Using Fourier-Motzkin-Elimination to Derive Capacity Models of Container Vessels

Mai Lise Ajspur
Rune Møller Jensen

IT University Technical Report Series **TR-2017-197**

ISSN 1600–6100 **January 2017**

ISSN 1600–6100

ISBN 978-87-7949-365-0

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web **`www.itu.dk`**

Using Fourier-Motzkin-Elimination to Derive Capacity Models of Container Vessels

Mai Lise Ajspur
Rune Møller Jensen

Abstract

Due to its high computational complexity, Fourier-Motzkin-Elimination (FME) is mainly known as a theoretical approach to determine feasibility of a linear program (LP). Current practical applications of FME in static program analysis and logic programming exploit that it is a transformation corresponding to existential quantification in logic, but large-scale variable elimination has not been attempted so far. In this report, we introduce a novel FME-based framework for massive variable elimination that takes advantage of the block structure found in many LP problems. Our objective is to simplify the LP by eliminating most of its variables. We show that this is possible for defining accurate capacity models for container vessels by deriving them from stowage models.

1 Introduction

Fourier-Motzkin-Elimination (FME) is a central theoretical approach to determine feasibility of linear programs (LPs), but it is seldom used in practice due to its high computational complexity. It is an algorithm that projects the polyhedron of feasible solutions of the LP one dimension at a time, such that the feasible values and interdependencies of the remaining variables in the resulting polyhedron are unaltered. In this way, FME computes an abstraction of the original LP corresponding to existential quantification in logic, where the value of the eliminated variable no longer is represented.

Projection is a fundamental operation that has been successfully applied in practice, in particular in static analysis of programs and logic programming, (see e.g. [BKM05] for applications and references). Here polyhedra represent numerical variable domains [CH78], and variables out of scope and other auxiliary variables are eliminated via projection. Likewise, projection is used in constraint query languages [Las90], and other potential applications of FME include coordination and/or negotiation situations as suggested in [LS08], and as an alternative solution method for parametric linear programming [JKM08].

We are not aware of any previous work, however, that applies FME for massive variable elimination, where say 99% of the variables are eliminated with the purpose to simplify the LP by removing most of its variables.This is, however, a relevant task. Consider for instance an economic model that achieves high accuracy through a large number of economic parameters at county level, but primarily is used to analyse these parameters at an aggregated national level. In this case, only a small fraction of the variables are of interest and the LP may be simplified by abstracting away (i.e., eliminating) the other low-level variables.

In this report, we consider a similar problem. In liner shipping, it is challenging to evaluate how the cargo capacity of a container vessel depends on the *cargomix*, i.e., the mixture of containers with different attributes that the vessel carries. The reason for this is that containers only can be stowed certain ways (e.g., $40'$ containers cannot be stowed on top of $20'$ containers and refrigerated containers must be stowed in slots with power plugs) while at the same time the vessel must fulfill seaworthiness requirements such as stability rules and stress force limits. The liner shipping companies need accurate capacity models to optimize decisions about cargo flow and uptake management on services, but today they rely on static capacity measures (e.g., [TT04, FC08, ZF13]) that are too inaccurate for practical usage [Del13].

On the other hand, accurate LP models for optimization of vessel stowage planning have been developed (e.g., [PDJB11, PDOJB12, Del13]). These models represent a feasible stowage plan as the number of containers of different type and weight class that the vessel can carry in each of its 100 or more storage areas. A capacity model is simply an abstraction of these stowage models, where the physical position of cargo is abstracted away. Since there are in the order of 100 storage areas on the vessel, this corresponds to eliminating 99% of the variables.

In this report, we introduce a novel hierarchically decomposed and parallelized FME algorithm for massive variable elimination. Similar to other recent FME frameworks (e.g., [SK05, LS08, SL12]), our method applies

Gauss-elimination (equality removal), removal of syntactic and quasi-syntactic redundancies, complete removal of redundancy in between variable eliminations, together with an approximation of the projection when needed. In addition, our framework includes preprocessing including removal of less strict inequalities. Our main contribution, however, is a hierarchical decomposition of the problem using auxiliary variables based on the block structure that often is observed in LPs [Wil78] as well as a sound, concurrent implementation of the redundancy removal.

Theoretically, the number of inequalities may grow double exponentially with the number of eliminated variables. Our experiments on deriving capacity models from stowage models of mega vessels, however, show only a modest growth in the number of inequalities when complete removal of redundant inequalities between each variable elimination is carried out (e.g., see Figures 10-12 in Chapter 5). As expected, the number of inequalities decrease in the end where most variables are removed. The resulting capacity models are small and useful in practice (e.g., see Table 5 in Chapter 5).

It is possible that the limited growth in non-redundant inequalities is unique for deriving capacity models, but since we mainly exploit the block structure of stowage models, we believe that similar results can be obtained in other domains.

The remainder of this report is organized as follows. In Chapter 2, we present the definitions and notation relating to inequality systems and projections that are used in this report. Chapter 3 outlines the basic algorithms used for achieving the projection, namely the classical Fourier-Motzkin elimination, Gauss-elimination, and some basic methods for preprocessing and removing redundant information (inequalities). In Chapter 4 we then describe the alterations and improvements made to these algorithms. Specifically we detail an altered and parallellized method for removing redundancy, how we combine the mentioned steps into an algorithm for projection, and we present how our system (as well as other block structured systems) can be decomposed to achieve a more efficient projection. Subsequently, Chapter 5 gives a short presentation of the considered stowage model(s) after which the results of projecting these are presented along with a few details regarding the implementation. Finally, we review and discuss related work in Chapter 6 before Chapter 7 concludes.

2 Definitions and notation

In this report we consider a finite set of variables $\mathcal{X}$ plus the set $\mathcal{IE}$ consisting of all inequalities and equalities over any subset of $\mathcal{X}$. An element $c \in \mathcal{IE}$ is thus an equality or inequality over a subset $VAR(c) \subseteq \mathcal{X}$ and is referred to as an (in)equality over $VAR(c)$. It can be written as

$$c : \sum_{x \in VAR(c)} co(x, c) \cdot x \odot_c rhs(c), \tag{1}$$

where $\odot_c \in \{=, \leq\}$ is the *relation* of c, $co(x, c) \in \mathbb{R}$ is the *coefficient* of x in c for any $x \in VAR(c)$, and $rhs(c) \in \mathbb{R}$ is the *right-hand-side* of c. Letting $co(x, c) = 0$ for all $x \in \mathcal{X} \setminus VAR(c)$, we can extend c to X for any set $X \subseteq \mathcal{X}$ by letting c_X be the (in)equality over X given by

$$c_X : \sum_{x \in X} co(x, c) \cdot x \odot_c rhs(c). \tag{2}$$

Assuming a given total order $\preceq$ on the variables in $\mathcal{X}$ we can order and name the variable of any subset $X \subseteq \mathcal{X}$ according to this order such that $X = \{x_1, x_2, \ldots, x_{|X|}\}$ and $x_1 \prec x_2 \prec \ldots \prec x_{|X|}$. Letting $\mathbf{co}(c_X) \stackrel{\text{def.}}{=} (co(x_1, c), \ldots, co(x_{|X|}, c))$ and $\mathbf{x}_X = (x_1, \ldots, x_{|X|})$, the constraint c_X can also be written using the dot-product as $c_X : \mathbf{co}(c_X) \cdot \mathbf{x}_X \odot_c rhs(c)$. By definition $c_{VAR(c)}$ equals c, so we let $\mathbf{co}(c) \stackrel{\text{def.}}{=} \mathbf{co}(c_{VAR(c)})$, and for convenience we leave out the subscript of $\mathbf{x}_X$ when the X is implied e.g. by the coefficient vector in the dot product.

We let $var(c)$ denote the variables whose coefficient in c is nonzero, i.e. $var(c) \stackrel{\text{def.}}{=} \{ x \in VAR(c) \mid co(x, c) \neq 0 \} \subseteq VAR(c)$, and we notice that we might have that $var(c) \subset VAR(c)$. When $VAR(c)$ is clear from context we allow ourselves to write c as $c : \sum_{x \in var(c)} co(x, c) \cdot x \odot_c rhs(c)$.

The set of points in $\mathbb{R}^{|VAR(c)|}$ that satisfies the (in)equality c is called the *feasible region* for c and is hence defined as

$$feas(c) \stackrel{\text{def.}}{=} \{ \mathbf{r} \in \mathbb{R}^{|VAR(c)|} \mid \mathbf{co}(c) \cdot \mathbf{r} \odot_c rhs(c) \}.$$

Since the feasible region of an (in)equality is just a set of points in a (multi-dimensional) Euclidian space, here the order of the variables, $\prec$, is important and needs to be explicitly or implicitly given or assumed.

In the following, an (in)equality system is a set S of (in)equalities over the same set of variables, $VAR(S)$, i.e. $S \subset \mathcal{IE}$ and $VAR(c) = VAR(S)$ for all $c \in S$. Therefore, when we union, intersect or subtract two (in)equality systems $S_1, S_2 \subset \mathcal{IE}$ to produce another (in)equality system S', it is required that all (in)equalities in S' are (in)equalities over the same set X. Hence we require that $VAR(S_1) = VAR(S_2)$ when we talk about the (in)equality systems $S_1 \cup S_2$, $S_1 \cap S_2$ or $S_1 \setminus S_2$. However, it is always possible to extend the (in)equalities in S_1 and S_2 to a common variable-set, and if $VAR(S_1) \neq VAR(S_2)$ we therefore let $S_1 \cup S_2 \stackrel{\text{def.}}{=} \{ c_X \mid c \in S_1 \} \cup \{ c_X \mid c \in S_2 \}$, where $X = VAR(S_1) \cup VAR(S_2)$, and similar for the other set-operations $\cap$ and $\setminus$.

We let $eqs(S)$ denote the set of equalities in S, i.e. $eqs(S) \stackrel{\text{def.}}{=} \{ c \in S \mid \odot_c \text{ is } = \}$, we define the used variables in S as $var(S) \stackrel{\text{def.}}{=} \cup_{c \in S} var(c)$, and we let $S_X \stackrel{\text{def.}}{=} \{ c_X \mid c \in S \}$ for any $X \subseteq \mathcal{X}$. The feasible region for S is the set of points in $\mathbb{R}^{|VAR(S)|}$ that satisfies *all* (in)equalities in S, i.e.

$$feas(S) \stackrel{\text{def.}}{=} \cap_{c \in S} feas(c).$$

It follows that if $S_1, S_2 \subseteq S$, then $feas(S_1 \cup S_2) = feas(S_1) \cap feas(S_2)$.

In linear programming, it is common to have bounds for some of the variables in an (in)equality system S. For our purpose, upper and lower bounds are modeled as inequalities as in (2) with $X = VAR(S)$ and a coefficient of ± 1. If they exist, the inequalities representing upper and lower bounds for x are referred to as $UBineq(x)$ and $lbineq(x)$, respectively, while $UB(x) \in \mathbb{R}$ and $lb(x) \in \mathbb{R}$ denote the upper, respectively lower bound of the variable x. That is, if the upper bound of x exists, then $UBineq(x)$ refers to the inequality $x \leq UB(x)$ in S, and if the lower bound of x exists, then $lbineq(x)$ refers to the inequality $-x \leq -lb(x)$ in S.

We say that two (in)equality systems S_1 and S_2 are *equivalent* and write $S_1 \cong S_2$ if $VAR(S_1) = VAR(S_2)$ and $feas(S_1) = feas(S_2)$.

An (in)equality $c \in S$ is *redundant* if c does not influence the feasible region for S, i.e. if $feas(S) = feas(S \setminus \{c\})$. The redundancy of an *in*equality $c : \mathbf{co}(c) \cdot \mathbf{x} \leq rhs(c)$ can be determined by maximizing its left-hand-side:

$$c \text{ is redundant iff } \max\{ \mathbf{co}(c) \cdot \mathbf{r} \mid \mathbf{r} \in feas(S \setminus \{c\}) \} \leq rhs(c). \tag{3}$$

The equality $c : \mathbf{co}(c) \cdot \mathbf{x} = rhs(c)$ is redundant iff both inequalities $\mathbf{co}(c) \cdot \mathbf{x} \leq rhs(c)$ and $-\mathbf{co}(c) \cdot \mathbf{x} \leq -rhs(c)$ are redundant. If the (in)equality c is *not* redundant, it is called *non-redundant*.

As described, the feasible region of S describes the combination of values for the variables in $VAR(S)$ that satisfy all the (in)equalities in S. However, for the sake of abstraction there are some variables $Y \subseteq VAR(S)$ whose value we are not interested in, we just want to know that a satisfying value for these variables exists. Therefore, we want to find a set of values P for the variables of interest ($VAR(S) \setminus Y$), such that when we have a set of values in P then we can extend it with values for the variables in Y in such a way that the collective set of values satisfy all (in)equalities in S. The projection of the feasible region of S gives us the largest set of values with this property.

Let S be an (in)equality system over X and let $Y \subseteq X$ be the set of variables that we want to eliminate. Without loss of generality we can assume that $X = \{x_1, \ldots, x_{|X|}\}$, $x_1 \prec \ldots \prec x_{|X|}$ and that $Y = \{x_i, x_{i+1}, \ldots, x_{|X|}\}$ for some $1 \leq i \leq |X|$. The projection of the polyhedron $feas(S)$ with respect to Y is then the polyhedron

$$proj_Y\big(feas(S)\big) \stackrel{\text{def.}}{=} \{ \mathbf{r} \in \mathbb{R}^{|X \setminus Y|} \mid \text{there exists an } \mathbf{r}' \in \mathbb{R}^{|Y|} \text{ such that } (\mathbf{r}, \mathbf{r}') \in feas(S) \}.$$

See Figure 1. Here $(\mathbf{r}, \mathbf{r}')$ denotes the vector $(r_1, \ldots, r_{|X \setminus Y|}, r'_1, \ldots, r_{|Y|})$ where $\mathbf{r} = (r_1, \ldots, r_{|X \setminus Y|})$ and $\mathbf{r}' = (r'_1, \ldots, r_{|Y|})$.

For convenience of notation we let $proj_{Y'}\big(feas(S)\big) = proj_{Y' \cap VAR(S)}(feas(S))$ for any $Y' \subseteq \mathcal{X}$.

We notice that for arbitrary subsets Y_1 and Y_2 of Y such that $Y_1 \dot\cup Y_2 = Y$,[1] we have that $proj_Y(P) = proj_{Y_1}(proj_{Y_2}(P))$.

In this report, we will only be interested in the projection of the feasible region of (in)equality systems, and these are the feasible sets of (other) (in)equality systems (see e.g. [Zie95]). What we are interested in are the

[1] $\dot\cup$ denotes disjoint union.

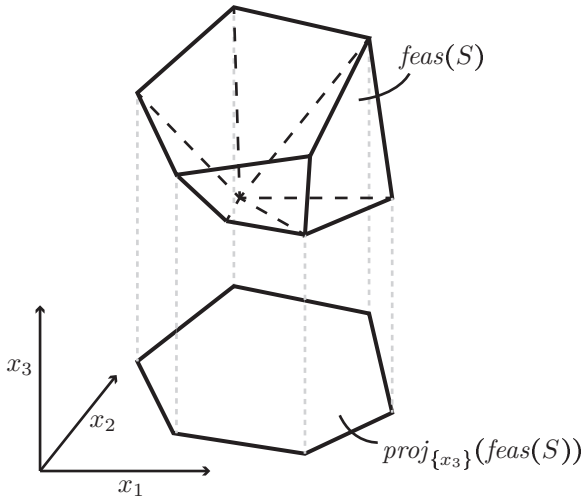


Figure 1: Projection of the polyhedron $feas(S)$ in $\mathbb{R}^3$ with respect to the third variable

dependencies among the non-projected variables, so we want to talk about a system that "generates" a certain projection. For this we define $PRS_Y(S)$ to be the set of (in)equality systems whose feasible set is the projection of S w.r.t. Y, i.e.

$$PRS_Y(S) \stackrel{\text{def.}}{=} \{\, S' \subset \mathcal{IE} \mid VAR(S') = VAR(S) \setminus Y \text{ and } feas(S') = proj_Y\big(feas(S)\big) \,\}.$$

If $S' \in PRS_Y(S)$ we say that S' is a *projection of S w.r.t. Y*.

We point out, that for arbitrary (in)equality systems $S_1, S_2 \subset \mathcal{IE}$ and $Y_1, Y_2 \subseteq \mathcal{X}$, the sets $proj_{Y_1}(feas(S_1))$ and $proj_{Y_2}(feas(S_2))$ are just sets of points in $\mathbb{R}^{|VAR(S_1)\setminus Y|}$ and $\mathbb{R}^{|VAR(S_2)\setminus Y|}$, respectively. As long as $|VAR(S_1)| = |VAR(S_2)|$ these sets can therefore be intersected, unioned or subtracted, but in order to have a meaningful interpretation of e.g. $proj_{Y_1}(feas(S_1)) \cap proj_{Y_2}(feas(S_2))$ in our setting, we must have that $VAR(S_1) \setminus Y_1 = VAR(S_2) \setminus Y_2$, and projections must be given according to the same, implicitly given order $\prec$ on $\mathcal{X}$.

For proving the correctness of our decomposition in Section 4.3 we need the following minor propositions.

Lemma 1. *Let $S, S_1, S_2 \subset \mathcal{IE}$ be (in)equality systems over X and let $Y \subseteq X$.*

1. *Assume that $var(S_1) \cap var(S_2) = \emptyset$. Then*

$$proj_Y\big(feas(S_1 \cup S_2)\big) = proj_Y\big(feas(S_1)\big) \cap proj_Y\big(feas(S_2)\big). \tag{4}$$

2. *Assume that $var(S_1) \cap Y = \emptyset$. Then*

$$proj_Y\big(feas(S_1 \cup S_2)\big) = feas\big((S_1)_{X\setminus Y}\big) \cap proj_Y\big(feas(S_2)\big). \tag{5}$$

3. *Let $X' \subseteq \mathcal{X}$ be a super set of X, i.e. $X \subseteq X'$, and let E be an (in)equality system such that $proj_Y\big(feas(S)\big) = feas(E)$. Then*

$$proj_Y(feas(S_{X'})) = feas(E_{X'\setminus Y}).$$

Proof. See Appendix. □

3 Basic Algorithm

3.1 Fourier-Motzkin-elimination

Traditionally, this method works on pure inequality systems (i.e. with no equalities), and we will describe it like that. Since, an (in)equality system easily can be translated into a pure inequality system, this does not cause any problems. The method is described below.

The method eliminates the variables in $Y \subseteq X$ by creating new inequality systems in a stepwise fashion. At each step, one of the variables in Y is eliminated from the current system to obtain the next system. A variable $x \in Y$ is projected from the pure inequality system S as follows (see e.g. [Imb93] or [Zie95]).

Firstly, the inequalities are divided into disjoint sets depending on the sign of the variable's coefficient in the constraints. Thus, we define

$$\begin{aligned} Pos_S(x) &\stackrel{\text{def.}}{=} \{\, c \in S \mid co(x,c) > 0 \,\}, \\ Neg_S(x) &\stackrel{\text{def.}}{=} \{\, c \in S \mid co(x,c) < 0 \,\}, \text{ and} \\ Zero_S(x) &\stackrel{\text{def.}}{=} \{\, c \in S \mid co(x,c) = 0 \,\}. \end{aligned}$$

To eliminate the variable x and construct the next inequality system, we first set aside $Zero_S(x)$, or rather $\{\, c_{X \setminus \{x\}} \mid c \in Zero_S(x) \,\}$. To this set we then add a new inequality for each combination of $c^+ \in Pos_S(x)$ and $c^- \in Neg_S(x)$. This new inequality is constructed by first normalizing c^+ and c^- such that their coefficients for x are plus and minus 1, respectively, and then the two normalized inequalities are added. In this inequality, the coefficient of x is zero, and restricting the inequality to $X \setminus \{x\}$ we then get the inequality $FM(x, c^+, c^-)$ over $X \setminus \{x\}$ given below.

$$\begin{aligned} &FM(x, c^+, c^-): \\ &\sum_{x' \in VAR(S) \setminus \{x\}} \left(\frac{1}{co(x,c^+)} co(x', c^+) + \frac{1}{co(x,c^-)} co(x', c^-) \right) \cdot x' \leq \frac{1}{co(x,c^+)} rhs(c^+) + \frac{1}{co(x,c^-)} rhs(c^-) \end{aligned} \quad (6)$$

The procedure is described in pseudocode in Algorithm 1.

Algorithm 1 Eliminating variables from an inequality system S using Fourier-Motzkin-elimination.

```
1:  function FM-ELIM(Inequality system S, variables to eliminate Y ⊆ VAR(S))
2:      S' ← S, Y' ← Y
3:      while Y' ≠ ∅ do
4:          x ← CHOOSEVARIABLETODELETE(Y', S')
5:          S' ← FM-ELIMVAR(S', x)
6:          Remove x from Y'
7:      return S'

8:  function FM-ELIMVAR(Inequality system S, variable to eliminate x ∈ var(S))
9:      Divide S into Pos_S(x), Neg_S(x) and Zero_S(x)
10:     S' ← { c_{VAR(S)\{x}} | c ∈ Zero_S(x) }
11:     for all c+ ∈ Pos_S(x) do
12:         for all c- ∈ Neg_S(x) do
13:             Add FM(x, c+, c-) to S'                          ▷ see (6)
14:     return S'
```

When eliminating a variable x_i, the inequalities in the constructed set S' are all implied by the inequalities in S. On the other hand, it also holds that if $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ satisfies the inequalities in S' then there exists an assignment to x_i such that $(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ satisfies the inequalities in S (e.g. [Duf74], Lemma 1). That is, $S' = proj_{\{x_i\}}(S)$.

A disadvantage of the procedure is its complexity. Due to the combination of inequalities in each step, in the worst case scenario where $|Pos_S(x)| = |Neg_S(x)| = \frac{|S|}{2}$ for the current system S, the number of inequalities in the succeeding system is $\frac{1}{4}|S|^2$. This implies that (both time and space) complexity is double-exponential. We also notice that the denser the system is, the more it will grow. It should, however, also be emphasized that not all inequalities in a succeeding system are necessarily non-redundant. In fact, the number of non-redundant inequalities can only grow in single exponential [Mon10].

The function CHOOSEVARIABLETODELETE in Algorithm 1 is a heuristic to determine which variable among the ones needed to be eliminated that should be eliminated first. In principle, there are many choices for heuristics, but the most commonly used heuristic is the greedy heuristic that aims at minimizing the size of the inequality

system in the next step [Duf74][2]. This is easy to calculate from the current system since the increase in size only depends on the size of the sets Pos_S and Neg_S for the variable in question. We have also chosen this heuristic, that is, the function CHOOSEVARIABLETODELETE(Y, S) returns

$$\operatorname*{argmin}_{y \in Y} \big(|Pos_S(y)||Neg_S(y)| - |Pos_S(y)| - |Neg_S(y)| \big).$$

3.2 Gauss-elimination

When our (in)equality system contains equalities, it is often an advantage to let them remain equalities instead of converting each one to two inequalities. Firstly, they are then easier to identify, and instead of doing FM-elimination on a variable in an equality, we can isolate it in the equality and substitute it in the other (in)equalities. By doing this, we avoid the increase in inequalities that an FM-elimination of the variable would otherwise (in most cases) cause. The downside is, that we make the system more dense, which FM-elimination, however, does as well. This is e.g. done in [SK05].

Given an equality c, isolating and substituting the variable $x \in var(c)$ in the (in)equality c' for which $x \in var(c')$ corresponds to scaling c' and c with $\frac{1}{|co(x,c')|}$ and $\frac{-sgn(x,c')}{co(x,c)}$, respectively[3], such that the two coefficients for x are plus and minus 1, and then adding the resulting (in)equalities. Restricting the resulting (in)equality to $VAR(S) \setminus \{x\}$ results in the inequality $GA(x, c, c')$ over $VAR(S) \setminus \{x\}$ given below.

$$GA(x,c,c') : \sum_{x' \in VAR(S) \setminus \{x\}} \left(\frac{-sgn(co(x,c'))}{co(x,c)} co(x',c) + \frac{1}{|co(x,c')|} co(x',c') \right) \cdot x' \\ \odot_{c'} \frac{-sgn(co(x,c'))}{co(x,c)} rhs(c) + \frac{1}{|co(x,c')|} rhs(c'). \quad (7)$$

In our method, we will define auxiliary variables (that should not be eliminated) and we will therefore have a number of equalities which we will use to make Gauss-elimination. The pseudocode for this step is given in Algorithm 2. Also for this procedure, a heuristic is used for choosing which variable - and in this case also which

Algorithm 2 Eliminating variables from an (in)equality system S using Gauss-elimination.

```
1:  function GAUSS-ELIM((In)equality system S, variables to be eliminated Y ⊆ VAR(S))
2:      S' ← S
3:      while Y ∩ var(eqs(S')) ≠ ∅ do
4:          (x, c) ← CHOOSEVARINEQUALITY(Y, S')
5:          S' ← GAUSS-ELIMVAR(S', c, x)
6:      return S'

7:      function GAUSS-ELIMVAR(System S, equality c, variable x ∈ var(c))
8:          S' ← ∅
9:          Remove c from S
10:         for all c' in S do
11:             if co(x, c') = 0 then
12:                 Add c'_{VAR(S)\{x}} to S'
13:             else
14:                 Add GA(x, c, c') to S'                    ▷ see (7)
15:         return S'
```

equality - should be used in the elimination. Again, there are many choices and we have decided to first find the variable (occurring in equations) that occurs the fewest times, and then find the shortest equation using this variable. I.e. the function CHOOSEVARINEQUALITY(Y, S) returns (y, e) where

$$y = \operatorname*{argmin}_{x \in var(eqs(S))} |\{\, c \in S \mid x \in var(c) \,\}| \text{ and } e = \operatorname*{argmin}_{c \in eqs(S).\, y \in var(c)} |var(c)|.$$

[2]This heuristic was useful for us, but though it is very commonly used, note that its success may depend on the specific problems considered; Imbert [Imb93] states in his conclusion to have found the related heuristic returning $\operatorname{argmin}_{y \in Y} |Pos_S(y)||Neg_S(y)|$ to be inferior to "any other choice, even at random".

[3]sgn is the sign-function that returns $+1$, -1 or 0 depending on the sign of its parameter.

3.3 Preprocessing

The purpose of preprocessing is to simplify the (in)equality system that we are dealing with, i.e. to reduce it by removing redundant inequalities that are easily identifiable, updating bounds for variables and substituting variables with implied values *such that* $feas(S)$ does not change. Likewise, preprocessing can be used to figure out at an early stage whether the problem is infeasible. However, we assume that the problem is feasible and do not spend time on checking this during the preprocessing step.

Our preprocessing part consists of the following steps, described below: removing empty (in)equalities and unused variables, updating bounds for (in)equalities with only one variable, substituting variables with fixed values, eliminating variables with only one sign occurring, updating bounds due to extreme values, removing linearly dependent (in)equalities [LHM93], and removing less strict (in)equalities. The basic preprocessing steps can be found in e.g. [BMW75], [AA95] and [Mar03].

When doing these steps, we take into consideration that not all variables should be eliminated, in fact there are some that we need to keep. This is particularly the case when we use auxiliary variables for decomposing our system (see Section 4.3 later). Therefore we do not remove variables that are not in Y, and we do not substitute them with a value during the preprocessing without keeping track of these substitutions.

3.3.1 Removing empty (in)equalities and unused variables

We remove from S all inequalities c for which $|var(c)| = 0$.

For all $x \in Y$ for which $x \notin var(S)$, we simply remove x from $VAR(S)$, i.e. we restrict S to $VAR(S) \setminus \{x\}$. Notice that we do not remove variables that we have not specifically expressed a desire to eliminate.

3.3.2 Updating bounds for (in)equalities with only one variable

For all inequalities c for which $|var(c)| = 1$, we have that c equals the inequality $co(x,c) \cdot x \odot_c rhs(c)$ for an $x \in X$. In this case, we update the bound(s) for x accordingly. That is, if $co(x,c) > 0$ and $\frac{rhs(c)}{co(x,c)} < UB(x)$, then c is removed from S and $UBineq(x)$ is replaced with the inequality $x \leq \frac{rhs(c)}{co(x,c)}$. Similarly, if $co(x,c) < 0$ and $\frac{rhs(c)}{co(x,c)} > lb(x)$, then c is removed from S and $lbineq(x)$ is replaced with the inequality $-x \leq -\frac{rhs(c)}{co(x,c)}$.

If c is an equality, both bounds are updated.

3.3.3 Substituting variables with fixed value

If $UB(x) = lb(x)$ for a variable $x \in X$, then we substitute x with the given value in all (in)equalities in S.

If the variable x is not in Y, meaning that we have not specified that it should be removed, then we do maintain the bounds of x in the system, i.e. we substitute x in all (in)equalities in S except for $lbineq(x)$ and $UBineq(x)$. The main reason for keeping such inequalities is that due to our decomposition of the system (see Section 4.3 later) the system might later be joined with another system where x is present and hence x's value should not "disappear".

3.3.4 Eliminating variables with only one sign occurring

If a variable x only occurs with a non-negative coefficient in S (excluding bounds) and does not occur in equalities, then we can substitute it with its lower bound if one exists; this actually corresponds to doing FM-elimination on x where $Neg_S(x)$ equals $\{lbineq(x)\}$. Notice that unlike the other described steps, this step changes the feasible area of S, though, this change is part of the overall goal of the whole procedure.

In other words, for all $x \in Y \cap P$ for which $lb(x)$ exists, where

$$P = \{\, x \in VAR(S) \setminus var(eqs(S)) \mid \forall c \in S \setminus \{lbineq(x)\} \,.\, co(x,c) \geq 0 \,\},$$

we substitute x with $lb(x)$ in all (in)equalities in S. If $x \in Y \cap P$ and x has no lower bound, we can just remove all inequalities c for which $x \in var(c)$, since this also corresponds to doing an FM-elimination of x. Notice, that we again only do this for variables that we have confirmed should be eliminated.

Likewise, if x has only non-positive coefficients in S, we can substitute is with its upper bound. That is, we substitute variables in

$$N = \{\, x \in VAR(S) \setminus var(eqs(S)) \mid \forall c \in S \setminus \{UBineq(x)\} \,.\, co(x,c) \leq 0 \,\}$$

with their upper bound if it exists and if the variable is in Y. As above, if a variable $x \in N \cap Y$ has no upper bound, then we simply remove all inequalities c for which $x \in var(c)$.

3.3.5 Updating bounds due to extreme values

For an (in)equality c in S we can – given appropriate bounds for the variables in $var(c)$ – calculate the highest and the lowest value that the left-hand-side of c can reach when variables are restricted to $feas(S)$ ([Mar03], [AA95]). The highest value is

$$high(c) = \sum_{x \in X. co(x,c) > 0} co(x,c) \cdot UB(x) + \sum_{x \in X. co(x,c) < 0} co(x,c) \cdot lb(x), \tag{8}$$

while the lowest value is

$$low(c) = \sum_{x \in X. co(x,c) > 0} co(x,c) \cdot lb(x) + \sum_{x \in X. co(x,c) < 0} co(x,c) \cdot UB(x). \tag{9}$$

If c is an equality, and $low(c) = rhs(c)$ ($high(c) = rhs(c)$) then all variables in $var(c)$ needs to be have the value of the bound that gives the lowest (highest) value in order to be in $feas(S)$, and we substitute the variables in $var(c)$ with these values in all (in)equalities in S.

If c is not an equality then we still substitute variables with the appropriate value if $low(c) = rhs(c)$. If $high(c) \leq rhs(c)$ then c will always be satisfied for variables in $feas(S \setminus \{c\})$, i.e. it is redundant and we will remove it.

Now assume x is a variable in $var(c)$. Staying within the feasible area given by c and also satisfying the bounds for the other variables in c can then imply tighter bounds on x.

As in (9) we can calculate the lowest reachable value for c (within $feas(S)$) when disregarding the contribution of x by not summing over x. We will call this value $low_x(c)$. We notice that if $low(c)$ exists, we have that $low_x(c) = low(c) - co(x,c) \cdot bound(x)$, where $bound(x) = lb(x)$ if $co(x,c) > 0$, and $bound(x) = UB(x)$ if $co(x,c) < 0$.

In order for $feas(c)$ to be non-empty it is then necessary that $co(x,c) \cdot x + low_x(c)$ is at most $rhs(c)$. That is, if $co(x,c) > 0$ then $x \leq \frac{rhs(c) - low_x(c)}{co(x,c)}$, and if $co(x,c) < 0$ then $x \geq \frac{rhs(c) - low_x(c)}{co(x,c)}$. We therefore update the upper of lower bound for x correspondingly.

This step is performed as in Algorithm 3. We say that $x \in var(c)$ has an appropriate high bound, if $UB(x)$ exists when $co(x,c) > 0$, and $lb(x)$ exists when $co(x,c) < 0$. Likewise, $x \in var(c)$ has an appropriate low bounds if $lb(x)$ exists when $co(x,c) > 0$, and $UB(x)$ exists when $co(x,c) < 0$.

We notice that the step "updating bounds for (in)equalities with only one variable" is a special case of the above.

3.3.6 Removing linearly dependent (in)equalities

Consider the two *different* (in)equalities c and c' in S. Assume that the left-hand-side of c is a multiple of the left-hand-side of c', i.e. there is an $\sigma \in \mathbb{R}$ such that $co(x,c) = \sigma \cdot co(x,c')$ for all $x \in var\{c,c'\}$. Given that S is feasible, then one of the (in)equalities are redundant [LHM93].

More concretely, we can conclude that c is redundant in the following cases.

- $c \notin eqs(S)$, $\sigma \geq 0$ and $\sigma \cdot rhs(c') \leq rhs(c)$.
- $c \notin eqs(S)$, $c' \in eqs(S)$, $\sigma < 0$ and $\sigma \cdot rhs(c') \leq rhs(c)$.
- $c, c' \in eqs(S)$ and $\sigma \cdot rhs(c') \leq rhs(c)$.

In all these cases it holds that if $\mathbf{r} \in feas(c')$ then

$$\mathbf{co}(c) \cdot \mathbf{r} = \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} \leq \sigma \cdot rhs(c') \leq rhs(c). \tag{10}$$

In the first two cases this means that $\mathbf{r} \in feas(c)$, and hence c is redundant. In the last case, since S is assumed feasible there is an $\mathbf{r}' \in feas(c) \cap feas(c')$, i.e. $rhs(c) = \mathbf{co}(c) \cdot \mathbf{r}' = \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r}' = \sigma \cdot rhs(c')$. Thus, $\mathbf{co}(c) \cdot \mathbf{r} = \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} = \sigma \cdot rhs(c') = rhs(c)$, and hence c is redundant.

Algorithm 3 Removing (in)equalities that are implied due to the bounds of its variables, and updating bounds of variables implied by the right-hand-side of the (in)equality (and the bound of the other variables).

function FORCEDBYBOUNDS((In)equality system S)
  $S' \leftarrow S$
  for $c \in S'$ **do**
    $(red, Subst) \leftarrow$ REDUNDANTDUETOBOUNDS?(S', c)
    if red **then**
      Remove c from S'
      for all $(x, v) \in Subst$ **do**
        Substitute x with v in all (in)equalities in S'
    else
      $(UBs, lbs) \leftarrow$IMPLIEDBOUNDS(S', c)
      for all $(x, b) \in UBs$ **do**
        Replace $UBineq(x)$ in S' with $x \leq b$
      for all $(x, b) \in lbs$ **do**
        Replace $lbineq(x)$ in S' with $-x \leq -b$
  return S'

function REDUNDANTDUETOBOUNDS?(S, $c \in S$)
  $red \leftarrow$ `false`, $Subst \leftarrow \emptyset$
  $high \leftarrow \infty$, $low \leftarrow -\infty$
  if x has an appropriate high bound for all $x \in var(c)$ **then**
    $high \leftarrow high(c)$ ▷ see (8)
  if x has an appropriate low bound for all $x \in var(c)$ **then**
    $low \leftarrow low(c)$ ▷ see (9)
  if $low = rhs(c)$ **then**
    $red \leftarrow$ `true`
    for all $x \in var\{c\}$ **do**
      Add (x, x's appropriate low bound) to $Subst$
  else if $c \in eqs(S)$ and $high = rhs(c)$ **then**
    $red \leftarrow$ `true`
    for all $x \in var\{c\}$ **do**
      Add (x, x's appropriate high bound) to $Subst$
  else if $high \leq rhs(c)$ **then**
    $red \leftarrow$ `true`
  return $(red, Subst)$

function IMPLIEDBOUNDS(S, $c \in S$)
  $UBs \leftarrow \emptyset$, $lbs \leftarrow \emptyset$
  for $x \in var(c)$ **do**
    if y has an appropriate low bound for all $y \in var(c) \setminus \{x\}$ **then**
      $b \leftarrow \frac{rhs(c) - low_x(c)}{co(x,c)}$
      if $co(x, c) > 0$ and $b < UB(x)$ in S **then**
        Add (x, b) to UBs
      else if $co(x, c) < 0$ and $b > lb(x)$ in S **then**
        Add (x, b) to lbs
  return (UBs, lbs)

If c is redundant because of an (in)equality c' as above, we say that c is *linearly dependent* compared to c'[4]. For a given pair of (in)equalities c and c' it is of course enough to check whether a specific value of σ, namely $\frac{co(x,c)}{co(x,c')}$ for the first variable in $var(c) \cap var(c')$, satisfies the given criteria; if $var(c) \cap var(c') = \emptyset$ then such an σ does not exist unless $var(c) = var(c') = \emptyset$, which is a case we disregard here since it should be taken case of when removing empty (in)equalities.

Linearly dependent (in)equalities are removed from the system S by using REMOVELINEARLYDEPD in Algorithm 4.

Algorithm 4 Removing linearly dependent (in)equalities from an (in)equality system S.

```
1: function RemoveLinearlyDepd((In)equality system S)
2:     S' ← S
3:     for all c in S' do
4:         for all c' in S' \ {c} do
5:             if c is linearly dependent compared to c' then
6:                 Remove c from S'
7:     return S'
```

3.3.7 Removing less strict inequalities

Consider again two different (in)equalities c and c' in S. Assume that all variables in $var(\{c, c'\})$ are non-negative (i.e. their lower bound are greater or equal to 0), and there exists a scalar $\sigma \geq 0$ such that

$$co(x, c) \leq \sigma \cdot co(x, c') \text{ for all } x \in var(\{c, c'\}) \text{ and } \sigma \cdot rhs(c') \leq rhs(c). \tag{11}$$

If $\mathbf{r} \in feas(c')$, then (11) implies that $\mathbf{co}(c) \cdot \mathbf{r} \leq \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} \leq \sigma \cdot rhs(c') \leq rhs(c)$, i.e. $\mathbf{r} \in feas(c)$ if $c \notin eqs(S)$. That is, c is redundant if it is not an equality. Assuming that S is feasible and that (11) holds while $c \in eqs(S)$, then we must have that the inequalities in (11) are in fact equalities; otherwise if $\mathbf{r} \in feas(S)$ then either $\mathbf{co}(c) \cdot \mathbf{r} < \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} \leq \sigma \cdot rhs(c') \leq rhs(c) = \mathbf{co}(c) \cdot \mathbf{r}$, or $\mathbf{co}(c) \cdot \mathbf{r} \leq \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} \leq \sigma \cdot rhs(c') < rhs(c) = \mathbf{co}(c) \cdot \mathbf{r}$, which are both contradictions. This again implies that c is redundant since $\mathbf{r} \in feas(c')$ implies $\mathbf{co}(c) \cdot \mathbf{r} = \sigma \cdot \mathbf{co}(c') \cdot \mathbf{r} = \sigma \cdot rhs(c') = rhs(c)$.

For given inequalities c and c' it is enough to check whether a specific value of σ satisfies (11) (see Proposition 1 below). If this is the case, we say that c is *less strict* than c'.

Proposition 1. *Let c and c' be given and assume that all variables in $var(\{c, c'\})$ are non-negative. If* (11) *hold for an $\sigma \geq 0$, then* (11) *holds for $\sigma' \geq 0$ given below.*

If there exists an $x \in var(c')$ such that $co(x, c') < 0$ then

$$\sigma' = \begin{cases} \min(m, \frac{rhs(c)}{rhs(c')}) & \text{if } rhs(c') > 0 \\ m & \text{otherwise} \end{cases}, \text{ where } m = \min_{\substack{x \in var(c'). \\ co(x,c') < 0}} \frac{co(x, c)}{co(x, c')}.$$

Otherwise, if $co(x, c') \geq 0$ for all $x \in var(c')$ and $var(c') \neq \emptyset$, then

$$\sigma' = \begin{cases} \max(m', \frac{rhs(c)}{rhs(c')}, 0) & \text{if } rhs(c') < 0 \\ \max(m', 0) & \text{otherwise} \end{cases}, \text{ where } m' = \max_{\substack{x \in var(c'). \\ co(x,c') > 0}} \frac{co(x, c)}{co(x, c')}.$$

Otherwise $\sigma' = \begin{cases} \max(\frac{rhs(c)}{rhs(c')}, 0) & \text{if } rhs(c') \neq 0 \\ 0 & \text{otherwise} \end{cases}$.

Proof. See Appendix. □

We implement the removal of less strict inequalities in a similar way to how the removal of linearly dependent inequalities are done. That is, we do the same as in Algorithm 4, where line 5 has been replaced with checks for whether c is less strict than c'. According to the above, we can check this by checking that all variables in $var(\{c, c'\})$ are non-negative, and (11) holds for σ given in Proposition 1.

[4]Note, [LHM93] differentiates and calls c syntactically redundant if $rhs(c) = \sigma \cdot rhs(c')$ (for positive σ), and quasi-syntactically redundant in the first case (first item) considered.

3.3.8 Overall preprocessing step

The overall preprocessing step then consists of doing all the above mentioned steps repeatedly in a cycle, as long as "something happens" in a cycle, that is, an (in)equality or variable has been removed or substituted with a value, or a bound has been updated.

Algorithm 5 Preprocessing an (in)equality system S. The variables in $VAR(S) \setminus Y$ should be kept.

1: **function** PREPROCESS((In)equality system S, variables $Y \subseteq VAR(S)$)
2: $S' \leftarrow S$
3: **do**
4: $Start \leftarrow S'$
5: $S' \leftarrow$REMOVEEMPTYINEQS(S', Y) ▷ See Section 3.3.1
6: $S' \leftarrow$UPDATEBOUNDSONEVAR(S', Y) ▷ See Section 3.3.2
7: $S' \leftarrow$SUBSTFIXEDVARS(S', Y) ▷ See Section 3.3.3
8: $S' \leftarrow$REMOVEUNUSEDVARS(S', Y) ▷ See Section 3.3.1
9: $S' \leftarrow$REMOVEVARSWITHONESIGN(S', Y) ▷ See Section 3.3.4
10: $S' \leftarrow$FORCEDBYBOUNDS(S', Y) ▷ See Algorithm 3
11: $S' \leftarrow$REMOVELINEARLYDEPD(S', Y) ▷ See Algorithm 4
12: $S' \leftarrow$REMOVELESSSTRICTINEQS(S', Y) ▷ See Section 3.3.7
13: **while** $S' \neq Start$
14: Return S'

3.4 Removing redundancy

As previously described, at each step of the FM-elimination, the number of (in)equalities will grow with $|Pos_S(x)| \cdot |Neg_S(x)| - |Pos_S(x)| - |Neg_S(x)|$ inequalities. Unless either $Pos_S(x)$ or $Neg_S(x)$ equals one, or both numbers equal two, this will account for an increase in the number of inequalities. For a large, dense system, the growth will be substantial, which prohibits it from use for practical purposes *if* the added inequalities are non-redundant (See e.g. [LHM93] and [LS08]). However, many of the added inequalities are potentially redundant, and it would greatly improve the applicability of the procedure if the addition of (too many) redundant inequalities can be (efficiently) avoided, or the redundant inequalities can be removed (efficiently) after they have been added, or both.

Theoretically, all redundant inequalities can be removed from an (in)equality system S by examining each inequality in turn and remove them when the property in (3) holds, see Algorithm 6. Notice, that this procedure does not examine equalities; this could of course have been implemented, but we chose not to, partly for reasons to be revealed later (see Section 4.3).

We notice that if c is a non-redundant inequality in S that lies in $Zero_S(x)$ for some variable x, then $c_{VAR(S)\setminus\{x\}}$ is also a non-redundant inequality in S' for all $S' \in PRS_{\{x\}}(S)$. Therefore, assuming that we have already removed redundant inequalities from the system S, it is therefore only necessary to check all the new inequalities being added to the system after each FM-elimination.

Algorithm 6 Removing redundant inequalities from an (in)equality system S. The maximum value in line 4 can be calculated with an lp-solver

1: **function** REMOVEREDUNDANTINEQS((In)equality system S, Inequalities to be checked $C \subseteq S \setminus eqs(S)$)
2: $S' \leftarrow S$
3: **for all** $c \in C$ **do**
4: **if** $\max\{\ \mathbf{co}(c) \cdot \mathbf{r} \mid \mathbf{r} \in feas(S' \setminus \{c\})\ \} \leq rhs(c)$ **then**
5: Remove c from S'
6: **return** S'

This method can then be called at "appropriate" times during the main algorithm, for example each time a certain number of inequalities have been added since last redundancy check, or possibly after every elimination.

4 Alterations and Improvements

4.1 Improved redundancy checks

4.1.1 Taking advantage of data's uncertainty

It is not uncommon that the coefficients and right-hand-sides in the considered (in)equality system comes from approximations and/or predictions of certain circumstances, and hence the exact vertices of the feasible area is not given with high accuracy. We will therefore allow slight discrepancies between the corners of the final solution space given by our algorithm and the actual projection. When calculating the maximum in line 4 in Algorithm 6 we will therefore allow the maximum to be within a given relative ϵ. That is, the condition in line 4 is replaced with the following:

$$\max\{\ \mathbf{co}(c) \cdot \mathbf{r} \mid \mathbf{r} \in \mathit{feas}(S \setminus \{c\})\ \} \leq \mathit{rhs}(c) + \epsilon \cdot |\mathit{rhs}(c)|. \tag{12}$$

Though, if $\mathit{rhs}(c) = 0$ then we instead require the maximum to be within a certain treshold, ϵ'.

If c satisfies the condition in (12), we call c *almost redundant*, while a c that satisfies the redundancy criteria in (3) is called *truly redundant*. Our definition of an almost redundant inequality is similar to the definition in [LS08] and [SL12]. In these paper, the authors similarly use a method for coarsening the boundary of the feasible area that relies on removing almost redundant inequalities, though it differs a bit from ours.

4.1.2 Concurrent redundancy checks

The overall setup is that a manager keeps track of $k > 1$ redundancy checkers to which it distributes the inequalities that need to be checked. The redundancy checkers run in parallel. Each checker starts by removing the inequalities from their own copy of the original system that the other checkers have found truly redundant so far. Then it checks if its assigned inequality is truly redundant compared to this system (see (3) on page 3) and tells the result to the manager when done. The manager then gives the idle checker a new inequality to check (if there are any left) and remembers to inform the other checkers if the examined inequality was redundant.

When using parallel redundancy checkers, it is important that we do *not* remove almost redundant inequalities, i.e. inequalities that satisfy (12). Otherwise, we might be in a situation as illustrated in Figure 2, where the two inequalities c and c' both are almost redundant (each due to the other one). Each of them could justifiably be removed *as long as* we keep the other one; removing both inequalities would result in a significantly different feasible area. Unfortunately, using parallel redundancy checkers, c and c' could be checked by different checkers which would then both report that their assigned inequality should be removed because it is almost redundant resulting in both inequalities being removed. Thus, we must use true redundancy for parallel redundancy checkers. For similar reasons, when doing parallel redundancy checks it is also required that S does not contain linearly dependent inequalities c and c' for which $\mathbf{co}(c) = \sigma \cdot \mathbf{co}(c')$ and $\mathit{rhs}(c) = \sigma \cdot \mathit{rhs}(c')$ for a positive σ, since c and c' in that case would define the same halfspace.

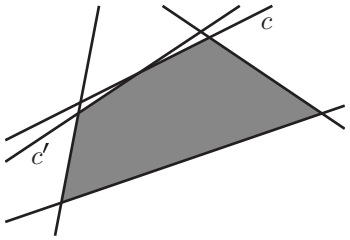


Figure 2: Both inequalities c and c' are almost redundant, but only one of them can be removed from the system.

However, we still let the checkers examine whether the inequalities are almost redundant (see (12)) and report this back to the manager who stores these. After having examined all the needed inequalities by use of parallel redundancy checkers, we then go through and check the almost redundant ones sequentially with a single redundancy checker, and remove - in turn - the ones that are still almost redundant. For the example in Figure 2 this would mean that which ever of c and c' is examined first is also removed, while the other is not.

The checkers also check and report on inequalities for which the optimization of its left hand side resulted in a bad condition number (also known as κ-value). This number is a measure for how much (small) differences in the input value (i.e. the coefficients and right-hand-side of the (in)equalities) effects the value of output (i.e. the

found optimal value); see e.g. [TBI07]. Essentially, the larger this number is, the less we can trust the result. We have chosen a threshold $\mathcal{K}$ that gives the limit for when we can trust the result. As the system gets smaller due to the removal of redundant inequalities, inequalities might be reexamined with a lower κ-value as a result, and therefore we also re-examine the inequalities that resulted in a bad κ-value when they were checked by the parallel redundancy checkers. Finally we also reexamine inequalities, for which the lp-solver was not able to solve the optimization of its left-hand-side.

The pseudocode for the redundancy checkers is listed in Algorithm 7, and the pseudocode for the manager is listed in Algorithm 8. When the manager receives results from the checkers (via calls to MANAGERESULT in line 21 in Algorithm 8), it finishes handling one call before handling the next call.

Algorithm 7 Initializing and running a redundancy checker. A checker can either check for true redundancy (used when doing concurrent redundancy checks) or not, in which case it checks for "almost redundancy" (for doing sequential redundancy checks).

```
1:  function CHECKREDUNDANCY(Manager m, (In)equality system S, c ∈ S, boolean truely)
2:      S' ← S, c' ← c, reds ← ∅
3:      while c' ≠ null do
4:          Remove reds and c' from S'
5:          max ← max{ co(c') · r | r ∈ feas(S') }
6:          Inspect solution:
7:              red ← false, recheck ← false
8:              κ ← GETCONDITIONNUMBER()
9:              if truely then
10:                 red ← ISTRUELYREDUNDANT?(max, rhs(c')) and κ ≤ K
11:                 recheck ← ISALMOSTREDUNDANT?(max, rhs(c'))
12:             else
13:                 red ← ISALMOSTREDUNDANT?(max, rhs(c')) and κ ≤ K
14:             recheck ← recheck or κ > K or UNSOLVED?()
15:         if ¬red then
16:             Add c' to S'
17:         (c', reds) ← m.MANAGERESULT(c', red, recheck)
18:     return

19: function ISTRUELYREDUNDANT?(max, rhs)
20:     return max ≤ rhs

21: function ISALMOSTREDUNDANT?(max, rhs)
22:     return ( rhs = 0 and max ≤ ε') or max ≤ rhs + ε · |rhs|
```

We notice that the inequalities that should be checked for redundancy is a subset of (in)equalities in S. Essentially, these should be the new inequalities added to the system after each FM-elimination as described earlier. However, the function is implemented such that it is also possible to check the whole system for redundant inequalities.

4.2 Combining strategies

For projecting an (in)equality system we combine the various strategies described so far in the method PROJECT presented below in Algorithm 9. It consists of steps of preprocessing the system, doing Gauss-elimination and doing FM-elimination on the system. After each FM-elimination of a variable some (timewise) "cheap" preprocessing steps are done, and we also make sure to remove linearly dependent (in)equalities as well. Hence CLEANUP consists of Algorithm 5 minus the calls to FORCEDBYBOUNDS and REMOVELESSSTRICTINEQS. The reason we do not simply use PREPROCESS is that the two above mentioned subprocedures in experience are more time consuming than the others and prior results showed a too little effect compared to the time taken.

After this, we remove redundant inequalities (we check only the newly added inequalities) using REMOVEREDUNDANCY from Algorithm 8. In REMOVEREDUNDANCY, each step of doing parallel or sequential redundancy

Algorithm 8 Managing the redundancy checkers. First a concurrent redundancy check is performed, followed by a sequential redundancy check. It is assumed that C', $reds$ and $rechecks$ can be accessed by any of the manager's functions in this algorithm.

```
1:  function REMOVEREDUNDANCY(System S, inequalities C ⊆ S ∖ eqs(S), number of checkers k)
2:      S' ← S, C' ← C, reds ← ∅, rechecks ← ∅, checkers ← ∅
3:      Do parallel redundancy check:
4:          do in parallel for i ← 1 to min{k, |C'|}
5:              Add new checker t to checkers
6:              Call t.CHECKREDUNDANCY(this, S', NEXTINEQTOCHECK(), true)
7:          until all checkers in checkers have returned

8:      S' ← S' ∖ reds, C' ← rechecks, reds ← ∅, rechecks ← ∅
9:      Do sequential redundancy check:
10:         if C' ≠ ∅ then
11:             t ← new checker, checkers ← {t}
12:             Call t.CHECKREDUNDANCY(this, S', NEXTINEQTOCHECK(), false)
13:     return S' ∖ reds

14: function NEXTINEQTOCHECK()
15:     if C' = ∅ then
16:         return null
17:     else
18:         c ← first inequality from C'
19:         Remove c from C'
20:         return c

21: function MANAGERESULT(Inequality c, boolean red, boolean recheck)
22:     if red then
23:         Add c to reds
24:     else if recheck then
25:         Add c to rechecks
26:     return (NEXTINEQTOCHECK(), reds)
```

checks, respectively, are potentially repeated a number of times (or until the system does not change) depending on the values in the considered model, since "off"-numbers (according to experience) are more likely to cause bad κ-values. If there are many inequalities resulting in bad κ-values, we would then like to test them again (after other redundant inequalities have been removed), since the removal of other inequalities can lead to a better κ-value when the inequality is re-evaluated. Due to the sometimes large number of inequalities that we do not know the redundancy staus of (because of a bad κ-value, that do not change during the repeated checks for redundancy), in some runs of the method we also start the FM-elimination-step in PROJECT with a full sequential redundancy check.

Algorithm 9 Overview of the method for projecting the variables Y from an (in)equality system S.

1: **function** PROJECT(S,Y)
2: $S' \leftarrow S$
3: $S' \leftarrow \text{PREPROCESS}(S', Y)$ ▷ See Algorithm 5
4: $S' \leftarrow \text{GAUSS-ELIM}(S', Y)$ ▷ See Algorithm 2
5: $S' \leftarrow \text{FM-ELIM*}(S', Y)$ ▷ FM-elimination with redundancy removal (see below)

6: **function** FM-ELIM*(S, Y)
7: $S' \leftarrow S, Y' \leftarrow Y$
8: **while** $Y' \neq \emptyset$ **do**
9: $x \leftarrow \text{CHOOSEVARIABLETODELETE}(Y', S')$
10: $S'' \leftarrow (Zero_{S'}(x))_{VAR(S') \setminus \{x\}}$
11: $S' \leftarrow \text{FM-ELIMVAR}(S', x)$ ▷ Algorithm 1
12: Remove x from Y'
13: $S' \leftarrow \text{CLEANUP}(S', Y)$
14: $S' \leftarrow \text{REMOVEREDUNDANCY}(S', S' \setminus S'', \text{available threads})$ ▷ Algorithm 8
15: **return** S'

4.3 Decomposition

Given the theoretical complexity of the Fourier-Motzkin-elimination procedure, we might not have great hope of using this to eliminate a large number of variables from a very large (in)equality system, at least not if the values of the (in)equalities are randomly generated. However, given an (in)equality system coming from a natural occurring problem, it is common that the constraints are *block structured* [Wil78], where groups of constraints are "local" for each subdomain of the problem, while other "global" or "transverse" constraints involve many or all of the local subdomains. We will in the following assume that the given problem has a block structure and exploit this fact.

Given a problem with block structure, the dense, "transverse" (in)equalities are problematic for Fourier-Motzkin-elimination, especially when they use variables that should not be eliminated, and particularly when they are equalities. Every time a variable in a transverse equality is eliminated, all the "local" (in)equalities using the variable *will be combined* with one of the two inequalities corresponding to the transverse equality. The result is that we get a large number of inequalities using all the variables of the large transverse equality *and* the variables from the local (in)equalities. Thus, not only do we increase the size of the problem, we also make it (much) more dense, both making the rest of the elimination process more time-consuming, and also increasing the time spend on finding redundant inequalities. Below we show how to avoid this.

4.3.1 Separating the (in)equality system

If possible, it would be an advantage to be able to separate our (in)equality system into a (preferably large) number of smaller (in)equality systems that have "nothing in common", meaning that there is no overlap between the sets of variables being used in the different subsystem. If this can be done, we can then solve each subsystem separately in parallel after which the resulting systems can be combined (see Lemma 1 item 1). As long as each subsystem is sufficiently small (such that it is can be projected in "reasonable" time) this may turn an insurmountable large problem into a projectable one. However, this is not always possible to do. Nonetheless, using the block

structure to decompose the system into smaller subsystems with only little interaction by separating the system's (in)equalities can still be useful.

We therefore consider k^0 disjoint subsets of $VAR(S)$ - $X_1, X_2, \ldots, X_{k^0}$ - and let $X_t \stackrel{\text{def.}}{=} VAR(S) \setminus var(\cup_{1 \le i \le k^0} X_i)$. For each $i \in \{1, \ldots, k^0\}$ we define S_i to be the (in)equality system that only uses variables in X_i, i.e. $S_i \stackrel{\text{def.}}{=} \{ c \in S \mid var(c) \subseteq X_i \}$, and we let $S_t = S \setminus \cup_{1 \le i \le k_0} S_i$ be the system consisting of the rest of the (in)equalities.

The result is that we have divided the system into k^0 *local* parts with disjoint sets of used variables, and a *transverse* part that contains (in)equalities using variables from more than one of the other local parts.

Our (in)equality system can thus be illustrated as in Figure 3.

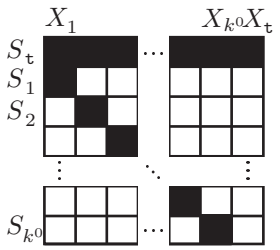


Figure 3: Separation of the (in)equalities of an (in)equality system into local parts and a transverse part. The figure shows the coefficient matrix for the system, where non-zero sections have been colored black.

It should be noticed that in general, this can be done in many ways, but in our case - and in many common cases - there is a natural way in which the local parts/local variables present themselves.

We likewise notice that for any (in)equality system, such partitions of variables and (in)equalities will always exist though they might not be very useful in practice; for any $X \subseteq VAR(S)$, we can make a partition of the (in)equalities as desired, $S = \{ c \in S \mid var(c) \subseteq X \} \dot\cup \{ c \in S \mid var(c) \nsubseteq X \}$. Intuitively, though, we have a better chance of success if S_t is as small as possible, as well as each S_i and/or X_i.

Example 4.1. To illustrate, let us consider the (in)equality system S consisting of the following (in)equalities from which we want to eliminate all variables but u, which is defined as a sum of the variables $x_1, y_1, x_2, y_2, x_3, y_3$:

$$sum : -u + x_1 + y_1 + x_2 + y_2 + x_3 + y_3 = 0$$

$$S_1 : \begin{cases} x_1 + 2 \cdot y_1 & \le 2 \\ -x_1 & \le 0 \\ -y_1 & \le 0 \end{cases}, \quad S_2 : \begin{cases} -x_2 - 3 \cdot y_2 & \le 1 \\ x_2 & \le 0 \\ y_2 & \le 0 \end{cases}, \quad S_3 : \begin{cases} -x_3 + y_3 & \le 1 \\ 2 \cdot x_3 - y_3 & \le 0 \\ -y_3 & \le 0 \end{cases}.$$

S_1, S_2 and S_3 are "local" subsystems only dealing with variables from $\{x_1, y_1\}$, $\{x_2, y_2\}$ and $\{x_3, y_3\}$, respectively. On the other hand, the equality defining u, sum, is a transverse equality. When eliminating any x or y variable, the system will be added (in)equalities using all variables (except the eliminated one) due to combinations with sum. △

Introducing auxiliary variables As previously mentioned it is preferable to perform elimination on a (local) subsystem such that only (in)equalities naturally "belonging" to that subsystem are produced, meaning that added (in)equalities uses no variables from other (local) subsystems. This can be achieved by the use of auxiliary variables.

Assume that $c : \sum_{x \in VAR(c)} co(x, c) \cdot x \odot_c rhs(c)$ belongs to S_t. For all $1 \le i \le k^0$ we then define a variable[5], $z^0_{c,i} \stackrel{\text{def.}}{=} \sum_{x \in X_i} co(x, c) \cdot x$, which can be written as an equality over $X_i \cup \{z^0_{c,i}\}$ as

$$Def(z^0_{c,i}) : -z^0_{c,i} + \sum_{x \in X_i} co(x, c) \cdot x = 0. \tag{13}$$

[5]Formally, we define the value of a variable in $\mathcal{X} \setminus VAR(S)$ which we will rename $z^0_{c,i}$, and so forth with all the following, defined variables.

Since $VAR(S) = X_1 \dot\cup \ldots \dot\cup X_{k^0} \dot\cup X_{\mathtt{t}}$ and $co(x,c) = 0$ for $x \in VAR(S) \setminus var(c)$, we therefore have that

$$\begin{aligned}\sum_{x \in VAR(c)} co(x,c) \cdot x &= \sum_{x \in VAR(S)} co(x,c) \cdot x \\ &= \Big(\sum_{1 \le i \le k^0} \Big(\sum_{x \in X_i} co(x,c) \cdot x \Big) \Big) + \sum_{x \in X_{\mathtt{t}}} co(x,c) \cdot x \\ &= \Big(\sum_{1 \le i \le k^0} z^0_{c,i} \Big) + \sum_{x \in X_{\mathtt{t}}} co(x,c) \cdot x,\end{aligned}$$

which means that using the $z^0_{c,i}$-variables, c can be rewritten as an (in)equality over $\cup_{1 \le i \le k_0} \{z^0_{c,i}\} \cup X_{\mathtt{t}}$ as

$$c^0_{decp} : \Big(\sum_{1 \le i \le k^0} z^0_{c,i} \Big) + \sum_{x \in X_{\mathtt{t}}} co(x,c) \cdot x \odot_c rhs(c). \tag{14}$$

From construction it therefore follows that values for $x \in VAR(S)$ that satisfies c can be extended with values for all $z^0_{c,i}$ such that c^0_{decp} and all the defining equalities (from (13)) are satisfied, and vice versa, if the latter (in)equalities are all satisfied then satisfying values for the variables in X will also satisfy c. That is,

$$feas(c) = proj_{\cup_{1 \le i \le k^0} \{z^0_{c,i}\}} \Big(feas(\{c^0_{decp}\} \cup \bigcup_{1 \le i \le k^0} \{Def(z^0_{c,i})\}) \Big). \tag{15}$$

Using auxiliary variables for all $c \in S_{\mathtt{t}}$ as above and adding the defining equalities $Def(z^0_{c,i})$ to the subsystem where they belong according to their variables, we can divide the (in)equality system S into a number of (in)equality systems, $S^0_1, \ldots, S^0_{k^0}, S^0_{\mathtt{t}}$, as described in the pseudocode below.

Algorithm 10 Separating an (in)equality system S according to a list of disjoint sets of variables $\mathbb{X} = (X_1, X_2, \ldots, X_{k^0})$, where each $X_i \subseteq VAR(S)$

function SEPARATEINEQS($S, \mathbb{X} = (X_1, X_2, \ldots, X_{k^0})$)
  $S_{\mathtt{t}} \leftarrow \{\, c \in S \mid var(c) \not\subseteq X_i \text{ for any } i \,\}$
  for $i \leftarrow 1$ to k^0 **do**
    $S^0_i \leftarrow \{\, c_{X_i} \mid c \in S, var(c) \subseteq X_i \,\}$
  $S^0_{\mathtt{t}} \leftarrow \varnothing$
  for all $c \in S_{\mathtt{t}}$ **do**
    for $i \leftarrow 1$ to k^0 **do**
      Add $Def(z^0_{c,i}) : -z^0_{c,i} + \sum_{x \in X_i} co(x,c) \cdot x = 0$ to S^0_i
    Add $c^0_{decp} : (\sum_{1 \le i \le k^0} z^0_{c,i}) + \sum_{x \in X_{\mathtt{t}}} co(x,c) \cdot x \odot_c rhs(c)$ to $S^0_{\mathtt{t}}$
  return $((S^0_1, \ldots, S^0_{k^0}), S^0_{\mathtt{t}})$

The (in)equality system consisting of the union of the systems $S^0_1, \ldots, S^0_{k^0}, S^0_{\mathtt{t}}$ will be referred to as the *separated system from* S, written $sep(S)$. This can be illustrated as in Figure 4. For ease of notation we let $Z^0 \stackrel{\text{def.}}{=} \cup_{1 \le i \le k^0} \{\, z^0_{c,i} \mid c \in S_{\mathtt{t}} \,\}$. We notice that for each constructed system S^0_i we have that $VAR(S^0_i) \subseteq X_i \cup Z^0$ and $VAR(S^0_t) = X_{\mathtt{t}} \cup Z^0$.

Example 4.2. Consider again the (in)equality system S from Example 4.1. As explained, separating S introduces auxiliary variables to prevent the immediate mix of variables from $var(S_1)$, $var(S_2)$ and $var(S_3)$ in new inequalities when any of the variables are eliminated. This is done by defining u as the sum of three auxiliary variables, z_1, z_2 and z_3[6], which in turn are defined as the sum of the variables in each respective subsystems. Hence $sep(S)$ is the following system.

$$sum^0_{decp} : -u + z_1 + z_2 + z_3 = 0$$

$$S^0_1 : \begin{cases} -z_1 + x_1 + y_1 & = 0 \\ x_1 + 2 \cdot y_1 & \le 2 \\ -x_1 & \le 0 \\ -y_1 & \le 0 \end{cases}, \quad S^0_2 : \begin{cases} -z_2 + x_2 + y_2 & = 0 \\ -x_2 - 3 \cdot y_2 & \le 1 \\ x_2 & \le 0 \\ y_2 & \le 0 \end{cases}, \quad S^0_3 : \begin{cases} -z_3 + x_3 + y_3 & = 0 \\ -x_3 + y_3 & \le 1 \\ 2 \cdot x_3 - y_3 & \le 0 \\ -y_3 & \le 0 \end{cases}.$$

[6]For clarity of the example we do not use the variable names as described in the section above.

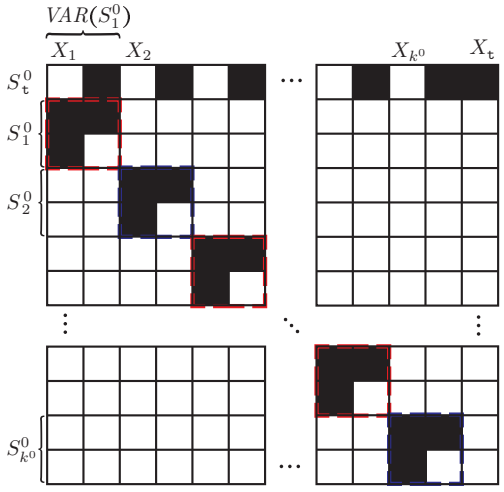


Figure 4: The separated system from the system in Figure 3. Auxiliary variables are used to prevent local constraints from different local parts to instantly "mix" when their variables are eliminated. Each system S_i^0, however, only corresponds to the system encased in coloured boxes.

When we eliminate x_1, y_1, x_2, y_2, x_3 and y_3 from S', we do not "mix" the local subsystems; this only happens when z_1, z_2 and z_3 afterward are eliminated.

Furthermore, when eliminating x_1 and y_1, we only need to do this with respect to the system S_1^0 (and similar for $var(S_2^0)$ and $var(S_3^0)$). The resulting projections from $PRS_{\{x_1,y_1\}}(S_1^0)$, $PRS_{\{x_2,y_2\}}(S_2^0)$ and $PRS_{\{x_3,y_3\}}(S_3^0)$, respectively, can then be added to sum^0 from which z_1, z_2 and z_3 can then be eliminated. △

In the end, the result is that each set S_i^0 now have one more (in)equality for each $c \in S_t$ compared to the original S_i, and the number of variables in $var(S_i^0)$ has increased with the same amount compared to X_i. But, more importantly, now $|var(S_i^0) \cap var(S_t^0)|$ equals $|S_t|$, and eliminating variables in $X_i \cap Y$ only will produce (in)equalities whose set of variables is disjoint from the variable sets of the other local (in)equality systems. Furthermore, eliminating the $z_{c,i}^0$-variables from the separated system will give us a system equivalent to the original system. Hence, eliminating the variables in $Y \subseteq VAR(S)$ from the original system is equivalent to eliminating Y and the defined z-variables, that is

$$PRS_Y(S) = PRS_{Y \cup Z^0}(sep(S)).$$

This follows from Lemma 2 below since by construction $VAR(S) \setminus Y = VAR(sep(S)) \setminus (Y \cup Z^0)$.

Lemma 2.

$$proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0}\big(feas(sep(S))\big).$$

Proof. See Appendix □

4.3.2 Splitting transverse constraints

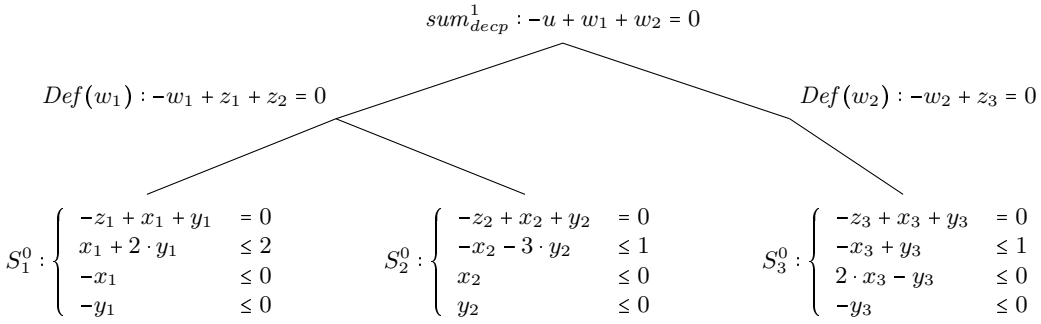
Separating the system as described above helps preventing the (in)equalities becoming too dense when eliminating variables from the local parts. However, we also need to eliminate the defined variables in Z^0, which there might be many of. Using auxiliary variables in a way similar to the one described above, we would therefore like to split the variables in Z^0 into groups and use auxiliary variables to prevent subsystems "mixing" with each other. Potentially, projecting the subsystems will again result in manageable systems that can then be put together.

We will exploit the fact that if the left-hand-side of a transverse (in)equality in S_t^0 consists of a sum of many terms, it can always be rewritten as a sum of fewer terms by use of auxiliary variables. For example

$$\begin{aligned} z_{c,1}^0 + z_{c,2}^0 + \ldots + z_{c,k^0}^0 &= (z_{c,1}^0 + z_{c,2}^0) + \ldots + (z_{c,k^0-1}^0 + z_{c,k^0}^0) \\ &= z_{c,1}^1 + \ldots + z_{c,k^1}^1, \end{aligned}$$

where $z_{c,1}^1 = z_{c,1}^0 + z_{c,2}^0$, $\ldots$, $z_{c,k^1}^1 = z_{c,k^0-1}^0 + z_{c,k^0}^0$. The latter sum can then again be rewritten into a sum of fewer terms, and so on.

Example 4.3. If we take the (separated) (in)equality system $sep(S)$ from Example 4.2, the variable u (which we want to keep) is now defined as a sum of three other variables (z_1, z_2, z_3) instead of the former six variables. Of course, this example is very small and a further splitting of the transverse constraints is most likely not necessary, but for the purpose of illustration, we can further split sum^0_{decp} such that the system obtained consists of the following (in)equalities (drawn in a tree structure)[7]:



To eliminate all variables but u from the system, we first find projections $E^0_1 \in PRS_{\{x_1,y_1\}}(S^0_1)$, $E^0_2 \in PRS_{\{x_2,y_2\}}(S^0_2)$, $E^0_3 \in PRS_{\{x_3,y_3\}}(S^0_3)$. Then we find projections $E^1_1 \in PRS_{\{z_1,z_2\}}(E^0_1 \cup E^0_2 \cup \{Def(w_1)\})$ and $E^1_2 \in PRS_{\{z_3\}}(E^0_3 \cup \{Def(w_2)\})$ and then we find a projection $E \in PRS_{\{w_1,w_2\}}(E^1_1 \cup E^1_2 \cup \{sum^1\})$. $\triangle$

To further divide the transverse (in)equalities, we use an "appropriate" partition $\mathbb{P}^1 = (P^1_1, \ldots, P^1_{k^1})$ of the indices of the local parts, $\{1, \ldots, k^0\}$. What is "appropriate" can vary, depending e.g. on the number of (in)equalities and variables in the subproblems as well as the underlying structure (if any) of the original problem. For each transverse (in)equality $c^0_{decp} : (\sum_{1 \leq i \leq k^0} z^0_{c,i}) + \sum_{x \in X_t} co(x,c) \cdot x \odot_c rhs(c)$ in S^0_t and each part $P^1_i \in \mathbb{P}^1$ we then define a variable to be the sum over the $z^0_{c,j}$-variables, whose index j is in that particular part P_i. That is, we define the following equality $Def(z^1_{c,i})$ over $\{z^1_{c,i}\} \cup \bigcup_{j \in P^i_i} \{z^0_{c,i}\}$.

$$Def(z^1_{c,i}) : -z^1_{c,i} + \sum_{j \in P^1_i} z^0_{c,j} = 0. \tag{16}$$

Using the defined variables we can then substitute in c^0_{decp} to get an (in)equality over $X_t \cup \bigcup_{1 \leq i \leq k^1} \{z^1_{c,i}\}$.

$$c^1_{decp} : (\sum_{1 \leq i \leq k^1} z^1_{c,i}) + \sum_{x \in X_t} co(x,c) \cdot x \odot_c rhs(c).$$

Letting $Z^1 \stackrel{\text{def.}}{=} \cup_{1 \leq i \leq k^1} \{ z^1_{c,i} \mid c \in S_t \}$ we can now split the newly added (in)equalities into k^1 "variable-disjoint" and one "transverse" (in)equality systems by defining $S^1_i \stackrel{\text{def.}}{=} \{ Def(z^1_{c,i}) \mid c \in S^0_t \}$ for each $1 \leq i \leq k^1$ (i.e. $VAR(S^1_i) = \bigcup_{c \in S^0_t} VAR(Def(z^1_{c,i})) \subseteq Z^0 \cup Z^1$) and $S^1_t \stackrel{\text{def.}}{=} \{ c^1_{decp} \mid c \in S^0_t \}$ (i.e $VAR(S^1_t) = \bigcup_{c \in S^0_t} VAR(c^1_{decp}) = X_t \cup Z^1$).

If $var(c^1_{decp})$ is still deemed too large, i.e. k^1 is too large, we can repeat this step - use a partition of the indices of the previous partition to define and use auxiliary variables - until the (in)equality c^K_{decp} only has a few terms, see Figure 5.

The procedure for splitting the transverse (in)equalities like this according to a partition at a given level l can be described by the following procedure SPLITTRANSVERSE in Algorithm 11.

At each step, we define auxiliary variables in terms of the previous level's auxiliary variables, and then we replace each decomposed transverse inequality from the previous level with another decomposed transverse inequality that uses the new auxiliary variables. The defining equalities are split into groups according to a partition of (the indices of) the previous level's variables.

Due to the definition of the new variables, Z^l, and the decomposed (in)equalities at the l'th level, it follows that eliminating the Z^l-variables from the (in)equalities defined thus far gives us an the (in)equality system consisting of all (in)equalities at the previous level; see Lemma 3 below.

[7] As with Example 4.2, for clarity, the variable names do not follow the described naming convention.

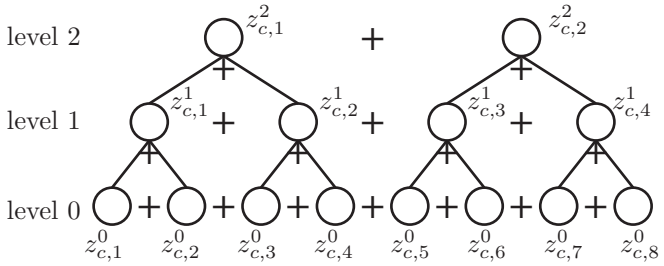


Figure 5: Splitting a transverse constraint $c^0_{decp} : \sum_{i=1}^{8} z^0_{c,i} + \sum_{x \in X_\text{t}} co(x,c) \cdot x \odot_c rhs(c)$ in S^0_t from a separated system using the partitions $\mathbb{P}^1 = \{1,2\} \dot\cup \{3,4\} \dot\cup \{5,6\} \dot\cup \{7,8\}$ and $\mathbb{P}^2 = \{1,2\} \dot\cup \{3,4\}$.

Algorithm 11 Splitting a transverse (in)equality at the l'th level, according to a partition $\mathbb{P}^l$ of the indices of the previous level.

function SPLITTRANSVERSE(Transverse (in)equalities S^0_t, partitions $\mathbb{P}^l = (P^l_1, \dots, P^l_{k^l})$)
  $S^l_\text{t} \leftarrow \varnothing$
  for $i \leftarrow 1$ to k^l **do**
    $S^l_i \leftarrow \varnothing$
    for all $c \in S^0_\text{t}$ **do**
      Add $Def(z^l_{c,i}) : -z^l_{c,i} + \sum_{j \in P^l_i} z^{l-1}_{c,j} = 0$ to S^l_i
    Add $c^l_{decp} : (\sum_{1 \le i \le k^l} z^l_{c,i}) + \sum_{x \in X_\text{t}} co(x,c) \cdot x \odot_c rhs(c)$ to S^l_t
  return $((S^l_1, \dots, S^l_{k^l}), S^l_\text{t})$

Lemma 3.

$$proj_{Z^l}\Big(feas(S^l_\text{t} \cup \bigcup_{1 \le i \le k^l} S^l_i)\Big) = feas(S^{l-1}_\text{t})$$

Proof. See Appendix. □

Solving the (in)equality system exploiting structure

Having separated the system and split the transverse (in)equalities, we can now solve the original problem, that is, eliminate the variables Y from the (in)equality system S. The result will be an (in)equality system, whose feasible region is the projection of Y from S, i.e. the system belongs to $PRS_Y(S)$ and will be expressed in the variables $X \setminus Y$. Though, instead of considering the whole system S and eliminating the variables in Y from this set, we will as previously mentioned take advantage of the decomposition of the problem.

The overall idea of the procedure is to use the constructed decomposition of the (in)equality system to make a tree structure of subsystems where a system S^l_i has the subsystems S^{l-1}_j as a child for all $j \in P^l_i$ (see Figure 6). The systems corresponding to the subtrees in this structure are then projected recursively (see Figure 7 later).

Given that the constraints of a given subproblem *in real life* deals with the same aspect of the problem in question, there is a good chance that the projected subsystem with the aid of some variables significant for that aspect can be expressed in only few constraints. By separating the system and splitting the transverse constraints, which both results in divisions into subproblems, we therefore hope to better take advantage of the structure of the problem.

Firstly, we let $\mathfrak{S}^l_i$ denote the (in)equality system consisting of S^l_i together with all it's descendants in this tree structure, see Figure 6.

Letting $k^{K+1} = 1$, $P^{K+1}_1 = \{1, \dots, k^K\}$, and $S^{K+1}_1 = S^K_\text{t}$, we formally define $\mathfrak{S}^l_i$ for all $0 \le l \le K+1$ and all $1 \le i \le k^l$ as

$$\mathfrak{S}^l_i = \begin{cases} S^l_i \cup \bigcup_{j \in P^l_i} \mathfrak{S}^{l-1}_j & \text{if } l > 0 \\ S^l_i & \text{if } l = 0 \end{cases}$$

Not surprisingly it is easily shown (see Lemma 4 below) that $\mathfrak{S}^{K+1}$ indeed equals the union of all the defined subsystems.

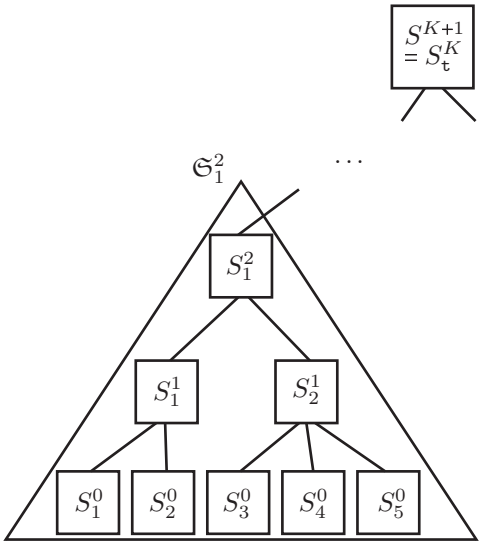


Figure 6: A part of the tree structure for an (in)equality system S. For the partitions it holds that $P^1_1, P^1_2 \in \mathbb{P}^1$ where $P^1_1 = \{1,2\}$ and $P^1_2 = \{3,4,5\}$, and $P^2_1 \in \mathbb{P}^2$ where $P^2_1 = \{1,2\}$.

Lemma 4.

$$S^K_t \cup \bigcup_{1 \le i \le k^0} S^0_i \cup \ldots \cup \bigcup_{1 \le i \le k^K} S^K_i = \mathfrak{S}^{K+1}_1$$

Proof. See Appendix. □

As expected, projecting all Y and Z-variables from the final system corresponds to projecting the Y variables from the original system. From this it follows that

$$PRS_Y(S) = PRS_{Y \cup Z^0 \cup \ldots \cup Z^K}(\mathfrak{S}^{K+1}_1) \tag{17}$$

since $VAR(S) \setminus Y = VAR(\mathfrak{S}^{K+1}_1) \setminus (Y \cup Z^0 \cup \ldots \cup Z^K)$. The following proposition proves the claim more formally.

Proposition 2.

$$proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0 \cup \ldots \cup Z^K}\big(feas(\mathfrak{S}^{K+1}_1)\big)$$

Proof. See Appendix. □

To obtain a system from $PRS_Y(S)$, we can therefore instead eliminate all Y and Z variables from the final system. Although this seems more troublesome, we can use the constructed tree structure to project the (in)equality system, such that a subsystem S^l_i is projected by first projecting each of its subtree's systems ($\mathfrak{S}^{l-1}_j$ for $j \in P^l_i$) recursively, add these projections to S^l_i and then finally project the result w.r.t. $(Y \cup Z^{l-1}) \cap S^l_i$. See Figure 7.

For each $0 \le l \le K$ and $1 \le i \le k^l$, we let $\mathfrak{E}^l_i$ be one of the systems in the (recursive) projection of S^l_i and its subsystems (see Figure 7).
More formally we define

$$\mathfrak{E}^l_i = \begin{cases} pick(PRS_{Z^{l-1}}(S^l_i \cup \bigcup_{j \in P^l_i} \mathfrak{E}^{l-1}_j)) & \text{if } l > 0 \\ pick(PRS_Y(S^l_i)) & \text{if } l = 0 \end{cases}, \tag{18}$$

where $pick : 2^{2^{\mathcal{IE}}} \to 2^{\mathcal{IE}}$ is an arbitrary choice-function, i.e. given a set of (in)equality systems M, it returns one of the (in)equality systems in M. As it turns out (see Lemma 5 later), the specific choice function is irrelevant for our purpose, so for now we just assume this function given.

The proposition below now gives us that projecting the Y- and Z-variables recursively as given in (18) results in the same as projecting all the variables from the final system.

Proposition 3.

$$proj_{Z^K \cup \ldots \cup Z^0 \cup Y}\big(feas(\mathfrak{S}^{K+1}_1)\big) = feas(\mathfrak{E}^{K+1}_1)$$

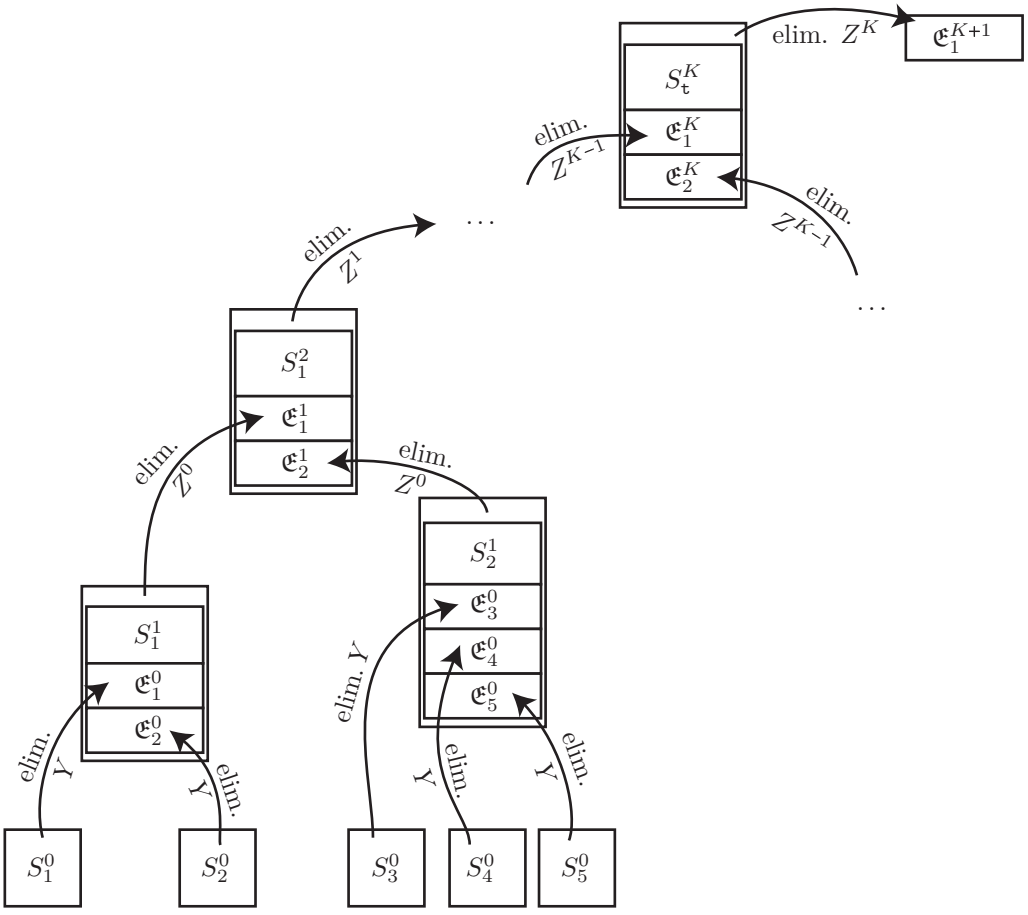


Figure 7: Using the tree structure from Figure 6 for an (in)equality system S to recursively project the variables $Y \subseteq VAR(S)$.

Proof. See Appendix. □

From Proposition 2 and Proposition 3 it follows that $feas(\mathfrak{E}_1^{K+1}) = proj_Y\big(feas(S)\big)$. It then follows from Lemma 5 below that the given choice function is irrelevant.

Corollary 1. *No matter in which way the system $\mathfrak{E}_i^l$ is chosen from $PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j\in P_i^l} \mathfrak{E}_j^{l-1})$ (for $l > 0$) or from $PRS_Y(S_i^l)$ (for $l = 0$) we have that*

$$\mathfrak{E}_1^{K+1} \in PRS_Y(S)$$

Lemma 5. *Let $p, p' : 2^{2^{\mathcal{IE}}} \to 2^{\mathcal{IE}}$ be such that $p(M), p'(M) \in M$ for all $M \in 2^{2^{\mathcal{IE}}}$. Then $\mathfrak{E}_i^l(p) \cong \mathfrak{E}_i^l(p')$ for all $0 \le l \le K+1$ and all $1 \le i \le k^l$, where*

$$\mathfrak{E}_i^l(p) \overset{def.}{=} \begin{cases} p\big(PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j\in P_i^l} \mathfrak{E}_j^{l-1}(p))\big) & \text{if } l > 0 \\ p\big(PRS_Y(S_i^l)\big) & \text{if } l = 0 \end{cases}, \text{ and}$$

$$\mathfrak{E}_i^l(p') \overset{def.}{=} \begin{cases} p'\big(PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j\in P_i^l} \mathfrak{E}_j^{l-1}(p'))\big) & \text{if } l > 0 \\ p'\big(PRS_Y(S_i^l)\big) & \text{if } l = 0 \end{cases}.$$

Proof. See Appendix. □

4.4 Projection framework using decomposition

Below in Algorithm 12 we present an algorithm for finding a projection of S w.r.t. Y exploiting the decomposition of a block structured (in)equality system. It producing the system $\mathfrak{E}_i^l$ according to the definition in (18) for some choice-function $pick$. The algorithm uses a subprocedure that returns a projection of the (in)equality system S' w.r.t. the set of variables $Y' \subseteq VAR(S')$. Our framework uses PROJECT from Algorithm 9 for this, but is not restricted to use this method; SOLVE can be combined with any other method that calculates a projection in $PRS_{Y'}(S')$.

Algorithm 12 Projecting the variables Y from an (in)equality system S by decomposing it. $\mathbb{X} = (X_1, \dots, X_{k^0})$ is a list of disjoint subsets of $VAR(S)$, and $\mathfrak{P} = (\mathbb{P}^1, \dots, \mathbb{P}^K)$ is a list of partitions. Each $\mathbb{P}^l$ is a partition of $\{1, \dots, k^{l-1}\}$, where $k^i = |\mathbb{P}^i|$ for $i > 0$ and $k^0 = |\mathbb{X}|$.

```
1:  function SOLVE(System S, variables Y ⊆ VAR(S), variable sets 𝕏, partitions 𝔓)
2:      (𝕊^0, S_t^0) ← SEPARATEINEQS(S, 𝕏)                    ▷ See Algorithm 10
3:      for l ← 1 to K do
4:          (𝕊^l, S_t^l) ← SPLITTRANSVERSE(S_t^0, ℙ^l)          ▷ See Algorithm 11
5:      𝕊^{K+1} = (S_t^K)
6:      return SOLVE-SUB(S_1^{K+1}, (𝕊^0, ..., 𝕊^{K+1}), 𝔓, Y, S_t^0)

7:  function SOLVE-SUB(S_i^l, (𝕊^0, ..., 𝕊^{K+1}), 𝔓, Y, S_t^0)
8:      if i = 0 then
9:          return PROJECT(S_i^l, Y)                           ▷ AlgorithmÂ¨9
10:     else
11:         S' ← S_i^l
12:         for all j ∈ P_i^l do
13:             Add SOLVE-SUB(S_j^{l-1}, (𝕊^0, ..., 𝕊^{K+1}), 𝔓, Y, S_t^0) to S'
14:         return PROJECT(S', Z^{l-1})                        ▷ AlgorithmÂ¨9
```

It should also be noted, that having the decomposition into smaller subsystems as above, it is also possible to parallelize the projection of the subproblems (either instead of or on top of the parallelization of the redundancy check). In this case, we would maintain a queue of systems, that is ready to be projected (and have not been so far), which is initialized with all leafs in the constructed tree-structure. When a system is projected, a counter for its parent is increased, and if the counter reaches the number of the node's children, it (i.e. the system with its projected child-systems) is put in the queue. Then the systems in the queue are solved independently by multiple solvers in parallel.

4.5 Nested structure

The method described in Algorithm 12 first makes the separated system and thereafter splits the transverse (in)equalities. It is however possible to also use these steps in a nested fashion; on a top-level, an (in)equality system might be divided into a transverse part and a number of local parts that in themselves can be divided into further local parts and a transverse part, and so on. See Figure 8. In such a case, the presented algorithms are of course modified accordingly.

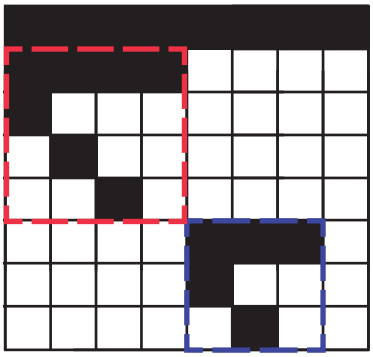


Figure 8: The inequality system in the figure can be divided into a transverse part and two local parts, namely the system enclosed within the red square and the system enclosed within the blue square. Each of the two subsystems can in themselves be solved using a decomposition into a transverse part and local parts.

We notice that it is up to the solver of the problem to identify the various local parts to best make use of the structure of the given problem. However, it might be possible, also at a syntactical level, to identify potential useful partitions of the (in)equalities. This identification process is, though, left to future research.

5 Stowage capacity case study

5.1 Stowage model

In this section we present the stowage model that is the point of origin for our projected capacity model. In other words, this model describes the (in)equality system which we want to project. The model is adapted from the PhD thesis of Delgado [Del13]; other models for stowage planning using actual vessel profiles can be found in e.g. [Pac12], [PDJB11] and [PDOJB12]. We will not go into details of the model in this report, but will only present the essentials; the reader is referred to [Del13] for further details.

Description

As shown in Figure 9a, the cargo space of a container vessel is divided into sections called *bays* that each are divided into a grid by stacks and tiers (Figure 9b). Each stack and tier constitute a *cell* that consists of two *slots*.

The containers transported on a vessel usually have a standard size that fits the cells and slots; the containers are normally (ISO standard) $8'$ wide, $8'6''$ high and $20'$, $40'$ or $45'$ long, though we do not consider the latter in this report. Accordingly, a cell can hold either two $20'$ containers (one in each slot) or one $40'$ container (Figure 9c). Further, some containers are refrigerated (*reefers*), and must be stowed at a power plug. The corners of the containers are constructed such that containers can be stacked, though $20'$ containers cannot be stacked on top of $40'$ containers due to the lack of corner supports in the middle of $40'$ containers (see Figure 9c).

The capacity of a container vessel is measured in *TEU* (Twenty-foot Equivalent Units), i.e. a standard $20'$ container as described above takes up one TEU, while a $40'$ container takes up two TEU. Each stack has a height and weight limit. Separate total weight limits for $20'$ and $40'$ containers exist, since only $20'$ containers rest on the middle support sockets of the stack, while the end sockets hold weight of both $20'$ and $40'$ containers (see Figure 9c).

Though containers physically are placed in specific slots, the stowage models considered here are themselves abstractions called master plans that specify how many containers of each *type* should be stowed in each subsection of each bay, referred to as a *location*. These locations emerge by dividing the bays vertically into sections on deck

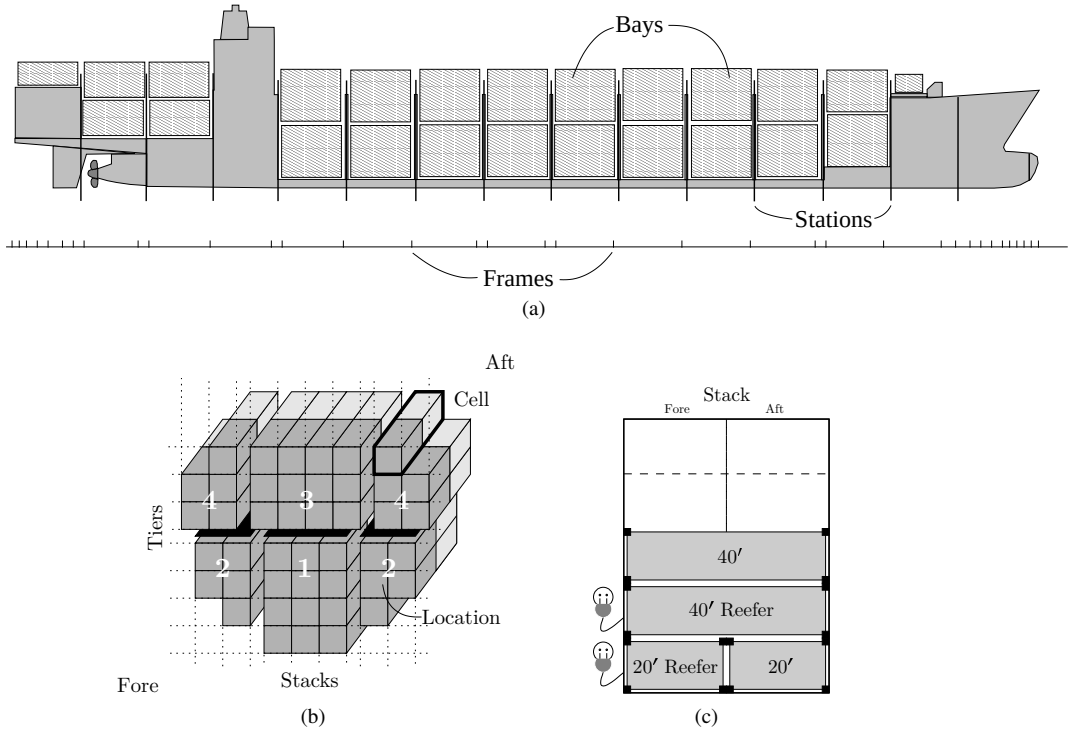


Figure 9: (a) The arrangement of bays in a small container vessel. (b) A bay divided into cells given by stacks and tiers. The cells are grouped into four locations. (c) A side view of a stack of containers with power plugs for reefer containers at bottom slots. Adapted from [Del13].

and below deck, respectively, and horizontally into a center section and symmetric side-sections, respectively; see Figure 9b which shows a bay divided into four location indicated by their number. The type of containers is here specified by the length, reefer-property and a weight class. The capacities for stacks translate to capacities for locations.

Besides the location-based capacity constraints, global hydrostatic constraints ensure the stability of the vessel. Stress forces arise as a result of gravitation acting downwards on the vessel and buoyancy acting upwards. The gravity forces are divided into three components: the light ship (i.e, the vessel mass without cargo and ballast water), cargo, and ballast water. Buoyancy forces are due to the vessel's displacement of water and hence depends on the varying (and irregular) shape of the hull, which is given at a set of reference points called *stations* (see Figure 9a). At these stations, the area submerged in water can be approximated by linear functions of the vessel's longitudinal center of gravity (*lcg*), and from this an approximation of the buoyancy force of the vessel between two consecutive stations can be calculated. [8]

There are three major stress forces: shear force, bending moment, and torsion moment. Limits on these stress forces are given for a set of points along the vessel called *frames* (see Figure 9a). In this work we only consider the limits on shear forces.

Sets, variables, constants

To formally describe the stacking and stability constraints of the considered stowage models, we use the sets, variables and constants summarized in Table 1, Table 2 and Table 3, respectively. Regarding the variables, we note that even though $x_{l,\tau}$ is a number of containers and hence a natural number, we model it as a reel number to ensure that the resulting model is an LP. Due to the large number of containers that can be stowed in each location, this approximation is sufficiently accurate in practice.

[8] A more accurate buoyancy model also takes the displacement of the vessel into account.

Sets

L	Set of locations.
F	Set of frames.
$F^{\mathtt{f}}, F^{\mathtt{a}} \subseteq F$	Sets of frames *fore* and *aft* where shear force limits must hold.
$S, S' = \{s_1, \ldots, s_S\}$	Number of stations and (ordered) set of stations.
T	Set of types of containers.
$T^{20}, T^{40}, T^{\mathtt{R}}, T^{20\mathtt{R}}, T^{40\mathtt{R}} \subseteq T$	Set of types that are $20'$ long, $40'$ long, reefers, $20'$ reefers, and $40'$ reefers, respectively.
BT	Set of ballast tanks.

Table 1: Sets used in the considered stowage models.

Decision variables

$x_{l,\tau} \in \mathbb{R}_0^+$	Number of containers of each type $\tau \in T$ to be stowed at each location $l \in L$.
$x_b \in \mathbb{R}_0^+$	The weight of each ballast tank $b \in BT$.

Auxilliary variables

$w_l \in \mathbb{R}_0^+$	The weight of all containers stowed at each location $l \in L$.
$x_\tau \in \mathbb{R}_0^+$	The total number of containers of each type $\tau \in T$.
$sf_f \in \mathbb{R}$	The shear force at each frame $f \in F$.
$lcg \in \mathbb{R}$	The longitudinal center of gravity.
$bf_s \in \mathbb{R}$	The buoyancy force at each section between station $s \in S' \setminus \{s_S\}$ and the next station.
$sf_f^{\mathtt{c},e}$, $sf_f^{\mathtt{bt},e}$, $sf_f^{\mathtt{bc},e} \in \mathbb{R}$	The contribution to the shear force for/aft ($e \in \{\mathtt{f}, \mathtt{a}\}$) each frame $f \in F$ from containers ($\mathtt{c}$)/ballast tanks ($\mathtt{bt}$)/buoyancy ($\mathtt{bc}$)

Table 2: Variables used in the considered stowage models.

Constants

Cap_l^{20}, Cap_l^{40}, $Cap_l^{\mathtt{TEU}}$, $Cap_l^{\mathtt{RS}}$, $Cap_l^{\mathtt{RC}}$, $Cap_l^{\mathtt{W20}}$, $Cap_l^{\mathtt{W40}}$	The capacity for each location $l \in L$, w.r.t. $20'$ containers, $40'$ container, TEU, reefer slots, reefer cells, weight of $20'$ containers, and weight of $40'$ containers, respectively.
W_τ	The weight of a type of container $\tau \in T$.
$B_s^{\mathtt{lcg}}$, $B_s^{\mathtt{C}}$	Coefficients for the linearization of the submerged area at each station $s \in S'$.
D_s	The distance between $s \in S' \setminus \{s_S\}$ and the next station.
$W_f^{\mathtt{f}}, W_f^{\mathtt{a}}$	The constant weight of the vessel fore/aft each frame $f \in F$ (light ship)
$P_{f,l}^e, P_{f,b}^e, P_{f,s}^e \in [0,1]$	The fraction of each location $l \in L$/ballast tank $b \in BT$/section between $s \in S' \setminus \{s_S\}$ and the next station that lies fore/aft ($e \in \{\mathtt{f}, \mathtt{a}\}$) each frame $f \in F$.
$UB(lcg)$, $lb(lcg)$, $UB(sf_f)$, $lb(sf_f)$, $UB(x_b)$, $UB(\mathtt{WBT})$, $lb(\mathtt{WBT})$	The upper and lower bounds for the lcg, shear force at each frame $f \in F$, the weight for each ballast tank $b \in BT$ (upper bound only), plus the weight of ballast tanks in total, respectively.

Table 3: Constants used in the considered stowage models

Constraints

Location-based capacity constraints Firstly, we have constraints that ensure that for each location of the vessel, the stowed containers are within the allowed capacities w.r.t. the number of $20'$ containers, $40'$ containers, TEUs, and the weight of the $20'$ and $40'$ containers, respectively. These constraint are modeled in the inequalities (19)-(23) below. Likewise, we have a constraint, (24), that ensures that the weight of a location is within limits, taken the different distribution of the weight of $40'$ containers, respectively $20'$ containers, within a slot into consideration; the weight of a $40'$ container is distributed on the four outer corner places of a slot while the weight of $20'$ containers also rest on the inner corners of the slot. Lastly, (25) and (26) ensure that each reefer container can be refrigerated, and that the total number of cells taken up by the reefer containers are within capacity, respectively.

$$\forall l \in L: \quad \sum_{\tau \in T^{20}} x_{l,\tau} \leq Cap_l^{20} \tag{19}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{40}} 2 \cdot x_{l,\tau} \leq Cap_l^{40} \tag{20}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{20}} x_{l,\tau} + 2 \cdot \sum_{\tau \in T^{40}} x_{l,\tau} \leq Cap_l^{\mathrm{TEU}} \tag{21}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{20}} W_\tau \cdot x_{l,\tau} \leq Cap_l^{\mathrm{W20}} \tag{22}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{40}} W_\tau \cdot x_{l,\tau} \leq Cap_l^{\mathrm{W40}} \tag{23}$$

$$\forall l \in L: \quad 0.5 \cdot \sum_{\tau \in T^{20}} W_\tau \cdot x_{l,\tau} + \sum_{\tau \in T^{40}} W_\tau \cdot x_{l,\tau} \leq Cap_l^{\mathrm{W40}} \tag{24}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{\mathrm{R}}} x_{l,\tau} \leq Cap_l^{\mathrm{RS}} \tag{25}$$

$$\forall l \in L: \quad \sum_{\tau \in T^{20\mathrm{R}}} 0.5 \cdot x_{l,\tau} + \sum_{\tau \in T^{40\mathrm{R}}} x_{l,\tau} \leq Cap_l^{\mathrm{RC}} \tag{26}$$

Defined variables and bounds The constraint in (27) defines the variables, x_τ, that specifies how many containers of a specific type τ is stowed on the vessel. These are the variables that we want the projected (in)equality system to be expressed in.

$$\forall \tau \in T: x_\tau = \sum_{l \in L} x_{l,\tau} \tag{27}$$

(28) below defines the total weight of the containers stowed in each location, since this number is used in further calculations of the hydrostatic constraints (see further below).

$$\forall l \in L: \quad w_l = \sum_{\tau \in T} W_\tau \cdot x_{l,\tau} \tag{28}$$

We also ensure that the number of containers placed at each location of each type as well as the amount of ballast in the ballast tanks is positive. For the ballast tanks, we also require that each tank's content is within limits, as well as the total content. These constraints are given in (30) and (31) below.

$$\forall l \in L, \tau \in T: \quad 0 \leq x_{l,\tau}. \tag{29}$$

$$\forall b \in BT: \quad 0 \leq x_b \leq UB(x_b) \tag{30}$$

$$lb(\mathtt{WBT}) \leq \sum_{b \in BT} x_b \leq UB(\mathtt{WBT}) \tag{31}$$

Hydrostatic constraints At each station, the submerged area of the cross-section has been linearized as a function of the lcg, and the buoyancy force for each section between consecutive stations are calculated as the average of the two areas times the distance between the two stations (32). Instead of calculating the vessel's lcg, we only require it to be within a given interval (33).

The shear forces must be within limits for a set of fore (F^{f}) and aft (F^{a}) frames. The shear force for a fore frame is the sum of resulting forces acting from the frame towards the stern, while the shear force for an aft frame is the sum of resulting forces acting from the frame towards the bow of the vessel, see (34). For each of the frames, we then require these shear forces to be within limits (35).

$$\forall s_i \in S' \setminus \{s_S\} : \; bf_{s_i} = \frac{1}{2} \cdot D_{s_i} \cdot (B^{\mathtt{C}}_{s_i} + lcg \cdot B^{\mathtt{lcg}}_{s_i} + B^{\mathtt{C}}_{s_{i+1}} + lcg \cdot B^{\mathtt{lcg}}_{s_{i+1}}) \tag{32}$$

$$lb(lcg) \leq lcg \leq UB(lcg) \tag{33}$$

$$\forall e \in \{\mathtt{f}, \mathtt{a}\}, f \in F^e : \; sf_f = W^e_f + \sum_{l \in L} w_l \cdot P^e_{f,l} + \sum_{b \in BT} x_b \cdot P^e_{f,b} + \sum_{s_i \in S' \setminus \{s_S\}} bf_{s_i} \cdot P^e_{f,s_i} \tag{34}$$

$$\forall f \in F : \; lb(sf_f) \leq sf_f \leq UB(sf_f) \tag{35}$$

5.2 Experimental results

To do a nested decomposition (see section 4.5), we will, though, divide the constraints in (34) in a part coming from the containers, ballast tanks and bouyancy, respectively, using auxiliary variables. Thus (34) is replaced with (36)-(39) below.

$$\forall e \in \{\mathtt{f}, \mathtt{a}\}, f \in F^e : sf^{\mathtt{c},e}_f = \sum_{l \in L} w_l \cdot P^e_{f,l} \tag{36}$$

$$\forall e \in \{\mathtt{f}, \mathtt{a}\}, f \in F^e : sf^{\mathtt{bt},e}_f = \sum_{b \in BT} x_b \cdot P^e_{f,b} \tag{37}$$

$$\forall e \in \{\mathtt{f}, \mathtt{a}\}, f \in F^e : sf^{\mathtt{bc},e}_f = \sum_{s_i \in S' \setminus \{s_S\}} bf_{s_i} \cdot P^e_{f,s_i} \tag{38}$$

$$\forall e \in \{\mathtt{f}, \mathtt{a}\}, f \in F^e : sf_f = W^e_f + sf^{\mathtt{c},e}_f + sf^{\mathtt{bt},e}_f + sf^{\mathtt{bc},e}_f \tag{39}$$

We have tested our methods on three different models, each including a different number of the constraints presented in the previous section and above. In all cases, we wanted a model capturing all the dependencies between the variables x_τ, and only those. Hence we project the variables $VAR(S) \setminus \{ x_\tau \mid \tau \in T \}$ where S is the system consisting of the (in)equalities corresponding to the constraints of the model.

We have used data from our industrial partner, Maersk, from a vessel with a capacity of 15.500 TEU, 91 locations, 27 ballast tanks and 33 stations. There are 25 frame points, however, we have only used one fore and one aft (in the two models considering hydrostatics). In all models, we considered 12 types, $T = \{20', 40'\} \times \{6t, 21t, 27t\} \times \{\mathtt{R}, \mathtt{NR}\}$, that are defined according to the length, the weight class and the reefer property of the containers.

In the two tables below we show which (in)equalities are present in the 3 (in)equality systems S_1, S_2 and S_3 that have been tested, and we state the size of each model given in number of (in)equalities, number of variables and the two numbers multiplied. Similarly we present the size of the projected (in)equality systems together with the approximate time taken to obtain those projections.

The (in)equality system S_1 includes 3 capacity constraints per location and shear force calculations at 2 frame points; S_2 includes all capacity constraints for each location but no calculations of shear forces. S_3 includes 6 capacity constraints per location and shear force calculations at the two frames.

(In)eq. system	Constraints included
S_1	(21), (24), (26), (27), (28), (29), (30), (31), (32), (33), (35), (36), (37), (38), (39)
S_2	(19), (20), (21), (22), (23), (24), (25), (26), (27), (28), (29)
S_3	(19), (20), (21), (24), (25), (26), (27), (28), (29), (30), (31), (32), (33), (35), (36), (37), (38), (39)

Table 4: Constraints included in test systems.

The tests have been done on a laptop with an Intel® Core™ i7-4600U-processor with a frequency of 2.10-3.3 GHz, 8GB RAM, and with 2 cores and 4 threads. The computer has also been used to perform other tasks while doing the projection, and 3 threads were used to run the parallel redundancy checks. The time used to make the projection is therefore not accurate, but still gives an indication of the running time (especially when these are compared across the test results).

(In)equality system	App. time	Size of system			Size of projected system		
		#(in)eqs	#vars	#ineqs · #vars	#(in)eqs	#vars	#ineqs · #vars
S_1	6 m	537	1092	586,404	26	12	312
S_2	18.5 h	922	1209	1,114,698	42	12	504
S_3	97 h	811	1171	949,681	106	12	1272

Table 5: Sizes of original and projected systems.

In all three test cases, the system consisting of the (in)equalities using $x_{l,\tau}$ and w_l were set aside; this is the system corresponding to the constraints (19)-(29) (for S_2) plus (36) (for S_1 and S_3). Then these systems were decomposed according to the locations, and the variables $x_{l,\tau}$ and w_l were eliminated, leaving a system over x_τ (for S_1), or x_τ and $sf_f^{c,e}$ (for S_1 and S_3). The decomposition was done such that $X_i = \{\, x_{l,\tau} \mid \tau \in T, l \in P_i^0 \,\} \cup \{\, w_l \mid l \in P_i^0 \,\}$ for each $1 \le i \le 45$, where

$$\begin{aligned}(P_1^0, P_2^0, \ldots, P_{45}^0) = \big(&\{0,1,2\},\{3,4\},\{5,6\},\{7,8\},\{9,10\},\{11,12\},\{13,14\},\{15,16\},\{17,18\},\\ &\{19,20\},\{21,22\},\{23,24\},\{25,26\},\{27,28\},\{29,30\},\{31,32\},\{33,34\},\\ &\{35,36\},\{37,38\},\{39,40\},\{41,42\},\{43,44\},\{45,46\},\{47,48\},\{49,50\},\\ &\{51,52\},\{53,54\},\{55,56\},\{57,58\},\{59,60\},\{61,62\},\{63,64\},\{65,66\},\\ &\{67,68\},\{69,70\},\{71,72\},\{73,74\},\{75,76\},\{77,78\},\{79,80\},\{81,82\},\\ &\{83,84\},\{85,86\},\{87,88\},\{89,90\},\{91\}\big).\end{aligned}$$

For S_1 and S_3, the list of partitions $\mathfrak{P}$ was $\mathfrak{P} = (\mathbb{P}^1, \mathbb{P}^2, \mathbb{P}^3, \mathbb{P}^4, \mathbb{P}^5)$, where

$$\begin{aligned}\mathbb{P}^1 = \big(&\{0,1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\},\\ &\{18,19\},\{20,21\},\{22,23\},\{24,25\},\{26,27\},\{28,29\},\{30,31\},\{32,33\},\\ &\{34,35\},\{36,37\},\{38,39\},\{40,41\},\{42,43\},\{44,45\}\big),\\ \mathbb{P}^2 = \big(&\{0,1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\},\\ &\{18,19\},\{20,21\},\{22\}\big),\\ \mathbb{P}^3 = \big(&\{0,1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{10,11\}\big),\\ \mathbb{P}^4 = \big(&\{0,1\},\{2,3\},\{4,5\}\big),\\ \mathbb{P}^5 = \big(&\{0\},\{1,2\}\big).\end{aligned}$$

For S_2, the partitions where $\mathfrak{P}' = (\mathbb{P}'^1, \mathbb{P}'^2, \mathbb{P}'^3, \mathbb{P}'^4, \mathbb{P}'^5)$, where

$$\begin{aligned}\mathbb{P}'^1 &= \mathbb{P}'^1,\\ \mathbb{P}'^2 &= \big\{\{0,1\},\{2,3\},\{4,5\},\{6,16\},\{8,9\},\{10,18\},\{12,13\},\{14,15\},\{7,17\},\{11,19\},\\ &\quad \{20,21\},\{22\}\big\},\\ \mathbb{P}'^3 &= \big\{\{0,2\},\{1,3\},\{4,6\},\{5,7\},\{8,10\},\{9,11\}\big\},\\ \mathbb{P}'^4 &= \big\{\{0,1\},\{2,4\},\{3,5\}\big\},\\ \mathbb{P}'^5 &= \mathbb{P}^5.\end{aligned}$$

The reason for having $\mathfrak{P}'$ like this (and not just "numerical") was to "pair" partitions according to the size of the produced (sub)projections such that some of the smaller projections were paired with some of the larger projection. The reason that some of the (sub)projections were smaller is partly that some of the locations have no reefer capacity, so $x_{l,\tau}$ for such a location and a τ that is a reefer type is quickly set to 0, which makes the projection smaller.

For S_2, making the above projection resulted in the required system expressed only in the x_τ-variables.

For both S_1 and S_3 we projected yet two more subsystems; the subsystem consisting of the (in)equalities relating to buoyancy ((32), (33) and (38)) were projected w.r.t. lcg and bf_s (leaving only the $sf_f^{\mathtt{bc},\mathtt{f}}$ and $sf_f^{\mathtt{bc},\mathtt{a}}$ variables), while the subsystem relating to ballast tanks ((30), (31) and (37)) were projected w.r.t. x_b (leaving the $sf_f^{\mathtt{bt},\mathtt{f}}$ and $sf_f^{\mathtt{bt},\mathtt{a}}$ variables). For both S_1 and S_3, the three projected subsystems were then joined together

with the (in)equalities (35) and (39), relating to the shear force. In this final system, we then eliminated the $sf_f^{\mathtt{c},e}$, $sf_f^{\mathtt{bt},e}$, $sf_f^{\mathtt{bc},e}$, and the sf_f-variables, which resulted in a system over the x_τ-variables as required.

To give the reader an idea of the how the number of (in)equalities and variables, respectively progress, Figure 10, Figure 11 and Figure 12 show the evolution of these numbers when projecting the mentioned final system stemming from S_3, which is the system producing the largest intermediary systems. Each "step" in these graphs corresponds to either the preprocessing or "clean-up" (as a whole), Gauss-elimination of one variable, FM-elimination of one variable, or a complete redundancy removal of the system. The first part where the number of (in)equalities is reasonable stable (when seen in this scale) corresponds to the preprocessing and Gauss-elimination part. This is followed by repeated steps of FM-elimination (the "peaks" in the number of (in)equalities), clean-up (the "flat" part after the peaks) and redundancy removal (the "valleys").

We note that S_3 is the most time-consuming system to project of the three, and the final system as shown in Figure 12 is by far the system (of all the subsystems that were projected) that produced the biggest intermediary systems.

5.3 Implementation notes

In the following we will give a few remarks regarding the specifics of the implementation.

The program was implemented in Java, and we use rationals for the coefficients and right-hand-sides. In the implementation, we have used a list, not a set, for representing the (in)equalities in a system. Thus, at a given point, the (in)equality system S might contain two (in)equalities c and c' for which $\mathbf{co}(c) = \mathbf{co}(c')$ and $rhs(c) = rhs(c')$. However, either c or c' will be removed in the preprocessing/clean up step (when removing linearly dependent (in)equalities), and this does not influence the correctness of the algorithm(s). We also normalize the (in)equalities in the inequality system such that the smallest absolute value of the coefficients is 1. This also makes it easier to detect linear dependencies.

ϵ in Section 4.1.1 is set to 0.01, $\epsilon' = 0.000001$, and $\mathcal{K} = 10^{14}$. The optimization software CPLEX from IBM was used as lp-solver.

We note that because we use the almost redundant criteria when doing (sequential) redundancy, the system S' returned by REMOVEREDUNDANCY in line 14 of Algorithm 9 is no longer completely identical to $proj_x(S)$; S' will most likely be an overapproximation of the projection. However, as previously mentioned, the parameters of a problem – and this indeed holds for the particular problem we have considered – are not necessarily very accurate, and small discrepancies between the returned projection and the actual projection is acceptable. It still remains to be shown that the overapproximation is not "too big" according to a given criterion.

Finally, we want to point out here, that the implemented program is non-deterministic due to a number of factors.

Firstly, the almost redundant-criteria for removing inequalities in the sequential redundancy removal means that the order in which the inequalities are checked, matters. For two runs (on the same input system) to result in the exact same projection, the variables must be projected in the same order, and at each variable elimination, the produced inequalities must be added in the same order (or checked for redundancy in the same order), and the redundancy check using CPLEX must result in the same yes/no answer (and κ-value).

Making use of parallel redundancy check means that when an inequality from a system is checked for redundancy in two different runs, another inequality might have been deemed redundant by another checker and hence have been removed in one run, while it is not in the other run (because the checker finished later). This can influence how long the check of the current inequality takes and influence the κ-value, so this effect can propagate. In the end, this might cause the set of removed (in)equalities to vary in the two runs altogether (because some inequalities in reality are redundant but cannot be determined to be so).

Likewise, we make use of an in-build mechanism in Java for parallelism, namely streams. These are used for example when using the heuristic for finding out which variable should be removed. Again, this can of course be avoided.

In conclusion, with careful ordering of variables e.t.c., settings of parameters for CPLEX and avoidance of parallelism, non-determinism could possibly be avoided, but will most likely also be less efficient.

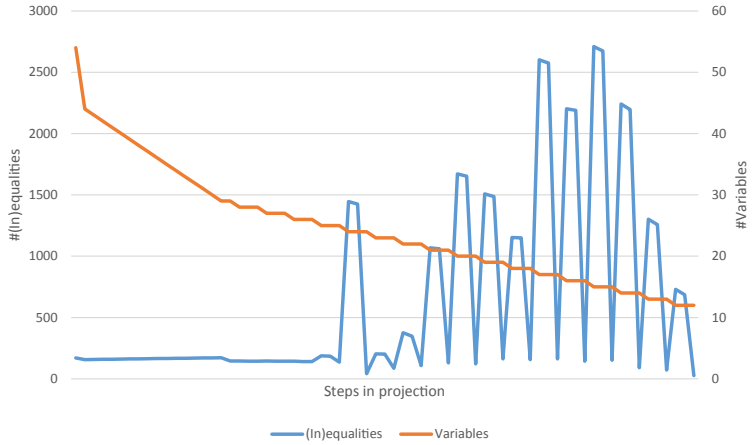


Figure 10: Projection of the final system stemming from S_1.

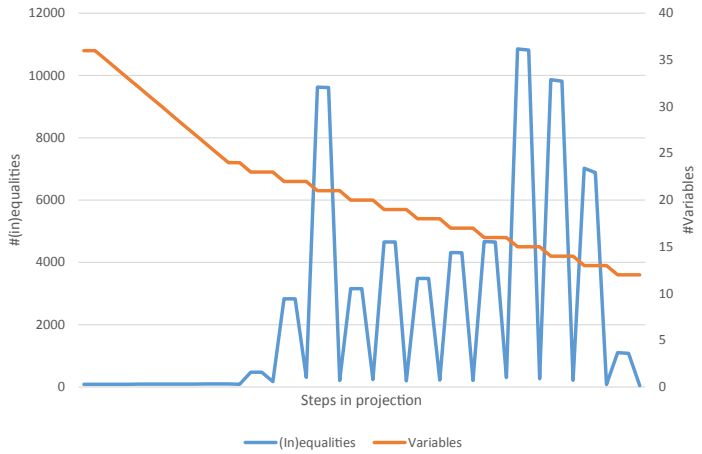


Figure 11: Projection of the final system stemming from S_2.

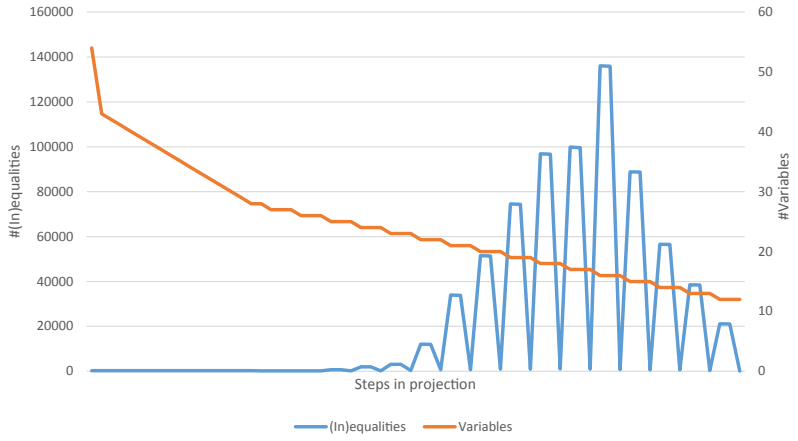


Figure 12: Projection of the final system stemming from S_3.

6 Related work and Discussion

Original method and additions The presented, basic method for eliminating variables from a system of linear inequalities was first provided by Fourier in 1824 [Fou26], while Motzkin later reintroduced the method in his Ph.D. thesis [Mot36]. Černikov later augmented Fourier's algorithm with computational simple rules, which were later rediscovered by Kohler [Koh67], to both prevent additions of (some) redundant inequalities and remove others after construction [Č63], and used together with these rules, the presented method is also referred to as the Fourier-Černikov method. These criteria-based rules rely on keeping track of an index set for each inequality $c \in S$, which contains (the index of) the inequalities from the original system that during the procedure have been combined to obtain c. However, though the proposed rules improve the method, they are incomplete, i.e. they do not detect or remove all redundancies (see e.g. [Imb93] or [LS08]).

The method has subsequently been thoroughly studied, e.g. by Kohler [Koh67], Duffin [Duf74] and Imbert [Imb90], [Imb93], and several additional rules were proposed in these papers to avoid the addition of redundant inequalities. Improving the run time of the procedure in various ways has also been studied, for example in [HLL92] and in [BZ15], where a faster method is proposed to check the slower of the two Černikov criteria.

However, as with the rules of Černikov and Kohler, the rules suggested by Duffin and Imbert do not identify and remove all redundant inequalities from an inequality system and it is not clear to which extend applying these methods are compatible with removing every redundant inequality by solving a linear programming problem; the Černikoc-rules are sound, but combining them with other sound redundancy removals (even such as removing duplicates) are in general unsound ([JMSY93], [HLL92], [Imb93], [Fou15]). [Imb93] further states that he does not know "a method which suppresses all redundancies, compatible with one of [the] Fourier elimination methods". However, *strictly redundant* inequalities, whose corresponding hyperplane does not intersect the feasible area of the system can be removed without compromising the soundness of the procedure with the additional rules [JMSY92].

Furthermore, these rules for avoiding additions of unnecessary inequalities will not be correct when our "almost redundant"-criteria is used for removing inequalities, and therefore we have chosen not to use the mentioned rules in our implementation. However, it would be possible to use them, without the added, full redundancy check, until the system e.g. reaches a certain size, at which it will then be fully reduced and the "index sets" reset, such that each inequality in this system is considered original.

Redundancy removal Separately, particularly within linear programming, work has also been done in the area of classifying and removing redundant inequalities in a inequality system as well as finding implicit equalities, e.g. [Tel83], [LHM93], [KLTZ83], [AA95], [Mat73] to name a few. Many of these (e.g. [Tel83] and the majority of the methods presented in [KLTZ83]) are to be performed within the simplex procedure (used to optimize an LP) or use the objective function and/or the optimal extreme point for the LP, and are hence not applicable for us. On the other hand, we have used cheap redundancy-identifications e.g. described in [AA95], [BMW75] and [Mar03] in our preprocessing and clean-up method, as well as the removal of linearly dependent inequalities from [LHM93].

Although it would be an advantage to be able to identify implicit equalities (inequalities that must hold as equalities in the feasible area) such that they can be used in Gauss-elimination, a prior implementation suggested that not much was gained when trying to identify them, while the (naïve) implementation was too time consuming. However, it is not impossible, that our implementation could benefit from an implementation of more sophisticated methods for redundancy detection and removal instead of or in addition to (prior to) the reasonable straightforward method applied here.

Methods for projection not based on Fourier-Motzkin-elimination Other methods exist for computing the projection of a feasible area of an (in)equality system, that are not based on the method by Fourier and Motzkin.

For example, in [HLL92], the authors describe a method (based on a method from [Las90]) that they recommend for dense systems, called the extreme point method. The method works by finding the extreme points of the polytope P defined by the convex combinations of constraints in S that eliminate the variables in Y. Since the method finds extreme points and hence inequalities in the projection space incrementally, the method can be used to approximate the projection. The method is consequently used as a supplement to Fourier-Motzkin-elimination, Gauss elimination and full redundancy removal in [SK05], when the system being projected becomes too dense and an approximation is required.

Huynh, Lassez and Lassez [HLL92] further describe a method (based on [LL90]), the convex hull method, in which the projection of an inequality system S is computed by successive refinements of an initial approximation of the projection. According to the authors, the complexity of their algorithm "depends essentially on the dimension of the projection of the output not the size of the input" [HLL92], and could therefore be an interesting alternative to Fourier-Motzkin-elimination in a case like the one we consider.

Another example is the method introduced in [JKM04], called equality set projection, which computes all facets of the projection by first finding a random facet and then iteratively computing all adjacent facets (without revisiting them) using a face-lattice. This method is recommended by the authors for polytopes with a low facet count and a high vertex count.

Yet another approach to projection is presented in the work by [JKM08] and [Fou15]. This work is based on parametric linear programming and the possibility to formulate a projection problem as a parametric programming problem. Hence techniques for finding the solution to the latter (e.g. a "parametric" version of simplex) can be used to find the projection.

It is an interesting direction for future research to see, if these methods are better than the currently used method for solving the problem described in this report, or whether they can be combined advantageously with our method. The described decomposition from Section 4.3 could for example still be used while each of the subproblems could be projected using any method for projection.

Fourier-Motzkin-based frameworks for projection In [SK05] Simon and King combine Fourier-Motzkin-elimination, Gauss-elimination, removal of linearly dependent inequalities and complete redundancy removal in a similar fashion as we have done, to project sparse systems, and they use the extreme point method of Huynh *et al.* [HLL92] to make approximations of the projection when this is necessary. Their method is implemented as part of an argument-size analyzer for logic programs and tested on a variety of these. Their elimination procedure is therefore not applied once, but instead multiple times during an analysis, and their method and results are therefore hard to compare to ours. The (in)equality systems operated on are rather sparse and quite small, while their objective is to do the analysis more efficiently (faster) than other methods (when using the polyhedral abstract domain for program analysis).

Lukatskii and Shapot [LS08], [SL12] describe and implement a projection method using Fourier-Motzkin-elimination augmented with Černikov's rules. They further use a techniques for full redundancy removal examining the solution matrix for a basic solution. Further they present and apply a method for "additional matrix clean up" where some almost redundant inequalities are removed. Their method for this is a little more elaborate than ours and involves a successive increase of the allowable deviation (corresponding to our ϵ) and a permissible maximal ratio between the number of inequalities in the current system compared to the original system. They perform tests on a prototype implementation, where the sized of the test ranges between 81 inequalities and 40 variables to 201 inequalities and 100 variables, i.e. the systems they project are much smaller than the ones we have considered; granted, their run time is of course also smaller than ours.

It is shown in [SL12] that if the polyhedron is solid, then the projection algorithm is stable, while for a singular polyhedron, small perturbations can lead to significant changes in the projection. However, no precise definitions of solid or singular polyhedra or of a stable algorithm are given, and it is not apparent which elements (Černikov rules, redundancy removal, additional matrix clean-up), the algorithm referred to in the proposition, contains. Given the examples shown in e.g. [JMSY93], that Černikov's rules together with other forms of redundancy removals are unsound in general, and since Lukatskii and Shapot exactly augment the Fourier-Černikov method with a procedure for removing all redundant inequalities and further use "additional matrix clean-up" it is unclear to the authors of this report whether their presented method works in general, especially since this question is not raised or relevant references mentioned.

7 Conclusion

In this report we have presented a framework for projecting large, block structured inequality systems. The method incorporates procedures for preprocessing (including removal of linearly dependent inequalities), Gauss-elimination, Fourier-Motzkin-elimination, parallel, full redundancy removal and adjustment of the edges of the projecting (i.e. an approximation) using sequential redundancy removal of "almost redundant" inequalities. Further, the method uses a novel decomposition of the input problem to exploit its block structure. This decomposition method is usable not only together with Fourier-Motzkin elimination but with any other projection method; just

replace PROJECT(S_i^l, Y) in line 9 and line 14 of the SOLVE-algorithm in Algorithm 12 with any other method producing a projection of S_i^l w.r.t. Y.

Further, we have applied the presented method to a problem within the domain of liner shipping as we have obtained capacity models of manageable size from much larger stowage models. The obtained models are small and use approximately 100 times fewer variables, yet provide a better representation of the inter-dependencies between the number of stowed containers of different types than the very simple capacity models that are currently used.

The results indicate that even though Fourier-Motzkin elimination has a bad time and space complexity, which often deems it unfit for practical use, it is possible to amend it to obtain projections of large, realistic size problems. The execution time for the test cases is still high and impractical for repeated use in online algorithm, but for the purpose of projecting a model (inequality system) once to obtain another, smaller model, the method seems viable.

7.1 Future work

As already mentioned in Chapter 6 there are several interesting directions for future work, including

- using better methods for finding and removing redundant inequalities;
- using criteria such as Černikov's (if compatible) to avoid the addition of some of the redundant inequalities;
- using other methods for approximating the projection, while naturally also evaluating if these are actually better;
- possibly find and use a better evaluation of when an approximation or adjustment of the projection is necessary.

Since our decomposition-approach also works for other projection methods, it is also evident to investigate whether it can be an advantage to combine this decomposition with other, potentially more efficient methods for projection.

A particular approach that could be interesting to pursue in our case is to add Černikov rules to the Fourier-Motzkin-procedure, and only look for (and remove) *strictly* redundant inequalities in parallel after each elimination. This should be done for each subproblem, while a full redundancy removal and/or approximation of the projection should be done when subproblems are combined at the "next level" of the decomposition, at which point the index-set would also be reset. Instead of using the sequential redundancy check to approximate the projection by removing almost redundant inequalities, it would here be possible to implement another method, potentially the extreme point method or the convex hull method, both described in [HLL92]. The approximation approach in [SL12] could also be considered, particularly using an increasing ϵ-value and a measure relatiing to the size of the original problem to determine when approximations should be done and to what extend.

Other interesting topics for further research includes developing better heuristics for choosing the order of the variables to be substituted and deleted, plus automatic detection of useful decompositions of a given problem.

References

[AA95] Erling D. Andersen and Knud D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995.

[BKM05] Florence Benoy, Andy King, and Fred Mesnard. Computing convex hulls with a linear solver. *Theory and Practice of Logic Programming*, 5(1-2):259–271, 2005.

[BMW75] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, 8(1):54–83, 1975.

[BZ15] S.I. Bastrakov and N. Yu. Zolotykh. Fast method for verifying chernikov rules in fourier-motzkin elimination. *Computational Mathematics and Mathematical Physics*, 55(1):160–167, 2015.

[CH78] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th ACM symposium on Principles of Programming Languages*, pages 84–96. ACM, 1978.

[Del13] Alberto Delgado. *Models and Algorithms for Container Vessel Stowage Optimization*. PhD thesis, IT University of Copenhagen, 2013.

[Duf74] Richard J. Duffin. *On Fourier's analysis of linear inequality systems*, pages 71–95. Springer, 1974.

[FC08] Cheng-Min Feng and Chia-Hui Chang. Optimal slot allocation in intra-asia service for liner shipping companies. *Maritime Economics & Logistics*, 10(3):295–309, 2008.

[Fou26] J. B. J. Fourier. Solution d'une question particuliére du calcul des inégalités, 1826.

[Fou15] Alexis Fouilhé. *Revisiting the abstract domain of polyhedra: constraints-only representation and formal proof*. PhD thesis, Université Grenoble Alpes, 2015.

[HLL92] Tien Huynh, Catherine Lassez, and Jean-Louis Lassez. Practical issues on the projection of polyhedral sets. *Annals of mathematics and artificial intelligence*, 6(4):295–315, 1992.

[Imb90] Jean-Louis Imbert. About redundant inequalities generated by Fourier's algorithm. In P. Jorrand, editor, *In Proceedings of the Fourth International Conference on Artificial Intelligence (AIMSA'90)*, pages 117–127, 1990.

[Imb93] Jean-Louis Imbert. Fourier's elimination: Which to choose? In *Proceedings of the First workshop on Principles and Practice of Constraint Programming*, pages 117—129, 1993.

[JKM04] Colin Jones, Eric C. Kerrigan, and Jan Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical report, Cambridge University Engineering Dept, 2004.

[JKM08] C.N. Jones, E.C. Kerrigan, and J.M. Maciejowski. On polyhedral projection and parametric programming. *Journal of Optimization Theory and Applications*, 138(2):207–220, 2008.

[JMSY92] Joxan Jaffar, Michael J. Maher, Peter J. Stuckey, and Roland H.C. Yap. Output in CLP. In *In Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS)*, pages 987–995, 1992.

[JMSY93] Joxan Jaffar, Michael J. Maher, Peter J. Stuckey, and Roland H.C. Yap. Projecting CLP(R) constraints. *New Generation Computing*, 11(3-4):449–469, 1993.

[KLTZ83] Mark H. Karwan, Vahid Lotfi, Jan Telgen, and Stanley Zionts. *Redundancy in Mathematical Programming. A State-of-the-Art Survey*, volume 206 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag Berlin Heidelberg, 1983.

[Koh67] David A. Kohler. Projections of convex polyhedral sets. Technical report, Operations Research Center, College of Engineering, 1967.

[Las90] Jean-Louis Lassez. Querying constraints. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 288–298. ACM, 1990.

[LHM93] Jean-Louis Lassez, Tien Huynh, and Ken McAloon. Simplification and elimination of redundant linear arithmetic constraints. In Frédéric Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming*, pages 73–87. MIT Press, Cambridge, MA, USA, 1993.

[LL90] Catherine Lassez and Jean-Louis Lassez. *Quantifier Elimination for Conjunctions of Linear Constraints Via a Convex Hull Algorithm*. IBM Thomas J. Watson Research Division, 1990.

[LS08] Alexander M. Lukatskii and Demetrius V. Shapot. A constructive algorithm for folding large-scale systems of linear inequalities. *Computational Mathematics and Mathematical Physics*, 48(7):1100–1112, 2008.

[Mar03] Istvan Maros. *Computational Techniques of the Simplex Method*, volume 61 of *International Series in Operations Research & Management Science*. Springer US, 2003.

[Mat73] Theodore H. Mattheiss. An algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities. *Operations Research*, 21(1):247–260, 1973.

[Mon10] David Monniaux. Quantifier elimination by lazy model enumeration. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Proceedings of the 22nd International Conference on Computer Aided Verification*, pages 585–599. Springer Berlin Heidelberg, 2010.

[Mot36] Theodore S. Motzkin. *Beiträge zur Theorie der Linearen Ungleichungen*. PhD thesis, Universität Zürich, 1936.

[Pac12] Dario Pacino. *Fast Generation of Container Vessel Stowage Plans: using mixed integer programming for optimal master planning and constraint based local search for slot planning*. PhD thesis, IT University of Copenhagen, 2012.

[PDJB11] Dario Pacino, Alberto Delgado, Rune M. Jensen, and Tom Bebbington. Fast generation of near-optimal plans for eco-efficient stowage of large container vessels. *International Joint Conference on Artificial Intelligence. Proceedings*, pages 286–301, 2011.

[PDOJB12] Dario Pacino, Alberto Delgado-Ortegon, Rune M. Jensen, and Tom Bebbington. An accurate model for seaworthy container vessel stowage planning with ballast tanks. *Lecture Notes in Computer Science*, 2012.

[SK05] Axel Simon and Andy King. Exploiting sparsity in polyhedral analysis. In Chris Hankin and Igor Siveroni, editors, *Proceedings of the 12th International Symposium in Static Analysis (SAS)*, pages 336–351. Springer Berlin Heidelberg, 2005.

[SL12] Demetrius V. Shapot and Alexander M. Lukatskii. Solution building for arbitrary system of linear inequalities in an explicit form. *American Journal of Computational Mathematics*, 2(01):1, 2012.

[TBI07] Lloyd N. Trefenthen and David Bau III. *Numerical Linear Algebra*, chapter Lecture 12: Conditioning and Condition number. Society for Industrial and Applied Mathematics (SIAM), 2007.

[Tel83] Jan Telgen. Identifying redundant constraints and implicit equalities in systems of linear constraints. *Management Science*, 29(10):1209–1222, 1983.

[TT04] Shin-Chan Ting and Gwo-Hshiung Tzeng. An optimal containership slot allocation for liner shipping revenue management. *Maritime Policy & Management*, 31(3):199–211, 2004.

[Č63] Sergei N. Černikov. Contraction of finite systems of linear inequalities (in Russian). *Doklady Akademiia Nauk SSSR*, 152:1075–1078, 1963.

[Wil78] H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, LTD, 1978.

[ZF13] Sebastian Zurheide and Kathrin Fischer. A revenue management slot allocation model for liner shipping networks. *Maritime Economics & Logistics*, 15(4):523, 2013.

[Zie95] Günter M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate texts in mathematics*. Springer, New York, 1995.

Appendix

Lemma 1. *Let $S, S_1, S_2 \subset \mathcal{IE}$ be (in)equality systems over X and let $Y \subseteq X$.*

1. *Assume that $var(S_1) \cap var(S_2) = \varnothing$. Then*

$$proj_Y\big(feas(S_1 \cup S_2)\big) = proj_Y\big(feas(S_1)\big) \cap proj_Y\big(feas(S_2)\big). \tag{4}$$

2. *Assume that $var(S_1) \cap Y = \varnothing$. Then*

$$proj_Y\big(feas(S_1 \cup S_2)\big) = feas\big((S_1)_{X \setminus Y}\big) \cap proj_Y\big(feas(S_2)\big). \tag{5}$$

3. *Let $X' \subseteq \mathcal{X}$ be a super set of X, i.e. $X \subseteq X'$, and let E be an (in)equality system such that $proj_Y\big(feas(S)\big) = feas(E)$. Then*

$$proj_Y(feas(S_{X'})) = feas(E_{X' \setminus Y}).$$

Proof. We start by showing item 1. Thus assume that $var(S_1) \cap var(S_2) = \emptyset$. Order the variables such that $x \prec y$ for all $x \in X \setminus Y$ and $y \in Y$.

First assume that $\mathbf{r} \in proj_Y\big(feas(S_1 \cup S_2)\big)$. Then there exists values for the variables in Y, $\mathbf{r}_Y$, such that $(\mathbf{r}, \mathbf{r}_Y) \in feas(S_1 \cup S_2)$, i.e. for all $c \in S_1 \cup S_2$ it holds that $(\mathbf{r}, \mathbf{r}_Y) \in feas(c)$. Hence $(\mathbf{r}, \mathbf{r}_Y) \in feas(S_1)$ and $(\mathbf{r}, \mathbf{r}_Y) \in feas(S_2)$, so $\mathbf{r} \in proj_Y(feas(S_1))$ and $\mathbf{r} \in proj_Y(feas(S_2))$. That is, $\mathbf{r} \in proj_Y\big(feas(S_1)\big) \cap proj_Y\big(feas(S_2)\big)$.

On the other hand, let $Y_1 = Y \cap var(S_1)$, $Y_2 = Y \cap var(S_2)$ and $Y_3 = Y \setminus (Y_1 \cup Y_2)$, so that $Y = Y_1 \dot\cup Y_2 \dot\cup Y_3$. Order the variables such that $x \prec y_1$, $y_1 \prec y_2$ and $y_2 \prec y_3$ for all $x \in \mathcal{X} \setminus Y$, $y_1 \in Y_1$, $y_2 \in Y_2$ and $y_3 \in Y_3$. Assume that $\mathbf{r} \in proj_Y\big(feas(S_1)\big) \cap proj_Y\big(feas(S_2)\big)$. Then $\mathbf{r} \in proj_Y(feas(S_1))$ and $\mathbf{r} \in proj_Y(feas(S_2))$. Hence there exists values $\mathbf{u}_1$ and $\mathbf{v}_1$ for the variables in Y_1, values $\mathbf{u}_2$ and $\mathbf{v}_2$ for the variables in Y_2, and values $\mathbf{u}_3$ and $\mathbf{v}_3$ for the variables in Y_3 such that $\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \odot_c rhs(c)$ for all $c \in S_1$, and $\mathbf{co}(c') \cdot (\mathbf{r}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \odot_{c'} rhs(c')$ for all $c' \in S_2$.

Now consider the vector $(\mathbf{r}, \mathbf{u}_1, \mathbf{v}_2, \mathbf{0})$. Since $co(x, c) = 0$ for all $x \in Y_2 \cup Y_3$ and $c \in S_1$, we have that

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}_1, \mathbf{v}_2, \mathbf{0}) &= \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c_{Y_1}) \cdot \mathbf{u}_1 + \mathbf{co}(c_{Y_2}) \cdot \mathbf{v}_2 + \mathbf{co}(c_{Y_3}) \cdot \mathbf{0} \\
&= \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c_{Y_1}) \cdot \mathbf{u}_1 + \mathbf{0} \cdot \mathbf{v}_2 + \mathbf{0} \cdot \mathbf{0} \\
&= \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c_{Y_1}) \cdot \mathbf{u}_1 + \mathbf{co}(c_{Y_2}) \cdot \mathbf{u}_2 + \mathbf{co}(c_{Y_3}) \cdot \mathbf{u}_3 \\
&= \mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \odot_c rhs(c)
\end{aligned}$$

for all $c \in S_1$. Likewise

$$\begin{aligned}
\mathbf{co}(c') \cdot (\mathbf{r}, \mathbf{u}_1, \mathbf{v}_2, \mathbf{0}) &= \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c'_{Y_1}) \cdot \mathbf{u}_1 + \mathbf{co}(c'_{Y_2}) \cdot \mathbf{v}_2 + \mathbf{co}(c'_{Y_3}) \cdot \mathbf{0} \\
&= \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{0} \cdot \mathbf{u}_1 + \mathbf{co}(c'_{Y_2}) \cdot \mathbf{v}_2 + \mathbf{0} \cdot \mathbf{0} \\
&= \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c'_{Y_1}) \cdot \mathbf{v}_1 + \mathbf{co}(c'_{Y_2}) \cdot \mathbf{v}_2 + \mathbf{co}(c'_{Y_3}) \cdot \mathbf{v}_3 \\
&= \mathbf{co}(c') \cdot (\mathbf{r}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \odot_{c'} rhs(c')
\end{aligned}$$

for all $c' \in S_2$. That is, there exists an $\mathbf{r}_Y$, namely $(\mathbf{u}_1, \mathbf{v}_2, \mathbf{0})$, such that $(\mathbf{r}, \mathbf{r}_Y) \in feas(S_1 \cup S_2)$, i.e. $\mathbf{r} \in proj_Y(feas(S_1 \cup S_2))$.

This shows item 1 of the lemma.

Now we will show item 2. Thus assume that $var(S_1) \cap Y = \emptyset$. Order the variables such that $x \prec y$ for all $x \in X \setminus Y$ and $y \in Y$.

First assume that $\mathbf{r} \in feas\big((S_1)_{X \setminus Y}\big) \cap proj_Y\big(feas(S_2)\big)$. Then it holds that $\mathbf{co}(c)_{X \setminus Y} \cdot \mathbf{r} \odot_c rhs(c)$ for all $c \in S_1$, and there exists an $\mathbf{r}_Y$, denoting values for the variables in Y, such that $(\mathbf{co}(c'_{X \setminus Y}), \mathbf{co}(c'_Y)) \cdot (\mathbf{r}, \mathbf{r}_Y) \odot_{c'} rhs(c')$ for all $c' \in S_2$. Since $var(S_1) \cap Y = \emptyset$, we have that $\mathbf{co}(c_Y) = \mathbf{0}$ for all $c \in S_1$, and hence $\mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} = \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c_Y) \cdot \mathbf{r}_Y = (\mathbf{co}(c_{X \setminus Y}), \mathbf{co}(c_Y)) \cdot (\mathbf{r}, \mathbf{r}_Y) \odot_c rhs(c)$ for all $c \in S_1$.

Thus $\mathbf{r}_Y$ is such that $(\mathbf{co}(c_{X \setminus Y}), \mathbf{co}(c_Y)) \cdot (\mathbf{r}, \mathbf{r}_Y) \odot_c rhs(c)$ for all $c \in S_1 \cup S_2$, i.e. there exists an $\mathbf{r}_Y$ such that $(\mathbf{r}, \mathbf{r}_Y) \in feas(S_1 \cup S_2)$. That is, $\mathbf{r} \in proj_Y\big(feas(S_1 \cup S_2)\big)$.

On the other hand, assume that $\mathbf{r} \in proj_Y\big(feas(S_1 \cup S_2)\big)$. Then there exists an $\mathbf{r}_Y$ such that $(\mathbf{co}(c_{X \setminus Y}), \mathbf{co}(c_Y)) \cdot (\mathbf{r}, \mathbf{r}_Y) \odot_c rhs(c)$ for all $c \in S_1 \cup S_2$, so $\mathbf{r} \in proj_Y(feas(S_2))$. Let $c \in S_1$ be arbitrary. Then $\mathbf{co}(c_Y) = \mathbf{0}$ and hence $\mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} = \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r} + \mathbf{co}(c_Y) \cdot \mathbf{r}_Y = (\mathbf{co}(c_{X \setminus Y}), \mathbf{co}(c_Y)) \cdot (\mathbf{r}, \mathbf{r}_Y) \odot_c rhs(c)$, so $\mathbf{r} \in feas(c_{X \setminus Y})$. I.e. $\mathbf{r} \in feas\big((S_1)_{X \setminus Y}\big)$. In conclusion we therefore have that $\mathbf{r} \in feas\big((S_1)_{X \setminus Y}\big) \cap proj_Y(feas(S_2))$.

Combined, this shows that $feas\big((S_1)_{X \setminus Y}\big) \cap proj_Y\big(feas(S_2)\big) \subseteq proj_Y\big(feas(S_1 \cup S_2)\big)$ and $proj_Y\big(feas(S_1 \cup S_2)\big) \subseteq feas\big((S_1)_{X \setminus Y}\big) \cap proj_Y\big(feas(S_2)\big)$, i.e. 2 in the lemma holds.

Finally we will show item 3. Thus assume that $X \subseteq X'$ and $proj_Y\big(feas(S)\big) = feas(E)$. We have that $X' = (X' \setminus X) \dot\cup (X \setminus Y) \dot\cup Y$, so order the variables in $\mathcal{X}$ such that $x' \prec x \prec y$ for all $x' \in X' \setminus X, x \in X \setminus Y$ and $y \in Y$.

First assume that $\mathbf{r} = (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \in feas(E_{X' \setminus Y})$. Hence for all $e \in E_{X' \setminus Y}$ it holds that $\mathbf{r} \in feas(e)$. Now take arbitrary $e' \in E$. For the extension $e'_{X' \setminus Y} \in E_{X' \setminus Y}$ we have that

$$\begin{aligned}
\mathbf{co}(e'_{X' \setminus Y}) \cdot \mathbf{r} &= \mathbf{co}(e'_{X' \setminus X}) \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(e'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} \\
&= \mathbf{0} \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(e'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} \\
&= \mathbf{co}(e') \cdot \mathbf{r}_{X \setminus Y}.
\end{aligned}$$

Since $\mathbf{r} \in feas(e'_{X' \setminus Y})$ and $\odot_{e'} = \odot_{e'_{X' \setminus Y}}$, we get that $\mathbf{co}(e'_{X' \setminus Y}) \cdot \mathbf{r} \odot_{e'} rhs(e'_{X' \setminus Y}) = rhs(e')$, so $\mathbf{co}(e') \cdot \mathbf{r}_{X \setminus Y} \odot_{e'} rhs(e')$. Hence $\mathbf{r}_{X \setminus Y} \in feas(e')$ for all $e' \in E$, so $\mathbf{r}_{X \setminus Y} \in feas(E) = proj_Y(feas(S))$ by the assumptions. Therefore there exists an $\mathbf{r}_Y$ such that $(\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S)$.

Now let $c \in S_{X'}$ be arbitrary. Then $c = c'_{X'}$ for a $c' \in S$, and $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y)$ applied to c is therefore

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) &= \mathbf{co}(c_{X' \setminus X}) \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(c_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} + \mathbf{co}(c_Y) \cdot \mathbf{r}_Y \\
&= \mathbf{0} \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} + \mathbf{co}(c'_Y) \cdot \mathbf{r}_Y \\
&= (\mathbf{co}(c'_{X \setminus Y}), \mathbf{co}(c'_Y)) \cdot (\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \\
&= \mathbf{co}(c') \cdot (\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \odot_{c'} rhs(c') = rhs(c)
\end{aligned}$$

since $(\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S)$ and $c' \in S$. Since $\odot_{c'} = \odot_c$, we therefore have that $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(c)$ for all $c \in S_{X'}$, and hence $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S_{X'})$, and $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \in proj_Y(feas(S_{X'}))$.

On the other hand assume that $\mathbf{r} = (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \in proj_Y(feas(S_{X'}))$ That means that there exists an $\mathbf{r}_Y$ such that $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S_{X'})$. Now take an arbitrary c' in S. Then $(\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y)$ applied to c' yields

$$\begin{aligned}
\mathbf{co}(c') \cdot (\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) &= \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} + \mathbf{co}(c'_Y) \cdot \mathbf{r}_Y \\
&= \mathbf{0} \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(c'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} + \mathbf{co}(c'_Y) \cdot \mathbf{r}_Y \\
&= \mathbf{co}(c'_{X'}) \cdot (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \odot_{c'} rhs(c')
\end{aligned}$$

since $c'_{X'} \in S_{X'}$, $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S_{X'})$, $\odot_{c'_{X'}} = \odot_{c'}$, and $rhs(c'_{X'}) = rhs(c')$. Hence $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(c'_{X'})$.

Thus, $(\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(c')$ for all $c' \in S$, so $(\mathbf{r}_{X \setminus Y}, \mathbf{r}_Y) \in feas(S)$. Hence $\mathbf{r}_{X \setminus Y} \in proj_Y(S) = feas(E)$.

Now take arbitrary $e \in E_{X' \setminus Y}$, i.e. $e = e'_{X' \setminus Y}$ for an $e' \in E$. Applying $(\mathbf{r}_{X' \setminus Y}, \mathbf{r}_{X \setminus Y})$ to e then gives us

$$\begin{aligned}
\mathbf{co}(e) \cdot (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) &= \mathbf{co}(e'_{X' \setminus Y}) \cdot (\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \\
&= \mathbf{co}(e'_{X' \setminus X}) \cdot \mathbf{r}_{X' \setminus X} + \mathbf{co}(e'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} \\
&= \mathbf{0} \cdot r_{X' \setminus X} + \mathbf{co}(e'_{X \setminus Y}) \cdot \mathbf{r}_{X \setminus Y} \\
&= \mathbf{co}(e') \cdot \mathbf{r}_{X \setminus Y} \odot_e rhs(e)
\end{aligned}$$

since $e' \in E$, $\mathbf{r}_{X \setminus Y} \in feas(E)$, $\odot_{e'} = \odot_e$, and $rhs(e') = rhs(e)$.

Hence $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \in feas(e)$ for all $e \in E_{X' \setminus Y}$, i.e. $(\mathbf{r}_{X' \setminus X}, \mathbf{r}_{X \setminus Y}) \in feas(E_{X' \setminus Y})$.

This shows item 3. □

Proposition 1. *Let c and c' be given and assume that all variables in $var(\{c, c'\})$ are non-negative. If* (11) *hold for an $\sigma \geq 0$, then* (11) *holds for $\sigma' \geq 0$ given below.*

If there exists an $x \in var(c')$ such that $co(x, c') < 0$ then

$$\sigma' = \begin{cases} \min(m, \frac{rhs(c)}{rhs(c')}) & \text{if } rhs(c') > 0 \\ m & \text{otherwise} \end{cases}, \text{ where } m = \min_{\substack{x \in var(c'). \\ co(x,c') < 0}} \frac{co(x, c)}{co(x, c')}.$$

Otherwise, if $co(x, c') \geq 0$ for all $x \in var(c')$ and $var(c') \neq \emptyset$, then

$$\sigma' = \begin{cases} \max(m', \frac{rhs(c)}{rhs(c')}, 0) & \text{if } rhs(c') < 0 \\ \max(m', 0) & \text{otherwise} \end{cases}, \text{ where } m' = \max_{\substack{x \in var(c'). \\ co(x,c') > 0}} \frac{co(x, c)}{co(x, c')}.$$

Otherwise $\sigma' = \begin{cases} \max(\frac{rhs(c)}{rhs(c')}, 0) & \text{if } rhs(c') \neq 0 \\ 0 & \text{otherwise} \end{cases}$.

Proof. Assume that σ satisfies (11). Then (11) implies that $\frac{co(x,c)}{co(x,c')} \leq \sigma$ for all x such that $co(x, c') > 0$, and $\sigma \leq \frac{co(x,c)}{co(x,c')}$ for all x such that $co(x, c') < 0$. Likewise, $\sigma \leq \frac{rhs(c)}{rhs(c')}$ if $rhs(c') > 0$, and $\sigma \geq \frac{rhs(c)}{rhs(c')}$ if $rhs(c') < 0$.

First we consider the case where $\{ x \mid co(x, c) < 0 \} \neq \emptyset$.

For all x for which $co(x, c') < 0$ we have that $0 \leq \sigma \leq \frac{co(x,c)}{co(x,c')}$, thus $m \geq 0$. If $rhs(c') > 0$ then $0 \leq \sigma \leq \frac{rhs(c)}{rhs(c')}$, so $\sigma' \geq 0$ by definition.

Now take an arbitrary $x \in var(c) \cup var(c')$. If $co(x,c') = 0$ then it follows from (11) that $co(x,c) \leq 0$, i.e. $co(x,c) \leq \sigma' \cdot co(x,c')$. If $co(x,c') < 0$ then $\frac{co(x,c)}{co(x,c')} \geq \sigma'$ by definition of σ', so $co(x,c) \leq \sigma' \cdot co(x,c')$. Finally, if $co(x,c') > 0$ then $\frac{co(x,c)}{co(x,c')} \leq \sigma'$; otherwise $\frac{co(x,c)}{co(x,c')} > \sigma' = m = \frac{co(y,c)}{co(y,c')}$ for a y such that $co(y,c') < 0$, or $\frac{co(x,c)}{co(x,c')} > \sigma' = \frac{rhs(c)}{rhs(c')}$ and $rhs(c') > 0$. But that means that either $\frac{co(x,c)}{co(x,c')} \leq \sigma \leq \frac{co(y,c)}{co(y,c')} < \frac{co(x,c)}{co(x,c')}$, or $\frac{co(x,c)}{co(x,c')} \leq \sigma \leq \frac{rhs(c)}{rhs(c')} < \frac{co(x,c)}{co(x,c')}$, which are both contradictions. Therefore $\frac{co(x,c)}{co(x,c')} \leq \sigma'$, and hence $co(x,c) \leq \sigma' \cdot co(x,c')$.

If $rhs(c') = 0$ then from (11) it follows that $rhs(c) \geq= 0$, i.e. $rhs(c) = 0 \geq \sigma' \cdot rhs(c') = 0$. If $rhs(c') > 0$ then by definition of σ', $\sigma' \leq \frac{rhs(c)}{rhs(c')}$, so $\sigma' \cdot rhs(c') \leq rhs(c)$. Finally, if $rhs(c') < 0$ then we must have that $\frac{rhs(c)}{rhs(c')} \leq \sigma'$ and hence $rhs(c) \geq \sigma' \cdot rhs(c')$. Otherwise $\frac{rhs(c)}{rhs(c')} > \sigma' = \frac{co(x,c)}{co(x,c')}$ for an x such that $co(x,c') < 0$. Hence (11) implies that $\frac{rhs(c)}{rhs(c')} \leq \sigma \leq \frac{co(x,c)}{co(x,c')} = \sigma' < \frac{rhs(c)}{rhs(c')}$, which is a contradiction.

We then consider the case where $co(x,c) \geq 0$ for all $x \in var(c') \neq \emptyset$.

By definition $\sigma' \geq 0$.

Take an arbitrary $x \in var(c) \cup var(c')$. If $co(x,c') = 0$ then $co(x,c) \leq 0 = \sigma' \cdot co(x,c')$. If $co(x,c') \neq 0$ then $co(x,c') > 0$. By definition $\sigma' \geq m'$, i.e. $\sigma' \geq \frac{co(y,c)}{co(y,c')}$ for all y such that $co(y,c') > 0$. Thus $\sigma \geq \frac{co(x,c)}{co(x,c')}$, and hence $co(x',c) \leq \sigma' \cdot co(x,c')$. If $rhs(c') = 0$, then again $rhs(c) \geq \sigma \cdot rhs(c') = 0 = \sigma' \cdot rhs(c')$. If $rhs(c') < 0$ then by definition $\sigma' \geq \frac{rhs(c)}{rhs(c')}$, i.e. $\sigma' \cdot rhs(c') \leq rhs(c)$. Finally, if $rhs(c') > 0$, then we must have that $\sigma' \cdot rhs(c') \leq rhs(c)$; otherwise $\sigma' \cdot rhs(c') > rhs(c)$, i.e. $\frac{rhs(c)}{rhs(c')} < \sigma' = \frac{co(x,c)}{co(x,c')}$ for an x such that $co(x,c') > 0$, or $\frac{rhs(c)}{rhs(c')} < \sigma' = 0$. In the former case we get that $\frac{co(x,c)}{co(x,c')} \leq \sigma \leq \frac{rhs(c)}{rhs(c')} < \sigma' = \frac{co(x,c)}{co(x,c')}$, and in the latter case $0 \leq \sigma \leq \frac{rhs(c)}{rhs(c')} < \sigma' = 0$, which are both contradictions.

In the case where $var(c') = \emptyset$, then we have that $\sigma' \geq 0$ by definition. (11) implies that $co(x,c) \leq 0$ for all $x \in var(\{c,c'\})$, so $co(x,c) \leq \sigma' \cdot co(x,c') = 0$. If $rhs(c') = 0$ then (11) implies that $rhs(c) \geq 0 = \sigma' \cdot rhs(c')$. If $rhs(c) \neq 0$ then either $\frac{rhs(c)}{rhs(c')} \geq 0$, or $\frac{rhs(c)}{rhs(c')} < 0$. In the former case we have by definition that $\sigma' \cdot rhs(c') = \frac{rhs(c)}{rhs(c')} \cdot rhs(c') = rhs(c)$. In the latter case, we must have that $rhs(c') \leq 0$, since otherwise $\frac{rhs(c)}{rhs(c')} < 0 \leq \sigma \leq \frac{rhs(c)}{rhs(c')}$ which is a contradiction. So $rhs(c') < 0$, which gives that $rhs(c) > 0$, and hence $\sigma' \cdot rhs(c') \leq 0 < rhs(c)$. □

Lemma 2.

$$proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0}\big(feas(sep(S))\big).$$

Proof. Let $X = VAR(S)$. Order the variables such that $x \prec y \prec z$ for all $x \in X \setminus Y, y \in Y$ and $z \in Z^0$.

First assume that $\mathbf{r}$ is a set of values for the variables in $X \setminus Y$ such that $\mathbf{r} \in proj_Y\big(feas(S)\big)$. This means that there exists values for the variables in $Y \subseteq X$, $\mathbf{u}$, such that the values $(\mathbf{r}, \mathbf{u})$ satisfy all (in)equalities $c \in S$. Hence $\sum_{x \in X \setminus Y} co(x,c) \cdot r_x + \sum_{y \in X \cap Y} co(y,c) \cdot u_y \odot_c rhs(c)$ for all $c \in S$, where r_x is the value of x in $\mathbf{r}$ for all $x \in X \setminus Y$ and u_y is the value of y in $\mathbf{u}$ for all $y \in Y$.

Now define $v_{z^0_{c,i}} = \sum_{x \in X_i \setminus Y} co(x,c) \cdot r_x + \sum_{y \in X_i \cap Y} co(y,c) \cdot u_y$ for all $z^0_{c,i} \in Z^0$. Let $\mathbf{v}$ be the vector of all values in $\cup_{1 \leq i \leq k^0} \{\ v_{z^0_{c,i}} \mid c \in S_\mathrm{t}\ \}$ in the order given by $\prec$.

Let $c \in S^0_\mathrm{t} \cup \bigcup_{1 \leq i \leq k^0} S^0_i = (S^0_\mathrm{t})_{X \cup Z^0} \cup \bigcup_{1 \leq i \leq k^0} (S^0_i)_{X \cup Z^0}$ be arbitrary. We will then show that $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ satisfy c, and hence $\mathbf{r} \in proj_{Y \cup Z^0}\big(feas(S^0_\mathrm{t} \cup \bigcup_{1 \leq i \leq k^0} S^0_i)\big) = proj_{Y \cup Z_0}\big(feas(S)\big)$.

If $c \in (S^0_i)_{X \cup Z^0}$ for an i, then by construction either c equals $c'_{X \cup Z^0}$ for a $c' \in S$ or c equals $Def(z^0_{c',i})$ for a $c' \in S_\mathrm{t}$. In the former case, $co(x,c) = co(x,c')$ for all $x \in X$, and $co(x,c) = 0$ for all $x \in Z^0$. $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ applied to c is therefore

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}, \mathbf{v}) &= \sum_{x \in X \setminus Y} co(x,c) \cdot r_x + \sum_{y \in Y} co(y,c) \cdot u_y + \sum_{z \in Z^0} co(z,c) \cdot v_y \\
&= \sum_{x \in X \setminus Y} co(x,c') \cdot r_x + \sum_{y \in Y} co(y,c') \cdot u_y + \sum_{z \in Z^0} 0 \cdot v_z \\
&= \sum_{x \in X \setminus Y} co(x,c') \cdot r_x + \sum_{y \in Y} co(y,c') \cdot u_y \\
&= \mathbf{co}(c') \cdot (\mathbf{r}, \mathbf{u}) \odot_{c'} rhs(c')
\end{aligned}$$

since $c' \in S$ is satisfied by $(\mathbf{r}, \mathbf{u})$. I.e. c is satisfied by $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ (since $\odot_c = \odot_{c'}$ and $rhs(c) = rhs(c')$).

In the latter case (c equals $Def(z^0_{c',i})$ for a $c' \in S_\text{t}$), c is the equality $-z^0_{c',i} + \sum_{x \in X_i} co(x,c') \cdot x = 0$ extended to $X \cup Z^0$. So $co(x,c) = co(x,c')$ for all $x \in X_i$, $co(z^0_{c',i}, c) = -1$, and $co(x,c) = 0$ for all $x \in X \cup Z_0 \setminus (X_i \cup \{z^0_{c',i}\})$. $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ applied to c is therefore

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}, \mathbf{v}) &= \sum_{x \in X \setminus Y} co(x,c) \cdot r_x + \sum_{y \in Y} co(y,c) \cdot u_y + \sum_{z \in Z^0} co(z,c) \cdot v_y \\
&= \sum_{x \in X_i \setminus Y} co(x,c') \cdot r_x + \sum_{y \in X_i \cap Y} co(y,c') \cdot u_y - 1 \cdot v_{z^0_{c',i}} \\
&= v_{z^0_{c',i}} - v_{z^0_{c',i}} = 0 = rhs(c).
\end{aligned}$$

I.e. c is satisfied by $(\mathbf{r}, \mathbf{u}, \mathbf{v})$.

Finally, if $c \in (S^0_\text{t})_{X \cup V^0}$ then by construction c equals c'^0_{decp} extended to $X \cup Z^0$ for a $c' \in S$, i.e. c equals the extension of $(\sum_{1 \leq i \leq k^0} z^0_{c',i}) + \sum_{x \in X_\text{t}} co(x,c') \cdot x \odot_{c'} rhs(c')$. Hence $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ applied to c is

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}, \mathbf{v}) &= \sum_{x \in X_\text{t} \setminus Y} co(x,c') \cdot r_x + \sum_{y \in X_\text{t} \cap Y} co(y,c') \cdot u_y + \sum_{1 \leq i \leq k^0} v_{z^0_{c',i}} \\
&= \sum_{x \in X_\text{t} \setminus Y} co(x,c') \cdot r_x + \sum_{y \in X_\text{t} \cap Y} co(y,c') \cdot u_y \\
&\quad + \sum_{1 \leq i \leq k^0} \Big(\sum_{x \in X_i \setminus Y} co(x,c') \cdot r_x + \sum_{y \in X_i \cap Y} co(y,c') \cdot u_y \Big) \\
&= \sum_{x \in X \setminus Y} co(x,c') \cdot r_x + \sum_{y \in Y} co(y,c') \cdot u_y \\
&= \mathbf{co}(c') \cdot (\mathbf{r}, \mathbf{u}) \odot_{c'} = rhs(c') = rhs(c),
\end{aligned}$$

since $X \setminus Y = (X_1 \setminus Y) \dot\cup \ldots \dot\cup (X_{k^0} \setminus Y) \dot\cup (X_\text{t} \setminus Y)$ and $(X_1 \cap Y) \dot\cup \ldots \dot\cup (X_{k^0} \cap Y) \dot\cup (X_\text{t} \cap Y) = X \cap Y = Y$. Since $\odot_{c'} = \odot_c$ we therefore have that c is satisfied by $(\mathbf{r}, \mathbf{u}, \mathbf{v})$. Hence $proj_Y\big(feas(S)\big) \subseteq proj_{Y \cup Z^0}\Big(feas\big(sep(S)\big)\Big)$.

On the other hand assume that $\mathbf{r}$ consists of the values for the variables in $X \setminus Y$ such that $\mathbf{r} \in proj_{Y \cup Z^0}\big(feas(sep(S))\big) = proj_{Y \cup Z^0}\big(feas(S^0_\text{t} \cup \bigcup_{1 \leq i \leq k^0} S^0_i)\big)$. Then there exists values $\mathbf{u}$ and $\mathbf{v}$ for the variables in Y and Z^0, respectively, such that $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ satisfies all (in)equalities c in $S^0_\text{t} \cup \bigcup_{1 \leq i \leq k^0} S^0_i$. To be able to refer to the values, let r_x be the value in $\mathbf{r}$ for the variable $x \in X \setminus Y$, let u_y be the value in $\mathbf{u}$ for $y \in Y$, and let v_z be the value in $\mathbf{c}$ for $z \in Z^0$.

Now take an arbitrary (in)equality c in S. Either $var(c) \subseteq X_i$ for an $0 \leq i \leq k^0$, or there is no such i. In the former case, by construction $c_{VAR(S^0_i)} \in S^0_i$ for an i, and hence $c_{X \cup Z^0} \in S^0_\text{t} \cup \bigcup_{1 \leq i \leq k^0} S^0_i$, and thus it is satisfied by $(\mathbf{r}, \mathbf{u}, \mathbf{v})$. This means that

$$\begin{aligned}
\mathbf{co}(c) \cdot (\mathbf{r}, \mathbf{u}) &= \sum_{x \in X \setminus Y} co(x,c) \cdot r_x + \sum_{y \in Y} co(y,c) \cdot u_y \\
&= \sum_{x \in X \setminus Y} co(x,c) \cdot r_x + \sum_{y \in Y} co(y,c) \cdot u_y + \sum_{z \in Z^0} 0 \cdot v_z \\
&= \sum_{x \in X \setminus Y} co(x, c_{X \cup Z^0}) \cdot r_x + \sum_{y \in Y} co(y, c_{X \cup Z^0}) \cdot u_y + \sum_{z \in Z^0} co(z, c_{X \cup Z^0}) \cdot v_z \\
&= \mathbf{co}(c_{X \cup Z^0}) \cdot (\mathbf{r}, \mathbf{u}, \mathbf{v}) \odot_{c_{X \cup Z^0}} rhs(c_{X \cup Z^0}) = rhs(c).
\end{aligned}$$

Since $\odot_c = \odot_{c_{X \cup Z^0}}$ this means that $(\mathbf{r}, \mathbf{u})$ satisfies c. Thus $\mathbf{r} \in proj_Y(feas(c))$.

In the latter case (i.e. there is no i such that $var(c) \subseteq X_i$), we get that for each $1 \leq i \leq k^0$, $Def(z^0_{c,i})$ extended to $X \cup Z^0$ belongs to $S^0_\text{t} \bigcup_{1 \leq i \leq k^0} S^0_i$ and are therefore satisfied, i.e. $(\mathbf{r}, \mathbf{u}, \mathbf{v})$ applied to $-z^0_{c,i} + \sum_{x \in X_i} co(x,c) \cdot x$ equals 0 for all i. That is, $-v_{z^0_{c,i}} + \sum_{x \in X_i \setminus Y} co(x,c) \cdot r_x + \sum_{y \in X_i \cap Y} co(y,c) \cdot u_y = 0$. Hence we must have that $v_{z^0_{c,i}} = \sum_{x \in X_i \setminus Y} co(x,c) \cdot r_x + \sum_{y \in X_i \cap Y} co(y,c) \cdot u_y$.

Likewise, c^0_{decp} is in S^0_t and it is satisfied, i.e. $\sum_{1 \leq i \leq k^0} v_{z^0_{c,i}} + \sum_{x \in X_\text{t} \setminus Y} co(x,c) \cdot r_x + \sum_{y \in X_\text{t} \cap Y} co(x,c) \cdot u_y \odot_c rhs(c)$.

From this we gather that

$$\begin{aligned}\mathbf{co}(c)\cdot(\mathbf{r},\mathbf{u}) &= \sum_{x\in X_\curlywedge\setminus Y} co(x,c)\cdot r_x + \sum_{y\in Y} co(y,c)\cdot u_y\\ &= \sum_{1\le i\le k^0}\Big(\sum_{x\in X_i\setminus Y} co(x,c)\cdot r_x + \sum_{y\in X_i\cap Y} co(y,c)\cdot u_y\Big)\\ &\quad + \sum_{x\in X_\curlywedge\setminus Y} co(x,c)\cdot r_x + \sum_{y\in X_\curlywedge\cap Y} co(y,c)\cdot u_y\\ &= \sum_{1\le i\le k^0} v_{z^0_{c,i}} + \sum_{x\in X_\curlywedge\setminus Y} co(x,c)\cdot r_x + \sum_{y\in X_\curlywedge\cap Y} co(y,c)\cdot u_y\\ &\odot_c rhs(c).\end{aligned}$$

That is, c is satisfied by $(\mathbf{r},\mathbf{u})$. Hence $\mathbf{r}\in proj_Y\big(feas(S)\big)$, and $proj_{Y\cup Z^0}\Big(feas\big(sep(S)\big)\Big)\subseteq proj_Y\big(feas(S)\big)$. □

Lemma 3.

$$proj_{Z^l}\Big(feas(S^l_\curlywedge\cup\bigcup_{1\le i\le k^l} S^l_i)\Big) = feas(S^{l-1}_\curlywedge)$$

Proof. First we will show that for all $c\in S^0_\curlywedge$ the following holds:

$$feas(c^{l-1}_{decp}) = proj_{Z^l}\Big(feas\big(\{c^l_{decp}\}\cup\bigcup_{1\le i\le k^l}\{Def(z^l_{c,i})\}\big)\Big). \tag{40}$$

Order the variables such that $x\prec z^l_{c,j}$ for all $x\in X_\curlywedge\cup\bigcup_{1\le j\le k^{l-1}}\{z^{l-1}_{c,j}\}$ and all $1\le j\le k^l$, and such that $z^l_{c,i}\prec z^l_{c,j}$ iff $i<j$.

First assume that $\mathbf{r}\in feas(c^{l-1}_{decp})$, i.e. $\mathbf{r}$ defines values for variables in $X_\curlywedge\cup\bigcup_{1\le i\le k^{l-1}}\{z^{l-1}_{c,i}\}$ that satisfy c^{l-1}_{decp}. Let therefore r_x be the value of x in $\mathbf{r}$ for all $x\in X_\curlywedge\cup\bigcup_{1\le i\le k^{l-1}}\{z^{l-1}_{c,i}\}$. For all $1\le i\le k^l$ define values for $z^l_{c,i}$ as follows: $v_{z^l_{c,i}}=\sum_{j\in P^l_i} r_{z^{l-1}_{c,i}}$ Let $\mathbf{v}=(v_{z^l_{c,1}},\dots,v_{z^l_{c,k^l}})$. Then $(\mathbf{r},\mathbf{v})$ applied to c^l_{decp} is

$$\begin{aligned}\sum_{1\le i\le k^l} v_{z^l_{c,i}} + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x &= \sum_{1\le i\le k^l}\Big(\sum_{j\in P^l_i} r_{z^{l-1}_{c,i}}\Big) + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x\\ &\overset{1}{=} \Big(\sum_{1\le i\le k^{l-1}} r_{z^{l-1}_{c,i}}\Big) + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x\\ &= \mathbf{co}(c^{l-1}_{decp})\cdot\mathbf{r}\odot_{c^{l-1}_{decp}} rhs(c^{l-1}_{decp}),\end{aligned}$$

since $\mathbf{r}\in feas(c^{l-1}_{decp})$. 1 holds since $\{1,\dots,k^{l-1}\}=P^l_1\dot\cup\dots\dot\cup P^l_{k^l}$. Thus $(\mathbf{r},\mathbf{v})$ satisfies c^l_{decp} (since $\odot_{c^l_{decp}}=\odot_{c^{l-1}_{decp}}$ and $rhs(c^l_{decp})=rhs(c^{l-1}_{decp})$).

Now let $1\le i\le k^l$ be arbitrary. Then $(\mathbf{r},\mathbf{v})$ applied to $Def(z^l_{c,i})$ is

$$-v_{z^l_{c,i}} + \sum_{j\in P^l_i} r_{z^{l-1}_{c,j}} = -v_{z^l_{c,i}} + v_{z^l_{c,i}} = 0 = rhs(Def(z^l_{c,i})).$$

Thus $(\mathbf{r},\mathbf{v})$ satisfies $Def(z^l_{c,i})$ for any $1\le i\le k^l$, too.

That is, $\mathbf{r}\in proj_{Z^l}\Big(feas\big(\{c^l_{decp}\}\cup\bigcup_{1\le i\le k^l}\{Def(z^l_{c,i})\}\big)\Big)$.

On the other hand, assume that $\mathbf{r}\in proj_{Z^l}\Big(feas\big(\{c^l_{decp}\}\cup\bigcup_{1\le i\le k^l}\{Def(z^l_{c,i})\}\big)\Big)$. Since $\{c^l_{decp}\}\cup\bigcup_{1\le i\le k^l}\{Def(z^l_{c,i})\}$ is a system over $X_\curlywedge\cup\bigcup_{1\le i\le k^l}\{z^l_{c,i}\}\cup\bigcup_{1\le j\le k^{l-1}}\{z^{l-1}_{c,j}\}$, $\mathbf{r}$ defines values for the variables in $X_\curlywedge\cup\bigcup_{1\le j\le k^{l-1}}\{z^{l-1}_{c,j}\}$, so let r_x be the value of x in $\mathbf{r}$ for all $x\in X_\curlywedge\cup\bigcup_{1\le j\le k^{l-1}}\{z^{l-1}_{c,j}\}$.

Then there exists values for the variables in Z^l, $\mathbf{u}$, such that $(\mathbf{r},\mathbf{u})$ satisfies c^l_{decp} and $Def(z^l_{c,i})$ for all $1\le i\le k^l$. Hence we have for all $1\le i\le k^l$ that $-u_{z^l_{c,i}}+\sum_{j\in P^l_i} r_{z^{l-1}_{c,j}}=0$, where $u_{z^l_{c,i}}$ is the value for $z^l_{c,i}$ in $\mathbf{u}$.

Now consider $\mathbf{r}$ applied to c^{l-1}_{decp}, which is an (in)equality over $X_\curlywedge\cup\bigcup_{1\le j\le k^{l-1}}\{z^{l-1}_{c,j}\}$:

$$\begin{aligned}\sum_{1\le i\le k^{l-1}} r_{z^{l-1}_{c,i}} + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x &= \sum_{1\le i\le k^l}\Big(\sum_{j\in P^l_i} r_{z^{l-1}_{c,i}}\Big) + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x\\ &= \sum_{1\le i\le k^l} u_{z^l_{c,i}} + \sum_{x\in X_\curlywedge} co(x,c)\cdot r_x\\ &\odot_{c^l_{decp}} rhs(c^l_{decp}),\end{aligned}$$

where the last (in)equality holds because c^l_{decp} is satisfied by $(\mathbf{r},\mathbf{u})$. Thus $\mathbf{r}$ satisfies c^{l-1}_{decp} since $\odot_{c^{l-1}_{decp}} = \odot_{c^l_{decp}}$ and $rhs(c^{l-1}_{decp}) = rhs(c^l_{decp})$. Hence we have shown (40).

Let $V = VAR(S^l_\mathtt{t} \cup \bigcup_{1\le i\le k^l} S^l_i) = X_\mathtt{t} \cup Z^l \cup Z^{l-1}$. Now the statement in the lemma follows since

$$\begin{aligned}
proj_{Z^l}\big(feas(S^l_\mathtt{t} \cup \bigcup_{1\le i\le k^l} S^l_i)\big) &= proj_{Z^l}\Big(feas\big((S^l_\mathtt{t})_V \cup \bigcup_{1\le i\le k^l} (S^l_i)_V\big)\Big)\\
&= proj_{Z^l}\Big(feas\big(\bigcup_{c\in S^0_\mathtt{t}} (\{c^l_{decp}\} \cup \bigcup_{1\le i\le k^l} \{Def(z^l_{c,i})\})_V\big)\Big)\\
&\overset{1}{=} \bigcap_{c\in S^0_\mathtt{t}} proj_{Z^l}\Big(feas\big((\{c^l_{decp}\} \cup \bigcup_{1\le i\le k^l} \{Def(z^l_{c,i})\})_V\big)\Big)\\
&\overset{2}{=} \bigcap_{c\in S^0_\mathtt{t}} feas\big((c^{l-1}_{decp})_{V\setminus Z^l}\big)\\
&= feas\big(\bigcup_{c\in S^0_\mathtt{t}} \{(c^{l-1}_{decp})_{V\setminus Z^l}\}\big) = feas\big((\bigcup_{c\in S^0_\mathtt{t}} \{c^{l-1}_{decp}\})_{V\setminus Z^l}\big)\\
&\overset{3}{=} feas(S^{l-1}_\mathtt{t}).
\end{aligned}$$

1 follows since from Lemma 1 item 1 since $var(\{c^l_{decp}\} \cup \bigcup_{i\le i\le k^l}\{Def(z^l_{c,i})\}) \cap var(\{c'^l_{decp}\} \cup \bigcup_{i\le i\le k^l}\{Def(z^l_{c',i})\}) = \varnothing$ for $c, c' \in S^0_\mathtt{t}$ where $c \neq c'$, 2 follows from Lemma 1 item 3 and (40) proven above, and 3 follows since $VAR(S^{l-1}_\mathtt{t}) = X_\mathtt{t} \cup Z^{l-1} = V \setminus Z^l$. □

Lemma 4.

$$S^K_\mathtt{t} \cup \bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^K} S^K_i = \mathfrak{S}^{K+1}_1$$

Proof. We show by induction on $0 \le l \le K$ that

$$\bigcup_{1\le i\le k^l} \mathfrak{S}^l_i = \bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^l} S^l_i \tag{41}$$

For $l = 0$ we have by definition that $S^0_i = \mathfrak{S}^0_i$ for all $1 \le i \le k^0$, so $\bigcup_{1\le i\le k^0} S^0_i = \bigcup_{1\le i\le k^0} \mathfrak{S}^0_i$. Thus (41) holds.

Now assume that $0 < l \le K$ and that (41) holds for all $0 \le l' < l$. Let $X = VAR(\bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^l} S^l_i) = \bigcup_{1\le i\le k^0} X_i \cup \bigcup_{1\le m\le l} Z^m$. Then

$$\begin{aligned}
\bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^l} S^l_i &= \big(\bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^l} S^l_i\big)_X\\
&= \big(\bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^{l-1}} S^{l-1}_i\big)_X \cup \big(\bigcup_{1\le i\le k^l} S^l_i\big)_X\\
&\overset{\text{IH}}{=} \big(\bigcup_{1\le i\le k^{l-1}} \mathfrak{S}^{l-1}_i\big)_X \cup \big(\bigcup_{1\le i\le k^l} S^l_i\big)_X\\
&= \big(\bigcup_{i\in P^l_1\cup\ldots\cup P^l_{k^l}} \mathfrak{S}^{l-1}_i\big)_X \cup \big(\bigcup_{1\le i\le k^l} S^l_i\big)_X\\
&= \big(\bigcup_{1\le i\le k^l} \bigcup_{j\in P^l_i} \mathfrak{S}^{l-1}_i\big)_X \cup \big(\bigcup_{1\le i\le k^l} S^l_i\big)_X\\
&= \bigcup_{1\le i\le k^l} \bigcup_{j\in P^l_i} (\mathfrak{S}^{l-1}_i)_X \cup \bigcup_{1\le i\le k^l} (S^l_i)_X\\
&= \bigcup_{1\le i\le k^l} (\bigcup_{j\in P^l_i} \mathfrak{S}^{l-1}_j \cup S^l_i)_X\\
&= (\bigcup_{1\le i\le k^l} \mathfrak{S}^l_i)_X = \bigcup_{1\le i\le k^l} \mathfrak{S}^l_i.
\end{aligned}$$

Thus (41) also holds for $l = K$, which means that

$$\mathfrak{S}^{K+1}_1 = S^{K+1}_1 \cup \bigcup_{j\in P^{K+1}_1} \mathfrak{S}^K_j = S^K_\mathtt{t} \cup \bigcup_{1\le j\le k^K} \mathfrak{S}^K_j = S^K_\mathtt{t} \cup \bigcup_{1\le i\le k^0} S^0_i \cup \ldots \cup \bigcup_{1\le i\le k^K} S^K_i.$$

□

Proposition 2.

$$proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0 \cup \ldots \cup Z^K}\big(feas(\mathfrak{S}_1^{K+1})\big)$$

Proof. We will show the proposition by showing that

$$proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0 \cup \ldots \cup Z^l}\Big(feas\big(S_\mathfrak{t}^l \cup \bigcup_{0 \le m \le l} \bigcup_{1 \le i \le k^m} S_i^m\big)\Big) \tag{42}$$

for all $0 \le l \le K$. Then (42) holds for K too, and hence $proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0 \cup \ldots \cup Z^K}\Big(feas\big(S_\mathfrak{t}^K \cup \bigcup_{0 \le m \le K} \bigcup_{1 \le i \le k^m} S_i^m\big)\Big) = proj_{Y \cup Z^0 \cup \ldots \cup Z^K}\big(feas(\mathfrak{S}_1^{K+1})\big)$ by Lemma 4.

The proof is done by induction on l.

For $l = 0$, Lemma 2 gives us that $proj_Y\big(feas(S)\big) = proj_{Y \cup Z^0}\Big(feas\big(S_\mathfrak{t}^0 \cup \bigcup_{1 \le i \le k^0} S_i^0\big)\Big)$ which is what we need.

For $0 < l \le K$ we assume that (42) holds for all $0 \le l' < l$. For ease of notation we let in the following $\mathbf{Z} \overset{\text{def.}}{=} Y \cup Z^0 \cup \ldots \cup Z^{l-1}$ and $X = VAR(S_\mathfrak{t}^l) \cup \bigcup_{0 \le m \le l} \bigcup_{1 \le i \le k^m} VAR(S_i^m)$. Then

$$\begin{aligned}
&proj_{Y \cup Z^0 \cup \ldots \cup Z^l}\Big(feas\big(S_\mathfrak{t}^l \cup \bigcup_{0 \le m \le l} \bigcup_{1 \le i \le k^m} S_i^m\big)\Big) \\
&= proj_{Y \cup Z^0 \cup \ldots \cup Z^l}\Big(feas\big((S_\mathfrak{t}^l)_X \cup \bigcup_{0 \le m \le l} \bigcup_{1 \le i \le k^m} (S_i^m)_X\big)\Big) \\
&= proj_{\mathbf{Z}}\Big(proj_{Z^l}\Big(feas\big(((S_\mathfrak{t}^l)_X \cup \bigcup_{1 \le i \le k^l} (S_i^l)_X) \cup (\bigcup_{0 \le m \le l-1} \bigcup_{1 \le i \le k^m} (S_i^m)_X)\big)\Big)\Big) \\
&\overset{*}{=} proj_{\mathbf{Z}}\Big(proj_{Z^l}\Big(feas\big((S_\mathfrak{t}^l \cup \bigcup_{1 \le i \le k^l} S_i^l)_X\big)\Big) \cap feas\big(\bigcup_{0 \le m \le l-1} \bigcup_{1 \le i \le k^m} (S_i^m)_{X \setminus Z^l}\big)\Big) \\
&\overset{\substack{\text{Lemma 1 item 3} \\ \text{+Lemma 3}}}{=} proj_{\mathbf{Z}}\Big(feas\big((S_\mathfrak{t}^{l-1})_{X \setminus Z^l}\big) \cap feas\big(\bigcup_{0 \le m \le l-1} \bigcup_{1 \le i \le k^m} (S_i^m)_{X \setminus Z^l}\big)\Big) \\
&= proj_{\mathbf{Z}}\Big(feas\big((S_\mathfrak{t}^{l-1})_{X \setminus Z^l} \cup \bigcup_{0 \le m \le l-1} \bigcup_{1 \le i \le k^m} (S_i^m)_{X \setminus Z^l}\big)\Big) \\
&= proj_{\mathbf{Z}}\Big(feas\big(S_\mathfrak{t}^{l-1} \cup \bigcup_{0 \le m \le l-1} \bigcup_{1 \le i \le k^m} S_i^m\big)\Big) \\
&\overset{\text{IH}}{=} proj_Y(S).
\end{aligned}$$

Above, $*$ follows from Lemma 1 item 2 since $var\big((S_i^m)_X\big) \cap Z^l = \varnothing$ for all $m \le l - 1$ and $0 \le i \le k^m$. By the principle of mathematical induction, (42) then holds for all $l \le K$. □

Proposition 3.

$$proj_{Z^K \cup \ldots \cup Z^0 \cup Y}\big(feas(\mathfrak{S}_1^{K+1})\big) = feas(\mathfrak{E}_1^{K+1})$$

Proof. We proof the proposition by showing that

$$proj_{Z^{l-1} \cup \ldots \cup Z^0 \cup Y}\big(feas(\mathfrak{S}_i^l)\big) = feas(\mathfrak{E}_i^l) \tag{43}$$

for all $0 \le l \le K + 1$ and all $1 \le i \le k^l$. This is done by induction on l.

Let $l = 0$ and $0 \le i \le k^0$ be arbitrary. Then by definition $\mathfrak{E}_i^0 \in PRS_Y(S_i^0)$, so $feas(\mathfrak{E}_i^0) = proj_Y(feas(S_i^0)) = proj_Y(feas(\mathfrak{S}_i^0))$.

For the induction step, let $0 < l \le K + 1$ and assume that (43) holds for all $0 \le l' < l$. Let $0 \le i \le k^l$ be arbitrary.

In the following we let $\mathbf{Z} = Z^{l-2} \cup Z^{l-3} \cup \ldots \cup Z^0 \cup Y$ and we let $X = VAR(\mathfrak{S}_i^l)$. Then

$$
\begin{aligned}
&proj_{Z^{l-1} \cup Z^{l-2} \cup \ldots \cup Z^0 \cup Y}\big(feas(\mathfrak{S}_i^l)\big) \\
&\quad = proj_{Z^{l-1} \cup \mathbf{Z}}\big(feas(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{S}_j^{l-1})\big) && \text{def. of } \mathfrak{S}_i^l \\
&\quad = proj_{Z^{l-1}}\Big(proj_{\mathbf{Z}}\Big(feas\big((S_i^l)_X \cup \bigcup_{j \in P_i^l} (\mathfrak{S}_j^{l-1})_X\big)\Big)\Big) \\
&\quad = proj_{Z^{l-1}}\Big(feas\big((S_i^l)_{X \setminus \mathbf{Z}}\big) \cap proj_{\mathbf{Z}}\Big(feas\big(\bigcup_{j \in P_i^l} (\mathfrak{S}_j^{l-1})_X\big)\Big)\Big) && \begin{array}{r}\text{Lemma 1 item 2,} \\ var((S_i^l)_X) \cap \mathbf{Z} = \emptyset\end{array} \\
&\quad = proj_{Z^{l-1}}\Big(feas\big((S_i^l)_{X \setminus \mathbf{Z}}\big) \cap \bigcap_{j \in P_i^l} proj_{\mathbf{Z}}\Big(feas\big((\mathfrak{S}_j^{l-1})_X\big)\Big)\Big) && \begin{array}{r}\text{Lemma 1 item 1,} \\ var((\mathfrak{S}_m^{l-1})_X) \cap var((\mathfrak{S}_{m'}^{l-1})_X) = \emptyset \\ \text{for } m \neq m'\end{array} \\
&\quad = proj_{Z^{l-1}}\Big(feas\big((S_i^l)_{X \setminus \mathbf{Z}}\big) \cap \bigcap_{j \in P_i^l} feas\big((\mathfrak{E}_j^{l-1})_{X \setminus \mathbf{Z}}\big)\Big) && \begin{array}{r}\text{Lemma 1 item 3,} \\ \text{Induction hypothesis}\end{array} \\
&\quad = proj_{Z^{l-1}}\Big(feas\big((S_i^l)_{X \setminus \mathbf{Z}} \cup \bigcup_{j \in P_i^l} (\mathfrak{E}_j^{l-1})_{X \setminus \mathbf{Z}}\big)\Big) \\
&\quad = proj_{Z^{l-1}}\Big(feas\big((S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1})_{X \setminus \mathbf{Z}}\big)\Big) \\
&\quad = proj_{Z^{l-1}}\Big(feas\big(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}\big)\Big) && \text{See below} \\
&\quad = feas(\mathfrak{E}_i^l). && \mathfrak{E}_i^l \in PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1})
\end{aligned}
$$

The second to last equality above holds because $VAR(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}) = X \setminus \mathbf{Z}$ which is proven below. Thus, (43) holds for l, and by the principle of mathematical induction it follows that (43) holds for $K+1$.

To conclude the proof, we now show by induction on l that the following holds for all $1 \leq l \leq K+1$ and $1 \leq i \leq k^l$.

$$VAR(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}) = VAR(\mathfrak{S}_i^l) \setminus (Y \cup \bigcup_{0 \leq m \leq l-2} Z^m) \tag{44}$$

For $l = 1$, we have for an arbitrary $1 \leq i \leq k^1$ that

$$
\begin{aligned}
VAR(S_i^1 \cup \bigcup_{j \in P_i^1} \mathfrak{E}_j^0) &= VAR(S_i^1) \cup \bigcup_{j \in P_i^1} VAR(\mathfrak{E}_j^0) \\
&= VAR(S_i^1) \cup \bigcup_{j \in P_i^1} (VAR(S_j^0) \setminus Y) && \mathfrak{E}_j^0 \in PRS_Y(S_j^0) \text{ by def.} \\
&= \big(VAR(S_i^1) \cup \bigcup_{j \in P_i^1} VAR(S_j^0)\big) \setminus Y && VAR(S_i^1) \cap Y = \emptyset \\
&= VAR\big(S_i^1 \cup \bigcup_{j \in P_i^1} S_j^0\big) \setminus Y \\
&= VAR(\mathfrak{S}_i^1) \setminus Y && \text{def. of } \mathfrak{S}_i^1 \\
&= VAR(\mathfrak{S}_i^l) \setminus (Y \cup \bigcup_{0 \leq m \leq l-1} Z^m).
\end{aligned}
$$

I.e. (44) holds for $l = 1$.

Now assume that $1 < l \leq K+1$ and that (44) holds for all $1 \leq l' < l$. Let $1 \leq i \leq k^l$ be arbitrary. Then similarly

$$\begin{aligned}
VAR(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}) &= VAR(S_i^l) \cup \bigcup_{j \in P_i^l} VAR(\mathfrak{E}_j^{l-1}) \\
&\overset{1}{=} VAR(S_i^l) \cup \bigcup_{j \in P_i^l} \Big(VAR(S_j^{l-1} \cup \bigcup_{k \in P_j^{l-1}} \mathfrak{E}_k^{l-2}) \setminus Z^{l-2} \Big) \\
&\overset{\text{IH}}{=} VAR(S_i^l) \cup \bigcup_{j \in P_i^1} \Big(\Big(VAR(\mathfrak{S}_j^{l-1}) \setminus (Y \cup \bigcup_{0 \leq m \leq l-3} Z^m) \Big) \setminus Z^{l-2} \Big) \\
&= VAR(S_i^l) \cup \bigcup_{j \in P_i^1} \Big(VAR(\mathfrak{S}_j^{l-1}) \setminus (Y \cup \bigcup_{0 \leq m \leq l-2} Z^m) \Big) \\
&\overset{2}{=} \Big(VAR(S_i^l) \cup \bigcup_{j \in P_i^1} VAR(\mathfrak{S}_j^{l-1}) \Big) \setminus (Y \cup \bigcup_{0 \leq m \leq l-2} Z^m) \\
&= VAR(\mathfrak{S}_i^l) \setminus (Y \cup \bigcup_{0 \leq m \leq l-2} Z^m)
\end{aligned}$$

1 holds because $\mathfrak{E}_j^{l-1} \in PRS_{Z^{l-2}}(S_j^{l-1} \cup \bigcup_{k \in P_j^{l-1}} \mathfrak{E}_k^{l-2})$ by definition, and 2 holds because $VAR(S_i^l) \cap (Y \cup \bigcup_{0 \leq m \leq l-2} Z^m) = \emptyset$.

Thus, (44) holds for all $1 \leq l \leq K+1$. □

Lemma 5. *Let $p, p' : 2^{2^{\mathcal{IE}}} \to 2^{\mathcal{IE}}$ be such that $p(M), p'(M) \in M$ for all $M \in 2^{2^{\mathcal{IE}}}$. Then $\mathfrak{E}_i^l(p) \cong \mathfrak{E}_i^l(p')$ for all $0 \leq l \leq K+1$ and all $1 \leq i \leq k^l$, where*

$$\mathfrak{E}_i^l(p) \overset{\text{def.}}{=} \begin{cases} p\big(PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p))\big) & \text{if } l > 0 \\ p\big(PRS_Y(S_i^l)\big) & \text{if } l = 0 \end{cases}, \text{ and}$$

$$\mathfrak{E}_i^l(p') \overset{\text{def.}}{=} \begin{cases} p'\big(PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p'))\big) & \text{if } l > 0 \\ p'\big(PRS_Y(S_i^l)\big) & \text{if } l = 0 \end{cases}.$$

Proof. By induction on l we will show that $feas(\mathfrak{E}_i^l(p)) = feas(\mathfrak{E}_i^l(p'))$ and $VAR(\mathfrak{E}_i^l(p)) = VAR(\mathfrak{E}_i^l(p'))$for all $1 \leq i \leq k^l$.

For $l = 0$ we have by definition that $\mathfrak{E}_i^0(p), \mathfrak{E}_i^0(p') \in PRS_Y(S_i^0)$ for all $1 \leq i \leq k^0$. Hence $VAR(\mathfrak{E}_i^0(p)) = VAR(S_i^0) \setminus Y = VAR(\mathfrak{E}_i^0(p'))$, and $feas(\mathfrak{E}_i^0(p)) = proj_Y(feas(S_0^i)) = feas(\mathfrak{E}_i^0(p'))$.

Now assume that $0 < l \leq K+1$, and assume that the induction hypothesis holds for all $0 \leq l' < l$. Let $1 \leq i \leq k^l$ be arbitrary. Let $X = VAR(S_i^l) \cup \bigcup_{j \in P_i^l} VAR(\mathfrak{E}_j^{l-1}(p)) \overset{\text{IH}}{=} VAR(S_i^l) \cup \bigcup_{j \in P_i^l} VAR(\mathfrak{E}_j^{l-1}(p'))$. Since $\mathfrak{E}_i^l(p) \in PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p))$ and $\mathfrak{E}_i^l(p') \in PRS_{Z^{l-1}}(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p'))$, we have that $VAR(\mathfrak{E}_i^l(p)) = X \setminus Z^{l-1} = VAR(\mathfrak{E}_i^l(p'))$. Further,

$$\begin{aligned}
feas(\mathfrak{E}_i^l(p)) &= proj_{Z^{l-1}}\Big(feas\big(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p)\big)\Big) \\
&= proj_{Z^{l-1}}\big(feas((S_i^l)_X) \cap \bigcap_{j \in P_i^l} feas(\mathfrak{E}_j^{l-1}(p)_X)\big) \\
&\overset{\text{IH}}{=} proj_{Z^{l-1}}\big(feas((S_i^l)_X) \cap \bigcap_{j \in P_i^l} feas(\mathfrak{E}_j^{l-1}(p')_X)\big) \\
&= proj_{Z^{l-1}}\Big(feas\big(S_i^l \cup \bigcup_{j \in P_i^l} \mathfrak{E}_j^{l-1}(p')\big)\Big) \\
&= feas(\mathfrak{E}_i^l(p')).
\end{aligned}$$

By the principle of mathematical induction this shows the proposition. □