A proof system for certifying symmetry and optimality reasoning in integer programming [⋆]

Jasper van Doornmalen[1], Leon Eifler[2], Ambros Gleixner[2,3], and Christopher Hojny[1]

[1] TU Eindhoven, The Netherlands, `{m.j.v.doornmalen,c.hojny}@tue.nl`
[2] Zuse Institute Berlin, Germany, `eifler@zib.de`
[3] HTW Berlin, Germany, `gleixner@htw-berlin.de`

Abstract. We present a proof system for establishing the correctness of results produced by optimization algorithms, with a focus on mixed-integer programming (MIP). Our system generalizes the seminal work of Bogaerts, Gocht, McCreesh, and Nordström (2022) for binary programs to handle any additional difficulties arising from unbounded and continuous variables, and covers a broad range of solving techniques, including symmetry handling, cutting planes, and presolving reductions. Consistency across all decisions that affect the feasible region is achieved by a pair of transitive relations on the set of solutions, which relies on the newly introduced notion of consistent branching trees. Combined with a series of machine-verifiable derivation rules, the resulting framework offers practical solutions to enhance the trust in integer programming as a methodology for applications where reliability and correctness are key.

1 Introduction

In this paper, we are concerned with general optimization problems of the form $\mathrm{OPT}(f, F) \coloneqq \min\{f(x) : x \in F\}$ for a real-valued function $f\colon \mathbb{R}^n \to \mathbb{R}$ and a feasible region $F \subseteq \mathbb{R}^n$ defined by a set of constraints. In particular, we are interested in the solution of mixed-integer programs (MIPs) to optimality, for which state-of-the-art algorithms are composed of a large array of different solving techniques [1,2,16,45,57]. Though their correctness is proven *on paper*, with today's tools it seems prohibitively difficult to verify that they are implemented correctly *in complex software*. In fact, there have been recurring reports of incorrect results in different types of solvers [3,13,19,34].

Instead of verifying correctness of an implementation, a more viable approach is to supplement the computed result with a *proof of correctness* [4,54] that can be independently and automatically verified. This approach has long been the standard in the SAT community [21,22,40,64], but there also exist proof-logging mechanisms for broader classes of problems such as in SMT solving [9,55], pseudo-boolean optimization [29,35], or exact MIP [14].

[⋆] The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (BMBF grant numbers 05M14ZAM, 05M20ZBM).

For MIP, no formal proof system has been developed to this date, and the solving techniques that are certifiable in [14] are severely limited, not covering any methods that remove parts of the feasible region from the search space, e.g., due to symmetry arguments. A seminal paper in this direction is [12], which describes a proof system to handle symmetry- and redundancy-based reasoning for pseudo-boolean optimization. Our work follows the same paradigm to develop a proof system that addresses the challenges posed by the presence of unbounded and continuous variables. As a result, we can certify the correctness of a wide range of static and dynamic, global and local symmetry handling methods, and many other state-of-the-art techniques from the MIP literature.

The overall strategy is to define a *configuration* as a snapshot of the derivations during the solving process. A trivial starting configuration is iteratively updated by a series of *rules* that are proven to establish valid derivations, eventually certifying optimality or infeasibility of $\mathrm{OPT}(f, F)$. To that end, in Sec. 2 we introduce the notion of a *consistent branching tree*, which is used to prevent inconsistencies among all derivations that remove parts of the feasible region and to define *validity* of a configuration. Vaguely speaking, any valid configuration admits the same optimal objective value as $\mathrm{OPT}(f, F)$. In Sec. 3, we present a set of validity-preserving transition rules that make it possible to certify optimality or infeasibility *sequentially*. In Sec. 4, we illustrate the framework for many MIP-techniques within LP-based branch-and-bound and discuss how certificates could be encoded and verified in practice. We conclude with an outlook in Sec. 5.

2 Trees, Configurations, and Validity

The central ingredient of our proof system, as in [12], is a reflexive and transitive relation on the solution space that defines a consistent direction for symmetry handling and optimality-based reductions. While in [12] this preorder remains mostly abstract, we provide a concrete construction designed to resolve the additional difficulties introduced by the presence of continuous and unbounded variables, and to capture the degrees of freedom due to locality in a branch-and-bound process. We require the following notational conventions.

In the abstract setting of Secs. 2 and 3, we consider constraints to be arbitrary subsets $C \in \mathcal{P}(\mathbb{R}^n)$, the power set of $\mathbb{R}^n$; $\overline{C} = \mathbb{R}^n \setminus C$ denotes its complement. For a set of constraints $\mathcal{C} \subseteq \mathcal{P}(\mathbb{R}^n)$, we write $\bigcap \mathcal{C}$ as a shorthand for $\bigcap_{C \in \mathcal{C}} C$. For $\tau = (i_1, \dots, i_k) \in \{\pm 1, \dots, \pm n\}^k$, we define by slight abuse of notation

$$\tau \colon \mathbb{R}^n \to \mathbb{R}^k, \tau(x) \coloneqq \big(\operatorname{sgn}(i_1)\, x_{|i_1|}, \dots, \operatorname{sgn}(i_k)\, x_{|i_k|}\big),$$

where $\operatorname{sgn}(\cdot)$ denotes the sign function, and $\tau(C) \coloneqq \{\tau(x) : x \in C\}$ for $C \subseteq \mathbb{R}^n$. We write $\tau \trianglelefteq \tau' = (i'_1, \dots, i'_{k'})$ if $i_j = i'_j$ for all $1 \leq j \leq k \leq k'$. We write $\delta^+(v)$ for the set of children of a node v in a directed graph, and $[a] \coloneqq \{j \in \mathbb{Z}_+ : j \leq a\}$.

Definition 1. *Let $\mathcal{C} \subseteq \mathcal{P}(\mathbb{R}^n)$, then we call $\mathcal{T} = (V, E, B, \sigma)$ a $\mathcal{C}$-*consistent branching tree *if the following conditions hold:*

(T1) (V, E) *is an arborescence with* $1 \leq |V| < \infty$ *and root* $r \in V$.
(T2) Each $v \in V$ *has attached a branching constraint* $B_v \subseteq \mathbb{R}^n$, *and* $B_r = \mathbb{R}^n$.
(T3) For all $u \in V$ *with* $\delta^+(u) \neq \emptyset$ *we have* $\bigcap \mathcal{C} \subseteq \bigcup_{v \in \delta^+(u)} B_v$.
(T4) Each $v \in V$ *has attached a list* $\sigma_v \in \{\pm 1, \dots, \pm n\}^{\ell_v}$, $\ell_v \in \{0, 1, \dots, n\}$, *of signed variable indices without duplicates in* $|(\sigma_v)_1|, \dots, |(\sigma_v)_{\ell_v}|$.
(T5) For all $u \in V$, $v \in \delta^+(u)$, *we have* $\sigma_u \trianglelefteq \sigma_v$.
(T6) For all $v \in V$ *and* $i \in [\ell_v]$, *we have* $\max\{\sigma_v(x)_i : x \in \bigcap \mathcal{C}\} < \infty$.
(T7) For all $u \in V$ *and* $v, w \in \delta^+(u)$, $v \neq w$, *we have* $\sigma_u(B_v) \cap \sigma_u(B_w) = \emptyset$.

For a node $u \in V$ with $\delta^+(u) \neq \emptyset$, (T3) and (T7) ensure that the children of u partition the feasible region of u; in particular, for every $x \in \bigcap \mathcal{C}$, there is a unique leaf of (V, E) whose feasible region contains x, denoted by $\operatorname{leaf}(x)$, see Lems. 1 and 2 in Sec. A.1. Hence, for $x, y \in \bigcap \mathcal{C}$, there exists a *deepest common node* $w =: \operatorname{dcn}(x, y)$ with x and y both contained in the feasible region of w. The partitioning condition (T7) is stronger and further implies that all $u \in V$ with $|\delta^+(u)| \geq 2$ have nontrivial σ_u. By (T5), also all successors v have $\ell_v \geq \ell_u \geq 1$. The boundedness condition (T6) holds if all variables contained in at least one σ_v have finite upper (lower) bound if contained with positive (negative) index.

Lem. 3 in Sec. A.1 shows that the relations $\succeq_{\epsilon,\mathcal{T}}$ and $\succ_{\epsilon,\mathcal{T}}$ defined next constitute a preorder resp. a strict order on the set of solutions in $\bigcap \mathcal{C}$:

Definition 2. *Let* $\mathcal{T}$ *be a* $\mathcal{C}$*-consistent branching tree,* $\epsilon > 0$, *and* $x, y \in \mathbb{R}^n$. *Let* $\succeq$ *and* $\succ_\epsilon$ *denote the standard resp. a strict lexicographic order given by*

$$x \succeq y :\Leftrightarrow x = y \vee x_i > y_i \text{ for the smallest index } i \text{ with } x_i \neq y_i,$$

$$x \succ_\epsilon y :\Leftrightarrow x \neq y \wedge x_i \geq y_i + \epsilon \text{ for the smallest index } i \text{ with } x_i \neq y_i.$$

Using $v = \operatorname{dcn}(x, y)$, *we then define the relations* $\succeq_{\epsilon,\mathcal{T}}$ *and* $\succ_{\epsilon,\mathcal{T}}$ *over* $\bigcap \mathcal{C}$ *by*

$$x \succeq_{\epsilon,\mathcal{T}} y :\Leftrightarrow \sigma_v(x) = \sigma_v(y) \vee \sigma_v(x) \succ_\epsilon \sigma_v(y),$$

$$x \succ_{\epsilon,\mathcal{T}} y :\Leftrightarrow \sigma_v(x) \succ_\epsilon \sigma_v(y).$$

With this, we arrive at the following central notion of a *configuration*:

Definition 3. *Let* $\mathcal{C}, \mathcal{D} \subseteq \mathcal{P}(\mathbb{R}^n)$, $g \colon \mathbb{R}^n \to \mathbb{R}$, $z \in \mathbb{R} \cup \{\infty\}$, $\epsilon > 0$, *and let* $\mathcal{T}$ *be a branching tree. A configuration* $\big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ *is called* (F, f)-valid *if:*

(V1) $\mathcal{T}$ *is* $\mathcal{C}$*-consistent and* $\epsilon > 0$.
(V2) If $z < \infty$, *there exists an* $x \in F$ *with* $f(x) \leq z$.
(V3) For all $\hat{z} < z$, *there exists an* $x \in F$ *with* $f(x) \leq \hat{z}$ *if and only if there exists an* $x \in \bigcap \mathcal{C}$ *with* $g(x) \leq \hat{z}$.
(V4) For every $x \in \bigcap \mathcal{C}$ *with* $g(x) < z$, *there exists a* $y \in \bigcap(\mathcal{C} \cup \mathcal{D})$ *with* $y \succeq_{\epsilon,\mathcal{T}} x$ *and* $g(y) \leq g(x)$.

As in [12], we call $\mathcal{C}$ the set of *core* and $\mathcal{D}$ the set of *derived constraints*. Thm. 1 states that the goal—to produce a valid configuration containing the contradiction $\emptyset \in \mathcal{D}$—yields a certificate of optimality (if $z < \infty$) or infeasibility (if $z = \infty$). While it seems out of reach to produce such a configuration and check its validity in one step, Thm. 2 provides a trivially valid starting configuration $\mathfrak{C}^0$. A proof in our system then becomes a *sequence of valid configurations* starting at $\mathfrak{C}^0$ and transitioning to richer configurations by iteratively applying a set of *machine-verifiable rules* that are *proven to preserve validity* in Sec. 3.

Theorem 1. *Let $\left(\mathcal{C},\mathcal{D},g,z,\mathcal{T},\epsilon\right)$ be an (F,f)-valid configuration with $\emptyset \in \mathcal{D}$. Then there exists no solution $x \in F$ with $f(x) < z$.*

Proof. Suppose there is an (F,f)-valid configuration $\left(\mathcal{C},\mathcal{D},g,z,\mathcal{T},\epsilon\right)$ with $\emptyset \in \mathcal{D}$, and $x \in F$ with $f(x) = \hat{z} < z$. By (V3), there is $\hat{x} \in \bigcap \mathcal{C}$ with $g(\hat{x}) \le \hat{z} < z$, and by (V4), there exists a $y \in \bigcap(\mathcal{C} \cup \mathcal{D})$. This contradicts $\bigcap(\mathcal{C} \cup \mathcal{D}) \subseteq \bigcap \mathcal{D} = \emptyset$. □

Theorem 2. *Let $\mathcal{F} \subseteq \mathcal{P}(\mathbb{R}^n)$, $F = \bigcap \mathcal{F}$, and $f\colon \mathbb{R}^n \to \mathbb{R}$. Then the* initial configuration $\mathfrak{C}^0 \coloneqq \left(\mathcal{F}, \emptyset, f, \infty, (\{r\}, \emptyset, \{r \mapsto \mathbb{R}^n\}, \{r \mapsto (\,)\}), 1\right)$ *is (F,f)-valid.*

Proof. (V2) holds due to $z = \infty$ and (V3) because $\bigcap \mathcal{C} = \bigcap \mathcal{F} = F$ and $g = f$. Since $\mathcal{D} = \emptyset$, we can always choose $y = x$ to satisfy (V4). Finally, the initial tree $(\{r\}, \emptyset, \{r \mapsto \mathbb{R}^n\}, \{r \mapsto (\,)\})$ consists only of a root node r with $B_r = \mathbb{R}^n$ and empty $\sigma_r = (\,)$. This tree trivially conforms to Def. 2, and $\epsilon = 1 > 0$. □

3 Validity-Preserving Transition Rules

The first two rules feature a special type of constraint called an *implication*. Let $\mathcal{A} \subseteq \mathcal{P}(\mathbb{R}^n)$ be a collection of *assumptions* and $C \subseteq \mathbb{R}^n$, then we define $[\mathcal{A} \rightsquigarrow C] \coloneqq \overline{\bigcap \mathcal{A}} \cup C$, i.e., $x \in [\mathcal{A} \rightsquigarrow C]$ if and only if $(\, x \in \bigcap \mathcal{A} \Rightarrow x \in C\,)$. In particular, any constraint C can be written as the implication $[\mathbb{R}^n \rightsquigarrow C]$.

Implicational derivation rule. This first rule allows to derive constraints that preserve all feasible, improving solutions in a subset of $\mathbb{R}^n$, i.e., under a (possibly empty) set of assumptions $\mathcal{A}$:

Theorem 3. *Let $\left(\mathcal{C},\mathcal{D},g,z,\mathcal{T},\epsilon\right)$ be an (F,f)-valid configuration. Let $C \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathcal{P}(\mathbb{R}^n)$. Then $\left(\mathcal{C},\mathcal{D} \cup \{[\mathcal{A} \rightsquigarrow C]\},g,z,\mathcal{T},\epsilon\right)$ is also (F,f)-valid, if*

$$C \supseteq \bigcap(\mathcal{C} \cup \mathcal{D} \cup \mathcal{A}) \cap \{x \in \mathbb{R}^n : g(x) < z\}. \tag{1}$$

Resolution rule. A particular form of the implicational derivation rule allows to derive new constraints by a disjunctive argument.

Corollary 1. *Let $\left(\mathcal{C},\mathcal{D},g,z,\mathcal{T},\epsilon\right)$ be an (F,f)-valid configuration. Consider sets A_1,A_2,C_1,$C_2 \subseteq \mathbb{R}^n$, and let $\mathcal{A}_1, \mathcal{A}_2$ be sets of constraints with*

$$[(\mathcal{A}_1 \cup \{A_1\}) \rightsquigarrow C_1]\,, [(\mathcal{A}_2 \cup \{A_2\}) \rightsquigarrow C_2] \in \mathcal{C} \cup \mathcal{D} \;\wedge\; A_1 \cup A_2 \supseteq \bigcap(\mathcal{C} \cup \mathcal{D}). \tag{2}$$

Then $\left(\mathcal{C},\mathcal{D} \cup \{[(\mathcal{A}_1 \cup \mathcal{A}_2) \rightsquigarrow (C_1 \cup C_2)]\},g,z,\mathcal{T},\epsilon\right)$ is also (F,f)-valid.

A typical example for resolution is the case where $\mathcal{A}_1 = \mathcal{A}_2$ contains the branching decisions up to a parent node in a search tree with exactly two children, and $C_1 = C_2$ is a constraint derived previously for both children.

Objective bound update rule. When an improving solution is known, this rule allows to update the value of z in a configuration:

Theorem 4. *Let $\big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and $x \in \bigcap \mathcal{C}$ with $g(x) = z' < z$. Then $\big(\mathcal{C}, \mathcal{D}, g, z', \mathcal{T}, \epsilon\big)$ is also (F, f)-valid.*

Objective function update rule. If the constraints imply equations, it may be desirable to substitute variables and update the objective function accordingly:

Theorem 5. *Let $\big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and $g' \colon \mathbb{R}^n \to \mathbb{R}^n$. If $\bigcap \mathcal{C} \subseteq \{x \in \mathbb{R}^n : g'(x) = g(x)\}$, then $\big(\mathcal{C}, \mathcal{D}, g', z, \mathcal{T}, \epsilon\big)$ is also (F, f)-valid.*

Redundance-based strengthening rule. The implicational derivation rule allows the removal of suboptimal solutions only once $z < \infty$ has been established by the objective bound update rule. The following rule allows the removal of feasible solutions independently of the objective bound if a so-called *witness* ω is known:

Theorem 6. *Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and let $C \subseteq \mathbb{R}^n$. If there exists an $\omega \colon \mathbb{R}^n \to \mathbb{R}^n$ such that*

$$\begin{aligned} x \in \bigcap(\mathcal{C} \cup \mathcal{D}) \wedge x \notin C \Rightarrow\ & \omega(x) \in \bigcap(\mathcal{C} \cup \mathcal{D} \cup \{C\}) \wedge \\ & \qquad \omega(x) \succeq_{\epsilon,\mathcal{T}} x \wedge g(\omega(x)) \leq g(x) \end{aligned} \tag{3}$$

holds for all $x \in \mathbb{R}^n$, then $\big(\mathcal{C}, \mathcal{D} \cup \{C\}, g, z, \mathcal{T}, \epsilon\big)$ is also (F, f)-valid.

Proof. (V1), (V2), and (V3) are invariant. To show that also (V4) continues to hold, let $x \in \bigcap \mathcal{C}$ with $g(x) < z$. We need to prove that there is a $y \in \bigcap(\mathcal{C} \cup \mathcal{D} \cup \{C\})$ with $y \succeq_{\epsilon,\mathcal{T}} x$ and $g(y) \leq g(x)$. By (V4) for $\mathfrak{C}$, there is $x' \in \bigcap(\mathcal{C} \cup \mathcal{D})$ with $x' \succeq_{\epsilon,\mathcal{T}} x$ and $g(x') \leq g(x)$. If $x' \in C$, we can choose $y = x'$. Otherwise, choose $y = \omega(x')$. Then (3) guarantees that $y \in \bigcap(\mathcal{C} \cup \mathcal{D} \cup \{C\})$, $y \succeq_{\epsilon,\mathcal{T}} x' \succeq_{\epsilon,\mathcal{T}} x$, and $g(y) \leq g(x') \leq g(x)$. By transitivity of $\succeq_{\epsilon,\mathcal{T}}$, this concludes the proof. □

Dominance-based strengthening rule. Thm. 6 is limited to cases where a single witness can repair removed solutions in one step such that they satisfy all core and derived constraints at once. The next rule allows more complex reasoning:

Theorem 7. *Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and let $C \subseteq \mathbb{R}^n$. If there exists an $\omega \colon \mathbb{R}^n \to \mathbb{R}^n$ such that*

$$x \in \bigcap(\mathcal{C} \cup \mathcal{D}) \wedge x \notin C \Rightarrow\ \omega(x) \in \bigcap \mathcal{C}\ \wedge\ \omega(x) \succ_{\epsilon,\mathcal{T}} x\ \wedge\ g(\omega(x)) \leq g(x) \tag{4}$$

holds for all $x \in \mathbb{R}^n$, then $\big(\mathcal{C}, \mathcal{D} \cup \{C\}, g, z, \mathcal{T}, \epsilon\big)$ is also (F, f)-valid.

Proof. Note that adding C to $\mathcal{D}$ keeps (V1), (V2), and (V3) invariant. To prove that (V4) holds if one replaces $\mathcal{D}$ by $\mathcal{D}' = \mathcal{D} \cup \{C\}$, let $x^0 \in \bigcap \mathcal{C}$ with $g(x^0) < z$. We need to show that there is $y \in \bigcap(\mathcal{C} \cup \mathcal{D}')$ with $y \succeq_{\epsilon,\mathcal{T}} x^0$ and $g(y) \leq g(x^0)$. We establish this by creating a (potentially infinite) sequence $(x^i)_{i=1}^k$ with

$$x^i := \begin{cases} y^i, & \text{if } i \text{ is odd for some } y^i \text{ from (V4) for } x = x^{i-1} \text{ in } \mathfrak{C}, \\ \omega(x^{i-1}), & \text{if } i \text{ is even,} \end{cases}$$

where $k = \inf\{i \in \mathbb{Z}_+ : x^i \in \bigcap(\mathcal{C} \cup \mathcal{D} \cup \{C\})\}$. This sequence is well-defined, which can be shown via induction by alternatingly applying (V4) and (4).

It suffices to prove $k < \infty$ as the alternating application of (V4) and (4) guarantees $x^k \succeq_{\epsilon,\mathcal{T}} x^{k-1} \succeq_{\epsilon,\mathcal{T}} \dots \succeq_{\epsilon,\mathcal{T}} x^0$ and $g(x^k) \leq g(x^{k-1}) \leq \dots \leq g(x^0)$, so we can choose $y = x^k$. To show finiteness, first note that (V4) and (4) imply

$$x^{2i} \succ_{\epsilon,\mathcal{T}} x^{2i-1} \succeq_{\epsilon,\mathcal{T}} x^{2i-2} \succ_{\epsilon,\mathcal{T}} \dots \succeq_{\epsilon,\mathcal{T}} x^2 \succ_{\epsilon,\mathcal{T}} x^1 \succeq_{\epsilon,\mathcal{T}} x^0$$

for all $i \in [k/2] = \{j \in \mathbb{Z}_+ : j \leq k/2\}$. Thus, Lem. 4 in the appendix shows $x^{2i} \succ_{\epsilon,\mathcal{T}} x^{2i-2}$ for all $i \in [k/2]$, i.e., for each $i \in [k/2]$, there is a $u_i \in V$ with $\sigma_{u_i}(x^{2i}) \succ_\epsilon \sigma_{u_i}(x^{2i-2})$.

For the sake of contradiction, suppose $k = \infty$. Since V in $\mathcal{T}$ is finite, also $\{\sigma_v : v \in V\}$ is finite. Transitivity of $\succ_{\epsilon,\mathcal{T}}$ and $k = \infty$ then implies that there exists $u \in V$ and a subsequence $(x^{i_j})_{j=1}^\infty$ with $\sigma_u(x^{i_{j+1}}) \succ_{\epsilon,\mathcal{T}} \sigma_u(x^{i_j})$ and $i_{j+1} > i_j \in 2\mathbb{Z}$ for all $j \in \mathbb{Z}_+$. Hence, for all $j \in \mathbb{Z}_+$, there is $t_j \in [\ell_u]$ such that

$$\sigma_u(x^{i_{j+1}})_{t_j} \geq \sigma_u(x^{i_j})_{t_j} + \epsilon \wedge \sigma_u(x^{i_{j+1}})_s = \sigma_u(x^{i_j})_s \text{ for all } s \in [t_j - 1].$$

Since ℓ_u is finite, there exists an index t that infinitely often determines the lexicographic difference. Among all such indices t, let t' be the smallest one.

By (T6), all entries in $\sigma_u(x)$, $x \in \bigcap\mathcal{C}$, are bounded from above. Hence, there must exist infinitely many j with $\sigma_u(x^{i_{j+1}})_{t'} < \sigma_u(x^{i_j})_{t'}$; otherwise, we would have $\lim_{j\to\infty} \sigma_u(x^{i_j})_{t'} = \infty$. For all such j, due to $\sigma_u(x^{i_{j+1}}) \succ_{\epsilon,\mathcal{T}} \sigma_u(x^{i_j})$, there must exist $s \in [t'-1]$ with $\sigma_u(x^{i_{j+1}})_s \geq \sigma_u(x^{i_j})_s + \epsilon$. This is a contradiction to t' being minimal. Consequently, $k < \infty$. This concludes the proof. □

In comparison to redundance-based strengthening, dominance-based strengthening is both weaker in the sense that $\omega(x)$ does not need to satisfy $\mathcal{D}$ and C directly, and stricter in the sense that ω must result in an increase w.r.t. the strict order $\succ_{\epsilon,\mathcal{T}}$. Note that in both rules, C may be an implication $[\mathcal{A} \rightsquigarrow C]$, e.g., a locally valid symmetry-breaking constraint for some node $v \in V$.

Epsilon shrinkage rule. The presence of ϵ in (V4) ensures consistent progress in the proof of Thm. 7. It needs to be fixed for each dominance-based strengthening, but intermediately, ϵ can be decreased to an arbitrarily small, positive value:

Theorem 8. *Let $(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon)$ be an (F, f)-valid configuration. If $0 < \epsilon' < \epsilon$, then $(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon')$ is also (F, f)-valid.*

Transfer rule. Derived constraints can be upgraded to core constraints. This may restrict future applications of dominance-based strengthening, but can be useful to facilitate an objective function update or the tree exchange rule below:

Theorem 9. *Let $(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon)$ be an (F, f)-valid configuration and let $C \in \mathcal{D}$. Then $(\mathcal{C} \cup \{C\}, \mathcal{D} \setminus \{C\}, g, z, \mathcal{T}, \epsilon)$ is also (F, f)-valid.*

Deletion rule. This rule allows to remove derived and redundant core constraints:

Theorem 10. *Let $\big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration, let $\mathcal{C}' \subseteq \mathcal{C}$ and $\mathcal{D}' \subseteq \mathcal{D}$. Then $\big(\mathcal{C}', \mathcal{D}', g, z, \mathcal{T}, \epsilon\big)$ is also (F, f)-valid, if (a) $\mathcal{C}' = \mathcal{C}$, or (b) $\mathcal{C}' = \mathcal{C} \setminus \{C\}$ for $C \supseteq \bigcap \mathcal{C}'$, or (c) $\mathcal{C}' = \mathcal{C} \setminus \{C\}$, $\sigma_v = (\,)$ for all $v \in V$, and $C \subseteq \mathbb{R}^n$ is derivable from $\big(\mathcal{C}', \emptyset, g, z, \mathcal{T}, \epsilon\big)$ by redundance-based strengthening.*

Tree exchange rule. The preorder $\succeq_{\epsilon,\mathcal{T}}$ based on the tree $\mathcal{T}$ is essential in guaranteeing that a valid configuration cannot contain contradictory derivations across different applications of redundance- or dominance-based strengthening. The following rule allows to install a new tree before any constraints have been derived, or when all derived constraints have been deleted or transferred to $\mathcal{C}$:

Theorem 11. *Let $\big(\mathcal{C}, \emptyset, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and let $\mathcal{T}'$ be a $\mathcal{C}$-consistent branching tree. Then $\big(\mathcal{C}, \emptyset, g, z, \mathcal{T}', \epsilon\big)$ is also (F, f)-valid.*

Dimension extension rule. Last, but not least, the dimension of the problem can be increased by introducing new variables that (initially) do not feature in any of the constraints, the objective function, or the preorder-defining tree:

Theorem 12. *Let $\big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be an (F, f)-valid configuration and let $\mathcal{C}' = \{C \times \mathbb{R} : C \in \mathcal{C}\}$, $\mathcal{D}' = \{C \times \mathbb{R} : C \in \mathcal{D}\}$, $g' \colon \mathbb{R}^{n+1} \to \mathbb{R}, g'(x_1, \ldots, x_{n+1}) = g(x_1, \ldots, x_n)$, and $\mathcal{T}' = (V, E, B', \sigma)$ with $B'_v = B_v \times \mathbb{R}$ for all $v \in V$. Then $\big(\mathcal{C}', \mathcal{D}', g', z, \mathcal{T}', \epsilon\big)$ is also (F, f)-valid.*

This allows to create extended formulations that help to derive new constraints, typically by redundance-based strengthening, as discussed in the next section.

4 Certificates for mixed-integer programs

Let (P) denote a mixed-integer program $\min\{c^T x : Ax \leq b,\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}$, where $m, p, n \geq 0$ are integers, $p \leq n$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, and $c \in \mathbb{Q}^n$. In the following section, we show how many popular MIP techniques can be certified within the framework of Secs. 2 and 3 and automatically verified.

Constraints in the abstract proof system were left to be arbitrary subsets of $\mathbb{R}^n$. In the MIP setting, the sets of core constraints $\mathcal{C}$ and derived constraints $\mathcal{D}$ consist of integrality restrictions on variables and linear inequalities or implications $[\mathcal{A} \rightsquigarrow C]$, where C and each $A \in \mathcal{A}$ is a linear inequality. In the initial configuration of Thm. 2, $\mathcal{C} = \mathcal{F}$ contains the rows of $Ax \leq b$, as well as $x_j \in \mathbb{Z}$ for all $j \in [p]$. The objective function is a linear function $f(x) = c^T x$. In $\mathcal{T}$, the branching decisions B_v, $v \in V$, are also linear inequalities.

While a configuration does not explicitly contain the value of a dual bound, it is natural that an LP-based branch-and-bound solver would maintain in $\mathcal{D}$ a set of implications $[\mathcal{A} \rightsquigarrow g(x) \geq d_{\mathcal{A}}]$ per subproblem $\mathcal{A}$, derived by the implicational derivation rule using aggregation of constraints with dual multipliers, and apply the resolution rule in order to derive globally valid dual bounds from these, much like in [14].

4.1 Presolving, Propagation, and Branch-and-Cut

A *cutting plane* for (P) is an inequality $\alpha^T x \leq \beta$ that is valid for the feasible region of (P), while separating some infeasible point x^*. Cuts that are not valid globally can be represented as an implication $[\mathcal{A} \rightsquigarrow C]$, where $\mathcal{A}$ contains the local constraints for which the cut is valid. Hence, it is clear that any cutting plane can—in the abstract proof system—be certified using the implication rule presented in Thm. 3. However, the complexity of this in general as hard as solving (P) [43]. Still, for a wide variety of cutting planes, efficient verification is possible if some additional information is provided.

Aggregation. If there exist multipliers $\lambda_i \geq 0$ such that $C \equiv \sum_i \lambda_i C_i$ for some linear constraints C_i, then C is redundant, and therefore the implication is trivially satisfied. It is clear that if the indices and multipliers are specified, a verifier can efficiently check that $C \equiv \sum_i \lambda_i C_i$ holds. If the multipliers are not specified, a verifier can solve an LP to find the correct multipliers and indices [27]. Therefore, we can efficiently verify the validity of cuts that are derived by aggregation.

Disjunctive cuts. A disjunctive cut is a cut $\alpha^T x \leq \beta$ for which there exists a disjunction $(\pi^T x \leq \pi_0) \vee (\pi^T x \geq \pi_0 + 1)$ such that the inequality is valid for both $\{x : Ax \leq b, \pi^T x \leq \pi_0\}$ and $\{x : Ax \leq b, \pi^T x \geq \pi_0 + 1\}$ [7,18,46]. Moreover, for all variables x_i that are not integer π_i is required to be 0. The verification for this type of cut is possible if the correct disjunction is provided in the certificate, which allows for a proof by aggregation for both sides of the disjunction. Afterwards, the validity of the complete cut can be verified by applying the resolution rule, see [27] for details. As shown in [20], an efficient procedure for verifying disjunctive cuts allows to verify many families of cuts including Chvátal-Gomory cuts [15], Gomory mixed-integer cuts [42], mixed-integer rounding cuts [56], or lift-and-project cuts [8]. In the appendix, we also provide examples for verification of simple knapsack [6] and flowcover cuts [60].

Extended Formulations. There are other types of cuts that cannot be characterized as a split cut, e.g., reformulation-linearization cuts [63]. However, using the dimension extension rule in combination with conic combinations of constraints, the needed extended formulation can be produced in the certificate.

Presolving and Propagation. Presolving is a vital part of solving MIPs [2,32,62], and often certifying a presolving step is straightforward. For example, if a variable bound is tightened by *constraint propagation* [62], the new variable bound can be verified by aggregation of existing constraints. Concretely, given some constraint $\alpha^T x \leq \beta$, and assume finite bounds on all variables $\ell_i \leq x_i \leq u_i, i \in [n]$, an activity-based upper bound for x_i (assuming $\alpha_i > 0$) can be computed as $x_i \leq {}^{(\beta - \sum_{\alpha_j \geq 0, j \neq i} \alpha_j \ell_j - \sum_{\alpha_j < 0, j \neq i} \alpha_j u_j)}/_{\alpha_i}$. An automatically verifiable proof by implicational derivation is to aggregate the original inequality $\alpha^T x \leq \beta$ with $-\alpha_j(x_j \leq u_j)$ for $\alpha_j < 0$ and with $\alpha_j(x_j \geq \ell_j)$ for $\alpha_j \geq 0$. Scaling the resulting inequality by $1/\alpha_i$ yields a proof of the new bound. Similar arguments can be made for many other presolving reductions, e.g., probing [2]. In the following, we mention several key presolving techniques that actively exclude feasible solutions of the original problem $\mathrm{OPT}(f, F)$.

Reduced cost fixing. This core technique dates back to the seminal paper of Dantzig, Fulkerson, and Johnson on the traveling salesman problem [23], and allows to tighten the bounds of a variable using information about the primal bound, the LP objective value, and the dual multipliers of the variable's bound constraints. Reduced cost fixing can be verified using implicational derivation (with $z < \infty$), see Sec. A.3.

Dominated columns. Dominated columns presolving [5, 33] fixes a variable x_k to one of its bounds, if there is another variable x_j with $c_j \leq c_k$, $a_{*j} \leq a_{*k}$, and x_j is integer whenever x_k is. If additionally local (i.e., derived) constraints need to be considered, we assume that they were added to $A \leq b$ for the sake of this paragraph. Such a fixing can be certified by the redundance-based strengthening rule. For the sake of simplicity, assume that x_j and x_k are bounded below by 0 and that x_j has no upper bound; in general, any lower bound and an implied upper bound is possible. Then, a witness is given by $\omega\colon \mathbb{R}^n \to \mathbb{R}^n$ with

$$\omega(x)_i = \begin{cases} x_i, & \text{if } i \notin \{j,k\}, \\ 0, & \text{if } i = k, \\ x_i + x_k, & \text{if } i = j. \end{cases}$$

To verify that (3) of Thm. 6 is satisfied for ω, it is straightforward to check that $c^T\omega(x) \leq c^T x$. As $a_{*j} \leq a_{*j}$, holds, any constraint satisfied by x is also satisfied by $\omega(x)$. This can be automatically verified with an aggregation proof similar as for constraint propagation. Furthermore, $\omega(x)$ trivially satisfies the new constraint $x_k = 0$. In presolving the tree $\mathcal{T}$ will typically be the initial tree, hence $\omega(x) \succeq_{\epsilon,\mathcal{T}} x$ is a tautology.

4.2 Symmetry Handling

Let γ be a permutation of $[n]$. For $x \in \mathbb{R}^n$, let $\gamma(x) = (x_{\gamma^{-1}(1)}, \dots, x_{\gamma^{-1}(n)})$. A permutation γ is a *symmetry* of a MIP (P) if $x \in \mathbb{R}^n$ is feasible for (P) if and only if $\gamma(x)$ is feasible, and $c^T x = c^T \gamma(x)$. The symmetries of a MIP form a group $\mathit{\Gamma}$, which is a subgroup of the symmetric group on $[n]$, denoted by $\mathcal{S}_n$.

Although this definition of symmetries is natural, finding all such symmetries is NP-hard [53]. One therefore usually restricts to symmetries that keep the MIP formulation invariant. That is, $\gamma \in \mathcal{S}_n$ is a *formulation symmetry* if there is a permutation $\pi \in \mathcal{S}_m$ of the constraints such that $\pi(b) = b$, $\gamma(c) = c$, $A_{\pi^{-1}(i),\gamma^{-1}(j)} = A_{i,j}$ for all $(i,j) \in [m] \times [n]$, and the variable types are preserved. They have the advantage that it is easy to verify if a pair (γ, π) defines a formulation symmetry, i.e., we can use them for verifying symmetry reductions.

To handle (formulation) symmetries $\mathit{\Gamma}$, many techniques have been discussed [24, 30, 44, 45, 48, 49, 51, 52, 58], and we show that four popular symmetry handling techniques can be derived via our framework: orbital branching [59], orbitopal reduction [10, 47], Schreier-Sims cuts (SST cuts) [50, 61], and lexicographic order based reductions [11, 31]. As shown in [25], these methods are compatible with adding constraints $\sigma_v(x) \succeq \sigma_v(\gamma(x))$ for all $\gamma \in \mathit{\Gamma}$ at the nodes v

of a consistent branching tree $\mathcal{T}$, the entries of σ_v being positive. Thus, to show that the aforementioned methods fit into our framework, it is sufficient to derive a representation of $\sigma_v(x) \succeq \sigma_v(\gamma(x))$ via linear inequalities for suitable σ_v.

There are many degrees of freedom when building σ_v, $v \in V$. One can select, e.g., (a) some permutation π of $[n]$ and assign $\sigma_v = \pi$ for all $v \in V$, or (b) if $v \in V \setminus \{r\}$ arises from its parent u by branching on x_i, one can extend σ_u by i if i is not present in σ_u, yet. Approach (a) creates globally valid symmetry handling inequalities (SHIs); approach (b) generates local SHIs that are adapted to the branching history, allowing to derive reductions earlier, cf. the discussion in [53]. As long as the tree is $\mathcal{C}$-consistent, all choices of σ_v allow to handle symmetries. Specialized symmetry handling methods might require a special structure of σ_v though, see below.

Lexicographic order based reduction. A classic approach to handle symmetries is to compute only solutions being lexicographic maximal representatives among a class of symmetric solutions. This can be enforced by deriving reductions from the relation $\sigma_v(x) \succeq \sigma_v(\gamma(x))$. For integer variables with bounded domain contained in $[L, U]$ of size $D = U - L$, this relation can be linearly expressed as

$$\sum_{i=1}^{\ell_v} (D+1)^{\ell_v - i} \cdot \sigma_v(x)_i \geq \sum_{i=1}^{\ell_v} (D+1)^{\ell_v - i} \cdot \sigma_v(\gamma(x))_i, \tag{5}$$

see, e.g., [31]. Prop. 1 in Sec. A.1 shows that (5) can be derived as local constraint at node v within our framework via dominance-based strengthening by selecting $\omega = \gamma$. Denote the corresponding derived constraint as $C_v^\gamma = [\mathcal{B}_v \rightsquigarrow (5)]$, where $\mathcal{B}_v = \{B_u : u \text{ is a node on the } r\text{-}v\text{-path in } \mathcal{T}\}$.

Typical reductions that are derived from $\sigma_v(x) \succeq \sigma_v(\gamma(x))$ are variable fixings in the binary case or variable bound tightenings in the integer case. Both type of reductions can be encoded as linear constraints. If a solver finds a lexicographic order based reduction at node v, e.g., $x_i \leq u_i$ for some $i \in [n]$ and $u_i \in \mathbb{Z}$, then this is the case because there is no $\bar{x}$ that is feasible for the MIP and satisfies $\bar{x}_i \geq u_i + 1$ and (5). Such a reduction can be derived in our framework by first deriving C_v^γ. In a second step, a CG-proof can be used to show that

$$\left\{ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \cap \bigcap \mathcal{B}_v : Ax \leq b,\ x_i \geq u_i + 1,\ x \text{ satisfies } (5) \right\} = \emptyset.$$

As CG-proofs are compatible with our framework, also $x_i \leq u_i$ can be derived.

Orbitopal reduction. While the previously discussed reductions are based on a single permutation, orbitopal reduction handles symmetries of an entire specially structured group $\varGamma$. Orbitopal reduction assumes the integer variables of a MIP to be arranged in an $s \times t$ matrix X and $\varGamma$ to operate on the variables by exchanging the columns of X, i.e., $\varGamma$ acts on the columns like $\mathcal{S}_t$. As lexicographic order based reductions, orbitopal reduction derives variable domain reductions.

In [25], we discuss that orbitopal reduction is implied by $\sigma_v(x) \succeq \sigma_v(\gamma(x))$ for all $\gamma \in \varGamma$, if σ_v can be partitioned into consecutive blocks of length t such that each block corresponds to exactly one row of X. Variable domain reductions, e.g.,

$X_{i,j} \leq u_{i,j}$ for some $(i,j) \in [s] \times [t]$ with $(i,j) \in \sigma_v$, can be derived as above: If all variables in X are bounded and integer, we derive (5) for all $\gamma \in \mathit{\Gamma}$ and use a CG-proof to show that no MIP solution satisfies $X_{i,j} \geq u_{i,j} + 1$. Deriving all inequalities (5) is intractable though as $\mathit{\Gamma}$ contains $t!$ permutations. But, in fact, it is sufficient to derive (5) for the $t-1$ permutations that swap variables of adjacent columns to derive the reductions of orbitopal reduction, see [45].

SST cuts. Let $i \in [n]$ and $\mathrm{orb}_{\mathit{\Gamma}}(i) = \{\gamma(i) : \gamma \in \mathit{\Gamma}\}$ be its *orbit*. Simple SHIs are $x_i \geq x_j$ for $j \in \mathrm{orb}_{\mathit{\Gamma}}(i)$ [49], but SHIs for different orbits can be incompatible. Compatibility can be ensured by adding these SHIs in rounds $i \in [n]$ for different $\mathit{\Gamma}_i$ per round [50, 61]. For $\hat{\sigma} \in \mathcal{S}_n$, round i adds $x_{\hat{\sigma}(i)} \geq x_j$ for all $j \in \mathrm{orb}_{\mathit{\Gamma}_i}(\hat{\sigma}(i))$, and $\mathit{\Gamma}_i = \{\gamma \in \mathit{\Gamma}_{i-1} : \gamma(\hat{\sigma}(j)) = \hat{\sigma}(j),\ j \in [i-1]\}$, where $\mathit{\Gamma}_0 = \mathit{\Gamma}$. These SHIs (called SST cuts) can be derived from $\hat{\sigma}(x) \succeq \hat{\sigma}(\gamma(x))$, $\gamma \in \mathit{\Gamma}$ [25].

SST cuts can be defined for arbitrary variable types. Via dominance-based strengthening a slightly weaker family of constraints, namely $x_i \geq x_j - \epsilon$ instead of $x_i \geq x_j$, can be derived. If $\epsilon < 1$ and the involved variables are integral, a rounding argument can be used to also derive $x_i \geq x_j$. For continuous variables, $x_i \geq x_j - \epsilon$ can be gradually made stronger by the epsilon shrinkage rule.

Orbital branching. To handle symmetries in binary programs, this branching rule creates two child nodes [59]. In one node, it enforces $x_i = 1$ for some $i \in [n]$; in the other node, it enforces $x_j = 0$ for all $j \in \mathrm{orb}_{\mathit{\Gamma}'}(i)$ for some subgroup $\mathit{\Gamma}'$ of $\mathit{\Gamma}$. These reductions can be derived from SST cuts [25] if σ_v is adapted to the branching decisions (cf. the discussion on σ above). Since SST cuts can be derived in our framework, so can the reductions by orbital branching.

4.3 Encoding and Verification

For pure binary programming, the feasibility of implementing and applying the seminal proof system in [12], which is the foundation of our work, has already been demonstrated with great success [29, 35–39]. For general MIP, the certificate system [14] provides a blueprint for encoding and verifying feasibility reasoning [26–28]. In the following, let us point out differences (and similarities) to both of these that need to be addressed when approaching an actual implementation of a verifier for MIP.

Both types of constraints in MIP are easily encoded: linear inequalities and integrality conditions. The latter may be part of the original model or derived from equations (represented natively or as pairs of inequalities) via integrality of coefficients and right-hand sides. But unlike in the pure binary case, locally valid inequalities over unbounded variables cannot generally be recast as globally valid inequalities. A suitable verifier hence needs a native notion of implications $[\mathcal{A} \rightsquigarrow C]$ as in [14]. As witnesses ω, affine maps $x \mapsto Qx + q$, $Q \in \mathbb{Q}^{n \times n}$, $q \in \mathbb{Q}^n$, are both sufficient and easy to encode. In many cases, Q will be a sparse permutation matrix, for which only off-diagonal elements need to be specified.

As described in [12], a preorder $x \succeq y$ on binary vectors can be elegantly encoded and verified as a system of linear inequalities in x and y. This is not possible in the presence of continuous and unbounded variables. Instead, the branching tree $\mathcal{T} = (V, E, B, \sigma)$ must be encoded as a generic graph structure with unique identifiers for B_v and σ_v attached to the nodes $v \in V$. To verify $\mathcal{C}$-consistency, checking (T1), (T2), (T4), and (T5) is elementary. Checking (T3) and (T7) is also simple when B_v are halfspaces given by linear inequalities. If the boundedness condition (T6) does not follow trivially from bounds on the variables, then it is the task of the certifier to provide derivations for the boundedness and transfer them to the core before installing the tree via Thm. 11.

The tree is then used to evaluate $\omega(x) \succeq_{\epsilon,\mathcal{T}} x$ in (3) and (4). This can be performed by starting at the root node and diving in the branching tree as long as x and $\omega(x)$ are known to be contained in the same child. Here, x is a partial solution with fixed values or tightened bounds according to the precondition $x \in \bigcap(\mathcal{C} \cup \mathcal{D}) \wedge x \notin C$ and elementary propagations of these constraints. If all variables x are fixed, then $\mathrm{dcn}(x, \omega(x))$ and the result of $\omega(x) \succeq_{\epsilon,\mathcal{T}} x$ can always be determined. For partial solutions, this evaluation may fail, but all techniques discussed in Sec. 4 allow for a witness ω such that it succeeds. Some of these techniques may require so-called subproofs, i.e., preliminary derivations to strengthen the preconditions, which can afterwards be deleted again from $\mathcal{D}$. One example is the inductive derivation of (5) given in Prop. 1 in Sec. A.1.

Finally, in its most basic form, a verifier may require the certifier to provide detailed justification for each derivation. In an advanced implementation, finding some of these justifications can be automated by techniques like reverse unit propagation [29, 41], computing dual multipliers by a rational LP solver [27], or heuristically detecting witnesses from the variable indices in the encoding of σ.

5 Outlook

In Sec. 4, we have outlined how the proof system presented in Secs. 2 and 3 covers a wide range of important methods implemented in state-of-the-art MIP solvers today, having to stop short only of a discussion of lifting techniques and infeasibility analysis, both of which typically require just implicational reasoning.

Beyond this, we are convinced that this system can be generalized further to make it more convenient to use and potentially (dis)cover different algorithms: (a) The σ_v may contain duplicate entries, and more generally could signify any set of dimension-reducing mappings for which a suitably generalized extension relation $\trianglerighteq$ holds. (b) The lexicographic order $\succeq$ used to compare $\sigma_v(x)$ could become any preorder with an ϵ-strict version that allows only finitely many strict increases on the bounded image spaces of the σ_v. (c) The need to install a fixed branching tree only known *after* solving could be eliminated by providing an update rule that certifies a dynamic refinement of $\mathcal{T}$ *while* solving. We know that such a rule in the style of Thm. 6 can be proven and works for the most common scenario of binary variable-based branching schemes, but a more general version seems to need inductive reasoning as in Thm. 7 and requires further research.

Acknowledgements. We wish to thank Alexander Hoen, Andy Oertel, and Jakob Nordström for many patient and insightful discussions on proof logging for binary programs.

References

1. Achterberg, T.: Constraint Integer Programming. Doctoral thesis, Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften, Berlin (2007). https://doi.org/10.14279/depositonce-1634
2. Achterberg, T., Bixby, R., Gu, Z., Rothberg, E., Weninger, D.: Presolve reductions in mixed integer programming. INFORMS Journal on Computing **32**, 473–506 (2019). https://doi.org/10.1287/ijoc.2018.0857
3. Akgun, O., Gent, I.P., Jefferson, C., Miguel, I., Nightingale, P.: Metamorphic testing of constraint solvers. In: International Conference on Principles and Practice of Constraint Programming (2018). https://doi.org/10.1007/978-3-319-98334-9_46
4. Alkassar, E., Böhme, S., Mehlhorn, K., Rizkallah, C., Schweitzer, P.: An introduction to certifying algorithms. it - Information Technology **53**(6), 287–293 (2011). https://doi.org/10.1524/itit.2011.0655
5. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Math. Program. **71**(2), 221245 (1995). https://doi.org/10.1007/BF01586000
6. Balas, E.: Facets of the knapsack polytope. Mathematical Programming **8**, 146–164 (1975). https://doi.org/10.1007/BF01580440
7. Balas, E.: Disjunctive programming. In: Hammer, P., Johnson, E., Korte, B. (eds.) Discrete Optimization II, Annals of Discrete Mathematics, vol. 5, pp. 3–51. Elsevier (1979). https://doi.org/10.1016/S0167-5060(08)70342-X
8. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math. Program. **58**(3), 295–324 (1993)
9. Barbosa, H., Reynolds, A., Kremer, G., Lachnitt, H., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Viswanathan, A., Viteri, S., Zohar, Y., Tinelli, C., Barrett, C.: Flexible proof production in an industrial-strength SMT solver. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning. pp. 15–35. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10769-6_3
10. Bendotti, P., Fouilhoux, P., Rottner, C.: Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem. Mathematical Programming **186**, 337–372 (2021). https://doi.org/10.1007/s10107-019-01457-1
11. Bestuzheva, K., Besançon, M., Chen, W.K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S.J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M.E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., Witzig, J.: Enabling research through the SCIP Optimization Suite 8.0. ACM Trans. Math. Softw. **49**(2) (2023). https://doi.org/10.1145/3585516
12. Bogaerts, B., Gocht, S., McCreesh, C., Nordström, J.: Certified dominance and symmetry breaking for combinatorial optimisation. J. Artif. Intell. Res. **77**, 1539–1589 (2023). https://doi.org/10.1613/JAIR.1.14296
13. Brummayer, R., Lonsing, F., Biere, A.: Automated testing and debugging of SAT and QBF solvers. In: Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing. p. 4457. SAT'10, Springer-Verlag, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14186-7_6
14. Cheung, K.K.H., Gleixner, A., Steffy, D.E.: Verifying integer programming results. In: Eisenbrand, F., Koenemann, J. (eds.) Integer Programming and Combinatorial Optimization. pp. 148–160. Springer International Publishing, Cham (2017)

15. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **4**(4), 305337 (1973). https://doi.org/10.1016/0012-365X(73)90167-2
16. Conforti, M., Cornuéjols, G., Zambelli, G.: Integer Programming. Springer (2014). https://doi.org/10.1007/978-3-319-11008-0
17. Conforti, M., Cornuéjols, G., Zambelli, G.: Integer programming, vol. 271. Springer (2014). https://doi.org/10.1007/978-3-319-11008-0
18. Cook, W., Kannan, R., Schrijver, A.: Chvátal closures for mixed integer programming problems. Math. Program. **47**(2), 155174 (1990). https://doi.org/10.1007/BF01580858
19. Cook, W.J., Koch, T., Steffy, D.E., Wolter, K.: A hybrid branch-and-bound approach for exact rational mixed-integer programming. Math. Program. Comput. **5**(3), 305–344 (2013). https://doi.org/10.1007/S12532-013-0055-6
20. Cornuéjols, G., Li, Y.: Elementary closures for integer programs. Operations Research Letters **28**(1), 1–8 (2001). https://doi.org/10.1016/S0167-6377(00)00067-5
21. Cruz-Filipe, L., Heule, M.J.H., Hunt, W.A., Kaufmann, M., Schneider-Kamp, P.: Efficient certified RAT verification. In: de Moura, L. (ed.) Automated Deduction – CADE 26. pp. 220–236. Springer International Publishing, Cham (2017)
22. Cruz-Filipe, L., Marques-Silva, J., Schneider-Kamp, P.: Efficient certified resolution proof checking. In: Proceedings, Part I, of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 10205. p. 118135. Springer-Verlag, Berlin, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54577-5_7
23. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Journal of the Operations Research Society of America **2**(4), 393–410 (1954). https://doi.org/10.1287/opre.2.4.393
24. van Doornmalen, J., Hojny, C.: Efficient propagation techniques for handling cyclic symmetries in binary programs (2022), (preprint at https://arxiv.org/abs/2203.00992)
25. van Doornmalen, J., Hojny, C.: A unified framework for symmetry handling (2023). https://doi.org/10.48550/arXiv.2211.01295
26. Eifler, L., Gleixner, A.: A computational status update for exact rational mixed integer programming. Mathematical Programming **197**, 793–812 (2023). https://doi.org/10.1007/s10107-021-01749-5
27. Eifler, L., Gleixner, A.: Safe and verified Gomory mixed integer cuts in a rational MIP framework (2023), accepted for SIAM Journal on Optimization.
28. Eifler, L., Gleixner, A., Pulaj, J.: A safe computational framework for integer programming applied to Chvátal's conjecture. ACM Transactions on Mathematical Software **48**(2), 1–12 (2022). https://doi.org/10.1145/3485630
29. Elffers, J., Gocht, S., McCreesh, C., Nordström, J.: Justifying all differences using pseudo-boolean reasoning. Proceedings of the AAAI Conference on Artificial Intelligence **34**(02), 1486–1494 (2020). https://doi.org/10.1609/aaai.v34i02.5507
30. Fischetti, M., Liberti, L.: Orbital shrinking. In: Mahjoub, A.R., Markakis, V., Milis, I., Paschos, V.T. (eds.) Combinatorial Optimization, Lecture Notes in Computer Science, vol. 7422, pp. 48–58. Springer Berlin Heidelberg (2012)
31. Friedman, E.J.: Fundamental domains for integer programs with symmetries. In: Dress, A., Xu, Y., Zhu, B. (eds.) Combinatorial Optimization and Applications, Lecture Notes in Computer Science, vol. 4616, pp. 146–153. Springer Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-73556-4_17
32. Fügenschuh, A., Martin, A.: Computational integer programming and cutting planes. In: Aardal, K., Nemhauser, G., Weismantel, R. (eds.) Discrete Optimiza-

tion, Handbooks in Operations Research and Management Science, vol. 12, pp. 69–121. Elsevier (2005). https://doi.org/10.1016/S0927-0507(05)12002-7
33. Gamrath, G., Koch, T., Martin, A., Miltenberger, M., Weninger, D.: Progress in presolving for mixed integer programming. Mathematical Programming Computation **7**, 367–398 (2015). https://doi.org/10.1007/s12532-015-0083-5
34. Gillard, X., Schaus, P., Deville, Y.: Solvercheck: Declarative testing of constraints. In: Schiex, T., de Givry, S. (eds.) Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11802, pp. 565–582. Springer (2019). https://doi.org/10.1007/978-3-030-30048-7_33
35. Gocht, S.: VeriPB. 10.5281/zenodo.3548582 (2019), version 0.1.0
36. Gocht, S., Martins, R., Nordström, J., Oertel, A.: Certified CNF Translations for Pseudo-Boolean Solving. In: Meel, K.S., Strichman, O. (eds.) 25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 236, pp. 16:1–16:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.SAT.2022.16
37. Gocht, S., McBride, R., McCreesh, C., Nordström, J., Prosser, P., Trimble, J.: Certifying solvers for clique and maximum common (connected) subgraph problems. In: Simonis, H. (ed.) Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Proceedings. pp. 338–357. Lecture Notes in Computer Science, Springer (2020). https://doi.org/10.1007/978-3-030-58475-7_20
38. Gocht, S., McCreesh, C., Nordström, J.: Subgraph isomorphism meets cutting planes: Solving with certified solutions. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. IJCAI'20 (2021)
39. Gocht, S., Nordström, J.: Certifying parity reasoning efficiently using pseudo-boolean proofs. Proceedings of the AAAI Conference on Artificial Intelligence **35**(5), 3768–3777 (2021). https://doi.org/10.1609/aaai.v35i5.16494
40. Goldberg, E., Novikov, Y.: Verification of proofs of unsatisfiability for CNF formulas. In: Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1. p. 10886. DATE '03, IEEE Computer Society, USA (2003)
41. Goldberg, E., Novikov, Y.: Verification of proofs of unsatisfiability for cnf formulas. In: Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1. p. 10886. DATE '03, IEEE Computer Society, USA (2003)
42. Gomory, R.E.: An algorithm for the mixed integer problem. Tech. rep., RAND Corporation, Santa Monica, CA (1960)
43. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, Algorithms and Combinatorics, vol. 2. Springer (1988). https://doi.org/10.1007/978-3-642-97881-4
44. Hojny, C.: Packing, partitioning, and covering symresacks. Discrete Applied Mathematics **283**, 689–717 (2020). https://doi.org/10.1016/j.dam.2020.03.002
45. Hojny, C., Pfetsch, M.E.: Polytopes associated with symmetry handling. Mathematical Programming **175**, 197–240 (2019). https://doi.org/10.1007/s10107-018-1239-7
46. Jeroslow, R.: Cutting-plane theory: Disjunctive methods. In: Hammer, P., Johnson, E., Korte, B., Nemhauser, G. (eds.) Studies in Integer Programming, Annals of Discrete Mathematics, vol. 1, pp. 293–330. Elsevier (1977). https://doi.org/10.1016/S0167-5060(08)70741-6
47. Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. Discrete Optimization **8**(4), 595–610 (2011). https://doi.org/http://dx.doi.org/10.1016/j.disopt.2011.07.001

48. Kaibel, V., Pfetsch, M.: Packing and partitioning orbitopes. Mathematical Programming **114**(1), 1–36 (2008)
49. Liberti, L.: Reformulations in mathematical programming: automatic symmetry detection and exploitation. Mathematical Programming **131**(1-2), 273–304 (2012). https://doi.org/10.1007/s10107-010-0351-0
50. Liberti, L., Ostrowski, J.: Stabilizer-based symmetry breaking constraints for mathematical programs. Journal of Global Optimization **60**, 183–194 (2014)
51. Margot, F.: Pruning by isomorphism in branch-and-cut. Mathematical Programming **94**(1), 71–90 (2002). https://doi.org/10.1007/s10107-002-0358-2
52. Margot, F.: Exploiting orbits in symmetric ILP. Mathematical Programming **98**(1–3), 3–21 (2003). https://doi.org/10.1007/s10107-003-0394-6
53. Margot, F.: Symmetry in integer linear programming. 50 Years of Integer Programming 1958–2008 pp. 647–686 (2010)
54. McConnell, R.M., Mehlhorn, K., Näher, S., Schweitzer, P.: Survey: Certifying algorithms. Computer Science Review **5**(2), 119–161 (2011), https://publications.cispa.saarland/940/
55. de Moura, L.M., Bjørner, N.S.: Proofs and refutations, and Z3. In: LPAR Workshops. vol. 418, pp. 123–132. Doha, Qatar (2008)
56. Nemhauser, G.L., Wolsey, L.A.: A recursive procedure to generate all cuts for 0-1 mixed integer programs. Math. Program. **46**(3), 379390 (1990). https://doi.org/10.1007/BF01585752
57. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons, Ltd (1988). https://doi.org/10.1002/9781118627372.ch1
58. Ostrowski, J., Anjos, M.F., Vannelli, A.: Modified orbital branching for structured symmetry with an application to unit commitment. Mathematical Programming **150**(1), 99–129 (2015). https://doi.org/10.1007/s10107-014-0812-y
59. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**(1), 147–178 (2011). https://doi.org/10.1007/s10107-009-0273-x
60. Padberg, M.W., Roy, T.J.V., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. Oper. Res. **33**, 842–861 (1985). https://doi.org/10.1287/opre.33.4.842
61. Salvagnin, D.: Symmetry breaking inequalities from the Schreier-Sims table. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 521–529. Springer International Publishing, Cham (2018)
62. Savelsbergh, M.: Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing **6** (11 1994). https://doi.org/10.1287/ijoc.6.4.445
63. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics **3**(3), 411–430 (1990). https://doi.org/10.1137/0403036
64. Wetzler, N., Heule, M., Jr., W.A.H.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Sinz, C., Egly, U. (eds.) Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8561, pp. 422–429. Springer (2014). https://doi.org/10.1007/978-3-319-09284-3_31

A Auxiliary Results and Technical Proofs

In this appendix, we provide auxiliary results as well as the technical proofs that we had deferred in the main part of the article.

A.1 Auxiliary Results

The first two results state that $\mathcal{C}$-consistent branching trees have the properties that one would expect from a tree generated by a branch-and-bound algorithm. That is, the children of a node u partition the feasible region of the subproblem at node u, and, for every feasible solution, we find exactly one leaf of the tree in which the solution is feasible. In the following results, we use the notation $\mathcal{B}_v \coloneqq \{B_u : u \text{ is a node on the } r\text{-}v\text{-path in } \mathcal{T}\}$.

Lemma 1. *Let $\mathcal{T}$ be a $\mathcal{C}$-consistent branching tree and $u \in V$ with $\delta^+(u) \neq \emptyset$. Then for all $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_u)$ there is exactly one $v \in \delta^+(u)$ with $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_v)$.*

Proof. Let $u \in V$ with $\delta^+(u) \neq \emptyset$ and let $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_u)$. By (T3), there is $v \in \delta^+(u)$ with $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_v)$. In particular, for every such $v \in \delta^+(u)$ we have $\sigma_u(x) \in \sigma_u(\bigcap(\mathcal{C} \cup \mathcal{B}_v)) \subseteq \sigma_u(B_v)$. Uniqueness of $v \in \delta^+(u)$ thus follows from (T7). □

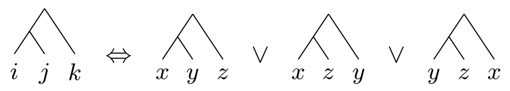
Lemma 2. *Let $\mathcal{T}$ be a $\mathcal{C}$-consistent branching tree. Then for all $x \in \bigcap \mathcal{C}$, there is exactly one leaf $v \in V$ with $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_v)$.*

Proof. Because of $B_r = \mathbb{R}^n$, $x \in \bigcap(\mathcal{C} \cup \mathcal{B}_r)$. The result follows by induction from Lem. 1 and finiteness of V in $\mathcal{T}$, see (T1). □

A crucial component of an (F, f)-valid configuration as defined in Def. 3 is the relation induced by a branching tree $\mathcal{T}$. The next results state that this relation and the corresponding strict relation define a preorder and strict order, respectively. Afterwards, we also show that a transitivity property between these two relations exists.

Lemma 3. *Let $\mathcal{T}$ be a $\mathcal{C}$-consistent branching tree. Then $\succeq_{\epsilon,\mathcal{T}}$ defines a preorder and $\succ_{\epsilon,\mathcal{T}}$ defines a strict order on $\bigcap \mathcal{C}$.*

Proof. Recall that $\succeq_{\epsilon,\mathcal{T}}$ defines a preorder, if it is reflexive and transitive. Reflexivity holds as $\sigma_{\mathrm{dcn}(x,x)}(x) = \sigma_{\mathrm{dcn}(x,x)}(x)$. For transitivity, let $x, y, z \in \bigcap \mathcal{C}$ with $y \succeq_{\epsilon,\mathcal{T}} x$ and $z \succeq_{\epsilon,\mathcal{T}} y$. If $v = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$, transitivity follows by transitivity of $=$ and $\succ_\epsilon$. In the following, we thus may assume that not all deepest common nodes are the same. Hence w.l.o.g. we can select pairwise distinct $i, j, k \in \{x, y, z\}$ such that $\mathrm{dcn}(i, k) = \mathrm{dcn}(j, k)$ and $\mathrm{dcn}(i, j)$ is a proper successor of $\mathrm{dcn}(i, k)$. We call this proper successor the *deepest node* and distinguish whether $\mathrm{dcn}(x, y)$, $\mathrm{dcn}(x, z)$, or $\mathrm{dcn}(y, z)$ is the deepest node:



First, let $v = \mathrm{dcn}(x, y)$ be the deepest node and $u = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$. Then, $y \succeq_{\epsilon,\mathcal{T}} x$ implies $\sigma_v(y) = \sigma_v(x) \lor \sigma_v(y) \succ_\epsilon \sigma_v(x)$, which in turn yields that $\sigma_u(y) = \sigma_u(x) \lor \sigma_u(y) \succ_\epsilon \sigma_u(x)$ by (T5) because u is an ancestor of v. Together with $z \succeq_{\epsilon,\mathcal{T}} y \Leftrightarrow \sigma_u(z) = \sigma_u(y) \lor \sigma_u(z) \succ_\epsilon \sigma_u(y)$, this shows $z \succeq_{\epsilon,\mathcal{T}} x$.

Second, let $v = \mathrm{dcn}(x, z)$ be the deepest node and $u = \mathrm{dcn}(y, z) = \mathrm{dcn}(x, y)$. Then, $y \succeq_{\epsilon,\mathcal{T}} x$ implies $\sigma_u(y) = \sigma_u(x) \lor \sigma_u(y) \succ_\epsilon \sigma_u(x)$, and $z \succeq_{\epsilon,\mathcal{T}} y$ yields that $\sigma_u(z) = \sigma_u(y) \lor \sigma_u(z) \succ_\epsilon \sigma_u(y)$. In fact, we claim that for both relations $\sigma_u(y) = \sigma_u(x)$ and $\sigma_u(z) = \sigma_u(y)$ cannot hold. If we are able to show this, $\sigma_u(y) \succ_\epsilon \sigma_u(x)$ and $\sigma_u(z) \succ_\epsilon \sigma_u(y)$ imply $\sigma_v(y) \succ_\epsilon \sigma_v(x)$ and $\sigma_v(z) \succ_\epsilon \sigma_v(y)$, respectively, by (T5). Relation $z \succeq_{\epsilon,\mathcal{T}} x$ then follows from transitivity of $\succ_\epsilon$.

To prove that $\sigma_u(y) = \sigma_u(x)$ cannot hold, observe that u is not a leaf as v is a proper successor of u. Since $u = \mathrm{dcn}(x, y)$, solutions x and y must be feasible at different children of u. By (T7), we conclude $\sigma_u(y) \neq \sigma_u(x)$, and analogously, $\sigma_u(z) \neq \sigma_u(y)$.

Third, let $v = \mathrm{dcn}(y, z)$ be the deepest node and $u = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z)$. Then, $z \succeq_{\epsilon,\mathcal{T}} y$ implies $\sigma_v(z) = \sigma_v(y) \lor \sigma_v(z) \succ_\epsilon \sigma_v(y)$. As in the first case, this yields $\sigma_u(z) = \sigma_u(y) \lor \sigma_u(z) \succ_\epsilon \sigma_u(y)$ by (T5) because u is an ancestor of v. Together with $y \succeq_{\epsilon,\mathcal{T}} x$, this shows $z \succeq_{\epsilon,\mathcal{T}} x$, concluding the proof that $\succeq_{\epsilon,\mathcal{T}}$ is a preorder on $\bigcap \mathcal{C}$.

To prove that $\succ_{\epsilon,\mathcal{T}}$ is a strict order on $\bigcap \mathcal{C}$, we need to show that it is irreflexive, antisymmetric, and transitive. Irreflexivity and antisymmetry follow from irreflexivity and antisymmetry of $\succ_\epsilon$. For transitivity, let $x, y, z \in \bigcap \mathcal{C}$ with $y \succ_{\epsilon,\mathcal{T}} x$ and $z \succ_{\epsilon,\mathcal{T}} y$. We pursue a similar strategy as above. If we have $v = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$, then $z \succ_{\epsilon,\mathcal{T}} x$ follows from transitivity of $\succ_\epsilon$ and that σ_v is used in all comparisons. Otherwise, we again distinguish the three cases above.

First, let $v = \mathrm{dcn}(x, y)$ be the deepest node and $u = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$. Then, $y \succ_{\epsilon,\mathcal{T}} x$ yields $\sigma_v(y) \succ_\epsilon \sigma_v(x)$. Thus, $\sigma_u(y) = \sigma_u(x) \lor \sigma_u(y) \succ_\epsilon \sigma_u(x)$ by (T5). From $z \succ_{\epsilon,\mathcal{T}} y$, we get $\sigma_u(z) \succ_{\epsilon,\mathcal{T}} \sigma_u(y)$. Combining this shows $z \succ_{\epsilon,\mathcal{T}} x$.

Second, let $v = \mathrm{dcn}(x, z)$ be the deepest node and $u = \mathrm{dcn}(y, z) = \mathrm{dcn}(x, y)$. As above, $y \succ_{\epsilon,\mathcal{T}} x$ implies $\sigma_u(y) \succ_\epsilon \sigma_u(x)$, and $z \succ_{\epsilon,\mathcal{T}} y$ yields $\sigma_u(z) \succ_\epsilon \sigma_u(y)$. Transitivity of $\succ_\epsilon$ then implies $\sigma_u(z) \succ_\epsilon \sigma_u(x)$. Since u is an ancestor of v, (T5) implies $\sigma_v(z) \succ_\epsilon \sigma_v(x)$, i.e, $z \succ_{\epsilon,\mathcal{T}} x$ holds.

Third, let $v = \mathrm{dcn}(y, z)$ be the deepest node and $u = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z)$. Then, $z \succ_{\epsilon,\mathcal{T}} y$ implies $\sigma_v(z) \succ_\epsilon \sigma_v(y)$, which again implies $\sigma_u(z) = \sigma_u(y) \lor \sigma_u(z) \succ_\epsilon \sigma_u(y)$. From $y \succ_{\epsilon,\mathcal{T}} x$, we derive $\sigma_u(y) \succ_\epsilon \sigma_u(x)$. Transitivity of $\succ_\epsilon$ yields $z \succ_{\epsilon,\mathcal{T}} x$. Relation $\succ_{\epsilon,\mathcal{T}}$ therefore defines a strict order. □

Lemma 4. *Let $\mathcal{T}$ be $\mathcal{C}$-consistent and let $x, y, z \in \bigcap \mathcal{C}$ be such that $z \succ_{\epsilon,\mathcal{T}} y$ and $y \succeq_{\epsilon,\mathcal{T}} x$ hold. Then, $z \succ_{\epsilon,\mathcal{T}} x$.*

Proof. If $v = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$, then the chain $z \succ_{\epsilon,\mathcal{T}} y \succeq_{\epsilon,\mathcal{T}} x$ implies $\sigma_v(z) \succ_\epsilon \sigma_v(y) = \sigma_v(x) \lor \sigma_v(z) \succ_\epsilon \sigma_v(y) \succ_\epsilon \sigma_v(x)$. In the former case, $z \succ_{\epsilon,\mathcal{T}} x$ follows immediately, and in the latter case it follows from transitivity of $\succ_\epsilon$. In the remainder of the proof, we follow the same strategy and terminology as in the proof of Lem. 3.

First, let $v = \mathrm{dcn}(x, y)$ be the deepest node and $u = \mathrm{dcn}(x, z) = \mathrm{dcn}(y, z)$. Then, $z \succ_{\epsilon,\mathcal{T}} y$ implies $\sigma_u(z) \succ_\epsilon \sigma_u(y)$, and $y \succeq_{\epsilon,\mathcal{T}} x$ yields $\sigma_v(y) = \sigma_v(x) \vee \sigma_v(y) \succ_\epsilon \sigma_v(x)$. Condition (T5) then implies $\sigma_u(y) = \sigma_u(x) \vee \sigma_u(y) \succ_\epsilon \sigma_u(x)$. Combining this with $\sigma_u(z) \succ_\epsilon \sigma_u(y)$ shows $z \succ_{\epsilon,\mathcal{T}} x$.

Second, let $v = \mathrm{dcn}(x, z)$ be the deepest node and $u = \mathrm{dcn}(y, z) = \mathrm{dcn}(x, y)$. From $z \succ_{\epsilon,\mathcal{T}} y$, we derive $\sigma_u(z) \succ_\epsilon \sigma_u(y)$, which implies $\sigma_v(z) \succ_\epsilon \sigma_v(y)$ by (T5). Moreover, analogously to the proof of lem. 3, we can show that $y \succeq_{\epsilon,\mathcal{T}} x$ implies $\sigma_u(y) \succ_\epsilon \sigma_u(x)$, i.e., $\sigma_u(y) = \sigma_u(x)$ cannot hold. Again by (T5), this implies $\sigma_v(y) \succ_\epsilon \sigma_v(x)$. Transitivity of $\succ_\epsilon$ then shows that $\sigma_v(z) \succ_\epsilon \sigma_v(x)$, and thus, $z \succ_{\epsilon,\mathcal{T}} x$ holds.

Third, let $v = \mathrm{dcn}(y, z)$ be the deepest node and $u = \mathrm{dcn}(x, y) = \mathrm{dcn}(x, z)$. From $y \succeq_{\epsilon,\mathcal{T}} x$, we conclude $\sigma_u(y) = \sigma_u(x) \vee \sigma_u(y) \succ_\epsilon \sigma_u(x)$. As above, $\sigma_u(y) = \sigma_u(x)$ cannot hold due to (T7), thus $\sigma_u(y) \succ_\epsilon \sigma_u(x)$ follows. By (T5), we obtain $\sigma_v(y) \succ_\epsilon \sigma_v(x)$. Combining this with $z \succ_{\epsilon,\mathcal{T}} y \Leftrightarrow \sigma_v(z) \succ_\epsilon \sigma_v(y)$, we conclude that $z \succ_{\epsilon,\mathcal{T}} x$ holds. □

The next lemma shows that $\mathcal{C}$-consistency of a branching tree is preserved if the set $\mathcal{C}$ of core constraints is extended.

Lemma 5. *Let $\mathcal{T}$ be a $\mathcal{C}$-consistent branching tree and $C \subseteq \mathbb{R}^n$. Then $\mathcal{T}$ is also $\mathcal{C} \cup \{C\}$-consistent.*

Proof. (T1), (T2), (T4), (T5), (T7) do not involve $\mathcal{C}$. Because $\bigcap \mathcal{C}$ is additionally intersected with C, the boundedness in (T6) continues to hold. Condition (T3) follows immediately, as $\bigcap(\mathcal{C} \cup \{C\}) \subseteq \bigcap \mathcal{C}$. □

To be able to show that symmetry reductions for MIPs can be derived within our framework, we require that a linear representation of the lexicographic order can be derived from our framework, which is achieved by the following result.

Proposition 1. *Let $(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon)$ be an (F, f)-valid configuration with branching tree $\mathcal{T} = (V, E, B, \sigma)$ such that all entries of σ_v, $v \in V$, are positive. Let γ be a symmetry of $(\bigcap \mathcal{C}, g)$ and $L \leq U$ be integers. Let $v \in V$ be such that x_i, $i \in \sigma_v$, is an integer variable with $L \leq x_i \leq U$. If $\epsilon \leq 1$, via dominance-based strengthening, one can derive the validity of inequality*

$$\sum_{i=1}^{\ell_v} (U - L + 1)^{\ell_v - 1} \sigma_v(x)_i \geq \sum_{i=1}^{\ell_v} (U - L + 1)^{\ell_v - 1} \sigma_v(\gamma(x))_i \tag{6}$$

at v. That is, if $C' = \{x \in \mathbb{R}^n : x \text{ satisfies } (6)\}$, one can derive $C = [\mathcal{B}_v \rightsquigarrow C']$.

We have designed the following proof in such a way that it is machine-verifiable. To keep the presentation simple, we do not mention explicitly how the different steps can be verified. But the idea is that case distinctions can be implemented as "subproofs" in which we add some assumptions characterizing the case. By using simple arguments in estimations, such as replacing a variable by their upper or lower bound, these subproofs either prove the desired statement or lead to a contradiction, showing thus the opposite statement.

Proof. To prove that we can derive C via dominance-based strengthening, we need to show that, if $x \in \bigcap(\mathcal{C} \cup \mathcal{D})$ and $x \notin C$, then there exists $\omega\colon \mathbb{R}^n \to \mathbb{R}^n$ with $g(\omega(x)) \leq g(x)$, $\omega(x) \in \bigcap \mathcal{C}$, and $\omega(x) \succ_{\epsilon,\mathcal{T}} x$. We claim that $\omega = \gamma$ serves as a witness.

Instead of proving the statement just for C, we show a stronger result. Let $\Delta = U - L + 1$ and

$$\sum_{i=1}^{k} \Delta^{k-1} \sigma_u(x)_i \geq \sum_{i=1}^{k} \Delta^{k-1} \sigma_u(\gamma(x))_i, \qquad k \in [\ell_v]. \tag{7}$$

Let $\hat{C}_v^j = \{x \in \mathbb{R}^n : x \text{ satisfies (7) for } k = j\}$ and $C_v^j = [\mathcal{B}_v \rightsquigarrow \hat{C}_u^j]$. We claim that we can derive C_v^j via dominance-based strengthening. If we can establish this claim, the assertion follows as $C = C_v^{\ell_v}$.

To prove this claim, we proceed by induction. Our induction hypothesis is that we can show, for all proper ancestors u of v, that we can derive C_u^j for all $j \in [\ell_u]$, and for node v, we can derive C_v^j for all $j \in [k-1]$ where $k < \ell_v$. Our goal is to show that we can also derive C_v^k. Note that the induction base case corresponds to $k = 1$. That is, the proof of the base case and the inductive step is the same. Moreover, as (7) is invariant under shifting the variable domains, we may assume w.l.o.g. that $L = 0$, which will simplify notation in the following.

In the inductive step, by the hypothesis, we may assume that, for all proper ancestors u of v and $j \in [\ell_u]$, we have $C_u^j \in \mathcal{D}$, and $C_v^j \in \mathcal{D}$ for all $j \in [k-1]$.[4] To apply dominance-based strengthening, let $\bar{x} \in \bigcap(\mathcal{C} \cup \mathcal{D})$ with $\bar{x} \notin C_v^k$. Since γ is a symmetry of $(\bigcap \mathcal{C}, g)$, we have $\gamma(\bar{x}) \in \bigcap \mathcal{C}$ and $g(\gamma(\bar{x})) = g(\bar{x})$. Thus, it remains to show $\gamma(\bar{x}) \succ_{\epsilon,\mathcal{T}} \bar{x}$.

Let $w = \mathrm{dcn}(\bar{x}, \gamma(\bar{x}))$. Following the definition of $\succ_{\epsilon,\mathcal{T}}$, we need to show that $\sigma_w(\gamma(\bar{x})) \succ_\epsilon \sigma_w(\bar{x})$. Since $\bar{x} \notin C_v^k$, we have $\bar{x} \in \bigcap \mathcal{B}_v$ and $\bar{x} \notin \hat{C}_v^k$. By the former, w and v lie on a common rooted path in $\mathcal{T}$. We distinguish whether w is a proper ancestor of v or not. In the latter case, note that $\sigma_w(\gamma(\bar{x})) \succ_\epsilon \sigma_w(\bar{x})$ follows from (T5) if we can show $\sigma_v(\gamma(\bar{x})) \succ_\epsilon \sigma_v(\bar{x})$.

First, suppose w is not a proper ancestor of v. By the induction hypothesis, $C_u^j \in \mathcal{D}$ for all $j \in [\ell_u]$ and proper ancestors u of v, as well as $C_v^j \in \mathcal{D}$ for all $j \in [k-1]$. For all $j \in [k-1]$, this in particular means that $\bar{x} \in \hat{C}_v^j$, because $\bar{x} \in \bigcap \mathcal{B}_v$. Suppose that there is one $j' \in [k-1]$ such that the inequality in $\hat{C}_v^{j'}$ is strictly satisfied. Among all such j', let j be the one that is minimal.

[4] This assumption is only needed in the proof. In practice, one would remove the previously derived constraints from $\mathcal{D}$ via the deletion rule after adding C_v^k.

Then, since all variables are non-negative by the assumption $L = 0$,

$$\begin{aligned}\sum_{i=1}^{k} \varDelta^{k-i}\sigma_v(\bar{x})_i \geq& \sum_{i=1}^{j-1} \varDelta^{k-i}\sigma_v(\bar{x})_i + \varDelta^{k-j}\sigma_v(\bar{x})_j \\ =& \sum_{i=1}^{j-1} \varDelta^{k-i}\sigma_v(\gamma(\bar{x}))_i + \varDelta^{k-j}\sigma_v(\bar{x})_j \\ \geq& \sum_{i=1}^{j-1} \varDelta^{k-i}\sigma_v(\gamma(\bar{x}))_i + \varDelta^{k-j}\sigma_v(\gamma(\bar{x}))_j + \varDelta^{k-j} \\ \geq& 1 + \sum_{i=1}^{k} \varDelta^{k-i}\sigma_v(\gamma(\bar{x}))_i.\end{aligned}$$

Here, the first inequality is due to $L = 0$. The third inequality holds as geometric sums satisfy $\varDelta^{k-j} = 1 + \sum_{i=j+1}^{k}(\varDelta - 1)\varDelta^i$, and $\sigma_v(\gamma(\bar{x}))_j \leq (U - L) = \varDelta - 1$. The second estimation holds as j is the smallest index for which (7) is a strict inequality; its inequality (7) thus reduces to $\sigma_v(\bar{x})_j > \sigma_v(\gamma(\bar{x}))_j$, which implies $\sigma_v(\bar{x})_j \geq 1 + \sigma_v(\gamma(\bar{x}))_j$ by integrality.

This inequality chain, however, is a contradiction to $\bar{x} \notin C_v^k$. Consequently, all inequalities in $\hat{C}_v^1, \dots, \hat{C}_v^{k-1}$ must be satisfied by $\bar{x}$ with equality. The inequality in $\hat{C}_v^k$ then reduces to $\sigma_v(x)_k \geq \sigma_v(\gamma(x))_v$. As such, since $\bar{x} \notin \hat{C}_v^k$, we have $\sigma_v(\gamma(\bar{x}))_k > \sigma_v(\bar{x})_k$. Again, by integrality, this implies

$$\sigma_v(\gamma(\bar{x}))_k \geq 1 + \sigma_v(\bar{x})_k \geq \epsilon + \sigma_v(\bar{x})_k.$$

As a consequence, $\sigma_v(\gamma(\bar{x})) \succ_\epsilon \sigma_v(\bar{x})$ follows.

It remains to consider the case that w is a proper predecessor of v. In this case, $\bar{x}$ and $\gamma(\bar{x})$ must be feasible at different children of w as w is the deepest common node. By (T7), $\sigma_w(\gamma(\bar{x}))_{\ell_w} < \sigma_w(\bar{x})_{\ell_w}$ or $\sigma_w(\gamma(\bar{x}))_{\ell_w} > \sigma_w(\bar{x})_{\ell_w}$. In the former case, exactly the same arguments as above can be used to show

$$\sum_{i=1}^{k} \varDelta^{k-i}\sigma_v(\bar{x})_i \geq \sum_{i=1}^{k} \varDelta^{k-i}\sigma_v(\gamma(\bar{x}))_i,$$

as σ_v is an extension of σ_w by (T5), which is a contradiction to $\bar{x} \notin C_v^k$. In the latter case, we find (again using the same arguments) that $\sigma_w(\gamma(\bar{x})) \succ_\epsilon \sigma_w(\bar{x})$. After considering all cases, $\gamma(\bar{x}) \succ_{\epsilon,\mathcal{T}} \bar{x}$ thus follows. □

A.2 Technical Proofs

This section provides the missing technical proofs of the rules that allow to adapt configurations while preserving their (F, f)-validity.

Proof of Thm. 3 (Implicational Derivation Rule) Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be the previously given configuration. (V1), (V2), and (V3) are trivially preserved since they do not depend on the set of derived constraints. For (V4), let $x \in \bigcap \mathcal{C}$ with $g(x) < z$. By validity of $\mathfrak{C}$, there exists a $y \in \bigcap(\mathcal{C} \cup \mathcal{D})$ with $y \succeq_{\epsilon,\mathcal{T}} x$ and $g(y) \leq g(x) < z$. It suffices to show $y \in [\mathcal{A} \rightsquigarrow C] = \overline{\bigcap \mathcal{A}} \cup C$. If $y \notin \bigcap \mathcal{A}$, this holds trivially. If $y \in \bigcap \mathcal{A}$, then Condition (1) implies

$$y \in \bigcap(\mathcal{C} \cup \mathcal{D} \cup \mathcal{A}) \cap \{x \in \mathbb{R}^n : g(x) < z\} \subseteq C \subseteq [\mathcal{A} \rightsquigarrow C].$$

In either case, (V4) is satisfied for the updated configuration. □

Proof of Corollary 1 (Resolution Rule) First, careful rewriting yields

$$\begin{aligned}
\bigcap(\mathcal{C} \cup \mathcal{D}) &\subseteq [(\mathcal{A}_1 \cup \{A_1\}) \rightsquigarrow C_1] \cap [(\mathcal{A}_2 \cup \{A_2\}) \rightsquigarrow C_2] \\
&= \Big(\overline{\bigcap(\mathcal{A}_1 \cup \{A_1\})} \cup C_1\Big) \cap \Big(\overline{\bigcap(\mathcal{A}_2 \cup \{A_2\})} \cup C_2\Big) \\
&\subseteq \Big(\overline{\bigcap(\mathcal{A}_1 \cup \{A_1\})} \cup C_1 \cup C_2\Big) \cap \Big(\overline{\bigcap(\mathcal{A}_2 \cup \{A_2\})} \cup C_1 \cup C_2\Big) \\
&= \Big(\overline{\bigcap(\mathcal{A}_1 \cup \{A_1\})} \cap \overline{\bigcap(\mathcal{A}_2 \cup \{A_2\})}\Big) \cup (C_1 \cup C_2) \\
&= \Big(\Big(\overline{\bigcap \mathcal{A}_1} \cup \overline{A_1}\Big) \cap \Big(\overline{\bigcap \mathcal{A}_2} \cup \overline{A_2}\Big)\Big) \cup (C_1 \cup C_2).
\end{aligned}$$

By this, every $x \in \bigcap(\mathcal{C} \cup \mathcal{D}) \subseteq A_1 \cup A_2$ is contained in at least one of the following three sets: in $C_1 \cup C_2$ or in $\overline{\bigcap \mathcal{A}_1}$ (if $x \in A_1 \setminus (C_1 \cup C_2)$) or in $\overline{\bigcap \mathcal{A}_2}$ (if $x \in A_2 \setminus (C_1 \cup C_2)$), i.e.,

$$x \in \overline{\bigcap \mathcal{A}_1} \cup \overline{\bigcap \mathcal{A}_2} \cup (C_1 \cup C_2) = [(\mathcal{A}_1 \cup \mathcal{A}_2) \rightsquigarrow (C_1 \cup C_2)] =: C.$$

Hence, C satisfies (1) for $\mathcal{A} = \emptyset$ and any value of z (including the value in the current configuration), and can be added to $\mathcal{D}$, preserving validity of the configuration. □

Proof of Thm. 4 (Objective Bound Update Rule) Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be the previously given configuration. (V1) is preserved because it does not involve the objective bound. Since $z' < \infty$, for (V2) we need to show that there exists an $x \in F$ with $f(x) \leq z'$. This is a direct consequence of (V3) for $\mathfrak{C}$. (V3) and (V4) continue to hold for the updated configuration because the objective bound becomes stricter and thus their statements become weaker. □

Proof of Thm. 5 (Objective Function Update Rule) Assume that the previously given configuration is $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$. (V1) and (V2) are not affected by the update of g. For (V3) let $\hat{z} < z$.

First, assume we have $x \in F$ with $f(x) \leq \hat{z}$. (V3) for $\mathfrak{C}$ ensures that there also exists $\hat{x} \in \bigcap \mathcal{C}$ with $g(\hat{x}) \leq \hat{z} < z$, and (V4) for $\mathfrak{C}$ gives us a $y \in \bigcap(\mathcal{C} \cup \mathcal{D})$

with $g(y) \leq g(\hat{x}) \leq \hat{z}$. The condition of the theorem implies that $g'(y) = g(y)$. All in all, $y \in \bigcap \mathcal{C}$ and $g'(y) \leq \hat{z}$, so the forward implication in (V3) is preserved.

For the reverse direction, assume we have $x \in \bigcap \mathcal{C}$ with $g'(x) \leq \hat{z} < z$. Then $g(x) = g'(x) \leq \hat{z}$ holds by the condition of the theorem. Now (V3) for $\mathfrak{C}$ ensures that there exists $x \in F$ with $f(x) \leq \hat{z}$.

To show (V4) for the updated configuration, let $x \in \bigcap \mathcal{C}$ with $g'(x) < z$. Again, the condition of the theorem implies $g(x) = g'(x) < z$. (V4) for $\mathfrak{C}$ yields a $y \in \bigcap(\mathcal{C} \cup \mathcal{D})$ with $y \succeq_{\epsilon,\mathcal{T}} x$ and $g(y) \leq g(x)$. Finally $g'(y) = g(y) \leq g(x) = g'(x)$ follows from the condition of the theorem. Hence, (V4) also holds for the updated configuration. □

Proof of Thm. 8 (Epsilon Shrinkage Rule) (V1) is trivially satisfied due to $\epsilon' > 0$. (V2) and (V3) are not affected when changing ϵ to ϵ'. (V4) becomes a weaker statement due to $\epsilon' < \epsilon$. □

Proof of Thm. 9 (Transfer Rule) Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be the previously given configuration. (V1) is preserved according to Lem. 5, (V2) does not depend on $\mathcal{C}$ and $\mathcal{D}$, and (V4) is preserved because $\bigcap(\mathcal{C} \cup \{C\}) \subseteq \bigcap \mathcal{C}$ and

$$\bigcap \big((\mathcal{C} \cup \{C\}) \cup (\mathcal{D} \setminus \{C\})\big) = \bigcap(\mathcal{C} \cup \mathcal{D}).$$

For (V3) let $z' < z$. First, assume we have $x \in F$ with $f(x) \leq z'$. (V3) for $\mathfrak{C}$ yields $x' \in \bigcap \mathcal{C}$ with $g(x') \leq z' < z$, and (V4) for $\mathfrak{C}$ gives us a $y \in \bigcap(\mathcal{C} \cup \mathcal{D}) \subseteq \bigcap(\mathcal{C} \cup \{C\})$ with $g(y) \leq g(x') \leq z'$. Hence, the forward implication in (V3) is preserved. For the reverse direction, assume we have $x \in \bigcap(\mathcal{C} \cup \{C\}) \subseteq \bigcap \mathcal{C}$ with $g(x) \leq z'$. Now (V3) for $\mathfrak{C}$ directly yields an $x \in F$ with $f(x) \leq z'$.

□

Proof of Thm. 10 (Deletion Rule) Let $\mathfrak{C} = \big(\mathcal{C}, \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ be the previously given configuration. For (a), (V1), (V2), and (V3) are invariant. (V4) becomes weaker, since $\bigcap(\mathcal{C} \cup \mathcal{D}') \supseteq \bigcap(\mathcal{C} \cup \mathcal{D})$.

For (b) and (c), it suffices to show that $\big(\mathcal{C}', \mathcal{D}, g, z, \mathcal{T}, \epsilon\big)$ is valid; then the validity of the derived constraint deletion follows from (a). The proof for (b) is trivial, since $\bigcap \mathcal{C}'$ and $\bigcap \mathcal{C}$ define the same set.

For proving (c), note that (V2) is invariant, and (V1) is trivially preserved since all σ_v are empty and the tree cannot have nodes with more than one child (and $\epsilon > 0$ is unchanged).

The forward direction of (V3) is trivial, since $\bigcap \mathcal{C} \subseteq \bigcap \mathcal{C}'$. For the reverse direction, we need to show that for any $\hat{z} < z$, if $x \in \bigcap \mathcal{C}'$ with $g(x) \leq \hat{z}$ then there exists $y \in F$ with $f(y) \leq \hat{z}$. If $x \in C$, then $x \in \bigcap \mathcal{C}$ and (V3) for $\mathfrak{C}$ guarantees us a suitable y. If $x \notin C$, then let ω be the witness provided alongside the application of the redundance-based strengthening rule. By (3) applied to $\big(\mathcal{C}', \emptyset, g, z, \mathcal{T}, \epsilon\big)$, i.e., with $\mathcal{C} = \mathcal{C}'$ and $\mathcal{D} = \emptyset$, we know that $\omega(x) \in \bigcap(\mathcal{C}' \cup \{C\}) = \bigcap \mathcal{C}$ and $g(\omega(x)) \leq g(x) \leq \hat{z}$. Now again, (V3) for $\mathfrak{C}$ guarantees us a suitable y.

For (V4), if $x \in C$ then the condition immediately follows from the validity of the previous configuration $\mathfrak{C}$ since $\bigcap(\mathcal{C}' \cup \mathcal{D}) \supseteq \bigcap(\mathcal{C} \cup \mathcal{D})$. If $x \notin C$, then we can once again apply $\omega(x) \in \bigcap \mathcal{C}$ and $g(\omega(x)) \leq g(x) \leq \hat{z}$, hence the condition immediately follows from (V4) for $\mathfrak{C}$. □

Proof of Thm. 11 (Tree Exchange Rule) (V1) holds because $\mathcal{T}'$ is assumed to be $\mathcal{C}$-consistent and $\epsilon > 0$. (V2) and (V3) are independent of the tree. Due to $\mathcal{D} = \emptyset$ and the reflexivity of the preorders, (V4) holds trivially for $y = x$, independently of the tree. □

Proof of Thm. 12 (Dimension Extension Rule) (V1) and (V2) are invariant under extension of the dimension. (V3) and (V4) on the new configuration are equivalent to (V3) and (V4) on the previous configuration after projecting out the newly added variable. □

A.3 Auxiliary examples of verification

We provide several more detailed examples of automatic verification of cutting planes.

Chvátal-Gomory cuts [15]. Given a valid inequality $\alpha^T x \leq \beta$ with $\alpha_i = 0$, for all $i > p$, it is valid to round down all coefficients, while also rounding down the right-hand-side, i.e., $\sum_{i=1}^{p} \lfloor \alpha_i \rfloor x_i \leq \lfloor \beta \rfloor$ is valid. This is a special case of a disjunctive cut, where the disjunction is given by $\sum_{i=1}^{p} \lfloor \alpha_i \rfloor x_i \leq \lfloor \beta \rfloor \vee \sum_{i=1}^{p} \lfloor \alpha_i \rfloor x_i \geq \lfloor \beta \rfloor + 1$ and the greater or equal part of the disjunction contains no points in the feasible region. In the previously existing certificate format for MIP called VIPR [14], CG-cuts are directly verified by checking the correctness of the rounding, as well as the integrality requirements. We point out that the interpretation as split cuts allows to verify a CG-cut without knowing the original inequality $\alpha^T x \leq \beta$, by solving an auxiliary LP to find the correct Farkas multipliers to show infeasibility of $\sum_{i=1}^{p} \lfloor \alpha_i \rfloor x_i \geq \lfloor \beta \rfloor + 1$.

Knapsack cover cuts [6]. Given a knapsack constraint $\sum_{j=1}^{n} a_j x_j \leq b$, where $x \in \{0,1\}^n, b \in \mathbb{Z}_{>0}, a_j \in \mathbb{Z}_{>0}$ for all $j \in [n]$, a *cover* is a subset $C \subseteq [n]$ such that $\sum_{j \in C} a_j > b$. The corresponding cover inequality is $\sum_{j \in C} x_j \leq |C| - 1$.

Any cover inequality can be verified as a split cut. We define the split disjunction $\sum_{j \in C} x_j \leq |C| - 1 \vee \sum_{j \in C} x_j \geq |C|$. For the first part of the disjunction, the cover inequality is trivially valid. For the second part, $\sum_{j \in C} x_j \geq |C|$ can be used to derive $x_j \geq 1$ for every $j \in C$ by a simple aggregation proof. Then $x_j \geq 1$ for all $j \in C$ can be aggregated using multipliers a_j to derive $\sum_{j \in C} a_j x_j \geq \sum_{j \in C} a_j > b$, which is a trivially verifiable contradiction to the knapsack constraint. Applying the resolution rule certifies the cover inequality as valid.

Flowcover cuts [60]. Flowcover cuts make use of a structure commonly found inside a MIP model, namely single-node flow sets, defined as

$$T := \left\{ (x,y) \in \{0,1\}^n \times \mathbb{R}^n_+ : \sum_{j=1}^{n} y_j \leq b, y_j \leq a_j x_j \text{ for } j \in [n] \right\}$$

with $0 < a_j \leq b$ for all $j \in [n]$. A set $C \subseteq \{1,\ldots,n\}$ is a *flow cover* of T if $\sum_{j\in C} a_j > b$. We denote by $(\cdot)^+ = \max\{0,\cdot\}$. The corresponding flow cover inequality is $\sum_{j\in C} y_j + \sum_{j\in C}(a_j - \lambda)^+(1-x_j) \leq b$, where $\lambda = \sum_{j\in C} a_j - b$.

We can verify a flow cover inequality as a split cut with the disjunction $\sum_{j\in C, a_j\geq\lambda}(1-x_j) \leq 0 \vee \sum_{j\in C, a_j\geq\lambda}(1-x_j) \geq 1$. In the first case, the inequality reduces to $\sum_{j=1}^{n} y_j \leq b$, which is valid by definition of T. In the second case, we can derive the following intermediate inequality:

$$\sum_{j\in C}(a_j-\lambda)^+(1-x_j) = \sum_{j\in C, a_j\geq\lambda} a_j(1-x_j) - \lambda \sum_{j\in C, a_j\geq\lambda}(1-x_j) \tag{8}$$
$$\leq \sum_{j\in C} a_j(1-x_j) - \lambda = b - \sum_{j\in C} a_j x_j \leq \left(b - \sum_{j\in C} a_j x_j\right)^+$$

The inequality holds, since $\sum_{j\in C, a_j\geq\lambda}(1-x_j) \geq 1$ by assumption, and since $a_j(1-x_j) \geq 0$. Then we derive the flow cover inequality using the additional disjunction $\sum_{j\in C} a_j x_j \leq b \vee \sum_{j\in C} a_j x_j \geq b+1$. In the first case, we find $\sum_{j\in C} y_j \leq \sum_{j\in C} a_j x_j = b - (b - \sum_{j\in C} a_j x_j)^+$, and adding the intermediate inequality (8) gives the flow cover inequality. In the second case, the inequality $\sum_{j\in C} y_j \leq b = b - (b - \sum_{j\in C} a_j x_j)^+$ holds and again adding the intermediate inequality (8) gives the flow cover inequality.

Reduced cost fixing. Assume we solved the LP relaxation of (P), with optimal objective value z_{MIP}, and assume that we already found an integer feasible solution with objective value $z_I < \infty$. Let x_j be a nonbasic variable at its lower bound, and for the sake of simplicity assume that $x_j \geq 0$ with reduced cost $\bar{c}_j \neq 0$. Then reduced cost fixing [17] states that the following inequality is valid:

$$x_j \leq \left\lfloor \frac{z_{MIP} - z_{LP}}{\bar{c}_j} \right\rfloor$$

Assume that x_j is an integer variable, and for the sake of simplicity, assume that all variables $x_i \geq 0$. The reduction of reduced cost fixing can be proven in the setting of our proof system using the implication rule. We make a simple aggregation proof, adding $c^T x \leq z_{MIP}$ and $-(y^T A)x \leq z_{LP}$, where y are the dual multipliers of the LP solution. This yields $(c - y^T A)x \leq z_{MIP} - z_{LP}$. Since the LP has been solved to optimality, we can aggregate out all variables except x_j and the inequality stays valid. Dividing by $\bar{c}_j = c_j - y^T A_j$ yields the desired inequality.

An analogous construction can be used to prove the validity of reduced cost fixing for variables at their upper bound.