A Note on Redundant Linear Constraints

Pascal Van Hentenryck[1]
Jean-Louis Imbert[2]

Technical Report No. CS-92-11
March 1992

[1]Brown University, Department of Computer Science Box 1910, Providence, RI 02912 USA
[2]GIA Parc Scientifique et Technologique de Luminy, 163 Avenue de Luminy, case 901, F-13288 Marseille Cedex 9, FRANCE

A Note on Redundant Linear Constraints

Jean-Louis Imbert
GIA Parc Scientifique et Technologique de Luminy,
163 Avenue de Luminy, case 901
F-13288 Marseille Cedex 9 (France)
Email: imbert@gia.univ-mrs.fr

Pascal Van Hentenryck
Brown University,
Department of Computer Science
Box 1910, Providence, RI 02912 (USA)
Email: pvh@cs.brown.edu

Abstract

Much research on redundancy has focused on identifying syntactic properties of redundant and non-redundant constraints to avoid resorting to costly semantic methods (e.g. optimization). In this paper, we study the detection of redundant linear constraints using a lexicographic solved form [8]. The lexicographic solved form is the basis of an incremental decision procedure, based on the simplex algorithm, for systems of equations, inequalities, and disequations. It enjoys the two important properties of (1) precluding syntactically the presence of implicit equalities and (2) being preserved by pivoting under a lexicographic pivoting rule. We show that the lexicographic solved form helps to identify new syntactic criteria for detecting redundant and non-redundant constraints. These criteria generalize and amplify existing results. In addition, we present a syntactic method to reduce a system of linear constraints which can be used before further applications of syntactic criteria or possibly semantic methods.

1 Introduction

Detection of redundancy in linear programs is important for many reasons. As mentioned by Telgen [7], redundant constraints can slow down the simplex algorithm by (1) increasing the number of iterations (e.g. there are more basic solutions), (2) increasing the computation time required for an iteration, and (3) requiring memory space that would be better used for other purposes. In addition, redundant constraints may exacerbate the problem of numerical errors. Recent interest in redundancy has also emerged from constraint logic programming [1, 3], a class of languages having constraint solving as its main operation. Redundant constraints arise naturally in constraint logic programming due to the nondeterministic nature of the language and the freedom left to the programmer. Also, operations like the printing of the results which require projection on a set of variables may potentially benefit from studies in redundancy.

There exists a simple (but rather costly) way of detecting redundant constraints which amounts to optimizing each constraint individually subject to the remaining constraints. Much research (see for example [7, 4]) has been devoted to avoiding these semantic methods and to identifying syntactic criteria to detect redundant constraints. In this paper, we continue this line of research and examine the use of the lexicographic solved form [8] for detecting redundant constraints. The lexicographic solved form is the basis of an incremental decision procedure, based on the simplex algorithm [2], for systems of equations, inequalities, and disequations. It enjoys the properties of (1) precluding the presence of implicit equalities (i.e. all implicit constraints of the form $< variable > = < constant >$ are made explicit) and (2) being preserved through pivoting under a lexicographic search rule. We show that the lexicographic solved form helps to identify a number of new syntactic criteria for detecting redundant and non-redundant constraints, generalizing and amplifying existing results. Moreover we provide a syntactic criteria to reduce the size of the system of constraints before further application of syntactic and/or semantic methods.

The rest of this paper is organized as follows: Sections 2 and 3 recall respectively relevant results in redundancy and results concerning the lexicographic solved form. Section 4 presents our new syntactic criteria. Section 5 illustrates them through a simple example. Section 6 contains the conclusion of this research and directions for further research.

2 Redundancy of Linear Constraints

We consider systems of linear inequalities

$$I \;=\; \{\; t_1 \geq 0, \ldots, t_n \geq 0 \;\}$$

where $t_i = a_{i0} + \sum_{j=1}^{m} a_{ij} x_j$ where the a_{ij} are scalars and the x_j variables. In the rest of this paper, we assume only satisfiable systems of inequalities for convenience.

The notion of redundant constraint can be formalized by the following two definitions.

Definition 1 The solution set of a system of inequalities $I \;=\; \{\; t_1 \geq 0, \ldots, t_n \geq 0 \;\}$ is the set

$$Sol(I) \;=\; \{\langle x_1, \ldots, x_n \rangle \in \Re^m \mid t_1 \geq 0, \ldots, t_n \geq 0\}.$$

Definition 2 An inequality C in I is redundant iff

$$Sol(I) = Sol(I \setminus \{C\}).$$

A basic result on redundancy is given by the following lemma [6].

Lemma 3 Let $I \;=\; \{\; t_1 \geq 0, \ldots, t_n \geq 0 \;\}$. The inequality $t_k \geq 0$ is redundant in I iff there exists nonnegative values $a_0, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n$ such that

$$t_k = a_0 + \sum_{j=1, j \neq k}^{n} a_i t_i.$$

Proof See [2, 6]. □
It is convenient to associate with a set of inequalities I a linear program.

Definition 4 Let I be a system of inequalities $\{ t_1 \geq 0, \ldots, t_n \geq 0 \}$. Its associated linear program $E = \{ x_1^+ = t_1^+, \ldots, x_n^+ = t_n^+ \}$ is defined as follows.

1. Transform I into a system of equations by introducing slack variables x_k^+ (i.e. variables that can only be assigned nonnegative values) giving
$$\{ t_1 - x_1^k = 0, \ldots, t_n - x_n^k = 0 \}.$$
2. Replace each variable in $\{t_1, \ldots, t_n\}$ by the difference of two new slack variables obtaining
$$\{ t_1^+ - x_1^k = 0, \ldots, t_n^k - x_n^k = 0 \}.$$
3. Rewrite the system to isolate $x_1^+, \ldots, x_n^+$.

Now detecting redundant constraints in I can be reduced to the problem of detecting redundant slack variables in its associated linear program E.

Definition 5 Let E be a linear program with slack variables $x_1^+, \ldots, x_m^+$. Slack variable x_k^+ is redundant in E iff there exist nonnegative values $c_0, \ldots, c_{k-1}, c_{k+1}, \ldots, c_m$ such that
$$E \Rightarrow x_k^+ = c_0 + \sum_{i=1, i \neq k}^{m} c_i x_i^k.$$

Lemma 6 Let I be a system of inequalities and E its associated linear program. Constraints $t_k \geq 0$ is redundant in I iff slack variable x_k^+ is redundant in E iff there exists no solution of E in which all variables are assigned nonnegative values except x_k^+ which is assigned a negative value.

Proof See [4, 7]. □

The rest of this paper thus focuses on the problem of detecting redundant or non-redundant slack variables. Both redundancy and non-redundancy are important results: the former eliminates a constraint while the latter removes the need for semantic methods on the constraints. Most existing criteria are in fact detecting non-redundancy.

Remark 7 In practice, it is better to rewrite the initial system into two subsystems, one of them only being a linear program. A set I can be transformed into
$$\{ y_1 = t_1', \ldots, y_r = t_r' \} \cup \{ x_1^+ = t_1^+, \ldots, x_s^+ = t_s^+ \}.$$
avoiding the replacement of the variables by the difference of two slack variables. This reduces the number of slack variables, the only ones remaining correspond directly to the initial constraints.

To present the relevant results on slack variables, it is necessary to introduce the solved form used in the simplex algorithm. For reasons that will become clear later on, we assume a total ordering $\gg$ on the variables.

Definition 8 A normalized linear term is an expression

$$a_0 + \sum_{i=1}^{n} a_i x_i^+$$

where $x_i^+ \gg x_{i+1}^+$ $(1 \leq i \leq n-1)$. a_0 is called the *constant* of the normalized linear term and the a_i $(1 \leq i \leq n)$ are called the *coefficients*. We also denote by $var(t)$ the set of variables appearing, with a non-zero coefficient, in a normalized linear term t.

Definition 9 A linear equation in *solved form* (SF for short) is an expression

$$x^+ = t^+ \quad .$$

where

- t^+ is a normalized linear term;
- $x^+ \notin var(t^+)$;
- the constant of t^+ is nonnegative.

The left-hand side variable is called a *basic* variable while the variables in the right-hand side are *non-basic* variables.

Definition 10 A set of linear equations $\{x_1^+ = t_1^+, \ldots, x_n^+ = t_n^+\}$ is in SF iff

- $x_i^+ = t_i^+$ is in SF $(1 \leq i \leq n)$;
- each basic variable appears only once in the whole system.

We are now ready to present three relevant results on slack variables. The first result concerns non-basic variables [4].

Lemma 11 Let E be a system of equations in SF. A non-basic slack variable x_k^+ is non redundant in E if, for each equation $x_j^k = a_{j0} + \sum a_{ji} x_i^+$ satisfying $a_{j0} = 0$, we have $a_{jk} \leq 0$.

Proof It suffices to assign 0 to all non-basic variables but x_k^+. Variable x_k^+ can then be assigned a sufficiently small negative value v_k (i.e. $|v_k| < min\{a_{j0}/a_{jk} \mid a_{jk} > 0\}$). □

The second result identifies the non-redundancy of a subset of the leaving variables [4].

Lemma 12 Let E be a system in SF. A basic slack variable x_k^+ is non-redundant in E iff there exists an entering variable x_e for which a_{k0}/a_{ke} is the greatest unique value for all values (a_{j0}/a_{je}) such that $a_{je} < 0$.

Proof It suffices to assign 0 to each non-basic variable $x_i^+ \neq x_l^+$ and $a_{j0} + \epsilon$ to x_e with ϵ sufficiently small. Then apply Lemma 6. □

The last result we mention is a simple way of detecting redundant constraints.

Lemma 13 Let E be a system in SF. A non-basic variable which has the unique positive coefficient in the right-hand side of an equation is redundant.

Proof This is a direct consequence of Lemma 5. □

3 The Lexicographic Solved Form

As mentioned previously, the lexicographic solved form was introduced as a basis for an incremental decision procedure for systems of equations and disequations on slack variables. To obtain such a procedure, it is important to discover implicit equalities (i.e. the fact that a variable is assigned a single value in all solutions). The property is not guaranteed by SF as illustrated by the example

$$\begin{cases} y_1^+ &= 0 + x_1^+ - x_2^+ \\ y_2^+ &= 0 - x_1^+ + x_2^+ \end{cases}$$

This set of equations is in SF but y_1 and y_2 can only be assigned the value 0.

The lexicographic solved form (LSF for short) [8] is a syntactic restriction on SF which precludes the presence of implicit equalities. To present LSF, we need some new notions. Recall that we are assuming a total ordering on the variables. In addition, we will take the convention that $x_i^+ \gg x_j^+$ iff $i < j$.

Definition 14 A vector $v \neq 0$ is called lexicographically positive (resp. negative), denoted $v \overset{L}{>} 0$ (resp. $v \overset{L}{<} 0$), if its first non-zero component is positive (resp. negative). We also denote by $v \overset{L}{\geq} 0$, the disjunction $v = 0 \vee v \overset{L}{>} 0$.

Definition 15 A vector v is said to be lexicographically greater than a vector u if $v - u \overset{L}{>} 0$.

Definition 16 A negative unit vector is a vector $(0, \ldots, 0, -1, 0, \ldots, 0)$.

We now associate a vector with a normalized linear term.

Definition 17 The vector v associated with a normalized linear term $a_0 + \sum_{i=1}^{n} a_i x_i^+$ is defined by $v = (a_0, a_1, \ldots, a_n)$.

This can be generalized to linear constraints in SF.

Definition 18 The vector v associated with a linear constraint

$$y^+ = a_0 + \sum_{i=1}^{n} a_i x_i^+$$

in SF where $x_{i-1}^+ \gg y^+ \gg x_i^+$ is defined by

$$v = (a_0, a_1, \ldots, a_{i-1}, -1, a_i, \ldots, a_n).$$

We are now in position to define the lexicographic solved form.

Definition 19 A linear equation C is in LSF iff

- C is in SF;
- the vector associated to C is either a negative unit vector[1] or lexicographically positive.

Definition 20 A set of linear equations $\{y_1^+ = t_1^+, \ldots, y_n^+ = t_n^+\}$ is in LSF iff

- $y_i^+ = t_i^+$ is in LSF $(1 \leq i \leq n)$;
- each basic variable appears only once in the whole set.

The basic difference between SF and LSF is that, for any constraint $y^+ = t^+$ in LSF, the constant or the first non-zero coefficient of t^+ is positive (except for assignments of a variable to zero). The variable associated with the first positive coefficient is called the *leading* variable in this paper. When the constant is non-zero, we take the convention of using a fictitious variable x_0^+ as the leading variable. For instance, the equations $x_4^+ = 2 - x_2^+ + x_5^+$ and $x_4^+ = x_2^+ - x_5^+$ are in LSF while the equation $x_2^+ = x_4^+ - x_5^+$ is not. The leading variables are respectively x_0^+ and x_2^+ for the first two equations.

One of the main results on LSF is as follows.

Lemma 21 A system E of linear equations in LSF does not contain any implicit equalities.

Proof See [8]. □

The other important property of LSF is that it is preserved through pivoting using a lexicographic pivoting rule. Recall that an iteration of the simplex algorithm consists of three steps.

1. Choosing a non-basic variable to enter the basis; this variable is called the *entering* variable.
2. Choosing a basic variable to leave the basis; this variable is called the *leaving* variable.
3. Pivoting to remove the leaving variable from and to introduce the entering variable into the basis.

The simplex algorithm leaves some freedom to choose the leaving variable. Let $\mathcal{E}$ be a set of equations in LSF which can be written as

$$\mathcal{E} = \{y_i^+ = a_{i0} + \sum_{j=1}^{n} a_{ij} x_j^+ \mid (1 \leq i \leq n)\}$$

[1] This is necessary to represent explicit assignments of the form $y^+ = 0$

Given an entering variable x_k^+, the simplex algorithm chooses the leaving variable y_l^+ such that

$$\frac{a_{l0}}{a_{lk}} = \max_{i \in S_k} \frac{a_{i0}}{a_{ik}}$$

where $S_k = \{i \mid 1 \leq i \leq n \text{ and } a_{ik} < 0\}$.

The simplex algorithm does not prescribe any specific rule to break ties. The *lexicographic* rule [2] is a specialization of the above rule which removes the freedom.

Definition 22 Let x_k^+ be the entering variable. Let $S_k = \{i \mid 1 \leq i \leq m \text{ and } a_{ik} < 0\}$. Let v_i be the vector associated with constraint i. Let u_i be v_i/a_{ik}. The *lexicographic rule* amounts to choosing the leaving variable y_l in such a way that u_l is lexicographically the greatest of all vectors, i.e.,

$$u_l = \text{lex} \max_{i \in S_k} u_i$$

Theorem 23 The lexicographic rule preserves LSF through pivoting.

Proof See [8]. □

Note finally that we can always change the lexicographic ordering as convenient provided that the system remains in LSF.

4 Redundancy With LSF

We now investigate the use of LSF for the detection of redundant and non-redundant constraints. In what follows, we always assume that the system E in LSF does not contain two equations which have similar right-hand sides up to multiplication by a constant. This kind of syntactic redundancy [5] can be easily detected.

Our first result is a generalization of Lemma 12 and shows that all leaving variables, for a given ordering, are non-redundant. The proof of the theorem is also the basis of most proofs in this paper.

Theorem 24 Let E be a system in LSF. Let $\gg_m$ a new total ordering on the variables such that

- $x_i^+ \gg_m x_j^+$ if x_i^+ is a non-basic variable and x_j^+ is a basic variable;
- $x_i^+ \gg_m x_j^+$ if x_i^+ and x_j^+ are both basic or both non-basic, and $x_i^+ \gg x_j^+$.

Then all leaving variables wrt ordering $\gg_m$ are non-redundant.

Proof

Let x_e^+ be the entering variable and y_k^+ the corresponding leaving variable. Let x_j^+ be the leading variable (possibly x_0^+) in the equation $y_k^+ = a_{k0} + \sum a_{kp} x_p^+$. Using Lemma 6, it is sufficient to find a solution to E assigning a negative value v_{m+k} to y_k^+ and nonnegative values $v_1, \ldots, v_{m+k-1}, v_{m+k+1}, \ldots, v_{m+n}$ to the other variables. The proof proceeds in four steps.

The first step is a normalization of the equations. Let

$$E = \{ y_i^+ = a_{i0} + \sum a_{ip} x_p^+ \mid 1 \leq i \leq n \ \& \ 1 \leq p \leq m \}.$$

The normalization produces a system

$$E' = \{ c_i y_i^+ = a'_{i0} + \sum a'_{ip} x_p^+ \mid 1 \leq i \leq n \ \& \ 1 \leq p \leq m \}.$$

such that $c_i > 0$, $a'_{ip} = c_i a_{ip}$ satisfying

$$0 < a'_{kj} \leq a'_{ij} \ \& \ a'_{ke} \leq a'_{ie}$$

for all equations in M_k with

$$M_k = \{ y_i^+ = a_{i0} + \sum a_{ip} x_p^+ \mid x_j^+ \text{ is the leading variable } \& \ a_{ie} < 0 \}.$$

This normalization is performed by choosing c_i as follows.

- $c_i = 1$ for equation not in M_k;
- $c_i = (a_{kj}/a_{ij})$ if $(a_{ij}/a_{ie}) = (a_{kj}/a_{ke})$;
- $a_{ij}a_{ke} < c_i a_{ij} a_{ie} < a_{kj}a_{ie}$ otherwise.

The second step assigns nonnegative values $v_j, \ldots, v_n$ such that

$$0 < \theta_{kj} < \theta_{ij}$$

for all equations $y_i^+ = a_{i0} + \sum a_{ip} x_p^+$, $i \neq k$ with

$$\theta_{lp} = \sum_{q=p}^{m} a_{lq}^+ v_q.$$

The assignment proceeds from m to j and makes sure that

1. $a'_{rp} > a'_{sp} \ \Rightarrow \ \theta_{rp} > \theta_{sp}$;
2. $a'_{rp} \neq 0 \ \Rightarrow \ a'_{rp}$ and θ_{rp} have the same sign.
3. $a'_{rp} = 0 \ \Rightarrow \ \theta_{rp} = \theta_{rp+1}$ if $p < m$ and $\theta_{rm} = 0$ otherwise.

The third step simply decreases the value of v_e so that θ_{kj} is the only negative value for a relevant set of equations (i.e. the set L below). Let

$$S_k = \{ y_i^+ = a_{i0} + \sum a_{ip} x_p^+ \mid 1 \leq i \leq n \ \& \ a_{ie} < 0 \}.$$

and L be the set of equations with a leading variable x_l^+ such that $l \geq j$. First note that any equation in S_k has a leading variable x_l^+ with $l \leq j$ by definition of the lexicographic pivoting rule. Hence, $L \cap S_k = M_k$ and increasing the value of v_e can only decrease the values θ_{ij} for those equations $y_i^+ = a_{i0} + \sum a_{ip} x_p^+$ in L which are in M_k. (Equations not in L are of no concern at this point since it is possible to increase their values arbitrarily). As the result of steps 1 and 2, we also know that $a'_{ke} \leq a'_{ie} < 0$, $0 < \theta_{kj} < \theta_{ij}$, and $\theta_{ke} \leq \theta_{ie} < 0$ for each equation i in M_k. Hence, it is possible to increase the value v_e in such a way that θ_{kj} is the only negative value for all equations in L.

The last step completes the proof by assigning values from $j-1$ to 0, as for instance in a way similar to step 2, using a fictitious variable x_0^+ for the constants $a_{10}, \ldots, a_{n0}$ if necessary. The final solution is obtained by dividing all values $v_1, \ldots, v_{m+n}$ by v_0. □

The next theorem is an important generalization of Lemma 11. It enables us to conclude the non-redundancy of non-basic variables which are never leading, a much weaker condition than the one imposed by Lemma 11.

Theorem 25 In a system in LSF, a non-basic variable which is never leading is non-redundant.

Proof Let x_i^+ be a non-basic variable which is never leading. We construct a solution of the system which assigns 0 to each non-basic variable x_j^+ lexicographically smaller than x_i^+ and assigning -1 to x_i^+. Then it is sufficient to proceed as in the last step of Theorem 24 and to apply Lemma 6. □

Example 26 The following table presents an application of the theorem.

	Cte	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+	x_6^+
y_1^+	+						
y_2^+	0	+					
y_3^+	0	0	0	+			
y_4^+	0	+					

In the table, a '+' denotes a positive coefficient and a space denotes any other value. The basic variables are on the left. Variables $x_2^+, x_4^+, x_5^+, x_6^+$ are non-redundant.

The next theorem considers leading variables. It indicates that, if we can find a variable to replace a leading variable, then the leading variable is non-redundant.

Theorem 27 Let x_i^+ and x_j^+ be two non-basic variables with $x_i^+ \gg x_j^+$. Assume that

1. $a_{sj} > 0$ in each equation s in which x_i^+ is leading.
2. $a_{kj} \geq 0$ for each equation k in which x_p^+ is leading with $x_i^+ \gg x_p^+ \gg x_j^+$. Then x_i^+ is non-redundant.

Proof Apply Theorem 25 with the same lexicographic ordering except that the order of variables x_i^+ and x_j^+ has been exchanged. □

Example 28 The following table presents an application of the theorem.

	Cte	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+	x_6^+
y_1^+	0	+				+	
y_2^+	0	0	0	+		+/0	
y_3^+	0	+				+	

In the table, a '+' denotes a positive coefficient and a space denotes any other value. Variables x_1^+ is non-redundant because of variable x_5^+.

The next result also deals with leading variables. It indicates that a leading variable always followed by a positive coefficient is non-redundant.

Theorem 29 Let x_i^+ be a non-basic variable. x_i^+ is non redundant if, in all equations in which x_i^+ is leading, the greatest variable with a non-zero coefficient which is smaller than x_i^+ has a positive coefficient.

Proof Apply Theorem 25 with the same lexicographic ordering except that x_i^+ becomes the smallest variable. □

Example 30 The following table presents an application of the theorem.

	Cte	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+	x_6^+
y_1^+	0	0	+	0	0	+	
y_2^+	0	0	0	+	+		
y_3^+	0	0	+	+		+	

In the table, a '+' denotes a positive coefficient and a space denotes any other value. Variables x_2^+ and x_3^+ are non-redundant.

The above theorems apply directly to the initial linear program associated with the set of inequalities. When no conclusion can be drawn for a particular variable, it is possible to reduce the size of the system to be considered. The above theorems, the reduction theorem, and possibly semantic methods can then be applied to the reduced system.

Theorem 31 [Reduction Theorem] Let E be a system of equations in which x_i^+ is a leading variable. x_i^+ is redundant in E iff x_i^+ is redundant in the system R defined as follows:

1. remove from E all equations whose leading variable (possibly the fictitious variable associated with the constant) is lexicographically greater than x_i^+ to obtain T;
2. remove from T all non-basic variables which never appear with a positive (> 0) coefficient to obtain R.

Proof

We first show that x_i^+ is non-redundant in E iff it is non-redundant in T. (The Only part) Since T is a subsystem of E, all solutions of E are also solutions of T. Hence, by Lemma 6, if x_i^+ is non-redundant in E, then it is non-redundant in T. (The If part) If x_i^+ is non-redundant in T, by Lemma 6, it is possible to find a solution T with a negative value for x_i^+ and nonnegative values for all other variables. This solution can be turned into a solution of E by assigning values to the variables appearing in E and not in T. It is sufficient to assign 0 to all variables which are lexicographically smaller than at least one non-basic variable in T. The other variables can be assigned as in step 4 of the proof of Theorem 24.

We now show that x_i^+ is non-redundant in T iff it is non-redundant in R. The if part is trivial since it suffices to complete a solution of R by assigning 0 to the variables in T and not in R. For the only part, consider a solution of T as suggested in Lemma 6. Decreasing the value of a variable occurring only with negative coefficients in T increases the value of the basic variables. Assigning 0 to them and removing them directly gives a solution to R and the theorem follows from Lemma 6. □

Example 32 Consider the initial system

	Cte	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+	x_6^+
y_1^+	+						
y_2^+	0	0	0	+		-	
y_3^+	0	+					
y_4^+	0	0	0	0	+	-	
y_5^+	0	0	0	+		0	

x_3^+ is redundant in the initial system iff x_3^+ is redundant in the system depicted in the following table:

	x_3^+	x_4^+	x_6^+
y_2^+	+		
y_4^+	0	+	
y_5^+	+		

5 A Complete Example

The following table presents a complete example. The empty spaces correspond to zero coefficients.

	Cte	x_1^+	x_2^+	x_3^+	x_4^+	x_5^+	x_6^+	x_7^+
x_8^+	2	1			3		2	1
x_9^+	4	-1	2	-2				
x_{10}^+	3	-2					-1	
x_{11}^+		2	-1				3	-1
x_{12}^+					1		2	-3
x_{13}^+	2	-1	1	-1				
x_{14}^+		1					2	-1
x_{15}^+						1		-3
x_{16}^+					2	1		-3
x_{17}^+	1	1	1	4			-2	
x_{18}^+				2	-1	-1	2	
x_{19}^+				1	-2	1		

Lemma 5 detects that x_8^+ is redundant. Lemma 12 detects that $x_{10}^+, x_{11}^+, x_{17}^+$, and x_{18}^+ are non-redundant. Theorem 24, which generalizes Lemma 12, concludes that $x_{13}^+, x_{15}^+, x_{19}^+$ (and

of course $x_{10}^+, x_{11}^+, x_{17}^+$, and x_{18}^+) are non-redundant. Lemma 11 detects that x_2^+ and x_7^+ are non-redundant. Theorem 25, generalizing Lemma 11, detects the non-redundancy of x_6^+ (in addition of those already detected by Lemma 11). Theorem 13 detects that x_5^+ is redundant. Theorem 27 detects that x_1^+ is non redundant. Theorem 29 detects the non redundancy of x_4^+.

We then use the reduction theorem for variable x_3^+. This gives us the system:

	x_3^+	x_4^+	x_5^+	x_6^+
x_{12}^+		1		2
x_{15}^+			1	
x_{16}^+		2	1	
x_{18}^+	2	-1	-1	2
x_{19}^+	1	-2	1	

Using Lemma 5, the first three rows can be removed, giving the following system:

	x_3^+	x_4^+	x_5^+	x_6^+
x_{18}^+	2	-1	-1	2
x_{19}^+	1	-2	1	

Then modifying the order on variables to enforce that $x_3^+ \gg x_6^+ \gg x_5^+ \gg x_4^+ \gg x_{18}^+ \gg x_{19}^+$, Theorem 29 concludes that x_3^+ is non-redundant. The only variables for which no conclusion was reached are x_9^+,x_{12}^+,x_{14}^+, and x_{16}^+. These variables will probably need to be checked using semantic methods.

6 Conclusion

In this paper, we considered the detection of redundant and non-redundant constraints using syntactic criteria. We showed that the use of the lexicographic solved form helps to generalize existing results and to identify new syntactic criteria. In addition, we presented a reduction criteria that can be applied to a system before further application of syntactic criteria, reduction, and/or semantic methods.

It is interesting to note that the problem of redundancy is sometimes connected to the problem of implicit equalities (see for example [7]). The lexicographic solved form seems to be an interesting tool to deal with both of these issues and it would be interesting to study other related applications. Note also that we have found more criteria dealing with non-basic variables than with basic variables. This is in accordance with the standard results in this domain. Similarly, it is easier to detect non-redundancy than redundancy.

It is too early to judge the practical benefit of these results. However, we believe that these results are interesting on their own since they improve our understanding of redundancy in linear programs and its connection with implicit equalities and solved forms of linear programs.

Acknowledgements

We would like to thank Jacques Cohen, Alain Colmerauer, and Timothy Hickey for interesting discussions and comments. This research was supported in part by the INRIA institute under grant 13001-5847 (Jean-Louis Imbert) and by the National Science Foundation under grant number CCR-9108032 and the Office of Naval Research under grant N00014-91-J-4052 ARPA order 8225 (Pascal Van Hentenryck).

References

[1] A. Colmerauer. An Introduction to Prolog III. *CACM*, 28(4):412–418, 1990.

[2] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[3] J. Jaffar and J-L. Lassez. Constraint Logic Programming. In *POPL-87*, Munich, FRG, January 1987.

[4] M.H. Karwan, V. Lofti, J. Telgen, and S. Zionts. *Redundancy in Mathematical Programming: a State-of-the-Art Survey*, volume 206 of *Lecture Notes in Economics and Mathematical Systems.* Springer Verlag, 1983.

[5] J-L. Lassez, T. Huynh, and K. McAloon. Simplification and Elimination of Redundant Arithmetic Constraints. In *Proceedings of the North-American Conference on Logic Programming (NACLP-89)*, Cleveland, Ohio, October 1989.

[6] A. Shrijver. *Theory of linear and integer programming.* Interscience Series in Discrete Mathematics and Optimization. Wiley, 1986.

[7] J. Telgen. Redundancy and linear programs. Mathematical Centre Tracts 137, Mathematisch Centrum, Amsterdam, 1981.

[8] P. Van Hentenryck and T. Graf. Standard Forms for Rational Linear Arithmetics in Constraint Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 1992. To Appear. A preliminary version was presented at The International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Fl, January 1990.