Redundancy Elimination with a Lexicographic Solved Form

J-L. Imbert[1]
P. Van Hentenryck[2]

Technical Report No. CS-95-02

Febuary, 1995

[1]Lab. Informatique de Clermont-Ferrand Complexe Universitaire des Cezaux F-63177 Aubiere Cedex (France)
[2]Dept.of Computer Science,Brown University Providence, RI 02912

Redundancy Elimination with a Lexicographic Solved Form

J-L. Imbert
Lab. Informatique de Clermont-Ferrand
Complexe Universitaire des Cezaux
F-63177 Aubiere Cedex (France)
imbert@glm.univ-bpclermont.fr

P. Van Hentenryck
Brown University
Box 1910
Providence, RI 02912, USA
pvh@cs.brown.edu

Abstract

Detection of redundant linear constraints is important for a variety of applications including linear programming, variable elimination, canonical forms, and convex hull algorithms to name a few. Systematic semantic methods (i.e. optimization) exist to identify all redundant constraints but they are generally computationally expensive. It is thus desirable to identify syntactic criteria to reduce the need for semantic methods.

This paper presents new syntactic criteria to detect redundant and non-redundant constraints. The results exploit a lexicographic solved form for linear programming (originally proposed to detect implicit equalities) which makes it possible to define syntactic criteria much stronger than those proposed in the past. In particular, one of our results shows that a system in a lexicographic solved form with n constraints and m non-basic variables has at most $min(2n-1, n+m)$ redundant variables. In addition, the paper presents new syntactic criteria to reduce the size of the constraint system before further applications of syntactic and semantic methods.

These theoretical results have been integrated in a complete system to detect redundant linear constraints. Experimental evaluation shows that our syntactic criteria decide the redundancy of about 90% of the constraints and may produce orders of magnitude improvement in efficiency over previous work whose criteria were deciding between 24% and 56% of the constraints. In addition, the benefits of the new results increase with the size and with the sparsity of the constraint system and are especially dramatic when the number of variables is large compared to the number of constraints.

1 Introduction

Detection of redundant linear constraints is important for a variety of applications including linear programming [12], variable elimination (e.g. [8, 6]), canonical forms (e.g [9]) and program analysis and verification of hybrid systems (e.g. [3, 5]) to name a few. Systematic semantic methods exist to identify all redundant constraints. They consist essentially in solving a linear program per constraint (i.e. optimizing the constraint subject to the other constraints) and hence they are generally computationally expensive. This has led to the investigation of syntactic methods to detect redundant linear constraints (e.g. [7, 10, 12]) in order to reduce the need for semantic methods.

The purpose of this paper is to propose new syntactic criteria to detect redundant linear constraints. The novel results are based on a lexicographic solved form for linear programming which

was proposed in [13] as a basis for a decision procedure for generalized linear constraints (including strict inequalities and $\neq$). The lexicographic solved form appears to be an effective tool to detect redundant linear constraints since our new syntactic criteria subsume and are much stronger than previous results. In particular, one of our results shows that, in a system in lexicographic solved form with n constraints and m non-basic variables, at most *min(2n-1,n+m)* variables are redundant. In addition, the paper presents new syntactic criteria to reduce the size of the constraint system before further applications of syntactic and semantic methods.

These theoretical results have been integrated in a complete system to detect redundant linear constraints. Experimental evaluation shows that our syntactic criteria decide the redundancy of about 90% of the constraints and may produce orders of magnitude improvement in efficiency over previous work whose criteria were deciding between 24% and 56% of the constraints. In addition, the benefits of the new results increase with the size and with the sparsity of the constraint system and are especially dramatic when the number of variables is large compared to the number of constraints.

The rest of this paper is organized as follows. Section 2 reviews some basic notion and formulates the problem precisely. Section 3 reviews existing syntactic criteria which are based on the simplex solved form. Section 4 gives an overview of the lexicographic solved form and presents the new syntactic criteria to detect redundant linear constraints based on this solved form. Section 5 presents the reduction results. Section 6 presents a complete example. Section 7 presents the experimental results. Section 8 concludes the paper.

2 Background

The purpose of this section is to formalize the notion of redundant linear constraints and to reduce the detection of redundant constraints to a simpler problem. For convenience, all systems of constraints in this paper are assumed to be satisfiable. Real variables are denoted by the letter x and real numbers by the letters a, c, v, both possibly subscripted and superscripted. The notion of redundant constraint can be formalized by the following two definitions.

Definition 1 Let I be a system of linear inequalities

$$\{\ t_1 \geq 0, \ldots, t_n \geq 0\ \}$$

where $t_i = a_{i0} + \sum_{j=1}^{m} a_{ij} x_j$. The solution set of I, denoted by *sol(I)*, is defined as follows:

$$Sol(I) \ = \ \{\langle v_1, \ldots, v_m \rangle \in \Re^m \mid a_{10} + \sum_{j=1}^{m} a_{1j} v_j \geq 0, \ldots, a_{n0} + \sum_{j=1}^{m} a_{nj} v_j \geq 0\}.$$

Definition 2 Let I be a system of linear inequalities. An inequality C in I is redundant iff

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

The following lemma is a basic result on redundancy [11].

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

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

Input: $I = \{ t_1 = 0, \ldots, t_n = 0 \}$ assumed satisfiable,
where t_i is a linear term over slack and non-slack variables.
Output: A system (E, S) equivalent to I.
begin
0. $E := \emptyset$; $S := \emptyset$;
1. for each constraint C in I,
   1.1. replace every variable x in C which also appears in an equation $x = t$ of E by t and simplify.
   1.2. if C is not trivially true and if there is a non-slack variable x in C,
   rewrite C as $x = t$ (with t not containing x) and insert it into E
   else (there are only slack variables in C)
   if C is not trivially true, insert it into S.
end

Figure 1: Transformation Algorithm

Proof See [4, 11]. □

The detection of redundant constraints is conveniently performed, not on the initial system of linear inequalities I, but rather on a linear program associated with I. For a system of inequalities $I = \{ t_1 \geq 0, \ldots, t_n \geq 0 \}$, such a linear program can be obtained in two steps:

1. transform I into a system of equations I' by introducing slack variables x_k^+:

$$I' = \{ t_1 - x_1^+ = 0, \ldots, t_n - x_n^+ = 0 \}.$$

2. apply the transformation algorithm depicted in Figure 1 on I' to obtain a system (E, S).

The system (E, S^+), where $S^+ = S \cup \{x_1^+ \geq 0, \ldots, x_n^+ \geq 0\}$, is equivalent to I. S^+ is a linear program associated with I.[1]

The above transformation enables us to reduce the detection of redundant constraints in I to the detection of redundant variables in S^+ since each variable in fact corresponds to an initial inequality.

Definition 3 Let S^+ be a linear program with variables $x_1, \ldots, x_m$. Variable x_k is redundant in S^+ iff there exist nonnegative values $a_0, \ldots, a_{k-1}, a_{k+1}, \ldots, a_m$ such that

$$S \Rightarrow x_k = a_0 + \sum_{i=1, i \neq k}^{m} a_i x_i.$$

Lemma 2 Let I be a system of linear inequalities and S^+ be a linear program associated with I. $t_k \geq 0$ is redundant in I if and only if variable x_k is redundant in S^+ if and only if there exists no solution of S in which all variables are assigned nonnegative values except x_k which is assigned a negative value.

[1] In the following, we always use S^+ to denote the system S augmented with the positivity requirements on all its variables.

Proof See [7, 12]. □

The rest of this paper focuses on the problem of detecting redundant or non-redundant variables. Detecting a non-redundant variable is in fact as important as detecting a redundant variable, since it precludes the need for semantic methods on the variable. Most existing criteria, and most of our new results in fact, detect non-redundancy. Note that Lemma 2 will be used heavily in the proofs of this paper.

3 Redundancy Detection with the Simplex Solved Form

The existing syntactic criteria for redundancy elimination are based on the simplex solved form of the linear program. We briefy recall the solved form and then present the results.

Definition 4 A linear term is an expression

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

a_0 is called the *constant* and the a_i $(1 \leq i \leq m)$ are called the *coefficients*. We use $var(t)$ to represent the set of variables appearing with a non-zero coefficient in the linear term t .

Definition 5 A linear equation in *solved form* (SF for short) is an expression

$$x_b = t$$

where t is a linear term, $x_b \notin var(t)$ and 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 called *non-basic* variables.

Definition 6 A set of linear equations $S = \{x_{b_1} = t_1, \ldots, x_{b_n} = t_n\}$ is in SF iff

- $x_{b_i} = t_i$ is in SF $(1 \leq i \leq n)$;
- each basic variable appears only once in S.

We are now in position to present three syntactic criteria for redundancy. The first result concerns non-basic variables [7].

Lemma 3 Let S^+ be a system of equations in SF. A non-basic variable x_k is non redundant in S^+ if, for each equation $x_{b_j} = a_{j0} + \sum a_{ji} x_i$ 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. $|v_k| < min\{a_{j0}/a_{jk} \mid a_{jk} > 0\}$). Then apply Lemma 2. □

The second result identifies the non-redundancy of a subset of the leaving variables [7] and uses the pivotiong rule of the simplex algorithm. 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 pivoting rule of the simplex algorithm requires that, for an entering variable x_e, the leaving variable x_l be such that the ratio a_{l0}/a_{le} be greater or equal to all ratios a_{k0}/a_{ke} satisfying $a_{ke} < 0$ $(1 \leq k \leq n)$. The lemma specifies that some leaving variables are non-redundant.

Lemma 4 Let S^+ be a system in SF. The basic variable x_{b_k} of row k is non-redundant in S^+ 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 $x_i \neq x_e$ and to assign $|a_{k0}/a_{ke}| + \epsilon$ to x_e with ϵ sufficiently small. Then apply Lemma 2. □

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

Lemma 5 Let S^+ 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 Definition 3. □

4 Redundancy Detection with the Lexicographic Solved Form

Our new results are based on another solved form, the lexicographic solved form [13], which was introduced to handle strict inequalities (e.g. $>$) and disequations ($\neq$). The lexicographic solved form enables us to obtain much stronger syntactic criterion than those obtained with SF. We first recall the relevant results on the lexicographic solved form and then present our new results.

4.1 The Lexicographic Solved Form

The lexicographic solved form is a specialization of SF which imposes a more restrictive positivity requirement. It assumes a total ordering $\gg$ on the variables. In the following, we assume, for simplicity, that $x_i \gg x_j$ iff $i < j$. We now introduce the basic notions underlying the lexicographic solved form.

Definition 7 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 (in the left to right order) is positive (resp. negative). We also denote by $v \overset{L}{\geq} 0$, the disjunction $v = 0 \vee v \overset{L}{>} 0$.

Definition 8 A vector v is said to be lexicographically greater than a vector u if $v - u \overset{L}{>} 0$.

Definition 9 A negative unit vector is a vector $(0, \ldots, 0, -1, 0, \ldots, 0)$.

Definition 10 The vector v associated with a linear constraint

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

in SF is defined by

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

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

Definition 11 A linear equation C is in lexicographic solved form (LSF for short) iff

- C is in SF;
- the vector associated with C is either a negative unit vector[2] or lexicographically positive.

Definition 12 A set of linear equations $S = \{x_{b_1} = t_1, \ldots, x_{b_n} = t_n\}$ is in LSF iff

- $x_{b_i} = t_i$ is in LSF $(1 \leq i \leq n)$;
- each basic variable appears only once in S.

The basic difference between SF and LSF is that, for any constraint $x_b = t$ 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. The following theorem states that LSF is indeed a solved form.

Theorem 1 A linear program is satisfiable if and only if it can be mapped into LSF.

Proof See [13]. □

The other important property of LSF is that it is preserved through pivoting provided that a lexicographic pivoting rule be used. Since our new criteria make use of the lexicographic pivoting rule, we briefly review the basic concepts. The *lexicographic* pivoting rule [4] is defined as follows.

Definition 13 Let

$$S = \{x_{b_i} = a_{i0} + \sum_{j=1}^{m} a_{ij} x_j \mid (1 \leq i \leq n)\}$$

be a set of equations in LSF, x_k be the selected entering variable. Denote by S_k the set $\{i \mid 1 \leq i \leq n \text{ and } a_{ik} < 0\}$, by 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 x_{b_l} in such a way that u_l is lexicographically the greatest of all vectors, i.e.,

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

[2] This is necessary to represent explicit assignments of the form $x_b = 0$

Theorem 2 The lexicographic rule preserves LSF through pivoting.

Proof See [13]. □

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

4.2 Redundancy Detection With LSF

We now investigate the use of LSF for the detection of redundant and non-redundant constraints. For convenience, we assume that the system S^+ in LSF does not contain two equations which have similar right-hand sides up to multiplication by a constant and up to the value of the constant. This kind of syntactic redundancy [10] can be easily detected.

Our first result is a generalization of Lemma 4 and shows that all leaving variables, for a given ordering about to be specified, are non-redundant. This theorem produces much stronger results than Lemma 4.

Theorem 3 Let S^+ 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 x_{b_k} the corresponding leaving variable. Assume that x_j is the leading variable (possibly x_0) in the equation $x_{b_k} = a_{k0} + \sum a_{kp}x_p$. Using Lemma 2, it is sufficient to find a solution to S assigning a negative value to x_{b_k} and nonnegative values to the other variables. The proof proceeds in four steps. The first step normalizes the equations to make it easy to assign a negative value to x_{b_k}. The second step assigns values to variables x_m down to x_j. The third step decreases the value assigned to x_e so that x_{b_k} becomes negative. The fourth step completes the assignment of variables (from x_{j-1} down to x_1) to make sure that all other variables are given positive values.

The first step is a normalization of the equations. Let

$$E \;=\; \{\; x_{b_i} = a_{i0} + \sum a_{ip}x_p \mid 1 \le i \le n \;\&\; 1 \le p \le m \;\}.$$

and

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

The normalization produces a system

$$E' \;=\; \{\; c_i y_i = a'_{i0} + \sum a'_{ip}x_p \mid 1 \le i \le n \;\&\; 1 \le p \le m \;\&\; c_i > 0 \;\&\; a'_{ip} = c_i a_{ip}\}$$

which satisfies

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

for all equations in M_k. 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

$$0 < \theta_{kj} < \theta_{ij} \quad (1 \leq i \leq m \ \& \ i \neq k)$$

where

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

The assignment proceeds from m to j and makes sure that $(j \leq p \leq m)$

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

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

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. It follows immediately that $L \cap S_k = M_k$. Moreover, increasing the value of v_e can only decrease the values θ_{ij} for those equations $x_{b_i} = a_{i0} + \sum a_{ip} x_p$ 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 can be viewed as a dramatic generalization of Lemma 3, giving a very strong syntactic criteria to detect non-redundant variables. 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 3. In particular, the theorem implies that, a system in LSF with n constraints and m non-basic variables has at most *min(2n-1,n+m)* redundant variables, showing that it is a powerful tool when the number of variables is large compared to the number of constraints. Note also that, when the system is in LSF, the theorem completely subsumes Lemma 3.

Theorem 4 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 3 and to apply Lemma 2. □

Corollary 1 A system in LSF with n constraints and m non-basic variables has at most *min(2n-1,n+m)* redundant variables.

Example 1 The following table presents an application of the theorem.

	Cte	x_1	x_2	x_3	x_4	x_5	x_6
x_7	+						
x_8	0	+					
x_9	0	0	0	+			
x_{10}	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 two theorems complement very well Theorem 4 and consider leading variables. The first theorem indicates that, if we can find a variable to replace a leading variable, then the leading variable is non-redundant.

Theorem 5 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 4 with the same lexicographic ordering except that the order of variables x_i and x_j has been exchanged. □

Example 2 The following table presents an application of the theorem.

	Cte	x_1	x_2	x_3	x_4	x_5	x_6
x_7	0	+				+	
x_8	0	0	0	+		+/0	
x_9	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 theorem indicates that a leading variable always followed by a positive coefficient is non-redundant.

Theorem 6 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 4 with the same lexicographic ordering except that x_i becomes the smallest variable of non-basic variables and basic variables the smaller than x_i. □

Example 3 The following table presents an application of the theorem.

	Cte	x_1	x_2	x_3	x_4	x_5	x_6
x_7	0	0	+	0	0	+	
x_8	0	0	0	+	+		
x_9	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.

5 Syntactic Reduction with the Lexicographic Solved Form

The above theorems apply directly to the linear program in LSF. Whenever no syntactic criteria applies, it is possible to reduce, in a syntactic way, the system to be considered. This may enable further applications of the syntactic criteria or reduce the size of the system which semantic methods will be applied to. This section presents two reduction theorems for non-basic and basic variables respectively.

Theorem 7 [Non-basic Reduction Theorem] Let S be a system of equations in which x_i is a leading variable. x_i is redundant in S^+ iff x_i is redundant in the system R^+ defined as follows:

1. remove from S 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 S iff it is non-redundant in T. (The only part) Since T is a subsystem of S, all solutions of S are also solutions of T. Hence, by Lemma 2, if x_i is non-redundant in S, then it is non-redundant in T. (The if part) If x_i is non-redundant in T, by Lemma 2, it is possible to find a solution of T with a negative value for x_i and nonnegative values for all other variables. This solution can be turned into a solution of S by assigning values to the variables appearing in S 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 3.

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 2. 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 2. □

Example 4 Consider the initial system

	Cte	x_1	x_2	x_3	x_4	x_5	x_6
x_7	+						
x_8	0	0	0	+		-	
x_9	0	+					
x_{10}	0	0	0	0	+	-	
x_{11}	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
x_8	+		
x_{10}	0	+	
x_{11}	+		

Theorem 8 [Basic Reduction Theorem] Let S be a system of equations in which x_j is a basic variable. x_j is redundant in S^+ iff x_j is redundant in the system R^+ defined as follows: Remove from S all equations whose leading variable (possibly the fictitious variable associated with the constant) is lexicographically greater than the leading variable of the equation in which x_j occurs to obtain R.

Proof Let x_i be the leading variable of the constraint where x_j is in basis. By modifying the lexicographic ordering to switch x_i and x_j, the role of the two variables can be switched as well: x_i becomes the basic variable and x_j becomes the leading variable. The constraint will remain in LSF for the new ordering. Moreover, after elimination of x_i, the remaining constraints are also in LSF for the new ordering, since

1. constraints whose leading variable has rank smaller than the rank of x_i are left unchanged;
2. constraints whose leading variable was x_i have now x_j as leading variable.
3. constraints whose leading variable has rank greater than the rank of x_i keep the same leading variable.

Hence, all theorems presented before apply. The reduced system obtained by applying Theorem 7 is simply the system R after a pivoting of x_i et x_j. □

Example 5 Consider the initial system

	Cte	x_1	x_2	x_3	x_4	x_5	x_6
x_7	+						
x_8	0	0	0	+		-	
x_9	0	+					
x_{10}	0	0	0	0	+	-	
x_{11}	0	0	0	+		0	

No reduction takes place for x_7. For x_9, the following reduction occurs:

	x_1	x_2	x_3	x_4	x_5	x_6
x_8	0	0	+		-	
x_9	+					
x_{10}	0	0	0	+	-	
x_{11}	0	0	+		0	

For x_8 et x_{11}, only the following subsystem needs to be considered:

	x_3	x_4	x_5	x_6
x_8	+		-	
x_{10}	0	+	-	
x_{11}	+		0	

Finally, for x_{10}, the reduced system is simply

	x_4	x_5	x_6
x_{10}	+	-	

6 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 3 detects that x_8 is redundant. Lemma 4 detects that x_{10}, x_{11}, x_{17}, and x_{18} are non-redundant. Theorem 3, which generalizes Lemma 4, concludes that x_{13}, x_{15}, x_{19} (and of course x_{10}, x_{11}, x_{17}, and x_{18}) are non-redundant. Lemma 3 detects that x_2 and x_7 are non-redundant. Theorem 4, generalizing Lemma 3, detects the non-redundancy of x_6 (in addition of those already detected by Lemma 3). Lemma 5 detects that x_5 is redundant. Theorem 5 detects that x_1 is non-redundant. Theorem 6 detects the non-redundancy of x_4.

At this point, we can use the non-basic 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 3, 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 6 concludes that x_3 is non-redundant.

An alternative consists in applying successively Theorem 7 to remove x_4, Definition 3 to remove x_{19}, Theorem 7 to remove x_5, and finally Theorem 6 to conclude that x_3 is non-redundant as shown in the following figure.

	x_3	x_6
x_{18}	2	2

The only variables which are left pending are x_9,x_{12},x_{14}, and x_{16}. These variables will probably need to be checked using semantic methods. However, for x_{14}, semantic methods apply on the sub-system:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_{11}	2	-1				3	-1
x_{12}				1		2	-3
x_{14}	1					2	-1
x_{15}					1		-3
x_{16}				2	1		-3
x_{18}			2	-1	-1	2	
x_{19}			1	-2	1		

and for x_{12} and x_{16}, they apply on

	x_4	x_5	x_6	x_7
x_{12}	1		2	-3
x_{15}		1		-3
x_{16}	2	1		-3

Remark 1 It is important to note that that the syntactic criteria depend on the lexicographic ordering. Hence, two different orderings can produce different results. For instance, by switching the rank of x_4 et x_5, Theorem 3, when applied with x_7 as entering variable, concludes that x_{12} is non-redundant (instead of x_{15}). Similarly, by ranking x_1 and x_2 immediately after x_7, Theorem 3 concludes that x_{14} is non-redundant.

7 Experimental Results

The above theoretical results have been implemented in a complete system to detect redundant constraints. The experimental results reported here use randomly generated constraint systems with various sparsity configurations and various ratios n/m, where n is the number of constraints and m is the number of non-basic variables. The complete results are reported in [1].

7.1 Applicability of the Theoretical Results

The first results we report are the applicability of the various theoretical results, i.e. how many times a given definition, lemma, or theorem can be used. Table 1 gives the results for a variety of parameter configurations. These results are not intended to be comprehensive (see [1] for a more complete coverage) but they give a good picture on the applicability of the results. In the table, `n` is the number of constraints, `m` is the number of variables, `%z` is the percentage of zeros, `s` is the number of linear programs generated, the next 9 columns describe the number of successful applications of the various definitions, lemmas, and theorems over all linear programs, `old` reports the number of successful applications of Definition 3 and Lemmas 3, 4, and 5 (i.e. previous work), while `new` reports the number of successful applications of Definition 3, Lemma 5, and Theorems 3, 4, 5, 6, 7, 8 and `v` is the total number of variables in all the systems. Table 2 reports the same results in percentage of the total number of variables. The following two remarks are important to interpret the results correctly.

- The results of Theorems 7 and 8 contain only the number of variables which, after reduction, are detected as redundant or non-redundant by using syntactical methods only. Of course, in the complete system, they are used to reduce the system before applying semantic methods whenever no semantic criteria applies.
- The definition, lemmas, and theorems are applied in the following order.

```
old:  Definition 3, Lemmas 3, 4, and 5;
new:  Theorem 4, Definition 3, Lemma 5, and Theorems 3, 6, 5, 7, and 8.
```

The application order is mainly used for efficiency reasons, since most theorems are independent. For instance, Definition 3 and Lemma 5 detect redundancy while Theorems 4 and 3 detects non-redundancy. Theorems 4, 5 and 6 on the one hand and Theorem 3 on the other hand are independent. Theorem 4 is independent from Theorems 5 and 6. Theorems 5 and 6 are not independent but the overlap should be small.

The results indicate the dramatic improvement produced by our new syntactic criteria. In all cases reported, the status of more than 73% of variables is decided using only syntactic criteria. In many cases, the status of more than 89% of the variables are decided. This contrasts with existing work where only 24% of the variables are decided in some cases and no more than 56% of the variables in all cases. Theorems 3 and 4 are particularly effective. More generally, the following comments can be made on each of the results.

- **Definition 3**: This result is particularly appropriate when the ratio n/m is large and/or when the system is very sparse. A typical example is the last line of the tables.

n	m	%z	s	d. 3	l. 3	l. 4	l. 5	t. 3	t. 4	t. 5	t. 6	t. 7	t. 8	old	new	v
15	30	25	50	1	142	288	0	332	665	64	1	20	18	431	1100	1500
15	30	50	50	10	86	256	9	404	585	85	8	62	85	361	1428	1500
15	30	75	50	151	199	285	95	399	446	117	15	47	110	740	1418	1500
15	45	75	50	19	275	444	21	582	1188	176	20	103	80	759	2189	2250
30	45	75	50	336	60	237	156	549	336	156	18	80	381	789	2012	2250
25	50	75	14	20	31	120	16	204	239	50	3	52	77	187	661	700
50	60	80	42	1063	30	122	206	362	42	153	4	25	444	1421	2268	2520

Table 1: Applicability of the Theoretical Results

n	m	%z	s	d. 3	l. 3	l. 4	l. 5	t. 3	t. 4	t. 5	t. 6	t. 7	t. 8	old	new
15	30	25	50	0	9	19	0	22	44	4	0	1	1	29	73
15	30	50	50	1	6	17	1	27	39	6	1	4	6	24	95
15	30	75	50	10	13	19	6	27	30	8	1	3	7	50	94
15	45	75	50	1	22	20	1	26	53	8	1	4	3	34	97
30	45	75	50	15	3	11	7	24	15	7	1	4	17	35	89
25	50	75	14	3	4	17	2	29	34	7	0	7	11	27	94
50	60	80	42	42	1	5	8	16	1	6	0	1	18	56	89

Table 2: Applicability of the Theoretical Results in Percentage

- **Lemma 3**: This lemma is relatively effective and can be tested very efficiently. However, when the system is in LSF, it is completely subsumed by Theorem 4 which is much stronger.
- **Lemma 4**: This lemma is completely subsumed by Theorem 3 which is much stronger.
- **Lemma 5**: This lemma can be tested very efficiently and produces good results when the system is very sparse. It gives poor results on dense systems.
- **Theorem 3**: This theorem can be tested reasonably efficiently and produces very good results in almost all cases. Its results are reasonably uniform over all configurations (detecting between 20% and 30% of non-redundant variables).
- **Theorem 4**: This theorem can be tested very efficiently and is almost always extremely effective. The smaller the ratio n/m, the more effective this result, as shown for instance in line 4 of the tables. The only cases where it is not very effective is when the ratio n/m is close to 1 as in the last line of the tables.
- **Theorems 5 and 6**: These theorems can be tested efficiently and can be seen as extensions of Theorem 4. Theorem 5 produces very good results in general and compensates the relative poor behaviour of Theorem 4 on problems where the ratio n/m is close to 1 as in the last line of the tables.
- **Theorems 7 and 8**: The cost of these theorems is high but still lower than the semantic methods. They are also very effective in reducing the size of the systems to consider and their effectiveness increases with sparsity.

n	m	%z	s	sem.	lem.	the.	sem/the	lem/the
15	30	25	50	1261	1117	729	1.74	1.53
15	30	50	50	808	472	438	2.09	1.84
15	30	75	50	196	148	62	3.26	2.46
15	45	75	50	721	561	151	4.77	3.71
30	45	75	50	1776	1644	819	2.16	2.01
25	50	75	14	918	808	438	2.97	2.51
50	60	80	42	822	335	276	2.41	2.17
15	100	80	50	2758	1933	172	16.03	11.24

Table 3: Efficiency of the Methods: Comparaison of the three Methods

7.2 Efficiency Results

We now report some efficiency results on the same sample problems. This should give an idea of the speed-ups that can be obtained from our results, although results clearly vary from one implementation to another. Table 3 reports the CPU time in seconds on a `Next` workstation using infinite precision numbers for three methods:

1. the `semantic` method (no syntactic criteria);
2. the `lemma` method using Definition 3 and the lemmas before applying semantic methods;
3. the `theorem` method using all results of this paper (including the reduction theorem) before applying semantic methods.

The table describes the same initial parameters as in previous tables. In addition, it reports the time taken by each of the methods and the ratio of the various methods. It indicates that the new results improve substantially existing work, the new system being more than 11 times faster on the last example. Moreover, the following conclusions seem to emerge

- **Number of constraints/number of non-basic variables:** The smaller this ratio, the larger the efficiency gap between the three methods. To confirm this hypothesis, we generate 250 sparse systems (75% of zeros) having between 30 and 100 variables. The following table shows the result normalized on the `theorem` method, clearly validating the hypothesis. When the ratio is 0.15, the `theorem` method is about 11 and 16 times faster than the `lemma` and `semantic` methods. This result is mainly due to Theorems 4 and 5.

n/m	the.	lem.	sem.
0.85	1	1.8	2.5
0.66	1	2.0	2.8
0.50	1	2.4	3.8
0.33	1	3.7	5.5
0.15	1	11	16

- **Sparsity:** The sparser the system, the larger the efficiency between the three methods. This is illustrated in the first three lines of Table 3.

- **Number of variables:** In general, the difference in efficiency between the three methods increase with the number of variables. Of course, specific results depend on the two parameters discussed above.

8 Conclusion

This paper reconsiders the problem of detecting redundant constraints. It proposes a number of new syntactic criteria to detect redundant and non-redundant constraints and two reduction theorems that can be used before further applications of the syntactic criteria and/or semantic methods. The criteria are all based on a lexicographic solved form which appears as a versatile tool to detect redundancy in linear programs. Experimental results indicate that the theoretical results apply much more often than previously proposed syntactic criteria, that they decide the redundancy of about 90% of the constraints in most cases (compared to between 24% and 56% in previous work) and that they may produce orders of magnitude improvement in efficiency. In addition, their effectiveness seems to improve with the sparsity and when the number of constraints decreases compared to the number of variables. In particular, Theorem 4 was shown to be a powerful tool whenever the number of constraints is small compared to the number of variables and can produce easily orders of magnitude improvement in these configurations. It also entails that a system in LSF with n constraints and m non-basic variable has at most $min(2n\text{-}1, n+m)$ redundant variables. Theorem 3 was shown to be effective on almost all cases.

It is interesting to note that the problem of redundancy is sometimes connected to the problem of implicit equalities (see for example [12]). The lexicographic solved form seems to be an interesting tool to deal with both of these issues since it was originally proposed as an efficient way of detecting implicit equalities inside the simplex algorithm. Hence, it would be interesting to study other related applications.

It is also worthwhile pointing out 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.

Finally, note that advanced linear programming software such as `CPLEX`$^{\texttt{TM}}$ include a number of transformations on the initial problem description (this is an optional feature which can be turned off if necessary). These transformations are syntactic and partly based on [2]; they include simple removal of columns and rows, detections of infeasibility or redundancy by bound reasoning, derivation of new bounds on the variables, and elimination of free variables. Our techniques are complementary to those; they do not work on the initial problem statement but rather on its solved form. It is thus conceivable that our techniques, because of their syntactic nature and efficient implementation, could find their way in these products as well.

Acknowledgements

We would like to thank Jacques Cohen, Alain Colmerauer, and Timothy Hickey for interesting discussions and comments and the reviewers for identifying a possible source of confusion in the paper. Special thanks to the `CPLEX` support staff for providing details on the `PreSolve` option of their software. Jean-Louis Imbert was partly supported by the INRIA institute under grant 13001-5847. pascal van Hentenryck was partly supported by the National Science Foundation

under a National Young Investigator Award and by the Office of Naval Research under grant Grant N00014-94-1-1153.

References

[1] J. Alziary de Roquefort. Les apports des méthodes syntaxiques pour la détection des redondances dans les systèmes d'inéquations linéaires. *Mémoire de DEA*, Laboratoire d'Informatique de Clermont-Ferrand, France, 1994.

[2] Brearly, Mitra, and Williams. Analysis of Mathematical Programming Problems Prior to Applying the Simplex Method *Mathematical Programming*, 8, p 54–83, 1975.

[3] P. Cousot and N. Halbwachs. Automatic Discovery of Linear Restraints among Variables of a Program In *Proceedings of the fifth Annual ACM Symposium on Principles of Programming languages*, Tucson, Arizona, January 23-25, 1978.

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

[5] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of Linear Hybrid Systems by Means of Convex Approximations In *Proceedings of the first Symposium on Static Analysis (SAS-94)*, Namur, Belgium, September, 1994.

[6] J.L. Imbert. About Redundant Inequalities Generated by Fourier's Algorithm. *AIMSA'90, Fourth International Conference on Artificial Intelligence: Methodology, Systems, Applications*, Albena-Varna, Bulgaria, September 1990

[7] 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.

[8] D.A. Kolher. *Projection of Convex Polyhedral Sets* Ph.D. Thesis, University of California, Berkeley, 1967.

[9] J.L. Lassez and K. McAloon. Applications of a Canonical Form for Generalized Linear Constraints. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, December 1988.

[10] J-L. Lassez, T. Huynh, and K. McAloon. Simplification and Elimination of Redundant Linear Arithmetic Constraints. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research*, p 73–87, MIT Press, Cambridge, USA, 1993.

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

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

[13] P. Van Hentenryck and T. Graf. Standard Forms for Rational Linear Arithmetics in Constraint Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 5(2-4), 1992.