Introducing Non-linear Parameters to the Polyhedron Model

Armin Größlinger
Fakultät für Mathematik und Informatik
Universität Passau

Martin Griebl
Fakultät für Mathematik und Informatik
Universität Passau

Christian Lengauer
Fakultät für Mathematik und Informatik
Universität Passau

Abstract

A mathematical model based on polyhedra (the so-called "polyhedron model") serves as a foundation for model based loop program transformation like automatic parallelization. One of the restrictions present in the current polyhedron model is the requirement that the coefficients of variables must be numeric constants. This has been hindering some recent developments which require parametric coefficients of variables. We show how such non-linear parameters can be introduced in the polyhedron model, using quantifier elimination in the real numbers as our main mathematical tool. We describe two approaches of obtaining algorithms for the generalized model. First, we point out how existing algorithms can be implemented for the generalized model. Quantifier elimination is employed in this approach to simplify arising case distinctions. We give Fourier-Motzkin elimination and the Simplex algorithm as examples of this approach. Second, we show how quantifier elimination can be used to solve some problems directly, e.g., by computing lexicographic maxima. We also demonstrate how to apply our methods to the frequently appearing case of tiling an index space with parametric tile size, and we present some performance results of the generalized algorithms we have implemented.

1 Introduction

In model based loop program transformation one frequently uses polyhedra to describe programs and transformations of them. In this mathematical model, one deals usually with linear inequality systems, in which the parameters may appear linearly in the additive term, but the coefficients of variables must be constants. One of the great advantages of the polyhedron model is that some important algorithms, like Fourier-Motzkin elimination, deliver results which do not contain case distinctions. This is due to the fact that the coefficients of the variables are fixed numbers and, since all the case distinctions in the Fourier-Motzkin algorithm are about signs of these coefficients (or coefficients derived from them), the conditions of the case distinctions can be evaluated statically (i.e., at compile time) when performing Fourier-Motzkin elimination.

Some problems cannot be expressed in the polyhedron model as just described. For example, it is not possible to specify the tiling of an index space with arbitrary tile shapes and parametric tile

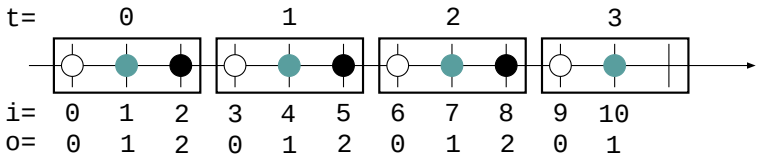


Figure 1: A tiled iteration domain

sizes. The reason is illustrated in Section 1.1. To describe such a parametric tiling (and other problems which require parameters as coefficients of variables), we have to use a generalized version of the polyhedron model in which the coefficients of variables may depend on parameters.

1.1 Parametric Tiling Example

As an example of a problem expressed in the generalized model, take a very simple one-dimensional tiling. Consider the index set defined by the following inequality system in the variable i for $n \geq 0$:

$$0 \leq i \leq n \tag{1}$$

We desire a tiling of this index set into tiles of size p (for $p \geq 1$), i.e., p adjacent points of the index set shall be contained in the same tile. This can be described by the following inequality system:

$$\begin{aligned} 0 \leq o \leq p-1 \\ i = p \cdot t + o \end{aligned} \tag{2}$$

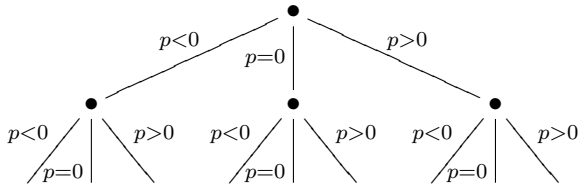
where variable t denotes the number of a tile and variable o is the offset of a point within its tile (in this case, from the left end of the tile). Figure 1 illustrates the tiled index set for $n = 10$ and $p = 3$. Obviously, p is a non-linear parameter in the inequality system formed by systems (1) and (2) since it appears as coefficient of variable t. To get an enumeration (using Fourier-Motzkin) of the tiled index space with a loop for the tile numbers t, and inside that loop another loop for the coordinate i, we have to eliminate the variables in the order o, i, t. We first solve $i = p \cdot t + o$ for o and eliminate o by substituting into the two inequalities. After that we solve the system for i:

$$\begin{aligned} 0 &\leq i \\ p \cdot t &\leq i \\ i &\leq n \\ i &\leq p \cdot t + p - 1 \end{aligned}$$

We eliminate i from this system by comparing lower bounds against upper bounds. This yields (after dropping two superfluous inequalities to simplify the calculation):

$$\begin{aligned} 1 - p \leq p \cdot t \\ p \cdot t \leq n \end{aligned}$$

To solve the system for t, we have to divide by coefficient p. In general, this would require a case distinction on the sign of p, since for $p < 0$ the relation symbols must be changed when dividing by p, and if $p = 0$ we cannot solve for t at all. A naïve implementation of Fourier-Motzkin would produce the following two-fold case distinction (since there are two inequalities with p as coefficient of t):



An implementation of an algorithm for the generalized polyhedron model should avoid such superfluous cases. The method we choose to solve this problem is quantifier elimination. Quantifier elimination allows one to check logical formulas like $\forall p\,(p \geq 1 \rightarrow p > 0)$. The formula $p \geq 1$ is the precondition on p stated above and $p > 0$ is one of the possible cases in the above case distinction. Given the formula $\forall p\,(p \geq 1 \rightarrow p > 0)$, quantifier elimination establishes that the formula is true, i.e., for every p (from the real numbers) $p \geq 1$ implies $p > 0$. In other words: the precondition $p \geq 1$ ensures that $p > 0$ is the only possible case in the above case distinction.

1.2 The Generalized Model

In the classical polyhedron model for loop parallelization, we deal with inequality systems with variables $x_1, \ldots, x_n$ and linear parameters $p_1, \ldots, p_m$ of the form

$$\sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{m} d_i p_i + e \geq 0$$

where $c_1, \ldots, c_n, p_1, \ldots, p_m, e \in \mathbb{Q}$. The previous section shows that parametric tiling of an index space cannot be described in this model. Therefore, we extend it to a generalized model which uses inequalities of the form:

$$\sum_{i=1}^{n} c_i x_i + d \geq 0$$

But now, in contrast to the classical model, the coefficients $c_1, \ldots, c_n, d$ are not fixed rational numbers, but may be symbolic constants; we call them parameters. Since the operations performed on the coefficients by the algorithms used in the polyhedron model include addition, subtraction, multiplication, and division, it is necessary to choose a field as the domain for the coefficients. We use fractions whose numerators and denominators are polynomials in the parameters. Formally, the domain of the coefficients $c_1, \ldots, c_n$ is denoted by $\mathbb{Q}(p_1, \ldots, p_m)$, which is defined as

$$\mathbb{Q}(p_1, \ldots, p_m) := \{\frac{a}{b} \mid a, b \in \mathbb{Q}[p_1, \ldots, p_m], b \neq 0\}$$

where $\mathbb{Q}[p_1, \ldots, p_m]$ denotes the ring of (multivariate) polynomials with indeterminates $p_1, \ldots, p_m$ over $\mathbb{Q}$.

1.3 Tree Representation of Case Distinctions

In general, results delivered by algorithms which handle input inequality systems with non-linear parameters cannot be represented without case distinctions. The example of $p \cdot x - 1 \geq 0$ illustrates this. If we assume nothing about the parameter p, solving $p \cdot x - 1 \geq 0$ for x yields three different solutions:

- $x \geq \frac{1}{p}$ if $p > 0$,
- false if $p = 0$,
- $x \leq \frac{1}{p}$ if $p < 0$.

We represent this case distinction as a decision tree. A leaf node of such a tree carries a result for a specific case, and an inner node n repesents some conditional. Each of the subtrees of n is applicable under a certain condition. We use the following definition for the decision tree data type:

$$
\begin{aligned}
&\mathbf{data}\ \mathit{Tree}\ \alpha\ =\ \mathit{Leaf}\ \alpha \\
&\quad \mid\ \mathit{SCond}\quad \mathit{Polynomial}\ (\mathit{Tree}\ \alpha)\ (\mathit{Tree}\ \alpha)\ (\mathit{Tree}\ \alpha) \\
&\quad \mid\ \mathit{EqCond}\ \ \mathit{Polynomial}\ (\mathit{Tree}\ \alpha)\ (\mathit{Tree}\ \alpha) \\
&\quad \mid\ \mathit{GeCond}\ \ \mathit{Polynomial}\ (\mathit{Tree}\ \alpha)\ (\mathit{Tree}\ \alpha) \\
&\quad \mid\ \mathit{FCond}\quad \mathit{QfFormula}\ (\mathit{Tree}\ \alpha)\ (\mathit{Tree}\ \alpha)
\end{aligned}
$$

The most important inner node types are:

- $\mathit{Leaf}\quad x$ represents a result with value x.
- $\mathit{SCond}\quad f\quad t_-\quad t_0\quad t_+$ represents a case distinction on the sign of the polynomial $f \in \mathbb{Q}[p_1, \ldots, p_m]$. t_- is applicable for $f < 0$, t_0 for $f = 0$, and t_+ for $f > 0$. The specialized constructors EqCond and GeCond are used when a binary case distinction is required for $f = 0$ against $f \neq 0$, and $f \geq 0$ against $f < 0$, respectively.
- $\mathit{FCond}\quad \varphi\quad t_\top\quad t_\bot$ represents a case distinction on an arbitrary quantifier-free logical formula φ. $t_\top$ applies if φ holds, and $t_\bot$ applies when φ does not hold.

The result of solving $p \cdot x - 1 \geq 0$ for x would be represented as

$$\mathit{SCond}\ p\ (\mathit{Leaf}\ x \leq \frac{1}{p})\ (\mathit{Leaf}\ \mathrm{false})\ (\mathit{Leaf}\ x \geq \frac{1}{p}).$$

2 Obtaining Generalized Algorithms via Program Transformation

In this section we explore ways to derive algorithms for the generalized polyhedron model from existing algorithms for the classical polyhedron model. We give an informal description of program transformation rules which generalize an existing algorithm. Then we look at necessary simplifications in the resulting decision trees.

2.1 New Algorithms by Program Transformation

Some algorithms (e.g., Fourier-Motzkin elimination and the Simplex algorithm) contain case distinctions on the signs of intermediate values (computed from the input values, i.e., the coefficients of the input inequalities). The general structure of such a case distinction is

$$\begin{array}{l}\textbf{case } \mathit{signum}\ f \textbf{ of}\\ \quad \mathit{Negative} \to t_-\\ \quad \mathit{Zero} \quad\ \to t_0\\ \quad \mathit{Positive} \to t_+\end{array}$$

where f is an expression derived from the input values of the algorithm by arithmetic operations. We restrict our consideration to the case that addition, subtraction, multiplication, and division in the real numbers are used, since this is essential for using real quantifier elimination to simplify the resulting decision trees (Section 2.2). If the input inequalities contain non-linear parameters, f is not a rational number but a rational function in the parameters, i.e., $f = \frac{f_1}{f_2}$ for $f_1, f_2 \in \mathbb{Q}[p_1, \ldots, p_m]$, $f_2 \neq 0$. It is generally impossible to decide which sign f has (since it depends on the values of the parameters, in general), so we modify the algorithm such that the above case distinction in the algorithm's *code* is replaced by a case distinction in the resulting *data structure*. We rewrite the above code to

$$\mathit{SCond}\ \ (f_1 \cdot f_2)\ \ t_-\ \ t_0\ \ t_+$$

Note that $sgn(f) = sgn(\frac{f_1}{f_2}) = sgn(f_1 \cdot f_2)$.
In addition to this transformation of case distinctions, we have to make some other changes in the algorithm:

- We have to replace expressions e which construct final results by $\mathit{Leaf}\ e$.
- If some function $f :: \alpha \to \beta$ is applied to an expression e which changes its type from α to $\mathit{Tree}\ \alpha$ during the generalization, we have to map f over the whole tree e, i.e., apply f to every leaf of e, by using some suitable combinator. The choice of the combinator depends on whether f has to be generalized to have the new type $\alpha \to \mathit{Tree}\ \beta$, or not [Grö03].

We do not formally apply this informally defined set of transformation rules to existing implementations of algorithms, since implementing the transformation system seems more difficult than implementing generalized versions of existing algorithms following the ideas of this transformation system. In addition, it may be desirable to deviate from strictly applying the transformation rules. In Fourier-Motzkin elimination, for example, we optimize the sets of lower and upper bounds by checking whether for two bounds b_1 and b_2 one of the relations $b_1 \leq b_2$ or $b_1 \geq b_2$ holds and remove the irrelevant bound. In the generalized Fourier-Motzkin elimination, this may depend on parameters, so the transformation system would generate a case distinction. But usually we do not want a case distinction in this case since, if neither $b_1 \leq b_2$ nor $b_1 \geq b_2$ holds, we simply keep both bounds b_1 and b_2 in the set of lower or upper bounds.

We have implemented a generalized Fourier-Motzkin elimination and the Simplex algorithm; see Section 5 for some practical applications of them.

2.2 Tree Simplification

The main challenge arising from the transformation system given in the last section is to simplify the decision trees constructed by generalized algorithms.

We use a top-down simplification procedure. The simplification starts at the root node of the decision tree with a *context* C which contains all the assumptions on the parameters, e.g., $C = \{p_1 \geq 0, p_2 > 4\}$. We illustrate the simplification procedure by looking at the node $n = SCond\, f\ t_-\ t_0\ t_+$. When the simplifier reaches this node, it checks whether the context implies one of the conditions $f < 0$, $f = 0$, or $f > 0$. If the context implies one of these conditions, the node n is replaced by the respective subtree of n and the simplification continues on that subtree. If the context implies none of these conditions, the node n is retained and the simplifier is applied recursively to the three subtrees t_-, t_0, t_+. For each of the subtrees the context is modified to contain the condition which makes the respective subtree applicable. For example, the simplification on the subtree t_- is performed with respect to the new context $C \cup \{f < 0\}$, since t_- is only applicable for $f < 0$. The full simplification procedure we have implemented [Grö03] handles special cases like detecting that $SCond$ can be replaced by $EqCond$ or $GeCond$, etc.

Checking whether the context C implies a certain condition is done by deciding, for example, the logical formula $\forall p_1 \cdots \forall p_m \left(\bigwedge C \rightarrow f < 0\right)$. Here $\bigwedge C$ denotes the conjunction of all formulas in C. "Deciding" means here to determine whether the formula holds in the real numbers, or not. We use quantifier elimination tools to do so (cf. Section 5).

3 Quantifier Elimination Based Algorithms

In addition to generalizing existing algorithms to non-linear parameters, we have also explored ways to solve some common problems in the generalized polyhedron model using new algorithms. We discovered that quantifier elimination is a mathematical tool which can be used for that.

As an example, we consider the lexicographic minimum of a polyhedron. To be able to apply quantifier elimination to a problem, we have to express the problem as a first-order logical formula with the operators $+$, $-$, $\cdot$, and the usual equality and ordering relations of the real numbers. Since this logical language does not contain a notion of lexicographic ordering, we have to define the lexicographic less-than relation $\preceq$ based on the standard ordering $<$ on the real numbers:

$$\begin{aligned} a_1 \preceq b_1 &:= a_1 \leq b_1 \\ (a_1, \bar{a}) \preceq (b_1, \bar{b}) &:= a_1 < b_1 \vee \left(a_1 = b_1 \wedge \bar{a} \preceq \bar{b}\right) \end{aligned}$$

A given polyhedron (which may depend on possibly non-linear parameters) can be described by a finite set of inequalities S in the variables $x_1, \ldots, x_n$. Without loss of generality, we assume that the inequalities in S are denominator-free (this can always be achieved by multiplying every inequality with the square of the common denominator of its coefficients). The conjunction $\bigwedge S$ of the inequalities in S describes the polyhedron (in dependence of the parameters). To find a formula describing the lexicographic minimality of a finite point $(x_1, \ldots, x_n)$, we translate the following property L into a logical formula:

> The point $(x_1, \ldots, x_n)$ is the lexicographic minimum of the given polyhedron if it

lies inside the polyhedron and it is lexicographically less than or equal to every other point $(y_1, \ldots, y_n)$ which also lies inside the polyhedron.

To express this property, we define $\varphi := \bigwedge S$ and $\psi := \varphi[x_1 := y_1, \ldots, x_n := y_n]$ for some new variables $y_1, \ldots, y_n$ (i.e., ψ is the same as φ with y_i instead of x_i). Then property L can be expressed by the formula μ:

$$\mu \;:=\; \varphi \wedge \forall y_1 \ldots \forall y_n \big(\psi \rightarrow (x_1, \ldots, x_n) \preceq (y_1, \ldots, y_n)\big)$$

The existence of a lexicographic minimum $(x_1, \ldots, x_n)$ is expressed by the formula:

$$\exists x_1 \cdots \exists x_n \, (\mu)$$

Some quantifier elimination tools can "solve" this problem by finding conditions under which values for $x_1, \ldots, x_n$ exist such that μ becomes true, and calculating such values for the variables $x_1, \ldots, x_n$. That is, the answer given by the quantifier elimination procedure is a set

$$\{(\gamma_i, [x_1 := t_{i,1}, \ldots, x_n := t_{i,n}]) \mid 1 \leq i \leq l\} \tag{3}$$

for some $l \in \mathbb{N}$, where γ_i is a quantifier-free logical formula in the parameters and $t_{i,1}, \ldots, t_{i,n}$ are expressions (i.e., terms) in the parameters describing the lexicographic minimum under the condition γ_i. This procedure is called "quantifier elimination with answer," in contrast to plain quantifier elimination which is used for decision tree simplification.

Example Consider the system

$$q \leq x_2 \leq p \cdot x_1$$

in the variables x_1, x_2 and the parameters p, q. For this system the formula μ is:

$$\begin{aligned} \mu \;:=\; & (q \leq x_2 \wedge x_2 \leq p \cdot x_1) \quad \wedge \\ & \forall y_1 \forall y_2 \big(q \leq y_2 \wedge y_2 \leq p \cdot y_1 \rightarrow x_1 < y_1 \vee (x_1 = y_1 \wedge x_2 \leq y_2)\big) \end{aligned}$$

The quantifier elimination's answer to the question $\exists x_1 \exists x_2 \, (\mu)$ is:

$$\{(p > 0, [x_1 := \frac{q}{p}, x_2 := q])\}$$

This means that, in the case $p > 0$, there is a finite lexicographic minimum, namely at $(\frac{q}{p}, q)$. Otherwise (i.e., for $p \leq 0$) there is no finite lexicographic minimum, since the polyhedron is either empty or unbounded.

The case distinction contained in an answer like in system (3) can easily be transformed into a decision tree using the $FCond$ constructor.

4 Special cases

Quantifier elimination can be the dominating factor for the overall computation time when applying a generalized algorithm. Therefore, it is desirable to find special cases where no need for quantifier elimination arises. We present here briefly one special case which can be exploited when generating loop nests which describe the tiling of an index space.

4.1 $K \cdot L$ decomposition

Lemma 1 *Let $A \cdot \bar{x} + a \geq 0$ be an inequality system in the variables $\bar{x} = x_1, \ldots, x_n$ and the parameters $p_1, \ldots, p_m$. When the coefficient matrix A can be written as a product $A = K \cdot L$, where K is a constant matrix and L is a lower triangular matrix (possibly containing parameters) such that the assumptions on the parameters imply that the diagonal entries of L are positive, then Fourier-Motzkin elimination of $A \cdot \bar{x} + a \geq 0$ does not lead to any case distinctions.*

The proof is left out due to space restrictions and can be obtained from the authors.

4.2 Tiling

For our experiments in Section 5 we use an index space tiling problem as example. To describe the tiling of an index space, we need the following information [AI91]:

- an index space described by an inequality system $S \cdot (x_1 \cdots x_n)^{\mathrm{T}} + s \geq 0$ in the variables $x_1, \ldots, x_n$,
- a tile shape described by an inequality system $T \cdot (o_1 \cdots o_n)^{\mathrm{T}} + t \geq 0$ in the variables $o_1, \ldots, o_n$, and
- vectors $l_1, \ldots, l_n$ which describe the translation between the base tile and other tiles; matrix $A = (l_1 \cdots l_n)$ is called a *lattice*.

The tiling is described by:

$$S \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + s \geq 0, \quad T \cdot \begin{pmatrix} o_1 \\ \vdots \\ o_n \end{pmatrix} + t \geq 0, \quad \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = A \cdot \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} + \begin{pmatrix} o_1 \\ \vdots \\ o_n \end{pmatrix} \tag{4}$$

where $(t_1, \ldots, t_n)$ are the coordinates of a tile in the tile space. In the classical polyhedron model, the lattice cannot contain parameters, since these would appear non-linearly in System (4). In our generalized model this is no problem. To obtain a tiling where the tiles are parallelepipeds (i.e., opposite sides are parallel) and the size depends on parameters, we choose linearly independent vectors $v_1, \ldots, v_n \in \mathbb{Q}^n$ which span the (unscaled) tile and use a lattice defined by

$$A = K \cdot \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \tag{5}$$

where $K = (v_1 \cdots v_n)$. The tile shape is defined by:

$$K^{-1} \cdot \begin{pmatrix} o_1 \\ \vdots \\ o_n \end{pmatrix} \geq 0, \qquad -K^{-1} \cdot \begin{pmatrix} o_1 \\ \vdots \\ o_n \end{pmatrix} + K^{-1} \begin{pmatrix} p_1 - 1 \\ \vdots \\ p_n - 1 \end{pmatrix} \geq 0 \tag{6}$$

Looking at the definition of A in Equality (5), it is easy to see that A satisfies the preconditions of Lemma 1. From that, one can deduce that tiling never leads to case distinctions.

5 Experiments

For our experiments we use a tiling example with

$$S = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix} \qquad s = \begin{pmatrix} 0 \\ 0 \\ n \end{pmatrix} \qquad K = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad A = K \cdot \begin{pmatrix} p_1 \\ p_1 \end{pmatrix}$$

where n is a (linear) parameter which determines the size of the index space, and p_1, p_2 are non-linear parameters which determine the size of the tiles. Figure 2 shows the index space, all non-empty tiles, and, as a representative, all index points inside the tile at $(0, 0)$ for $n = 24$, $p_1 = 7$, $p_2 = 4$. We have solved the following problems:

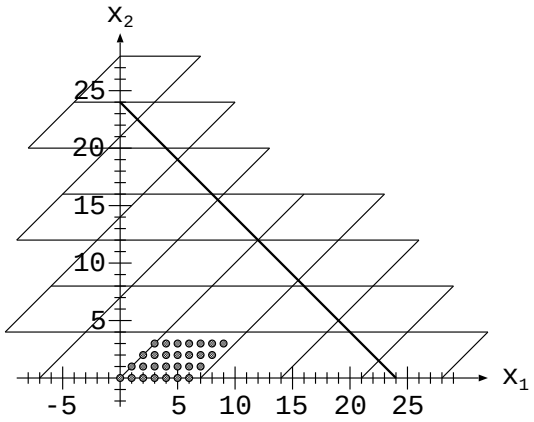


Figure 2: Tiling example

(1) Find an enumeration of the tiled index space for which the loops for t_1 and t_2 are outside and the loops for x_1 and x_2 are inside.

(2) Enumerate the communication partners (receiving tiles) for a given sending tile in the given index space for a dependence $(x_1, x_2) \mapsto (x_1 + 2, x_2 + 1)$.

Problem (1) requires the projection of the six-dimensional input system[1] onto the dimensions (t_1, t_2, x_1, x_2). Problem (2) starts with a 12-dimensional input system (six dimensions for the sender and six for the receiver) and requires the calculation of a projection to the four dimensions (t_1, t_2, t'_1, t'_2), where the receiver coordinates (t'_1, t'_2) are enumerated in dependence of the sender coordinates (t_1, t_2); explicit bounds for the sender coordinates need not be computed.

We ran the experiments on an AMD Athlon™ 1700+ processor (1467 MHz) with 512 MB RAM. We have implemented our generalized Fourier-Motzkin and Simplex algorithms in Haskell and used the Glasgow Haskell Compiler (GHC) version 6.2.1. As quantifier elimination tools we use

[1] Because of the two equalities in the system, the polyhedron defined by the (in)equalities is just four-dimensional. Our Fourier-Motzkin and Simplex implementations take advantage of these equalities to simplify the computation.

the REDLOG package [DS97] of the commercial computer algebra system REDUCE [Red] (version 3.7), the freely available tool QEPCAD [Qep], (version B 1.21) and our own implementation of a simple version of cylindrical algebraic decomposition (CAD) [ACM98]. REDUCE and QEPCAD are invoked as external tools by our implementation, i.e., there is some startup cost to pay for every invocation of them. The startup time for QEPCAD is approximately 50ms, since every call starts QEPCAD as an external process. REDUCE is only started once (as an external process) and the communication with it takes place over pipes. The overhead for communicating with REDUCE is therefore only approximately 1.5ms for every call. The CAD algorithm integrated in our system is not capable of performing a complete quantifier elimination, but it suffices to decide the logical formulas emerging during decision tree simplification. Due to its integration in our system, there is no startup time when our CAD procedure is used.

	(1)		
	tot. time	non-QE time	QE calls
REDLOG	0.5s	0.1s	35
QEPCAD	1.7s		
our CAD	0.6s		
Lemma 1	0.1s		8

	(2)		
	tot. time	non-QE time	QE calls
REDLOG	2.6s	1.9s	277
QEPCAD	14.3s		
our CAD	7.0s		
Lemma 1	2.0s		63

Table 1: Running times of our Fourier-Motzkin experiments

Tables 1 and 2 show the total running time, the time spent in the algorithm (i.e., not in calls to a quantifier elimination tool), and the number of calls to the quantifier elimination procedure of the different algorithms (Fourier-Motzkin, Simplex-based Lexmax, Quantifier elimination based Lexmax) for the two problems.

The first three rows of Table 1 show a comparison of REDLOG, QEPCAD, and our CAD implementation. In general (not only in the two examples shown here), REDLOG is the fastest quantifier elimination tool most of the time; only in cases which generate simple case distinctions (i.e., with polynomials of low degree) our CAD implementation was a bit faster due to the missig startup costs with our CAD procedure. If we take the startup costs for QEPCAD into account, we can estimate that QEPCAD's performance is comparable to REDLOG's, but for technical reasons we have not implemented a low-overhead connection to QEPCAD.

The "Lemma 1" row of Table 1 shows the results when Lemma 1 is used to eliminate quantifier elimination calls for decision tree simplification (the eight calls to the quantifier elimination are to suppress superfluous bounds during Fourier-Motzkin; REDLOG has been used to solve these eight problems).

		Lexmax	QE-Lexmax
(1)	total time	1.2s	2min 25s
	non-QE time	0.3s	–
	QE calls	409	1
	pivoting steps	22	–
(2)	total time	3.2s	heap overflow
	non-QE time	1.3s	
	QE calls	858	
	pivoting steps	34	

Table 2: Running times of our Simplex experiments

The Simplex algorithm constructs case distinctions with more complicated polynomials than Fourier-Motzkin. This leads to problems with REDLOG's quantifier elimination procedure, since it cannot handle polynomials of arbitrary degrees in the input formulas. Therefore, our implementation tries REDLOG first and, if REDLOG fails, the problem is solved with QEPCAD.

Table 2 compares an algorithm to find the lexicographic maximum ("Lexmax") based on our generalized Simplex algorithm (together with the just described heuristics for the choice of the quantifier elimination tool) to quantifier elimination with answer ("QE-Lexmax") as described in Section 3. Unfortunately, we could not find a way to formulate the problem such that quantifier elimination with answer performed comparably with the generalized Simplex algorithm.

6 Conclusion

We have outlined that quantifier elimination facilitates an extension of the classical polyhedron model to a generalized model with non-linear parameters. We have pursued two different approaches to develop algorithms for the generalized model. First, we can take an existing algorithm for the classical polyhedron model and derive an algorithm for the generalized polyhedron model from it by program transformation. In general, the results computed by such generalized algorithms are big case distinctions. Quantifier elimination can be used to reduce considerably the number of branches in the decision trees representing these case distinctions. Second, quantifier elimination can be used to directly solve some problems like finding lexicographic minima, since these problems can be described by first-order logical formulas.

The experiments on the use of quantifier elimination with answer to solve some problems directly (and some other problems [Grö03]) suggest that, in general, better performance is obtained by generalizing an existing specialized algorithm (like the Simplex algorithm or Fourier-Motzkin elimination) than by using quantifier elimination with answer.

In some special cases it is not necessary to use quantifier elimination to simplify the decision trees constructed by generalized algorithms. The case we have presented is when the coefficient matrix of an inequality system can be written as the product of two matrices with certain properties, Fourier-Motzkin elimination can be perfomed without making case distinctions. This is the case, for example, when Fourier-Motzkin elimination is applied to an inequality system describing the

tiling of an index space with parallelepipeds as tiles which are scaled by parametric factors in each dimension.

The applicability of the techniques we have presented rests on the assumption that the coefficients of the input inequality systems are (fractions of) polynomials in the parameters and that the arithmetic operations performed by the algorithms are addition, subtraction, multiplication, and division. This is necessary to guarantee that the conditions arising during simplification of the case distinctions constructed by the generalized algorithms can be decided using real quantifier elimination. As a consequence, we cannot solve "integral" problems, e.g., it is not possible to use the generalized Simplex algorithm to implement a (generalized) branch-and-bound algorithm to find integral optima, since $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ operations are not available in $\mathbb{Q}(p_1, \ldots, p_m)$ and would lead, during decision tree simplification, to formulas for which no decision procedure exists. The question of whether $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ can be introduced in the generalized model, or not, is a subject for further research.

Acknowledgements This work is supported by the BFHZ and the DAAD through exchange projects.

References

[ACM98] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical Algebraic Decompositions I: The Basic Algorithm. In Bob F. Caviness and Jeremy R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 136–151. Springer-Verlag, 1998.

[AI91] Corinne Ancourt and François Irigoin. Scanning Polyhedra with DO Loops. *Third ACM SIGPLAN Symposium on Priciples & Practice of Parallel Programming*, 26(7):39–50, July 1991.

[DS97] Andreas Dolzmann and Thomas Sturm. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31:2–9, June 1997.

[Grö03] Armin Größlinger. Extending the Polyhedron Model to Inequality Systems with Non-linear Parameters using Quantifier Elimination. Diploma thesis, Universität Passau, September 2003. `http://www.infosun.fmi.uni-passau.de/cl/arbeiten/groesslinger.ps.gz`.

[Qep] `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`.

[Red] `http://www.zib.de/Symbolik/reduce/`.