An analysis of Quantified Linear Programs

K. Subramani *
LDCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

Abstract

Quantified Linear Programming is the problem of checking whether a polyhedron specified by a linear system of inequalities is non-empty, with respect to a specified quantifier string. Quantified Linear Programming subsumes traditional Linear Programming, since in traditional Linear Programming, all the program variables are existentially quantified (implicitly), whereas, in Quantified Linear Programming, a program variable may be existentially quantified or universally quantified over a continuous range. On account of the alternation of quantifiers in the specification of a Quantified Linear Program (QLP), this problem is non-trivial. QLPs represent a class of Declarative Constraint Logic Programs (CLPs) that are extremely rich in their expressive power. The complexity of Quantified Linear Programming for arbitrary constraint matrices is unknown. In this paper, we show that polynomial time decision procedures exist for the case in which the constraint matrix is Totally Unimodular (TUM). We also provide a taxonomy of Quantified Linear Programs, based on the structure of the quantifier string and discuss the computational complexities of the constituent classes.

1 Introduction

Quantified Linear Programming was a term coined in [Sub00], in discussions with [Joh] and [Imm] to describe linear programming problems in which one or more of the variables are universally quantified over some domain and the rest of the variables are existentially quantified in the usual sense. Quantified Linear Programs (QLPs) are extremely expressive in that they can be used to express a number of schedulability specifications in real-time systems [Sub02]. In a typical real-time programming language such as the Maruti Programming Language (MPL) [MKAT92, LTCA89], it is necessary to specify constraints between various tasks that make up the schedule; constraints such as relative timing constraints cannot be captured through precedence graphs and necessitate the use of QLPs [Sub02]. While no hardness result is known for the general problem, in this paper we show that the recognition problem is in `PSPACE` . The main contributions of this paper are as follows:

1. Description of the QLP framework in general form
2. Development of new proof techniques to analyze the decidability of QLPs,
3. Development of a new sufficiency condition guaranteeing the polynomial time convergence of a given QLP,
4. Development of a taxonomy scheme for QLPs, based on their quantifier strings.

The rest of this paper is organized as follows: Section §2 formally describes the Quantified Linear Programming problem. The motivation for our analysis and related work are discussed in Section §3. Section §4 describes an algorithm for the Quantified Linear Programming problem, while Section §5 proves the correctness of the same. In Section §6, we analyze the complexity of the algorithm discussed in Section §4, for the case in which the constraint matrix is totally unimodular (TUM). In Section §7, we provide a taxonomy of QLPs based on the structure of

*This research has been supported in part by the Air Force Office of Scientific Research under Grant F49620-02-1-0043

the quantifier string; we show that the class of **E-QLPs** is solvable in polynomial time (using a modification of the algorithm in Section §4), regardless of the structure of the constraint matrix and that the class of **F-QLPs** is `coNP-complete`, in general. We conclude in Section §8 by summarizing our results and discussing a number of problems for future research.

2 Problem Statement

We are interested in deciding the following query:

$$\mathbf{G} : \exists x_1 \in [a_1, b_1]\ \forall y_1 \in [l_1, u_1]\ \exists x_2 \in [a_2, b_2]\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_n \in [a_n, b_n]\ \forall y_n \in [l_n, u_n] \\ \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\ ,\ \vec{\mathbf{x}} \geq \vec{\mathbf{0}}? \tag{1}$$

where

- $\mathbf{A}$ is an $m \times 2 \cdot n$ matrix called the constraint matrix,
- $\vec{\mathbf{x}}$ is a $n-$vector, representing the control variables (these are existentially quantified)
- $\vec{\mathbf{y}}$ is a $n-$vector, representing the variables that can assume values within a pre-specified range; i.e., component y_i has a lower bound of l_i and an upper bound of u_i (these are universally quantified);
- $\vec{\mathbf{b}}$ is an $m-$vector

The pair $(\mathbf{A}, \vec{\mathbf{b}})$ is called the *Constraint System*. Without loss of generality, we assume that the quantifiers are strictly alternating, since we can always add dummy variables (and constraints, if necessary) without affecting the correctness or complexity of the problem.

The string $\exists x_1 \in [a_1, b_1]\ \forall y_1 \in [l_1, u_1]\ \exists x_2 \in [a_2, b_2]\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_n \in [a_n, b_n]\ \forall y_n \in [l_n, u_n]$ is called the quantifier string of the given QLP and is denoted by $\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})$. The length of the quantifier string, is denoted by $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})|$ and it is equal to the dimension of $\mathbf{A}$. Note that the range constraints on the existentially quantified variables can be included in the constraint matrix $\mathbf{A}$ ($x_i \in [a_i, b_i]$ can be written as $a_i \leq x_i,\ x_i \leq b_i$) and thus the generic QLP can be represented as:

$$\mathbf{G} : \exists x_1\ \forall y_1 \in [l_1, u_1]\ \exists x_2\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_n\ \forall y_n \in [l_n, u_n] \\ \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}} \tag{2}$$

It follows that the QLP problem can be thought of as checking whether a polyhedron described by a system of linear inequalities $(\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}})$ is non-empty vis-a-vis the specified quantifier string (say $\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})$). The pair $< \mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}}), (\mathbf{A},\ \vec{\mathbf{b}}) >$ is called a *Parametric Polytope*. In other words, Quantified Linear Programming is concerned with checking the non-emptiness of Parametric Polytopes, just as traditional linear programming is concerned with checking the non-emptiness of simple polytopes. For the rest of this paper, we shall assume that the generic QLP has the form described by System (2), so that the analysis is simplified. Accordingly, we observe that in a QLP, the dimension of the constraint matrix $\mathbf{A}$ and hence the length of the quantifier string is always even.

2.1 Constraint Satisfaction and Model Verification

Definition 2.1 *Let* $\mathbf{G} =< \mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}}), (\mathbf{A},\ \vec{\mathbf{b}}) >$ *represent an arbitrary Parametric Polytope (QLP) in the form specified by System (2). We say that* $\mathbf{G}$ *is true or non-empty if there exists an* $x_1 \in \Re$*, such that for all* $y_1 \in [l_1, u_1]$ *(which could depend upon* x_1*), there exists an* $x_2 \in \Re$ *(which could depend upon* y_1*), there exists a* $y_2 \in [l_2, u_2]$ *(which could depend upon* x_1 *and* x_2*) and so on such that* $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$*, where* $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_n]^T$ *and* $\vec{\mathbf{y}} = [y_1, y_2, \ldots, y_n]^T$*.*

In order to better understand Definition (2.1), we resort to the following 2-person game. Let $\mathbf{X}$ denote the existential player and $\mathbf{Y}$ denote the Universal player. The game is played in a sequence of $2 \cdot n$ rounds, with $\mathbf{X}$ making his i^{th} move, x_i, in round $2 \cdot i - 1$ and $\mathbf{Y}$ making his i^{th} move, y_i, in round $2 \cdot i$. The initial constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ is referred to as the initial board configuration. The following conventions are followed in the game:

1. $x_i, y_i \in \Re,\ y_i \in [l_i, u_i]\ i = 1, 2, \ldots, n$,
2. The moves are strictly alternating, i.e., $\mathbf{X}$ makes his i^{th} move before $\mathbf{Y}$ makes his i^{th} move, before $\mathbf{X}$ makes his $(i+1)^{th}$ move and so on,
3. When either player makes a move, the configuration of the board changes; for instance, suppose that $\mathbf{X}$ makes the first move as 5. The current configuration is then transformed from $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ to $\mathbf{A}' \cdot [\vec{\mathbf{x}'}\ \ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq \vec{\mathbf{b}'}$, where $\mathbf{A}'$ is obtained from $\mathbf{A}$, by dropping the first column, $\vec{\mathbf{x}'} = [x_2, x_3, \ldots, x_n]^T$ and $\vec{\mathbf{b}'} = \vec{\mathbf{b}} - 5 \cdot \vec{\mathbf{a_1}}$, with $\vec{\mathbf{a_1}}$ denoting the first column of $\mathbf{A}$.
4. The i^{th} move made by $\mathbf{X}$, viz., x_i may depend upon the current board configuration as well as the first $(i-1)$ moves made by $\mathbf{Y}$; likewise, y_i may depend upon the current board configuration and the first i moves made by $\mathbf{X}$.
5. Let $\vec{\mathbf{x_1}}$ denote the *numerical* vector of the n moves made by $\mathbf{X}$; $\vec{\mathbf{y_1}}$ is defined similarly. If $\mathbf{A} \cdot [\vec{\mathbf{x_1}}\ \ \vec{\mathbf{y_1}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$, then $\mathbf{X}$ is said to have won the game; otherwise, the game is a win for $\mathbf{Y}$. It is important to note that the game as described above is non-deterministic in nature, in that we have not specified *how* $\mathbf{X}$ and $\mathbf{Y}$ make their moves. Further, if it is possible for $\mathbf{X}$ to win the game, then he will make the correct sequence of moves; likewise, if $\mathbf{X}$ cannot win the game, then corresponding to every sequence of moves that he makes, $\mathbf{Y}$ has a corresponding sequence of moves to ensure that at least one constraint in the constraint system is violated. (See [Pap94, HO02].)
6. From the above discussion, it is clear that the moves made by $\mathbf{X}$ will have the following form:

$$\vec{\mathbf{x}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots, y_{n-1})]^T \tag{3}$$

where c_1 is a constant and $x_i = f_{i-1}(y_1, y_2, \ldots, y_{i-1})$ captures the dependence of x_i on the first $(i-1)$ moves of $\mathbf{Y}$.

Likewise, the moves made by $\mathbf{Y}$ have the following form:

$$\vec{\mathbf{y}} = [g_1(x_1), g_2(x_1, x_2), \ldots, g_n(x_1, x_2, \ldots, x_n)]^T \tag{4}$$

The $f_i()$ and $g_i()$ are Skolem functions.

The following phrases are equivalent and will be used interchangeably for the rest of this paper:

(a) $\vec{\mathbf{x}}$ is a solution vector of $\mathbf{G}$,
(b) $\vec{\mathbf{x}}$ is a model for $\mathbf{G}$,
(c) $\vec{\mathbf{x}}$ satisfies $\mathbf{G}$,
(d) $\vec{\mathbf{x}}$ is appropriate for $\mathbf{G}$,
(e) $\vec{\mathbf{x}} \in \mathbf{G}$,
(f) $\vec{\mathbf{x}}$ is a winning strategy for $\mathbf{G}$.

Note that corresponding to any game (QLP) $\mathbf{G}$, either the Existential player ($\mathbf{X}$) has a winning strategy against all strategies employed by the Universal player ($\mathbf{Y}$) or (mutually exclusively) the Universal player ($\mathbf{Y}$) has a winnging strategy, against all strategies employed by the Existential player. However a specific strategy $\vec{\mathbf{x}}$, for $\mathbf{X}$ may or may not be winning; if it is not a winning strategy for $\mathbf{X}$, then $\mathbf{Y}$ has a winning strategy $\vec{\mathbf{y}}(\vec{\mathbf{x}})$, *corresponding to* $\vec{\mathbf{x}}$.

Suppose that a solution vector $\vec{\mathbf{x}}$ in the form described by Equation (3) is given; then the above model verification algorithm requires an **infinite precision** Alternating Turing Machine, since there is no guarantee that the guessed values have polynomial size or even finite size! We shall show in Section §5 that the space required for each guess is polynomial in the sizes of $\mathbf{A}$ and $\vec{\mathbf{b}}$, if $\mathbf{A}$ and $\vec{\mathbf{b}}$ are rational; thus the Alternating Turing Machine verifying the appropriateness of $\vec{\mathbf{x}}$ for $\mathbf{G}$ runs in polynomial time.

3 Motivation and Related Work

Quantified Linear Programs represent a rich language that is ideal for expressing schedulability specifications in real-time scheduling [Sak94, GPS95, Cho97, Cho00, Sub00, Sub01], although the term Quantified Linear Programming is being used for the first time in this paper. The scheduling QLPs that are considered in the above citations have two restrictions, which were both used in the development of polynomial time algorithms, viz. (a) a total ordering on the Existentially quantified variables, i.e., $x_1 \leq x_2 \leq \ldots \leq x_n$, and (b) all constraints are "standard", i.e., strict difference constraints. Note that difference constraints are constraints of the form $x_i + y_i \leq x_j + y_j + c$ and hence can be represented by a constraint graph [Sub01]. Although Restriction (a) is a component of the real-time scheduling problems that were studied, it must be noted that it does make the problem easy. *In our framework, there is no order on the variables, other than what can be deduced from the constraints.* Further, the constraints we consider are far more general, in that the restriction that we place on the constraint matrix is total unimodularity, as opposed to the restriction that the constraint system is representable as a network graph. Observe that difference constraints form a subset of the class of totally unimodular constraint systems [Sch87, NW99]. The work in this paper represents the first attempt to study Quantified Linear Programming as an independent mathematical programming paradigm.

The connections between logic and optimization have been addressed at great depth in [Hoo00] and [CH99]. Constraint Logic Programming resulted from the combination of the declarative aspects of First-order Logic and procedural aspects of Programming [CD00, Col86]. Quantified Boolean formulae have been studied extensively from both the theoretical and the practical perspectives, within the Artificial Intelligence community [CGS98, CG$^+$97]. We remark that all the above paradigms are essentially discrete; to the best of our knowledge, our work represents the first effort to consider logic programming in conjunction with real-valued (continuous) variables. At their heart, QLPs represent a subset of Constraint Logic Programs, that combine the expressiveness of First Order Logic with the power of Linear Programming. It is worth noting that our framework subsumes the Bi-level Programming problems discussed in [BA93, VC94].

4 The Quantifier Elimination Algorithm

Algorithm (4.1) represents our strategy to decide Query (1). It proceeds by eliminating one variable at a time, until we are left with precisely one variable, viz. x_1. If there is a feasible choice for x_1, then the QLP is feasible, since we can then work backwards to build a solution for each of the variables x_2 to x_n; otherwise it is not. A variation of this algorithm (for restricted constraint classes) was proposed in [GPS95] and analyzed; our proof explicitly uses techniques (such as the notion of non-determinism in the form of 2-person games) that make it significantly more general and is completely different from all previous work.

The chief idea is that the constraint matrix $\mathbf{A}$ undergoes a series of transformations through variable elimination techniques; in the following section, we shall argue that each of these transformations is solution preserving.

Observe that given a variable x_i, every constraint involving x_i must be representable as a constraint of the form $x_i \leq ()$ or (mutually exclusively) $x_i \geq ()$, where the $()$ notation represents the function that is created by transposing the rest of the variables to the Right Hand Side. For instance, consider the constraint $l_1 : 2 \cdot x_1 + y_{23} \leq 8$. Clearly l_1 can be written in the form $x_1 \leq 4 - \frac{1}{2} \cdot y_{23}$; in this case, the function $4 - \frac{1}{2} \cdot y_{23}$ is represented by $()$. When a variable (existential or universal) is eliminated the (possibly) new constraints that result are called *derived constraints*. These derived constraints could: (a) Make an existing constraint redundant, (b) be redundant themselves, or (c) Create an inconsistency. In cases (a) and (b), the appropriate constraints are eliminated through Prune-Constraints(), whereas in case (c), the system is declared infeasible (through Check-Inconsistency()).

Function QLP-DECIDE $(\mathbf{A}, \vec{\mathbf{b}})$

1: $\mathbf{A'_{n+1}} = \mathbf{A}; \vec{\mathbf{b}}'_{\mathbf{n+1}} = \vec{\mathbf{b}}$
2: **for** $(i = n$ **down to** $2)$ **do**
3: $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}) =$ ELIM-UNIV-VARIABLE $(\mathbf{A'_{i+1}}, \vec{\mathbf{b}}'_{\mathbf{i+1}}, y_i, l_i, u_i)$
4: $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}) =$ ELIM-EXIST-VARIABLE $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}, x_i)$
5: **if** (CHECK-INCONSISTENCY()) **then**
6: **return (false)**
7: **end if**
8: PRUNE-CONSTRAINTS()
9: **end for**
10: $(\mathbf{A'_1}, \vec{\mathbf{b}}'_{\mathbf{1}})$ =ELIM-UNIV-VARIABLE $(\mathbf{A'_2}, \vec{\mathbf{b}}'_{\mathbf{2}}, y_1, l_1, u_1)$
11: {After the elimination of y_1, the original system is reduced to a one-variable system, i.e., a series of intervals on the x_1-axis. We can therefore check whether this system provides an interval or declares an inconsistency. An interval results if after the elimination of redundant constraints, we are left with $x_1 \geq a, x_1 \leq b, a \leq b$; an inconsistency results if we are left with $x_1 \geq a, x_1 \leq b, b < a$.}
12: **if** $(a \leq x_1 \leq b, \quad a, b \geq 0, \quad a \leq b)$ **then**
13: `System is feasible`
14: **return**
15: **else**
16: `System is infeasible`
17: **return**
18: **end if**

Algorithm 4.1: A Quantifier Elimination Algorithm for deciding Query **G**

Function ELIM-UNIV-VARIABLE $(\mathbf{A}, \vec{\mathbf{b}}, y_i, l_i, u_i)$

1: {Every constraint involving the variable y_i can be re-written in the form $y_i \leq ()$ or (exclusively) $y_i \geq ()$, i.e., in a way that the coefficient of y_i is $+1$.}
2: Substitute $y_i = l_i$ in each constraint that can be written in the form $y_i \geq ()$
3: Substitute $y_i = u_i$ in each constraint that can be written in the form $y_i \leq ()$
4: Create the new coefficient matrix $\mathbf{A}'$ and the new vector $\vec{\mathbf{b}}'$ after the requisite manipulations
5: **return**$(\mathbf{A}', \vec{\mathbf{b}}')$

Algorithm 4.2: Eliminating Universally Quantified variable $y_i \in [l_i, u_i]$

Function ELIM-EXIST-VARIABLE $(\mathbf{A}, \vec{\mathbf{b}}, x_i)$

1: Form the set $L_{\leq}$ of every constraint that can be written in the form $x_i \leq ()$. If $x_i \leq m_j$ is a constraint in $(\mathbf{A}, \vec{\mathbf{b}})$, m_j is added to $L_{\leq}$.
2: Form the set $L_{\geq}$ of every constraint that can be written in the form $x_i \geq ()$. Corresponding to the constraint $x_i \geq n_k$ of $(\mathbf{A}, \vec{\mathbf{b}})$, n_k is added to $L_{\geq}$.
3: Form the set $L_{=}$ of every constraint that does not contain x_i
4: $\mathcal{L} = \phi$.
5: **for** each constraint $m_j \in L_{\leq}$ **do**
6: **for** each constraint $n_k \in L_{\geq}$ **do**
7: Create the new constraint $l_{kj} : n_k \leq m_j$; $\mathcal{L} = \mathcal{L} \cup l_{kj}$.
8: **end for**
9: **end for**
10: Create the new coefficient matrix $\mathbf{A}'$ and the new vector $\vec{\mathbf{b}}'$, to include all the constraints in $\mathcal{L} = \mathcal{L} \cup L_{=}$, after the requisite manipulations.
11: **return**$(\mathbf{A}', \vec{\mathbf{b}}')$

Algorithm 4.3: Eliminating Existentially Quantified variable x_i

Example (1): Let us say that the universally quantified variable $y_5 \in [3,5]$ is eliminated from the constraint set: $l_1 : y_5 + x_1 \leq 14$, $l_2 : x_1 \leq 12$. Using Algorithm (4.2), we get the constraints $l_1 : x_1 \leq 9$, $l_2 : x_1 \leq 12$. Clearly l_2 is redundant and can be eliminated.

Example (2): Let us say that the existentially quantified variable x_4 is eliminated from the constraint set: $l_1 : x_4 \leq 7$, $l_2 : -x_4 \leq -8$. We get the constraint $0 \leq -1$, which is clearly an inconsistency.

5 Correctness

We use induction on the length of the quantifier string of $\mathbf{G}$, i.e., $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})|$ and hence on the dimension of $\mathbf{A}$. Observe that as defined in System (2), $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})|$ is always even.

Lemma 5.1 *Algorithm (4.1) correctly decides feasibility of a QLP having $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2$*

Proof: Observe that a QLP in which $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2$, must be of the form:

$$\exists x_1 \forall y_1 \in [l_1, u_1] \ \vec{\mathbf{g}} \cdot x_1 + \vec{\mathbf{h}} \cdot y_1 \leq \vec{\mathbf{b}} \tag{5}$$

When the QLP represented by System (5) is input to Algorithm (4.1), Step (10 :) is executed first. The ELIM-UNIV-VARIABLE() procedure converts System (5) into a QLP of the form

$$\exists x_1 \ \ \vec{\mathbf{g}} \cdot x_1 \leq \vec{\mathbf{b'}}, \tag{6}$$

where

$$\begin{aligned} b'_i &= b_i - u_1 \cdot h_i, \ \ if \ h_i > 0 \\ &= b_i - l_1 \cdot h_i, \ \ if \ h_i < 0 \\ &= b_i, \ \ otherwise \end{aligned}$$

Note that System (6) represents an intersection of intervals that are closed on one side and extending infinitely on the other; the i^{th} constraint in System (6), is either of the form $x_1 \leq c_i$ (when $g_i > 0$) or (mutually exclusively) of the form $x_1 \geq c_i$ (when $g_i < 0$), where $c_i = \frac{b'_i}{g_i}$. Let $a = \max_{c_i}\{x_1 \geq c_i\}$ and $b = \min_{c_i}\{x_1 \leq c_i\}$. It follows that applying ELIM-EXIST-VARIABLE() to System (6) results in an interval of the form $a \leq x_1 \leq b$, if $a \leq b$ or an inconsistency otherwise. Let us say that System (6) has a solution $x_1 = c_1$; we need to show that System (5) has a solution. Pick any constraint $p_i : g_i \cdot x_1 + h_i \cdot y_1 \leq b_i$ in System (5). Assume that $h_i > 0$. Since c_1 is a solution to System (6), we know that

$$\begin{aligned} g_i \cdot c_1 &\leq b'_i \\ &\leq b_i - u_1 \cdot h_i, \ \ since \ \ h_i > 0 \\ \Rightarrow g_i \cdot c_1 &\leq b_i - l_1 \cdot h_i, \ \ since \ l_1 \leq u_1 \\ \Rightarrow \forall y_1 \in [l_1, u_1] \ \ g_i \cdot c_1 &\leq b_i - l_1 \cdot h_i \end{aligned}$$

It follows that

$$\exists x_1 \forall y_1 \in [l_1, u_1] \ \ g_i \cdot x_1 + h_i \cdot y_1 \leq b_i$$

is true. Parallel arguments hold for the cases when $h_i = 0$ and $h_i < 0$. Since the constraint p_i was picked arbitrarily, it follows that $x_1 = c_1$ satisfies all the constraints of System (5), i.e., it is a valid solution for System (5).

Now let us consider the case when System (5) has a solution, say $x_1 = c_1$. Pick an arbitrary constraint $p_i : g_i \cdot x_1 \leq b'_i$ in System (6). The corresponding constraint in System (5) is of the form $p'_i : g_i \cdot x_1 + h_i \cdot y_1 \leq b_i$ Since $x_1 = c_1$ satisfies p'_i, we have

$$\begin{aligned} g_i \cdot c_1 + h_i \cdot l_1 &\leq b_i \\ g_i \cdot c_1 + h_i \cdot u_1 &\leq b_i \end{aligned}$$

Hence the corresponding constraint in System (6), i.e., p_i is satisfied, since the RHS of p_i is either $b_i - h_i \cdot l_1$ or $b_i - h_i \cdot u_1$. Since the constraint p_i was chosen arbitrarily, it follows that $x_1 = c_1$ satisfies every constraint in System (6).

We have thus proved the base case of the induction. □

We now assume that Algorithm (4.1) decides correctly decides QLPs, when $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2 \cdot (n-1)$. Lemma (5.2) and Lemma (5.3) consider a QLP, where $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2 \cdot n$.

Lemma 5.2 *Let*

$$\mathbf{L} : \exists x_1 \, \forall y_1 \in [l_1, u_1] \, \exists x_2 \, \forall y_2 \in [l_2, u_2] \; \ldots \exists x_n \forall y_n \in [l_n, u_n] \;\; \mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

and

$$\mathbf{R} : \exists x_1 \, \forall y_1 \in [l_1, u_1] \, \exists x_2 \, \forall y_2 \in [l_2, u_2] \; \ldots \exists x_{n-1} \forall y_{n-1} \in [l_{n-1}, u_{n-1}] \exists x_n \;\; \mathbf{A'} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y'}}]^{\mathbf{T}} \leq \vec{\mathbf{b'}},$$

where $\vec{\mathbf{y'}} = [y_1, y_2, \ldots, y_{n-1}]^T$ *and* $(\mathbf{A'}, \vec{\mathbf{b'}})$ *is the constraint system that results after calling Algorithm (4.2) on* $(\mathbf{A}, \vec{\mathbf{b}}, y_n, l_n, u_n)$. *Then* $\mathbf{L} \Leftrightarrow \mathbf{R}$.

Proof: Let $(\mathbf{X_L}, \; \mathbf{Y_L})$ denote the Existential and Universal players of game $\mathbf{L}$ and let $(\mathbf{X_R}, \; \mathbf{Y_R})$ denote the Existential and Universal Players of game $\mathbf{R}$ respectively. Let

$$\vec{\mathbf{x_L}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots y_{n-1})]^T$$

be a model for $\mathbf{L}$. We shall show that $\vec{\mathbf{x_L}}$ is also a model for $\mathbf{R}$.

Let us assume the contrary and say that $\vec{\mathbf{x_L}}$ is not a model for $\mathbf{R}$. From the discussion in Section §2.1, we know that $\mathbf{Y_R}$, has a winning strategy $\vec{\mathbf{y_R}}(\vec{\mathbf{x_L}})$, corresponding to the strategy $\vec{\mathbf{x_L}}$. Note that $\vec{\mathbf{y_R}}$ has $n-1$ components, i.e., player $\mathbf{Y_R}$ makes only $n-1$ moves. (We could add a column $\vec{\mathbf{0}}$ to $\mathbf{A'}$, so that $\mathbf{Y_R}$ makes all n moves.)

Consider the complete set of moves made by $\mathbf{X_R}$ and $\mathbf{Y_R}$ to decide $\mathbf{R}$, with $\mathbf{X_R}$ playing, as per $\vec{\mathbf{x_L}}$ and $\mathbf{Y_R}$ playing, as per $\vec{\mathbf{y_R}}(\vec{\mathbf{x_L}})$; let $\vec{\mathbf{x_1}}^{\mathbf{R}}$ and $\vec{\mathbf{y_1}}^{\mathbf{R}}$ denote the corresponding numerical vectors. Since $\vec{\mathbf{y_R}}(\vec{\mathbf{x_L}})$ represents a winning strategy, there is at least one constraint in the system $\mathbf{A'} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y'}}]^{\mathbf{T}} \leq \vec{\mathbf{b'}}$ of QLP $\mathbf{R}$ that is violated. Let $p'_i : \vec{\mathbf{a'_i}} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y'}}]^{\mathbf{T}} \leq b'_i$ represent a violated constraint; we thus have $\vec{\mathbf{a'_i}} \cdot [\vec{\mathbf{x_1}}^{\mathbf{R}} \;\; \vec{\mathbf{y_1}}^{\mathbf{R}}]^{\mathbf{T}} > b'_i$. Let p_i denote the corresponding constraint in the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ of QLP $\mathbf{L}$. Consider the following two cases:

1. p_i does not contain y_n - In this case, the constraints p_i and p'_i are identical. Consequently, the violation of p'_i implies the violation of p_i. It follows that $\vec{\mathbf{y_R}}(\vec{\mathbf{x_L}})$ is also a winning strategy for $\mathbf{Y_L}$, contradicting the existence of $\vec{\mathbf{x_L}}$ as a model for $\mathbf{L}$.

2. p_i contains y_n - We assume that p_i can be written in the form $y_n \geq ()$. Thus p'_i was obtained from p_i by substituting $y_n = l_n$. Now consider the strategy

   $$\vec{\mathbf{y_L}} = [\vec{\mathbf{y_R}}(\vec{\mathbf{x_L}}), l_n]^T$$

   for the Universal player $\mathbf{Y_L}$ of the QLP $\mathbf{L}$. Since the constraint p'_i is violated, $\vec{\mathbf{y_L}}$ causes a violation of p_i as well, establishing that $\vec{\mathbf{y_L}}$ is a winning strategy for $\mathbf{Y_L}$ and contradicting the hypothesis. A parallel argument holds for the case when p_i can be written in the form $y_n \leq ()$.

It follows that $\vec{\mathbf{x_L}}$ is also a model for $\mathbf{R}$; we have thus shown that $\vec{\mathbf{x}} \in \mathbf{L} \Rightarrow \vec{\mathbf{x}} \in \mathbf{R}$.

We now proceed to prove the converse. Let

$$\vec{\mathbf{x_R}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots, y_{n-1})]^T$$

be a model for $\mathbf{R}$. We shall show that it is also a model for $\mathbf{L}$. Assume the contrary and say that it is not a model for $\mathbf{L}$. It follows that $\mathbf{Y_L}$ has a winning strategy $\vec{\mathbf{y_L}}(\vec{\mathbf{x_R}})$, corresponding to $\vec{\mathbf{x_R}}$.

Consider the complete set of moves made by $\mathbf{X_L}$ and $\mathbf{Y_L}$ to decide $\mathbf{L}$, with $\mathbf{X_L}$ playing, as per $\vec{\mathbf{x_R}}$ and $\mathbf{Y_L}$ playing, as per $\vec{\mathbf{y_L}}(\vec{\mathbf{x_R}})$; let $\vec{\mathbf{x_1}}^{\mathbf{L}}$ and $\vec{\mathbf{y_1}}^{\mathbf{L}}$ denote the corresponding numerical vectors. Since $\vec{\mathbf{y_L}}(\vec{\mathbf{x_R}})$ represents a winning strategy, there is at least one constraint in the system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ of QLP $\mathbf{L}$ that is violated. Let $p_i : \vec{\mathbf{a_i}} \cdot [\vec{\mathbf{x}}\ \vec{\mathbf{y}}]^{\mathbf{T}} \leq b_i$ represent a violated constraint; we thus have $\vec{\mathbf{a_i}} \cdot [\vec{\mathbf{x_1}}^{\mathbf{L}}\ \vec{\mathbf{y_1}}^{\mathbf{L}}]^{\mathbf{T}} > b_i$. Let p'_i denote the corresponding constraint in the constraint system $\mathbf{A'} \cdot [\vec{\mathbf{x}}\ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq \vec{\mathbf{b}'}$ of QLP $\mathbf{R}$. Consider the following two cases:

1. p_i does not contain y_n - In this case, the constraints p_i and p'_i are identical. Consequently, the violation of p_i implies the violation of p'_i. It follows that $\vec{\mathbf{y_L}}(\vec{\mathbf{x_R}})$ is also a winning strategy for $\mathbf{Y_R}$, contradicting the existence of $\vec{\mathbf{x_R}}$ as a model for $\mathbf{R}$.

2. p_i contains y_n - We assume that p_i can be written in the form $y_n \geq ()$. Thus p'_i was obtained from p_i by substituting $y_n = l_n$. Since $\vec{\mathbf{x_R}}$ ensured that this constraint was satisfied for $y_n = l_n$, it follows that this constraint will be satisfied for all values of $y_n \in [l_n, u_n]$. In other words, there is no legal move that $\mathbf{Y_L}$ can make that would cause this constraint to be violated. Thus, p_i, cannot exist and since p_i was chosen arbitrarily, it follows that no such constraint can exist. A parallel argument works for the case in which p_i is of the form $y_n \leq ()$. Hence, $\mathbf{Y_L}$ does not have a winning strategy, corresponding to $\vec{\mathbf{x_R}}$, i.e., $\vec{\mathbf{x_R}} \in \mathbf{L}$.

We have thus shown that $\vec{\mathbf{x}} \in \mathbf{R} \Rightarrow \vec{\mathbf{x}} \in \mathbf{L}$. The lemma follows. □

Lemma 5.3 *Let*

$$\mathbf{L} : \exists x_1\ \forall y_1 \in [l_1, u_1]\ \exists x_2\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_n\ \ \mathbf{A} \cdot [\vec{\mathbf{x}}\ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

and

$$\mathbf{R} : \exists x_1\ \forall y_1 \in [l_1, u_1]\ \exists x_2\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_{n-1} \forall y_{n-1} \in [l_{n-1}, u_{n-1}]\ \ \mathbf{A'} \cdot [\vec{\mathbf{x'}}\ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq \vec{\mathbf{b}'},$$

where $\vec{\mathbf{y}'} = [y_1, y_2, \ldots, y_{n-1}]^T$, $\vec{\mathbf{x}'} = [x_1, x_2, \ldots, x_{n-1}]^T$ *and* $(\mathbf{A'}, \vec{\mathbf{b}'})$ *is the constraint system that results after calling Algorithm (4.3) on* $(\mathbf{A}, \vec{\mathbf{b}}, x_n)$*. Then* $\mathbf{L} \Leftrightarrow \mathbf{R}$.

Proof: As in Lemma (5.2), we let $(\mathbf{X_L},\ \mathbf{Y_L})$ denote the Existential and Universal players of game $\mathbf{L}$ and let $(\mathbf{X_R},\ \mathbf{Y_R})$ denote the Existential and Universal Players of game $\mathbf{R}$ respectively.

Let

$$\vec{\mathbf{x_L}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots y_{n-1})]^T \in \mathbf{L}$$

be a model for $\mathbf{L}$.

We shall show that

$$\vec{\mathbf{x_R}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-2}(y_1, y_2, \ldots y_{n-2})]^T$$

is a model for $\mathbf{R}$. Note that $\vec{\mathbf{x_R}}$ has been obtained from $\vec{\mathbf{x_L}}$ by truncating the last component.

Let us assume the contrary and say that $\vec{\mathbf{x_R}}$ is not a model for $\mathbf{R}$. It follows that $\mathbf{Y_R}$ has a winning strategy $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$, corresponding to $\vec{\mathbf{x_R}}$. Consider the complete set of moves made by $\mathbf{X_R}$ and $\mathbf{Y_R}$ to decide $\mathbf{R}$, with $\mathbf{X_R}$ playing as per $\vec{\mathbf{x_R}}$ and $\mathbf{Y_R}$ playing as per $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$; let $\vec{\mathbf{x_1}}^{\mathbf{R}}$ and $\vec{\mathbf{y_1}}^{\mathbf{R}}$ be the corresponding nnumerical vectors. Likewise, consider the complete set of moves made by $\mathbf{X_L}$ and $\mathbf{Y_L}$ to decide game $\mathbf{L}$, with $\mathbf{X_L}$ playing as per $\vec{\mathbf{x_L}}$; let $\vec{\mathbf{x_1}}^{\mathbf{L}}$ and $\vec{\mathbf{y_1}}^{\mathbf{L}}$ be the corresponding numerical vectors.

Since $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$ is a winning strategy, at least one constraint in the system $\mathbf{A'} \cdot [\vec{\mathbf{x'}}\ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq \vec{\mathbf{b}'}$ is violated. Let $p_i : \vec{\mathbf{a'_i}} \cdot [\vec{\mathbf{x'}}\ \vec{\mathbf{y}'}]^{\mathbf{T}} \leq b'_i$ represent a violated constraint; we thus have $\vec{\mathbf{a'_i}} \cdot [\vec{\mathbf{x_1}}^{\mathbf{R}}\ \vec{\mathbf{y_1}}^{\mathbf{R}}]^{\mathbf{T}} > b'_i$. Note that $\vec{\mathbf{x_1}}^{\mathbf{R}}$, $\vec{\mathbf{y_1}}^{\mathbf{R}}$ and $\vec{\mathbf{y_1}}^{\mathbf{L}}$ are $(n-1)$ dimensional vectors, while $\vec{\mathbf{x_1}}^{\mathbf{L}}$ is an n-dimensional vector. We use the notation $\vec{\mathbf{a_i}}(n)$ to indicate the n^{th} element of the vector $\vec{\mathbf{a_i}}$.

We consider the following 2 cases:

1. p_i appears in identical form in $\mathbf{L}$ - We first rewrite p_i as: $\vec{\mathbf{g'_i}} \cdot \vec{\mathbf{x'_i}} + \vec{\mathbf{h'_i}} \cdot \vec{\mathbf{y'_i}} \leq b'_i$. Let p'_i be the constraint in $\mathbf{L}$ that corresponds to p_i. Observe that p'_i is constructed from p_i as follows: $p'_i : \vec{\mathbf{g_i}} \cdot \vec{\mathbf{x}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y}'} \leq b_i$, where $b_i = b'_i$, $\vec{\mathbf{g_i}} = [\vec{\mathbf{g'_i}},\ 0]^T$, $\vec{\mathbf{h_i}} = \vec{\mathbf{h'_i}}$, $\vec{\mathbf{x}} = [\vec{\mathbf{x}'},\ x_n]^T$.

Since $\vec{\mathbf{x_L}}$ is a model for $\mathbf{L}$, it is a winning strategy for $\mathbf{X_L}$ and hence $\vec{\mathbf{a_i}} \cdot [\vec{\mathbf{x_1}}^{\mathbf{L}} \ \vec{\mathbf{y_1}}^{\mathbf{L}}]^{\mathbf{T}} \leq b_i$, where $\vec{\mathbf{a_i}} = [\vec{\mathbf{g_i}} \ \vec{\mathbf{h_i}}]^{\mathbf{T}}$. Thus, p_i (or more correctly p_i') is satisfied in $\mathbf{L}$, irrespective of the guess made for x_n by $\mathbf{X_L}$. It follows that if p_i is violated in $\mathbf{R}$, then $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$ is also a winning strategy for $\mathbf{Y_L}$, contradicting the existence of $\vec{\mathbf{x_L}}$ as a model for $\mathbf{L}$.

2. p_i was created by the fusion of two constraints $l_i : m_i \leq x_n$ and $l_j : x_n \leq n_j$, in $\mathbf{L}$ to get $m_i \leq n_j$, as per Algorithm (4.3) - We rewrite the constraint l_i as $\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x'}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y'}} + b_i' \leq x_n$ and the constraint l_j as $x_n \leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x'}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y'}} + b_j'$, where $\vec{\mathbf{x'}} = [x_1, x_2, \ldots, x_{n-1}]^T$ and $\vec{\mathbf{y'}} = [y_1, y_2, \ldots, y_{n-1}]^T$. Since $\vec{\mathbf{x_L}}$ is a model for $\mathbf{L}$, we know that

$$\begin{aligned} \vec{\mathbf{g_i}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{L}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{L}} + b_i' &\leq \vec{\mathbf{x_1}}^{\mathbf{L}}(n) \\ \vec{\mathbf{x_1}}^{\mathbf{L}}(n) &\leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{L}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{L}} + b_j', \end{aligned} \tag{7}$$

where $\vec{\mathbf{x_1'}}^{\mathbf{L}}$ denotes the first $(n-1)$ components of $\vec{\mathbf{x_1}}^{\mathbf{L}}$; likewise for $\vec{\mathbf{y_1}}'^{\mathbf{L}}$. Since $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$ represents a winning strategy for $\mathbf{Y_R}$, we know that

$$\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{R}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{R}} + b_i' > \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{R}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{R}} + b_j' \tag{8}$$

It follows immediately that $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$ is also a winning strategy for $\mathbf{Y_L}$, since if System (8) is true, there cannot exist a $\vec{\mathbf{x_1}}^{\mathbf{L}}(n) \in \Re$ for $\mathbf{X_L}$ to guess that would make System (7) true. Thus, if $\vec{\mathbf{y_R}}(\vec{\mathbf{x_R}})$ is a winning strategy for $\mathbf{Y_R}$, $\vec{\mathbf{x_L}}$ cannot be a model for $\mathbf{L}$.

We have shown that corresponding to a model for $\mathbf{L}$, there exists a model for $\mathbf{R}$; we now need to show the converse.

Let

$$\vec{\mathbf{x_R}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-2}(y_1, y_2, \ldots y_{n-2})]^T$$

be a model for $\mathbf{R}$. We need to show that $\vec{\mathbf{x_R}}$ can be suitably extended to a model for $\mathbf{L}$.

Let $S_1 = \{m_1 \leq x_n, m_2 \leq x_n, \ldots, m_p \leq x_n\}$ denote the set of constraints in $\mathbf{L}$ that can be written in the form $x_n \geq ()$. Likewise, let $S_2 = \{x_n \leq n_1, x_n \leq n_2, \ldots, x_n \leq n_q\}$ denote the set of constraints that can be written in the form $x_n \leq ()$. Consider the following Skolem function describing x_n: $\max_{i=1}^{p} m_i \leq x_n \leq \min_{j=1}^{q} n_j$. We claim that $\vec{\mathbf{x_L}} = [\vec{\mathbf{x_R}}, \ x_n]^T$ is a model for $\mathbf{L}$.

Assume the contrary and say that $\vec{\mathbf{x_L}}$ (as defined above) is not a model for $\mathbf{L}$; it follows that $\mathbf{Y_L}$ has a winning strategy $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$, corresponding to $\vec{\mathbf{x_L}}$. Consider the complete set of moves made by $\mathbf{X_L}$ and $\mathbf{Y_L}$ to decide $\mathbf{L}$, with $\vec{\mathbf{X_L}}$ playing as per $\vec{\mathbf{x_L}}$ and $\mathbf{Y_L}$ playing as per $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$; let $\vec{\mathbf{x_1}}^{\mathbf{L}}$ and $\vec{\mathbf{y_1}}^{\mathbf{L}}$ denote the corresponding numerical vectors. Likewise, consider the complete set of moves made by $\mathbf{X_R}$ and $\mathbf{Y_R}$ to decide $\mathbf{R}$, with $\mathbf{X_R}$ playing as per $\mathbf{x_R}$; let $\vec{\mathbf{x_1}}^{\mathbf{R}}$ and $\vec{\mathbf{y_1}}^{\mathbf{R}}$ denote the corresponding numerical vectors. Since $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$ is a winning strategy for $\mathbf{Y_L}$, there is at least one constraint in the system $\mathbf{A} \cdot [\vec{\mathbf{x}} \ \vec{\mathbf{y'}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ that is violated. Let p_i denote the violated constraint. We consider the following 2 cases:

1. p_i appears in identical form in $\mathbf{R}$ - Observe that x_n cannot appear in p_i and hence as argued above, $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$ is also a winning strategy for $\mathbf{Y_R}$, contradicting the existence of $\vec{\mathbf{x_R}}$ as a model for $\mathbf{R}$.

2. p_i is a constraint of the form $l_i : m_i \leq x_n \in S_1$ - Consider a constraint of the form $l_j : x_n \leq n_j \in S_2$. In this case, the constraint $m_i \leq n_j$ is part of the QLP $\mathbf{R}$.

   We rewrite the constraint l_i as $\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x'}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y'}} + b_i' \leq x_n$ and the constraint l_j as $x_n \leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x'}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y'}} + b_j'$, where $\vec{\mathbf{x'}} = [x_1, x_2, \ldots, x_{n-1}]^T$ and $\vec{\mathbf{y'}} = [y_1, y_2, \ldots, y_{n-1}]^T$. Since $\vec{\mathbf{x_R}}$ is a model for $\mathbf{R}$, we know that

$$\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{R}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{R}} + b_i' \leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{R}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{R}} + b_j' \tag{9}$$

   Since $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$ represents a winning strategy for $\mathbf{Y_L}$, we know that

$$\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x_1'}}^{\mathbf{L}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y_1'}}^{\mathbf{L}} + b_i' > \vec{\mathbf{x_1}}^{\mathbf{L}}(n) \tag{10}$$

This means that $\mathbf{X_L}$ could not find a $\vec{\mathbf{x_1}}^{\mathbf{L}}(n) \in \Re$ such that

$$\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x'_1}}^{\mathbf{L}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y'_1}}^{\mathbf{L}} + b'_i \leq \vec{\mathbf{x_1}}^{\mathbf{L}}(n), \text{ and} \tag{11}$$

$$\vec{\mathbf{x_1}}^{\mathbf{L}}(n) \leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x'_1}}^{\mathbf{L}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y'_1}}^{\mathbf{L}} + b'_j \tag{12}$$

This means that

$$\vec{\mathbf{g_i}} \cdot \vec{\mathbf{x'_1}}^{\mathbf{L}} + \vec{\mathbf{h_i}} \cdot \vec{\mathbf{y'_1}}^{\mathbf{L}} + b'_i \not\leq \vec{\mathbf{g_j}} \cdot \vec{\mathbf{x'_1}}^{\mathbf{L}} + \vec{\mathbf{h_j}} \cdot \vec{\mathbf{y'_1}}^{\mathbf{L}} + b'_j \tag{13}$$

It follows immediately that $\vec{\mathbf{y_L}}(\vec{\mathbf{x_L}})$ is also a winning strategy for $\mathbf{Y_R}$, since this strategy would cause System (9) to fail. Hence $\vec{\mathbf{x_R}}$ cannot be a model for $\mathbf{R}$.

A similar argument works for the case, when the violated constraint p_i is of the form $l_j : x_n \leq n_j \in S_2$.

□

Theorem 5.1 *Algorithm (4.1) correctly decides query* $\mathbf{G}$ *of System (2).*

Proof: We have shown that eliminating the n^{th} universally quantified variable and the n^{th} existentially quantified variable of a QLP, with $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2 \cdot n$ using Algorithm (4.1), is solution preserving, for arbitrary n. As a consequence of the elimination, we are left with a QLP, having $|\mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}})| = 2 \cdot (n-1)$. Applying the principle of mathematical induction, we conclude that Algorithm (4.1) correctly decides an arbitrary QLP. □

Remark 5.1 *Algorithm (4.3) has been described as the Fourier-Motzkin elimination procedure in the literature [NW99, DE73].*

It is now easy to see that:

Theorem 5.2 *Query (2) can be decided by an Alternating Turing Machine in polynomial time, if* $\mathbf{A}$ *and* $\vec{\mathbf{b}}$ *are rational.*

Proof: From the ELIM-UNIV-VARIABLE() procedure, we know that the guess of $\mathbf{Y}$ for y_i can be confined to $\{l_i, u_i\}$ (instead of (possibly) non-rational values in $[l_i, u_i]$). Observe that the ELIM-EXIST-VARIABLE() procedure can be implemented through repeated pivoting; pivoting has been shown to be rationality preserving in [Sch87]. Thus, in round 1, $\mathbf{X}$ guesses a rational x_1, while in round 2, $\mathbf{Y}$ must guess from $\{l_1, u_1\}$. After round 2, we are left with a new polyhedron in 2 dimensions less than the original polyhedron. Once again, we note that x_2 can be guessed rational and y_2 must be rational. It follows that if the QLP is feasible, $\mathbf{X}$ can guess a rational vector $\vec{\mathbf{x}}$ for the vector $\vec{\mathbf{y}}$ that is guessed by $\mathbf{Y}$, taking into account the alternations in the query. Thus, the language of QLPs can be decided in Alternating polynomial time which is equivalent to saying that QLPs can be decided in `PSPACE`. □

6 Analysis

To calculate the running time of the algorithm, we note that the elimination of a universally quantified variable *does not* increase the number of constraints and hence can be carried out in $O(m \cdot n)$ time (assuming that the matrix currently has m constraints and $O(n)$ variables), while the elimination of an existentially quantified variable could cause the constraints to increase from m to m^2. In fact, [Sch87] provides a pathological constraint set, in which the Fourier-Motzkin elimination procedure results in the creation of $O(m^{2^k})$ constraints after eliminating k variables.

Let us now consider the case of totally unimodular matrices, i.e., assume that $\mathbf{A}$ is a TUM.

Definition 6.1 *A matrix* $\mathbf{A}$ *is said to be totally unimodular (TUM) if the determinant of every square submatrix* $\mathbf{A}'$ *belongs to the set* $\{0, 1, -1\}$.

Lemma 6.1 *The class of totally unimodular matrices is closed under Fourier-Motzkin elimination.*

Proof: For details of Fourier-Motzkin implementation, see [VR99]. Assume that we are eliminating variable x_1. Let us denote :

- $I^+ = \{i : \mathbf{A}_{i1} = +1\}$
- $I^- = \{i : \mathbf{A}_{i1} = -1\}$
- $I^0 = \{i : \mathbf{A}_{i1} = 0\}$

Assuming, neither I^+ nor I^- are empty (if I^+ is empty we can simply discard the inequalities in I^- and vice versa), pick $k \in I^+$ and $l \in I^-$. Add the inequality $\mathbf{A_k} \cdot \vec{\mathbf{x}} \leq b_k$ to the inequality $\mathbf{A_l} \cdot \vec{\mathbf{x}} \leq b_l$. The variable x_1 is eliminated, after each (k, l) pair is processed in this fashion. The key point is that the elimination is a variation of a pivot operation i.e., adding a row to a multiple of another row. In this case the multiple is always $\{+1\}$. Further, we know that the class of totally unimodular matrices is closed under these operations (See Pg. 540, Proposition 2.1 of [NW99]). The lemma follows. □

Lemma 6.2 *Given a totally unimodular matrix* $\mathbf{A}$ *of dimensions* $m \times n$*, for a fixed* n*,* $m = O(n^2)$*, if each row is unique.*

Proof: The above lemma was proved in [Ans80, AF84]. □

Observe that the elimination of a universally quantified variable trivially preserves total unimodularity since it corresponds to deleting a column from the constraint matrix.

Lemma 6.3 *Given an* $m \times n$ *totally unimodular constraint matrix, Algorithm (4.1) runs in polynomial time.*

Proof: We have observed previously that eliminating a universally quantified variable does not increase the number of constraints and hence can be implemented in time $O(m \cdot n)$, through variable substitution. Further, since it corresponds to simple deletion of a column, the matrix stays totally unimodular. From Lemma (6.2), we know that there are at most $O(n^2)$ non-redundant constraints. Eliminating an existentially quantified variable, could create at most as $O(n^4)$ constraints, since $m = O(n^2)$ [VR99]. A routine to eliminate the redundant constraints can be implemented in time $n \times O(n^4 \cdot \log n^4) = O(n^5 \cdot \log n)$ through a sort procedure. (There are $O(n^4)$ row vectors in all; comparing two row vectors takes time $O(n)$.) The procedures PRUNE-CONSTRAINTS() and CHECK-INCONSISTENCY() work only with single variable constraints and hence both can be implemented in time $O(m \cdot n)$. Thus a single iteration of the i loop in Algorithm (4.1) takes time at most $O(n^5 \cdot \log n)$ and hence the total time taken by Algorithm (4.1) to decide a given QLP is at most $O(n^6 \cdot \log n)$. □

Remark 6.1 *We could use a variation of* RADIX-SORT()*, to achieve sorting in time* $O(n \times n^4)$*, to give a total running time of* $O(n^6)$.

7 A taxonomy of QLPs

In the preceding sections, we have shown that a fully general QLP, i.e., a QLP with unbounded alternation can be decided in polynomial time, if the constraint matrix has the property of being totally unimodular. Total unimodularity is a property of the constraint structure; in this section, we classify QLPs based on the structure of the quantifier string and obtain some interesting results.

Definition 7.1 *An* **E**-*QLP is a QLP in which all the existential quantifiers precede the universal quantifiers, i.e., a QLP of the form:*

$$\exists x_1 \exists x_2 \ldots \exists x_n \; \forall y_1 \in [l_1, u_1] \forall y_2 \in [l_2, u_2], \ldots \forall y_n \in [l_n, u_n] \;\; \mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

Theorem 7.1 **E**-*QLPs can be decided in polynomial time.*

Proof: We need to modify Algorithm (4.1) so that ELIM-UNIV-VARIABLE() is called n times so as to eliminate the universally quantified variables. From the argument in Section §6, we know that this can be accomplished in $O(n \times m \cdot n) = O(m \cdot n^2)$ time. (Note that all the universally quantified variables can be eliminated in a single pass in time $O(m \cdot n)$.) The resultant QLP is a standard Linear Programming problem in n variables and m constraints, which can be solved in polynomial time [Vai87]. □

Definition 7.2 *An* **F**-*QLP is a QLP in which all the universal quantifiers precede the existential quantifiers, i.e., a QLP of the form:*

$$\forall y_1 \in [l_1, u_1] \; \forall y_2 \in [l_2, u_2], \ldots \forall y_n \in [l_n, u_n] \; \exists x_1 \; \exists x_2 \ldots \exists x_n \;\; \mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

Lemma 7.1 *The* **F**-*QLP recognition problem is in* coNP.

Proof: A Non-deterministic Turing Machine guesses values for $y_1, y_2, \ldots, y_n$, with $y_i \in \{l_i, u_i\}$. The constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ is transformed into a polyhedral system of the form $\mathbf{G} \cdot \vec{\mathbf{x}} \leq \vec{\mathbf{b}'}$. We can use a linear programming algorithm that runs in polynomial time to verify that the polyhedral system is empty, i.e., $\mathbf{G} \cdot \vec{\mathbf{x}} \not\leq \vec{\mathbf{b}'}$. Thus the "no" instances of **F**-QLPs can be verified in polynomial time. □

In order to prove the coNP-Hardness of **F**-QLPs, we need the concept of an inverse reduction.

Definition 7.3 *A problem P_1 is said to inversely reduce to problem P_2, written $P \leq_I P_2$, if there exists a Turing machine computable function $f()$, such that $x \in P_1 \Leftrightarrow f(x) \notin P_2$.*

Inverse reductions are useful in the following way: If P_1 is complete for a class $\mathcal{C}$ and $f()$ is computable in logarithmic space, then P_2 is complete for the class $co - \mathcal{C}$.

Lemma 7.2 *The* **F**-*QLP recognition problem is* coNP-Hard.

Proof: Let P_1 be the $MAX2SAT$ problem: *Given a 2SAT formula $\phi = \phi_1 \wedge \phi_2 \ldots \phi_m$ on the literals $\{y_1, \bar{y_1}, y_2, \bar{y_2}, \ldots, y_n, \bar{y_n}\}$, is there an assignment such that the number of satisfied clauses is greater than or equal to k?*

MAX2SAT has been shown to be NP-complete in [Pap94].

The **F-QLP** (P_2) is constructed as follows: Each clause ϕ_i is replaced by a pair of linear constraints as per the rules below:

1. If $\phi_i = (y_j, y_k)$ add the 2 constraints: $(1 - y_j) + x_i \geq 1$ and $(1 - y_k) + x_i \geq 1$;
2. If $\phi_i = (\bar{y_j}, y_k)$ add the 2 constraints: $y_j + x_i \geq 1$ and $(1 - y_k) + x_i \geq 1$;
3. If $\phi_i = (y_j, \bar{y_k})$ add the 2 constraints: $(1 - y_j) + x_i \geq 1$ and $y_k + x_i \geq 1$;
4. If $\phi_i = (\bar{y_j}, \bar{y_k})$ add the 2 constraints: $y_j + x_i \geq 1$ and $y_k + x_i \geq 1$;

Note that the clause set ϕ has been replaced by a set of $2 \cdot m$ linear constraints; call this set $\mathcal{Z}_1$. The second set of linear constraints that are added to the **F**-QLP are the box constraints on $\vec{\mathbf{x}}$, i.e., $0 \leq x_i \leq 1, \; i = 1, 2, \ldots, m$; call this set $\mathcal{Z}_2$.

Finally, we add the aggregate constraint $\sum_{i=1}^{m} x_i \leq k-1$ ($\mathcal{Z}_3$) and the quantifier string $\forall y_1 \in [0,1]\ \forall y_2 \in [0,1] \ldots \forall y_n \in [0,1]\ \exists x_1 \exists x_2 \ldots \exists x_n$. Thus the **F**-QLP is completely specified as:

$$\forall y_1 \in [0,1]\ \forall y_2 \in [0,1] \ldots \forall y_n \in [0,1]\ \exists x_1\ \exists x_2 \ldots \exists x_n \quad \mathcal{Z}_1 \wedge \mathcal{Z}_2 \wedge \mathcal{Z}_3$$

Note that all the constraint sets are linear.

Consider an assignment to the clause set ϕ; let us say that clause ϕ_i is satisfied by this assignment. Assume that ϕ_i has the form $(y_j, \bar{y_k})$; the other 3 cases can be argued in identical fashion. The corresponding constraints in the **F**-QLP are: (i) $(1-y_j)+x_i \geq 1$ and (ii) $y_k + x_i \geq 1$. Without loss of generality assume that $y_j =$**true** in P_i. In P_2, the setting $y_j = 1$ forces x_i to be at least 1, in order to satisfy (i). Extending this argument, we see that whenever an assignment satisfies a clause ϕ_i, the variable x_i is forced to be set to at least 1 in the corresponding setting in the **F**-QLP. Thus if an assignment satisfies k or more clauses in P_1, at least k of the x_i variables are set to 1, violating the aggregate constraint and hence the **F**-QLP cannot be true.

Likewise, consider the case where the clause $\phi_i = (y_j, \bar{y_k})$ is not satisfied by an assignment $\vec{\mathbf{y}}$. To make ϕ_i false, y_j must be **false** and y_k must be **true**. Thus the corresponding linear constraints in the **F**-QLP are satisfied using the $\vec{\mathbf{y}}$ values only (i.e., $y_j = 0$ and $y_k = 1$) and x_i can be set to 0. Hence, if all assignments to ϕ satisfy at most $k-1$ clauses, we need to set at most $(k-1)$ x_i values to 1, in each assignment, i.e., the **F**-QLP is true. □

Theorem 7.2 *The* **F***-QLP recognition problem is* `coNP-complete`.

Proof: Follows from Lemma (7.1) and Lemma (7.2). □

Definition 7.4 *A QLP, which is neither an* **E***-QLP nor an* **F***-QLP is called a* **G***-QLP.*

As mentioned before, the complexity of recognizing **G**-QLPs is unknown, although we strongly suspect that it is `PSPACE-complete`.

8 Conclusion

In this paper, we described a new problem called Quantified Linear Programming. Although the complexity of this problem for general constraint matrices is unknown, we showed that in the case in which the constraint matrix is totally unimodular, the problem can be decided in polynomial time. As part of our solution strategy, we independently proved that the class of totally unimodular matrices is closed under Fourier-Motzkin elimination. We also provided a taxonomy for QLP problems and analyzed the complexity of 2 special cases.

We make a couple of interesting observations:

1. The Fourier-Motzkin procedure provides an alternative technique to prove that Linear Programs specified by a rational system of inequalities have rational solutions. This proof is much simpler than the proof used in [PS82], which requires the computation of sizes of subdeterminants of the constraint matrix!

2. The `coNP-completeness` of the recognition problem for **F**-QLPs is remarkable in that QLPs are continuous valued mathematical programs, whereas `NP-completeness` is essentially a discrete concept. A similar result was obtained by Vavasis in [Vav91] for the Quadratic Programming problem.

There are a number of open problems, that we believe merit further investigation:

1. What is the complexity of **G**-QLPs?

2. Are there constraints structures other than TUMs, for which Algorithm (4.1) converges in polynomial time?

3. What is the complexity of **G**-QLP(2)? QLP(i) refers to the case in which the existential support (the number of existentially quantified variables) of every constraint is at most 2.

References

[AF84] R.P. Anstee and M. Farber. Characterizations of totally balanced matrices. *J. Algorithms*, 5:215–230, 1984.

[Ans80] R.P. Anstee. Properties of (0,1)-matrices with no triangles. *J. of Combinatorial Theory (A)*, 29:186–198, 1980.

[BA93] O. Ben-Ayed. Bilevel linear programming. *Computers and Operations Research*, 20:485–501, 1993.

[CD00] Alain Colmerauer and T. Dao. Expressiveness of full first order constraints in the algebra of finite or infinite trees. In *Proceedings of Constraint Programming*, September 2000.

[CG+97] M. Cadoli, , M. Giovanardi, A. Giovanardi, and M. Schaerf. Experimental analysis of the computational cost of evaluating quantified boolean formulae. In *Lecture Notes in Artificial Intelligence*, 1997.

[CGS98] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *AAAI-98*, July 1998.

[CH99] V. Chandru and J. N. Hooker. *Optimization Methods for Logical Inference*. Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., 1999.

[Cho97] Seonho Choi. *Dynamic Time-based scheduling for Hard Real-Time Systems*. PhD thesis, University of Maryland, College Park, jun 1997.

[Cho00] Seonho Choi. Dynamic time-based scheduling for hard real-time systems. *Journal of Real-Time Systems*, 2000.

[Col86] A. Colmerauer. *Logic Programming and Its Applications*, chapter Theoretical Model of PrologII, pages 181–200. Ablex Series in Artificial Intelligence. Ablex Publishing Corporation, 1986.

[DE73] G. B. Dantzig and B. C. Eaves. Fourier-Motzkin Elimination and its Dual. *Journal of Combinatorial Theory (A)*, 14:288–297, 1973.

[GPS95] R. Gerber, W. Pugh, and M. Saksena. Parametric Dispatching of Hard Real-Time Tasks. *IEEE Transactions on Computers*, 1995.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer-Verlag, New York, 2002.

[Hoo00] J.N. Hooker. *Logic-Based methods for Optimization*. Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., 2000.

[Imm] Neil Immerman. Personal Communication.

[Joh] D.S. Johnson. Personal Communication.

[LTCA89] S. T. Levi, S. K. Tripathi, S. D. Carson, and A. K. Agrawala. The `Maruti` Hard Real-Time Operating System. *ACM Special Interest Group on Operating Systems*, 23(3):90–106, July 1989.

[MKAT92] D. Mosse, Keng-Tai Ko, Ashok K. Agrawala, and Satish K. Tripathi. Maruti: An Environment for Hard Real-Time Applications. In Ashok K. Agrawala, Karen D. Gordon, and Phillip Hwang, editors, *Maruti OS*, pages 75–85. IOS Press, 1992.

[NW99] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1999.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice Hall, 1982.

[Sak94] Manas Saksena. *Parametric Scheduling in Hard Real-Time Systems*. PhD thesis, University of Maryland, College Park, June 1994.

[Sch87] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1987.

[Sub00] K. Subramani. *Duality in the Parametric Polytope and its Applications to a Scheduling Problem*. PhD thesis, University of Maryland, College Park, August 2000.

[Sub01] K. Subramani. Parametric scheduling - algorithms & complexity. In et. al. Burkhard Moniem, editor, *Proceedings of the* 8^{th} *International Conference on High-Performance Computing (Hi-PC)*, volume 2228 of *Lecture Notes in Computer Science*, pages 36–46. Springer-Verlag, December 2001.

[Sub02] K. Subramani. A specification framework for real-time scheduling. In W.I. Grosky and F. Plasil, editors, *Proceedings of the* 29^{th} *Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, volume 2540 of *Lecture Notes in Computer Science*, pages 195–207. Springer-Verlag, November 2002.

[Vai87] P. M. Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations. In Alfred Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 29–38, New York City, NY, May 1987. ACM Press.

[Vav91] S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

[VC94] L. Vicente and P. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[VR99] V.Chandru and M.R. Rao. Linear programming. In *Algorithms and Theory of Computation Handbook, CRC Press, 1999*. CRC Press, 1999.