VE Combine with the upper bound $-x_i \geq -1$, we get $p \geq c - 1$ in S_{i-1} and hence, since $M_{i-1} \models S_{i-1}$, we have $1 + M_{i-1}(p) \geq c$. If $1 + M_{i-1}(p) = c$ then by construction we have $M_i(x_i) = 1$ and $M_i \models C$. If $1 + M_{i-1}(p) > c$ then $M_{i-1}(p) \geq c$ and even if $M_i(x_i) = 0$ we have $M_i \models C$.

B) C is of the form $-ax_i + p \geq c$ with $a > 0$. By construction $M_i(x_i) = d - M_i(q)$ for some $x_i + q \geq d$ in S_i, so we need to prove $-ad + aM_i(q) + M_i(p) \geq c$. By a Ordered VE Combine of a times $x_i + q \geq d$ and C we get that $aq + p \geq ad + c$ is in S_{i-1}, hence by I.H. $aM_i(q) + M_i(p) \geq c + ad$.

C) C is of the form $ax_i + p \geq c$ with $a > 1$.

C1) If $a|p$ then by Ordered Exact Divide $x_i + p/a \geq \lceil c/a \rceil$ is in S_i. By case A) $M_i(x_i) + M_i(p/a) \geq \lceil c/a \rceil$ and hence $aM_i(x_i) + aM_i(p/a) \geq c$ and hence $M_i \models C$.

C2) Otherwise, let x_j be the maximal variable in p whose coefficient b is not a multiple of a. Then C is of the form $ax_i + q + bx_j + q' \geq c$, where $a|q$.

C2a) If $M_{i-1}(x_j) = 1$, do a Upper 1 Combine getting $ax_i + q + (b-r)x_j + q' \geq c - r$.

C2b) If $M_{i-1}(x_j) = 0$, do a Lower 0 Combine getting $ax_i + q + (b+r)x_j + q' \geq c$.

In both cases a constraint D is obtained that is in S_i, and such that $M \models D$ iff $M \models C$. By doing this for every variable in p whose coefficient is not a multiple of a, we obtain a constraint as in case C1), proving $M_i \models C$. □

5 Examples and counterexamples

In the previous sections we have given completeness proofs attempting to minimize the applicability of Combine, by limiting it to the cases where one of the premises has coefficient 1, where moreover the factors n, m of Combine are fully determined, and where a specific variable is either eliminated or increased by a precise amount. Moreover Divide is applied only to divide by a, the coefficient of the maximal variable, and only if the polynomial is exactly divisible by a.

A natural question arises: can non-variable eliminating applications of Combine be avoided altogether? The answer is negative even in the 0-1 case and even with the more general Combine and Divide:

Example 1. Let S consist of $3x + 2y \geq 1$, $-3x + 2y \geq -2$, $2x - 3y \geq -2$ and $-2x - 3y \geq -4$ and the four bounds for $0 \leq x, y \leq 1$, inspired by (and equivalent to) the well-known unsatisfiable example of four clauses that cannot be refuted by resolution *without factoring*: $x \vee y$, $\overline{x} \vee y$, $x \vee \overline{y}$ and $\overline{x} \vee \overline{y}$. Their standard representation as constraints

$x + y \geq 1$, $-x + y \geq 0$, $x - y \geq 0$, and $-x - y \geq -1$ is easily refuted, but all refutations are effectively blocked by representing the clauses in a non-standard way with a clever choice of coefficients $\pm 2, \pm 3$ (adjusting the rhs accordingly). For example, from $3x + 2y \geq 1$ and $-3x + 2y \geq -2$ a variable-eliminating Combine infers $4y \geq -1$, which can only be Divided into the completely uninformative $y \geq 0$. One easily verifies that no contradiction can be derived by variable-eliminating Combine and Divide. □

Example 2. Let S be as in Example 1. It is instructive to see how the inference rules of Theorem 3 refute S without doing much redundant work. Define the ordering by $x > y$, so $x_2 = x$ and $x_1 = y$. Ordered VE Combine can only be used to infer the tautology $0 \geq 1$ from the bounds for x of y, or with $x \geq 0$ and either $-3x + 2y \geq -2$ or $-2x - 3y \geq -4$, yielding the uninformative $2y \geq -2$ and $-3y \geq -4$, respectively. Lower 0 Combine and Upper 1 Combine can only be applied with $3x + 2y \geq 1$ and a bound for y in S, and with $2x - 3y \geq -2$ and a bound for y in S. With $-y \geq -1$ this yields the uninformative $3x \geq -1$ and the informative $2x - 4y \geq -3$, or $x - 2y \geq -1$ after an application of Ordered Exact Divide. With $y \geq 0$ this yields the informative $3x + 3y \geq 1$, or $x + y \geq 1$ after an application of Ordered Exact Divide, and the uninformative $2x \geq -2$. We have now inferred two new constraints, $x - 2y \geq -1$ and $x + y \geq 1$ in which the variable x has coefficient 1, which opens up for new applications of Ordered VE Combine. Indeed, an Ordered VE Combine of $-3x + 2y \geq -2$ and $x + y \geq 1$ yields $5y \geq 1$, or $y \geq 1$ after an application of Ordered Exact Divide. Moreover, an Ordered VE Combine of $-2x - 3y \geq -4$ and $x - 2y \geq -1$ yields $-7y \geq -6$, which contradicts $y \geq 1$, formally by another application of Ordered VE Combine. □

The next question that arises is whether our rules for the 0-1 case are sufficient for the integer case, when *suitably generalized*, i.e., the right premises in Upper 1 Combine and Lower 0 Combine are the initial bounds for x_j in S, say $-x_j \geq -u_j$ and $x_j \geq l_j$, and the right-hand sides of the conclusion of Upper 1 Combine and Lower 0 Combine are $c - ru_j$ and $c + rl_j$, respectively. The answer is no, as we will see in the next example.

Example 3. Let S be as in Example 1 with the only difference that the upper bound $-y \geq -1$ is replaced by $-y \geq -2$. Therefore, the analysis of the previous example largely applies here. Hence we focus on the differences. The upper bound $-y \geq -2$ can only be applied by Upper 1 Combine. With $3x + 2y \geq 1$ this gives the uninformative $3x \geq -3$. With $2x - 3y \geq -2$ it gives $2x - 4y \geq -4$, and after Ordered Exact Divide, $x - 2y \geq -2$. From the latter we can infer $-2y \geq -3$ by an application of Ordered VE Combine with $-x \geq -1$. By applying Ordered Exact Divide, we can indeed infer a new *derived* upper bound $-y \geq -1$. However, as remarked after the definition of Upper 1 Combine and Lower 0 Combine, derived bounds cannot be used in these two rules, thus blocking the refutation of the previous example. Indeed, with some effort, doing an exhaustive case analysis the reader can now verify that no contradiction can be derived. □

6 Related Work

It is well-known that from any run of a CDCL-based SAT solver on an unsatisfiable instance one can in fact extract a resolution refutation: at each conflict, by resolution a

new clause is derived that enables a backjump, which finally leads to a conflict at decision level zero, giving the empty clause. Proving this formally requires a termination argument based on a well-founded ordering on CDCL states (see e.g., [15]).

We start by the observation that, in a similar way, it is possible to extract a completeness proof of Combine and Divide for bounded ILP from Jovanovic and De Moura's Cutsat procedure [12] (whose termination is based on [15]). But it gives a far longer and more complicated proof than ours, less insight-providing, and it does not lead to our restrictive rules nor to ordered rules. Note however that the Cutsat procedure was not devised with the purpose of providing a completeness proof; it is an elegant and highly non-trivial procedure, the first complete practical CDCL cutting-planes-based solver for ILP.

A similar situation exists for several procedures for Pseudo-Boolean constraint solving [2, 7, 8, 6, 4, 20, 3], see also the handbook chapter [18]. These procedures perform variable-eliminating Combine steps, and in order to always be able to backjump, they sometimes need to weaken a constraint, e.g., into a cardinality constraint (i.e., a constraint having only unit coefficients) or into a clause, known to be a logical consequence for semantic reasons. These various versions of semantic weakening can be rephrased as sequences of Combine steps and Divide steps. Some papers make these steps explicit; it also follows from the consequence-finding completeness results [13, Theorem 2.14] and [19, Corollary 23.2b]. An independent syntactic proof of this semantic weakening and a formal termination argument together constitute a completeness proof obtained from these procedures.

Hooker has done pioneering work, first proving consequence-finding completeness of a form of cutting planes for cardinality constraints [9]. Later he extended it to arbitrary 0-1 linear constraints, although assuming a subprocedure for finding certain entailed constraints (in particular this is possible for cardinality constraints) [10, 11].

Work on proof complexity of cutting planes, such as [17], hardly relates to ours, since it essentially concentrates on showing that (no) short (in various senses) proofs exist for certain problem families, rather than on completeness of cutting planes. E.g., Cook et al. [5] rely on Schrijver [19] for completeness.

Last but not least, let us again mention the immense body of work on the theory of (I)LP in e.g., Schrijver [19], already described in Section 1.

7 Conclusions and further work

We have given short and –we think– insight-providing completeness proofs for general bounded ILPs as well as 0-1 ones, including new (to the best of our knowledge) results for less prolific and hence more easily automatizable systems including ordered ones.

Given the kind of induction used in our proofs it would not be difficult to extend these results to include *abstract redundancy* notions as in, e.g., [1, 16] for (first-order) resolution and paramodulation. This would allow one to remove constraints that follow from (finitely many) *smaller* constraints w.r.t. a given well-founded ordering $\succ$ on constraints. The proof is then by contradiction from the existence of a $\succ$-minimal constraint that is false in the constructed model. Such an abstract redundancy notion covers many

concrete criteria. To give just one simple example, a constraint $2x + 4y + z \geq 3$ can be removed (or needs not be generated) if, say, $x + 2y \geq 2$ and $z \geq 0$ are already present.

Some questions arise from the proof complexity point of view. We have proved completeness for inference systems that are particular cases of Combine and Divide but that are far more "restrictive" than these. But there is usually a trade-off: such restrictive systems tend to become less "efficient" in terms of minimal proof length. Which problem families (if any) do have short (polynomial) proofs by Combine and Divide but not by our rules? Does this have any practical consequences for CDCL-based ILP provers? If so, are there any more "controllable" appropriate intermediate systems?

8 Acknowledgments

Both authors have been supported by the Spanish MEC/MICINN project SweetLogics under the grant TIN2010-21062-C02-01 and by the project DAMAS, Spanish MINECO under the grant TIN2013-45732-C4-3-P.

References

1. Leo Bachmair and Harald Ganzinger. Resolution. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers and MIT Press, 2001.
2. Peter Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. Technical report, Max-Plank-Institut fur Informatik, Saarbrucken, 1995. Technical Report MPII952003.
3. Daniel Le Berre and Anne Parrain. The sat4j library, release 2.2. *JSAT*, 7(2-3):59–6, 2010.
4. Donald Chai and Andreas Kuehlmann. A fast pseudo-boolean constraint solver. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 24(3):305–317, 2005.
5. W. Cook, C. Coullard, and Gy. Turan. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.
6. Heidi E. Dixon. *Automating Psuedo-Boolean Inference within a DPLL Framework*. PhD thesis, University of Oregon, 2004.
7. Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-boolean satisfiability solver. In *Procs 18th National Conf on Artificial Intelligence*, pages 635–640, 2002.
8. Heidi E. Dixon, Matthew L. Ginsberg, David K. Hofer, Eugene M. Luks, and Andrew J. Parkes. Implementing a generalized version of resolution. In *Procs 19th National Conf on Artificial Intelligence*, pages 55–60, 2004.
9. John N Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12:217–239, 1988.
10. John N Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6:271–286, 1992.
11. John N Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, 2000.
12. Dejan Jovanovic and Leonardo Mendonça de Moura. Cutting to the chase - solving linear integer arithmetic. *J. Autom. Reasoning*, 51(1):79–108, 2013.
13. G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1999.
14. Robert Nieuwenhuis. The IntSat method for integer linear programming. In *Principles and Practice of Constraint Programming, 20th International Conference, CP 2014, LNCS 8656*, pages 574–589.

15. Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM, JACM*, 53(6):937–977, 2006.
16. Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–444. Elsevier Science and MIT Press, 2001.
17. P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.
18. Olivier Roussel and Vasco M. Manquinho. Pseudo-boolean and cardinality constraints. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in AI and Applications*, pages 695–733. IOS Press, 2009.
19. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.
20. Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-boolean SAT solver. *JSAT*, 2(1-4):165–189, 2006.