Dense Time and Temporal Constraints With $\neq$*

Manolis Koubarakis
Computer Science Division
Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73
Athens GREECE
koubarak@theseas.ntua.gr

Abstract

In some temporal reasoning systems, inequations can give rise to disjunctions of inequations when variable elimination is performed. Motivated by this observation, we extend previous research on temporal constraints by considering disjunctions of inequations (under the assumption that time is dense.) We present results on consistency checking, canonical forms and variable elimination for this new class of temporal constraints.

1 INTRODUCTION

In recent years temporal reasoning has received much attention from the artificial intelligence and database community (see [SG88, Sno90] for surveys). This is a rather natural trend since reasoning about time is essential in many applications (e.g., natural language understanding, planning, scheduling etc.). In the artificial intelligence community in particular, some researchers have introduced binary constraint networks as reasoning tools for problems involving temporal constraints [All83, VKvB89, LM88, DMP91, KL91, Mei91]. Much of this work has its origins in previous research on general constraint satisfaction problems [Mon74, Mac77, DP88]. We follow this line of work and extend it by considering a more expressive class of temporal constraints.

In our work, time is assumed to be linear, dense and unbounded. Points are the only time entities. Temporal information is represented in terms of *inequalities* and *disjunctions of inequations*. Two kinds of inequalities are allowed: (i) bounds on the distance between two time points e.g., $t_1 - t_2 \leq 5$, and (ii) bounds on the position of a time point on the time-line e.g., $t_1 \geq 5$. *Inequations* are constraints of the form $\alpha \neq \beta$. Similarly with inequalities, two kinds of inequations are allowed: $t_1 - t_2 \neq r$ or $t \neq r$. Although inequations have been considered in the past in the context of binary constraint networks (e.g., by [VKvB89, Mei91]), disjunctions of inequations have been neglected. However, if one considers temporal formalisms or systems with more expressive representation/query languages (e.g., TMM [DM87], Telos [MBJK90] and the model of [Kou92b]), then one cannot deal with binary inequation constraints and avoid non-binary ones. In such systems there are queries that can only be answered by eliminating variables from a set

*Appears in: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), pages 24-35, Morgan Kaufmann, October 1992.

of temporal constraints. But when variable elimination is performed, an inequation can give rise to a disjunction of inequations.[1]

The basic properties of disjunctions of linear inequations have already been studied by [LM89] who considered generalized linear constraints (i.e., linear inequalities and disjunctions of linear inequations). [LM89] observed that disjunctions of inequations can be treated independently of one another for deciding consistency. An important consequence of this fact is that one can decide in polynomial time whether a set of temporal constraints is consistent. Building on the results of [LM89], we were able to make the following contributions:

- We applied the results of [LM89] to the case of temporal constraints (section 3). In particular, we developed algorithms for deciding consistency and computing canonical forms for sets of temporal constraints.
- We developed a variable elimination algorithm for temporal constraints (section 4). This algorithm can also be applied to the more expressive class of generalized linear constraints.
- We showed that the presence of disjunctions of inequations makes variable elimination intractable (theorem 4.4). However, for the limited class of simple temporal constraints (which corresponds to the full point algebra of [VKvB89]) variable elimination is tractable (corollary 5.1). The proof of this result is indirect and demonstrates that a decomposable constraint set equivalent to a given set of simple temporal constraints can be found in polynomial time (theorem 5.1). The decomposability of sets of simple temporal constraints has been an open problem since [vB90a].

The organization of the paper is as follows. Section 2 introduces temporal constraints. Section 3 discusses consistency checking and canonical forms for temporal constraints. Section 4 presents our results on variable elimination. Section 5 discusses variable elimination for simple temporal constraints. Section 6 summarizes our contributions and presents some directions for future research. Finally, appendix A contains the proofs of the most important results.

2 TEMPORAL CONSTRAINTS

At first, we present a language $\mathcal{L}$ for talking about time. We consider time to be linear, dense and unbounded. *Points* will be our only time entities. *Interval* information will be represented in terms of endpoint information. Points are identified with the rational numbers but our results still hold if points are identified with the reals.

We begin by giving a formal definition of $\mathcal{L}$. $\mathcal{L}$ is a first order calculus with equality. The logical symbols of $\mathcal{L}$ include parentheses, quantifiers, sentential connectives, a countably infinite set of variables V and $=$ (the equality symbol). The non-logical symbols of $\mathcal{L}$ include a countably infinite set of constants (the rational numerals), a function symbol $-$ of arity 2 and a predicate symbol $<$ of arity 2. The symbols of $\mathcal{L}$ are interpreted with respect to the structure $\mathcal{T}$ which embodies our assumptions for time. $\mathcal{T}$ assigns to each constant a member of the set Q (i.e., a rational number). To function symbol $-$, $\mathcal{T}$ assigns the function $-_{\mathsf{Q}}$ which is the subtraction operation over the rationals. To predicate symbol $<$, $\mathcal{T}$ assigns the relation $<_{\mathsf{Q}}$ ("less than") over the rationals.

A *variable assignment* v is a function $v : V \rightarrow \mathsf{Q}$ where V is the set of variables of $\mathcal{L}$. We use $\mathcal{T} \models \phi[v]$ to denote that the structure $\mathcal{T}$ *satisfies* ϕ with variable assignment v. Similarly,

[1] Van Beek makes a very similar observation in the context of decomposable point networks [vB90a].

$\mathcal{T} \models \sigma$ denotes that the sentence σ is *true* in the structure $\mathcal{T}$. We will also write $\phi \models \psi$ whenever $\mathcal{T} \models \phi \supset \psi$. Satisfaction and truth for the structure $\mathcal{T}$ are defined in the usual way [End72].

We will now restrict our attention to certain formulas of $\mathcal{L}$ which will be used to express temporal information. *Temporal constraints* are formulas of $\mathcal{L}$ which fall under any of the following categories:

- *Inequalities*: $t_1 - t_2 \leq r$ where t_1, t_2 are variables and r is a constant (t_1 or t_2 can be omitted).[2]
- *Disjunctions of inequations*:

$$t_1 - t_1' \neq r_1 \ \vee \ \cdots \ \vee \ t_n - t_n' \neq r_n$$

  where $t_1, \cdots, t_n, t_1', \cdots, t_n'$ are variables and $r_1, \cdots, r_n$ are constants (t_i or t_i' can be omitted).
- *true* and *false* with the obvious semantics.

Using the above constraints equalities can be written as conjunctions of two inequalities. Similarly, strict inequalities can be written as conjunctions of an inequality and an inequation.

A temporal constraint of the form $t_1 - t_2 \leq r$ or $t_1 - t_2 \neq r$ will be called *atomic*. A temporal constraint of the form $t_1 \leq t_2$ or $t_1 \neq t_2$ is called *simple*. Simple temporal constraints have the same expressive power with the full point algebra of [VKvB89]. If c is a disjunction of inequations then $\overline{c}$ denotes the *complement* of c i.e., the conjunction of equations obtained by negating c.

For a set C of temporal constraints in variables $x_1, \ldots, x_n$ the *solution set* of C is defined as $Sol(C) = \{(\tau_1, \ldots, \tau_n) : \ \tau_i \in \mathbf{Q} \text{ and for every } c \in C, \ \mathcal{T} \models c[x_1 \leftarrow \tau_1, \ldots, x_n \leftarrow \tau_n]\}$. Each member of $Sol(C)$ is called a *solution* of C. If C is a set of equalities in n variables, the solution set of C is an affine subset of $\mathbf{Q}^n$. If C is a set of inequalities in n variables, the solution set of C is a convex polyhedron in $\mathbf{Q}^n$. If C is a set of disjunctions of inequations, the solution set of C is $\mathbf{Q}^n \setminus Sol(\{\overline{c} : \ c \in C\})$.

A set of temporal constraints is called *consistent* or *satisfiable* if and only if $Sol(C) \neq \emptyset$. Two sets of constraints are called *equivalent* if and only if they have the same solution set. A set of constraints is the algebraic counterpart of the logical conjunction of its members. Thus, in the following sections, we will frequently mix the terms "set of constraints" and "conjunction of constraints".

Inequality constraints have so far been studied with the help of constraint networks [LM88, vB90b, DMP91, KL91, Mei91]. A *binary constraint network* is a directed graph whose nodes represent variables and whose arcs represent binary constraints between these variables. For example, if the constraints between variables x_i and x_j are $x_i - x_j \leq 3$ and $x_j - x_i \leq 0$ then arc $j \rightarrow i$ is labeled with 3 and arc $i \rightarrow j$ is labeled with 0. Inequalities with only one variable are accomodated in these networks with the introduction of an auxiliary variable $x_0 = 0$.[3]

The notions of solution, consistency and satisfiability can also be defined for constraint networks in the obvious way. Two networks are called *equivalent* if they have the same set of solutions. Let us now assume that N and N' are networks with the same set of variables X. If $x_i, x_j \in X$ then $c_N(x_i, x_j)$ will denote the conjunction of the constraints between x_i and x_j in N. We will say that N is *tighter* than N' (denoted $N \subseteq N'$) if and only if for each pair of variables $x_i, x_j \in X$, $c_N(x_i, x_j) \models c_{N'}(x_i, x_j)$. For each network N there is a unique equivalent network N' which is minimal with respect to $\subseteq$. This network is called the *minimal network*. If C is a set of

[2] For conjunctions of inequalities, we will usually write $\alpha \leq \beta \leq \gamma$ instead of $\alpha \leq \beta \ \wedge \ \beta \leq \gamma$.

[3] [DMP91] call this graph the *distance graph* of the constraints. They reserve the term binary constraint network for a different but equivalent representation.

inequalities then $NET_m(C)$ denotes its minimal network representation. If N is a network then $Constraints(N)$ will denote the set of constraints represented by N.

Let us now compare the class of temporal constraints considered in this paper to the classes considered previously by other researchers. Inequalities have been investigated thoroughly in the past [DMP91, VKvB89, Mei91, KL91] but inequations have received less attention. Inequations of the form $t_1 \neq t_2$ have been considered by [vB90a] in the context of point networks. Inequations of the form $t_1 - t_2 \neq r$ or $t \neq r'$ (r, r' are real constants) can be represented in the binary constraint networks of [DMP91] and the generalized temporal constraint networks of [Mei91]. In addition, [Mei91] studies inequations of the form $t \neq r$ (r a real constant) in the context of point networks with *almost-single-interval domains*. To the best of our knowledge, no one has considered disjunctions of inequations so far. Nevertheless, disjunctions of inequations are useful for the representation of several temporal situations as it is demonstrated by the following examples:

- Jobs $J1$ and $J2$ should not both start at the time job J starts: $J_s \neq J1_s \lor J_s \neq J2_s$
- It took robots A and B at most five minutes to finish job J; however, one of the robots worked for a shorter time:
  $A_{end} - A_{begin} \leq 5,\ B_{end} - B_{begin} \leq 5,$
  $A_{end} - A_{begin} \neq 5 \lor B_{end} - B_{begin} \neq 5$
- Intervals i and j are not equal:[4]
  $i_L \neq j_L \lor i_R \neq j_R$
- Let us assume that the following information is given to a natural language understanding system. "David and Murray arrived at the accident scene together. Murray left at least five minutes later than Bill. Bill left at least three minutes later than David. John and Bill did not leave the scene together." If we assume that $scene(p, t_L, t_R)$ represents the fact that person p was at the accident scene for an interval whose endpoints are t_L and t_R then this information can be represented by the following first order logic formulas (in clausal form):
  $scene(david, d_L, d_R),\ scene(murray, m_L, m_R),$
  $scene(john, j_L, j_R),\ scene(bill, b_L, b_R),$
  $d_L < d_R,\ m_L < m_R,\ b_L < b_R,\ j_L < j_R,$
  $d_L = m_L,\ m_R - b_R \geq 5,\ b_R - d_R \geq 3,\ b_R \neq j_R.$

  In these formulas, *david*, *murray*, *john* and *bill* are constants different from each other (unique names assumption [Rei80]) but $d_L,\ d_R, \ldots, b_L,\ b_R$ are Skolem constants characterized only by the given constraints.

  Let us now assume that the query "What are the *possible* times that John, David and Murray were at the scene ?" is posed. To answer this query, we have to compute all 6-tuples $(J_L,\ J_R,\ D_L,\ D_R,\ M_L,\ M_R)$ which satisfy the following constraints:
  $D_L < D_R,\ M_L < M_R,\ b_L < b_R,\ J_L < J_R,$
  $D_L = M_L,\ M_R - b_R \geq 5,\ b_R - D_R \geq 3,\ b_R \neq J_R.$

  Obviously this answer set is *infinite*. In the temporal deductive database model of [Kou92b] this query will be answered by returning a *finite representation* of the answer set.[5] This representation is computed by *eliminating* b_L and b_R from the above set of constraints to

[4] In this and the following example i_L and i_R denote the left and right endpoint of any interval i respectively.

[5] The TMM system [DM87] also offers a limited form of this query answering capability. But TMM's representation language does not support inequation constraints.

obtain:
$D_L < D_R,\ M_L < M_R,\ J_L < J_R,\ D_L = M_L,$
$M_R - D_R \geq 8,\ J_R - D_R \neq 3\ \vee\ J_R - M_R \neq -5.$

The last example is important. It shows that an inequation can give rise to a disjunction of inequations when variable elimination is performed. This fact motivated the work presented in the following sections.

3 DECIDING CONSISTENCY AND COMPUTING CANONICAL FORMS FOR TEMPORAL CONSTRAINTS

In this section, we discuss two problems: (i) deciding whether a set of temporal constraints is consistent, and (ii) computing the first and second canonical form for a consistent set of temporal constraints. Since most of our results are consequences of [LM89], our presentation will be rather terse. The reader can consult [LM89] and [Kou92a] for proofs and more details.

The following result of [LM89] shows that disjunctions of inequations can be treated independently of one another for deciding consistency.

Lemma 3.1 *(Independence of disjunctions of inequations.) Let C_i be a set of inequalities and C_n be a set of disjunctions of inequations. The set $C_i \cup C_n$ is consistent if and only if C_i is consistent and each disjunction of inequations in C_n is consistent with the inequalities in C_i.*

This lemma is very important since it allows one to avoid the combinatorial explosion resulting from translating inequations into disjunctions of strict inequalities.

We will now define the first canonical form for temporal constraints. In a temporal reasoning system (e.g., TMM or a system based on the model of [Kou92b]), a canonical form can be useful for answering queries, standardizing output and determining the equivalence of two sets of temporal constraints. In previous work on inequality constraints the role of a canonical form was played by the minimal network representation [DMP91]. Before we proceed, we need a few definitions.

A set of equalities in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *solved form* if and only if it is

$$\{y_1 = x_{j_1} + r_1,\ y_2 = x_{j_2} + r_2, \ldots, y_m = x_{j_l} + r_m\}$$

where $y_1, \ldots, y_m, x_1, \ldots, x_k$ are distinct variables, $\{x_{j_1}, \ldots, x_{j_l}\} \subseteq \{x_1, \ldots, x_k\}$ and $r_1, \ldots, r_m$ are constants. The variables $y_1, \ldots, y_m$ are called the *eliminable* or *bound* variables and $x_1, \ldots, x_k$ are called the *parameters* of the solved form. A disjunction of inequations is in solved form if and only if its complement is in solved form.

A disjunction of inequations c is called C_i*-precise* (or simply *precise* when C_i is understood) if and only if it is not implied by the inequalities in C_i, and the affine closure of $Sol(C_i) \cap Sol(\overline{c})$ is $Sol(\overline{c})$ (the latter requirement simply says that c is given in its smallest possible dimension).

A constraint c is *redundant* in the set of constraints C if and only if $Sol(C) = Sol(C - \{c\})$.

A set C of temporal constraints in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *first canonical form* if and only if it does not contain redundant constraints and consists of (i) a set of equalities C_e in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ in solved form, (ii) a set of inequalities C_i in variables $x_1, \ldots, x_k$ which does not contain any implicit equalities,[6] and (iii) a set of C_i-precise disjunctions of inequations in variables $x_1, \ldots, x_k$ in solved form.

[6] If C is a set of inequalities then $x_i - x_j \leq r \in C$ is called an *implicit equality* if and only if $C \models x_i - x_j = r$.

Algorithm First_Canonical_Form

Input: A set of temporal constraints $C = C_e \cup C_i \cup C_n$ where C_e is a set of equalities, C_i is a set of inequalities and C_n is a set of disjunctions of inequations.

Output: INCONSISTENT if C is inconsistent, otherwise a constraint set equivalent to C in first canonical form.

Method:
Step 1: Detect possible inconsistencies in $C_e \cup C_i$.
$N := NET_m(C_e \cup C_i)$.
If N contains an inconsistency then
    Return INCONSISTENT
EndIf

Step 2: Find all implicit/explicit equalities of $C_e \cup C_i$ and return them in solved form.
$C_e^1 := Equalities(N)$.

Step 3: Eliminate all bound variables from C_i and C_n.
$C_i^1 := \emptyset$; $C_n^1 := \emptyset$
For every equation $y = x + r$ in C_e^1 do
    For every $c \in C_i$ do
        If y appears in c then
            Substitute $x + r$ for y in c
            Simplify c and add it to C_i^1
        EndIf
    EndFor
    For every $c \in C_n$ do
        If y appears in c then
            Substitute $x + r$ for y in c; Simplify c
            If c evaluates to $0 \neq 0$ then
                Return INCONSISTENT
            Else
                Add c to C_n^1
            EndIf
        EndIf
    EndFor
EndFor

Step 4: Remove redundant inequalities.
Call $Redundant\_Inequalities(C_i^1)$

Figure 1: Computing the first canonical form

Algorithm First_Canonical_Form (continued)

Step 5: Determine which disjunctions of inequations are not implied by the inequalities in C_i^1. Transform these constraints into precise ones.

Initialize C_n^2 to $\emptyset$
For each $c \in C_n^1$ do
    Negate c to obtain a set of equalities E
    $N := NET_m(E \cup C_i^1)$
    If there is an inconsistency in N then
        Discard c (c is a tautology or it is
        already implied by C_i^1)
    Else
        $E' := Equalities(N)$
        Negate the equalities of E' to obtain
        a precise constraint c. Add c to C_n^2.
    EndIf
EndFor

Step 6: Remove redundancy from C_n^2.
Call $Redundant\_Inequations(C_n^2)$

Step 7: Return the normal form of C.
Return $(C_e^1,\ C_i^1,\ C_n^2)$

Figure 2: Computing the first canonical form

Figures 1 and 2 present an algorithm for computing the first canonical form of a set of temporal constraints. We now discuss this algorithm in detail and calculate its time complexity.

In this and subsequent sections, complexity bounds for our algorithms are calculated in terms of *cardinality, size* and *number of variables* of constraint sets. The cardinality of a set of constraints C is denoted by $|C|$. The number of variables in a set of constraints C is denoted by $v(C)$. The size of a constraint set C, which is denoted by $\|C\|$, is the sum of the sizes of its elements. The size of a constraint is the number of atomic constraints appearing in its disjunctive normal form. For example, $\|x_1 - x_2 \leq 3\| = 1$, $\|2 \leq x_1 - x_2 \leq 3\| = 2$ and $\|x_1 \neq x_2 \ \vee \ x_1 \neq x_3\| = 2$. The following lemma will be useful in calculating complexity bounds. Its proof is an easy consequence of the previous definitions.

Lemma 3.2
1. *For every set of constraints C, $v(C) \leq 2\,\|C\|$.*
2. *If C is a non-redundant set of inequalities then $|C| \leq v(C)^2 + v(C)$.*
3. *If C is a set of atomic constraints then $\|C\| = |C|$.*
4. *If C is a set of disjunctions of inequations in solved form then $\|C\| \leq |C|\,v(C)$.*

Initially, the algorithm discovers whether there is an inconsistency in the constraints $C_e \cup C_i$. This step takes $O(v(C_e \cup C_i)^3)$ time. Step 2 computes the solved form of all explicit and implicit equalities in the constraints. This is done by algorithm *Equalities* which can be found in [Kou92a]. This step takes time $O(v(C_e \cup C_i)^3)$ and generates at most $\left|C_e^1\right| = O(v(C_e \cup C_i))$ equalities. Step 3 eliminates all bound variables of C_e^1 from $C_i \cup C_n$. This step takes $O(v(C_e \cup C_i)(|C_i| + |C_n|\,v(C_n)))$ time. Step 4 calls algorithm *Redundant_Inequalities* (given in [Kou92a]) to remove redundant inequalities. This step takes $O(|C_i|\,v(C_i)^3)$ time. Step 5 rewrites each disjunction of inequations in its precise form. This step takes $O(|C_n|\,(v(C_i \cup C_n)^3)$ time. Step 6 calls algorithm *Redundant_Inequations* (given in [Kou92a]) removes redundancy from disjunctions of inequations. This step takes $O(|C_n|^2\,v(C_n)^2)$ time.

Using lemma 3.2, we can now conclude that the algorithm takes $O(\|C\|^4)$ time. Thus we have proved the following theorem.

Theorem 3.1 *Deciding consistency and computing the first canonical form of a set of temporal constraints can be done in $O(\|C\|^4)$ time.*

Example 3.1 For the constraint set $\{x_6 - x_3 = 7,\ x_5 \leq x_1,\ x_1 \leq x_5,\ x_5 \leq x_3,\ x_3 \leq x_2,\ x_3 - x_2 \leq 4, x_1 \neq x_2\}$, *First_Canonical_Form* returns $\{x_5 = x_1,\ x_6 = x_3 + 7,\ x_1 - x_3 \leq 0,\ x_3 - x_2 \leq 0,\ x_2 - x_1 \neq 0 \ \vee \ x_3 - x_1 \neq 0\}$.

The following uniqueness theorem is a consequence of [LM89]. The assumption underlying the proof of the theorem is that there is a fixed order of variables and this order is taken into account consistently when equations or inequations are transformed into solved form (e.g., in example 3.1 the lowest numbered vertex is always chosen to be a parameter).

Theorem 3.2 *For equivalent sets of constraints in the same set of variables, the algorithm First_Canonical_Form outputs identical canonical forms.*

The next theorem calculates the complexity of deciding whether a constraint is implied by a constraint set in first canonical form.

Theorem 3.3 *Let us assume that the first canonical form of C is (C_e, C_i, C_n). Then the following statements are true. For statements 2 and 3 we assume that x_i and x_j are parameter variables.*
1. *$C \models x_i - x_j = r$ if and only if $C_e \models x_i - x_j = r$. This entailment can be checked in $O(|C_e|)$*

time.
2. $C \models x_i - x_j \leq r$ *if and only if* $C_i \models x_i - x_j \leq r$*. This entailment can be checked in* $O(v(C_i)^3)$ *time.*
3. If c *is a disjunction of inequations,* $C \models c$ *if and only if* $C_i \models c$ *or* $C_n \models c'$ *and* c' *is the precise form of* c*. This entailment can be checked in* $O(v(C_i)^3 + |C_n|\, v(c)v(C_n))$ *time.*

Let us now assume that C is a set of inequalities and compare the first canonical form of C with the minimal network $NET_m(C)$. The network $NET_m(C)$ can be built in $O(v(C)^3)$ time. It contains redundant constraints, but this allows us to determine in constant time whether an inequality is implied by C. On the other hand, the first canonical form of C can be built in $O(|C|\, v(C)^3)$. It does not contain redundancy and requires $O(v(C)^3)$ time for answering the same query (theorem 3.3). This suggests that the following combination of the two representations can be advantageous.

A set C of temporal constraints in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *second canonical form* if and only if it consists of (i) a non-redundant set of equalities C_e in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ in solved form, (ii) a set of inequalities C_i in variables $x_1, \ldots, x_k$ such that C_i does not contain implicit equalities and $C_i = Constraints(NET_m(C_i))$, and (iii) a set of C_i-precise disjunctions of inequations in variables $x_1, \ldots, x_k$ in solved form. The second canonical form can be built in $O(\|C\|^4)$ time by a variation of the algorithm for the canonical form. In this case, determining whether an inequality is implied by C can be done in constant time. Determining whether an equality or a disjunction of inequations is implied by C is as for the first canonical form.

4 VARIABLE ELIMINATION FOR TEMPORAL CONSTRAINTS

In this section, we present an algorithm for eliminating a variable from a set of temporal constraints. Variable elimination is the algebraic counterpart of the quantifier elimination operation of Mathematical Logic and the projection operation of Geometry. We will say that the variable x_1 has been *eliminated* from a set of constraints C with variables $x_1, x_2, \ldots, x_n$ if C is transformed into a new set of constraints C' such that x_1 does not appear in C' and every solution $(x_2^0, x_3^0, \ldots, x_n^0)$ of the constraints in C' can be extended to a solution $(x_1^0, x_2^0, \ldots, x_n^0)$ of the constraints in C.

Let us first sketch a variable elimination algorithm for inequality constraints. This algorithm, which is due to Fourier [Sch86], is based on the following observation. Any non-strict inequality involving a variable x can be written in the form $x \leq r_u$ or $x \geq r_l$ i.e., it gives an upper or a lower bound on x. Thus if we are given two inequalities, one of the form $x \leq r_u$ and the other of the form $x \geq r_l$, we can eliminate x and obtain the inequality $r_l \leq r_u$. Obviously, $r_l \leq r_u$ is a logical consequence of the given inequalities. In addition, any solution of $r_l \leq r_u$ can be extended to a solution of the given inequalities (simply by choosing for x any value between the values of r_l and r_u). Following this observation, Fourier's elimination algorithm forms all pairs $x \leq r_u$ and $x \geq r_l$, eliminates x and returns the resulting constraints.

The same goal could have been achieved using the *adaptive consistency* techniques discussed by [DP88] in the context of finding backtrack-free solutions to binary constraint satisfaction problems. If we want to eliminate variables $x_1, \ldots, x_m$ from a set of inequalities C represented by a binary constraint network N then we could run the algorithm Adaptive-consistency of [DP88] on N with arguments $x_1, \ldots, x_m$. In the resulting network the set of constraints between variables $x_{m+1}, \ldots, x_n$ is equivalent to the set of constraints generated by Fourier's algorithm.

Fourier's algorithm can be extended to cope with disjunctions of inequations. The extension is based on the following observation. Consider the constraints $x \leq r_u, x \geq r_l$ and $x \neq r \ \vee \ \phi$

where ϕ is a disjunction of inequations or *false*. Without loss of generality we can assume that x does not appear in ϕ.[7] Let us eliminate x from the first two constraints to obtain $r_l \leq r_u$. Now consider a solution v of $r_l \leq r_u$. If v satisfies ϕ then it can be extended to a solution of the original constraints. If v does not satisfy ϕ then it can be extended to a solution of the original constraints if and only if r, r_l and r_u do not share the same value. As a result, the constraint $\neg\phi \supset r \neq r_u \vee r \neq r_l$ or equivalently $\phi \vee r \neq r_u \vee r \neq r_l$ must be included in the resulting set of constraints not involving x. Note that this constraint is also a consequence of the original constraints. Thus, if the variable to be eliminated is x, our algorithm reduces to the following statement: For all triples $x \leq r_u, x \geq r_l$ and $x \neq r \vee \phi$ of input constraints, add $r_l \leq r_u$ and $\phi \vee r \neq r_u \vee r \neq r_l$ in the output constraints. Thus as in lemma 3.1, disjunctions of inequations can be processed independently of one another when variable elimination is performed.

Example 4.1 Let us assume we want to eliminate variable x_1 from the set of constraints $\{x_3 \leq x_1,\ x_5 \leq x_1,\ x_1 \leq x_2,\ x_5 \neq x_1,\ x_4 \neq x_1\}$. Eliminating x_1 gives the inequalities $\{x_3 \leq x_2,\ x_5 \leq x_2\}$ and the disjunctions of inequations $\{x_5 \neq x_3 \vee x_5 \neq x_2,\ x_5 \neq x_5 \vee x_5 \neq x_2,\ x_4 \neq x_3 \vee x_4 \neq x_2,\ x_4 \neq x_5 \vee x_4 \neq x_2\}$. The latter set is equivalent to $\{x_5 \neq x_2,\ x_4 \neq x_3 \vee x_4 \neq x_2\}$.

Figure 3 gives our variable elimination algorithm. For simplicity, the algorithm assumes that one-variable inequalities are given as two-variable inequalities (with the introduction of an auxiliary variable x_0 which is understood to be equal to zero). With very few changes the same algorithm can be applied to the generalized linear constraints of [LM89]. The following proposition shows that the algorithm is correct. Its proof is an easy generalization of the previous discussion and can be found in [Kou92a].

Theorem 4.1 *Let C be a set of constraints in variables $x_1, x_2, \ldots, x_n$ and C' be the set of constraints produced by algorithm Variable_Elimination when x_1 is eliminated. If $(x_1^0, x_2^0, \ldots, x_n^0)$ is a solution of C then $(x_2^0, x_3^0, \ldots, x_n^0)$ is a solution of C'. Conversely, every solution $(x_2^0, x_3^0, \ldots, x_n^0)$ of C' can be extended to a solution $(x_1^0, x_2^0, \ldots, x_n^0)$ of C.*

Although the input to *Variable_Elimination* is in first canonical form, it is possible that its output is not. This can happen only if an eliminated variable is a parameter which appears only in $C_i \cup C_n$. When this variable is eliminated, C_i' does not contain implicit equalities (see [LHM89] for a proof of this) but may contain redundant constraints. For example, assume C_i is $\{x_2 \leq x_1,\ x_1 \leq x_3,\ x_2 \leq x_5,\ x_5 \leq x_3\}$ and the eliminated variable is x_1. Then C_i' is $\{x_2 \leq x_3,\ x_2 \leq x_5,\ x_5 \leq x_3\}$ and obviously $x_2 \leq x_3$ is redundant. There are also cases when the constraints C_n' are not precise or irredundant. For example, assume that $C_i \cup C_n$ is $\{x_3 - x_2 \leq 2,\ x_2 - x_1 \leq 3,\ x_7 \geq 5,\ x_3 \leq 0,\ x_2 \neq x_7\}$. After x_2 is eliminated, we are left with $\{x_3 - x_1 \leq 5,\ x_7 \geq 5,\ x_3 \leq 0,\ x_3 - x_7 \neq 2 \vee x_1 - x_7 \neq -3\}$. It is easy to see now that $x_3 - x_7 \neq 2$ is implied by the inequality constraints. Consequently, if the output of *Variable_Elimination* must be returned in first canonical form, redundancy must be eliminated from C_i', each $c \in C_n'$ must be made precise and finally redundancy must be eliminated from C_n' (in this order!).

Fortunately, in the remaining cases the output of *Variable_Elimination* is in first canonical form as it is demonstrated by the following theorem (its proof can be found in [Kou92a]).

Theorem 4.2 *Let the inputs to algorithm Variable_Elimination be $C_e \cup C_i \cup C_n$ and X. If all variables of X are bound or parameter variables of the input set C_e then the output sets C_i and C_n are irredundant. Moreover, the former set does not contain implicit equalities and the latter consists only of precise constraints.*

[7] The constraint $x \neq r \vee \phi$ is equivalent to $\neg(x = r \wedge \neg\phi)$ where $\neg\phi$ is a conjunction of equalities. If we substitute r for x in $\neg\phi$, we obtain an equivalent constraint.

Algorithm Variable_Elimination

Inputs: A set of constraints $C = C_e \cup C_i \cup C_n$, in variables from the set X, in first canonical form. A set of variables E to be eliminated from C.

Output: A new set of constraints in variables from the set $E \setminus X$.

Method:

```
Initialize C'_e, C'_i and C'_n to ∅.
For each variable x ∈ X do
    If x is the bound variable of equation e_i ∈ C_e then
        Assign (C_e − {e_i}) to C_e, C'_i to C_i, C'_n to C_n
        Assign ∅ to C'_e, C'_i and C'_n
    ElseIf x is a parameter in equation c_i : y_i = x + r of C_e then
        Rewrite c_i as x = y_i − r
        For each constraint c in C_e − {c_i} (resp. C_i, C_n) do
            Substitute y_i − r for x in c
            Add the resulting constraint to C'_e (resp. C'_i, C'_n)
        EndFor
        Assign C'_e to C_e, C'_i to C_i, C'_n to C_n
        Assign ∅ to C'_e, C'_i and C'_n
    Else (the variable x appears only in C_i ∪ C_n)
        If there is at least one constraint x − x_j ≤ r_j in C_i then
            If there is at least one constraint x_k − x ≤ r_k in C_i then
                For each constraint x_k − x ≤ r_k in C_i do
                    For each constraint x − x_j ≤ r_j in C_i (j ≠ k) do
                        Add constraint x_k − x_j ≤ r_k + r_j in C'_i
                        For each constraint c ∈ C_n which contains x do
                            Let c be x − x_s ≠ r_s ∨ φ such that φ does not involve x
                            Simplify x_k − x_s ≠ r_k + r_s ∨ x_j − x_s ≠ r_s − r_j ∨ φ
                            and add it to C'_n.
                        EndFor
                    EndFor
                EndFor
                Add all constraints of C_i which do not contain x to C'_i
                Add all constraints of C_n which do not contain x to C'_n
            Else
                Add all constraints of C_i which do not contain x to C'_i
                Add all constraints of C_n which do not contain x to C'_n
            EndIf
        Else
            Add all constraints of C_i which do not contain x to C'_i
            Add all constraints of C_n which do not contain x to C'_n
        EndIf
        Assign C'_e to C_e, C'_i to C_i, C'_n to C_n
        Assign ∅ to C'_e, C'_i and C'_n
    EndIf
EndFor
Output C_e ∪ C_i ∪ C_n
```

Figure 3: An algorithm for variable elimination

In what follows, we will assume that when a variable is eliminated by *Variable_Elimination*, the resulting constraint set $C'_e \cup C'_i \cup C'_n$ is transformed into first canonical form before proceeding with the elimination of the rest of the variables.[8] Let us now investigate the complexity of *Variable_Elimination*.

Lemma 4.1 *Let $C = C_i \cup C_n$ be a set of temporal constraints. If m variables appearing in C_i are eliminated using algorithm Variable_Elimination and the resulting set is $C' = C'_i \cup C'_n$ then $|C'_i| \leq 2(v(C_i) - m)^2$ and $|C'_n| \leq |C_n|(v(C_i) - 1)^{2m}$.*

Theorem 4.3 *The algorithm Variable_Elimination takes time polynonial in the size of the input constraint set and exponential in the number of eliminated variables.*

The following theorem shows that variable elimination is intractable. The source of this intractability is the fact that the size of the output constraint set can be exponential in the number of eliminated variables. The following section examines a tractable case.

Theorem 4.4 *For every m, k such that $k \geq 4m$ we can find a set of temporal constraints C of size $||C|| \leq k$ and m variables of C such that when these variables are eliminated the resulting constraint set cannot be represented by fewer than 2^m disjunctions of inequations and $2m$ inequalities.*

Let us now consider variable elimination when the input constraint set is in second canonical form. In this case, elimination of one (or many) variables is trivial if these variables appear *only* in the inequality constraints. This is a consequence of the fact that the minimal network for a set of inequality constraints is decomposable (this concept will be explained fully in the following section) [DMP91]. In the remaining cases, the situation is as for the first canonical form.

5 VARIABLE ELIMINATION FOR SIMPLE TEMPORAL CONSTRAINTS

In this section, we show that for simple temporal constraints variable elimination is tractable. Our proof is indirect and is based on the fact that a set of simple temporal constraints can be transformed into an equivalent *decomposable* constraint set in time polynonial in its size. This indirect proof is interesting in its own right since it answers an open problem of [vB90a].

A set of constraints C is called *decomposable* if and only if it is λ-decomposable for every λ, $1 \leq \lambda \leq v(C)$ [DMP91]. A set of constraints is λ-decomposable if and only if every valuation $u = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0\}$ of $\lambda - 1$ variables which satisfies the constraints applicable to $x_1, \ldots, x_{\lambda-1}$ (namely, those involving only these variables) can be extended to a valuation $u' = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0, x_\lambda \leftarrow x_\lambda^0\}$ (for any variable x_λ) such that u' satisfies the constraints applicable to $x_1, \ldots, x_\lambda$.

The algorithm *Decompose* in figure 4 transforms a set of simple temporal constraints into an equivalent decomposable constraint set (which is not necessarily simple). The algorithm proceeds in three steps. At first, all equalities and inequalities which are consequences of C_e and C_i are computed by *Deductive_Closure*. This algorithm is not presented. For the case of C_e it is trivial; for the case of C_i the path consistency algorithm of [vB90a] (for the point algebra without $\neq$) can be used to achieve the same effect. The set returned by *Deductive_Closure* is decomposable [vB90a]. In its second step, *Decompose* computes a decomposable constraint set $C'_i \cup C'_n$ equivalent

[8] In practice, it must be more efficient to simply remove redundancy every time a variable is eliminated and postpone the transformation of inequation constraints to precise ones until the end of the algorithm.

to $C_i \cup C_n$. This is performed in a controlled manner to avoid generating redundant constraints (see the following lemmas). Finally, *Decompose* takes into account the equalities in C'_e and generates the constraints implied by the ones in $C'_i \cup C'_n$. This is done by function *Implied_Constraints* which is not presented since it is straightforward: C_{new} will contain one constraint for each constraint $c \in C'_i \cup C'_n$ in parameters $x_1, \ldots, x_l (l \leq 5)$ and each combination of (bound or parameter) variables $z_1, \ldots, z_l$ such that $z_i = x_i \in C'_e,\ i = 1, \ldots, l$.

To show that the algorithm is correct, we need the following lemmas. Appendix A contains the proof for lemma 5.2. The proofs for the rest of the lemmas are similar.

Lemma 5.1 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^1 such that z has not been marked and ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^1$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

Lemma 5.2 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^2 such that ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^2 \cup C_n^3$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

Lemma 5.3 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^3 such that ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^3$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

The following lemma is used in computing the complexity of algorithm *Decompose*.

Lemma 5.4 *If $C = C_e \cup C_i \cup C_n$ is a set of simple temporal constraints in first canonical form then Decompose returns a constraint set $C'_e \cup C_{in}$ such that*

$$|C'_e| \leq \frac{1}{2}(v(C_e)^2 - v(C_e)) \text{ and } |C_{in}| \leq |C_e|^5 \, v(C_i)^4 \, |C_n| .$$

We will now state the main result of this section.

Theorem 5.1 *The algorithm Decompose takes time $O(|C_e|^5 \, v(C_i)^4 \, |C_n|)$ and computes a decomposable constraint set equivalent to the input one.*

In fact, the previous theorem overestimates the running time of *Decompose*. As can be seen from the proof of lemma 5.4, a more precise upper bound is $O(k^5 v(C_i)^4 \, |C_n|)$ where k is the maximum number of equations of C_e which involve the same parameter variable.

Variable elimination is a trivial operation for decomposable sets of constraints. If C is a decomposable set of constraints then variables $x_1, \ldots, x_m$ can be eliminated from C by simply deleting all constraints of C which involve these variables. However, lemma 5.4 implies that even in the case of simple temporal constraints, a decomposable constraint set can contain a rather big number of disjunctions of inequations. Therefore this representation will probably be avoided in practise.

The above observation has also the following consequence. If a variable is eliminated from a set of simple temporal constraints, using *Variable_Elimination*, the first canonical form of the resulting set has size smaller than the size of the set obtained by first invoking *Decompose* and then eliminating this variable. The following corollary is now obvious.

Corollary 5.1 *For simple temporal constraints, algorithm Variable_Elimination takes time polynomial in the size and the number of variables of the input constraint set (thus, it also takes time polynomial in the number of eliminated variables).*

Algorithm Decompose

Input: A set of simple temporal constraints $C = C_e \cup C_i \cup C_n$ in first canonical form.

Output: A decomposable set of constraints equivalent to C.

Method:
$C'_e := Deductive\_Closure(C_e)$
$C'_i := Deductive\_Closure(C_i)$

$C^1_n := \emptyset$; $C^2_n := \emptyset$
For every $c: \ x_i \neq x_k$ in C_n do
  For every pair of constraints $x_l \leq x_i$ and $x_i \leq x_m$ in C'_i do
    Let c' be $x_k \neq x_l \ \vee \ x_m \neq x_l$ simplified
    Mark x_k in c'
    Add c' to C^1_n
  EndFor
  For every pair of constraints $x_l \leq x_k$ and $x_k \leq x_m$ in C'_i do
    Let c' be $x_i \neq x_l \ \vee \ x_m \neq x_l$ simplified
    Add c' to C^2_n
  EndFor
EndFor

$C^3_n := \emptyset$
For every $c: \ x_k \neq x_l \ \vee \ x_m \neq x_l$ in C^1_n such that x_k is marked do
  For every pair of constraints $x_t \leq x_k$ and $x_k \leq x_s$ in C'_i do
    Let c' be $x_t \neq x_l \ \vee \ x_s \neq x_l \ \vee \ x_m \neq x_l$ simplified. Add c' to C^3_n
  EndFor
EndFor
$C'_n := C^1_n \cup C^2_n \cup C^3_n$

$C_{new} := Implied\_Constraints(C'_e, C'_i \cup C'_n)$

Output $C'_e \cup C_{new}$

Figure 4: The algorithm *Decompose*

6 CONCLUSIONS

This paper discusses consistency checking, canonical forms and variable elimination for temporal constraints. Our results extend previous work on temporal constraints by considering disjunctions of inequations. This extension has been motivated by the fact that in certain temporal formalisms (e.g., [DM87, Kou92b]) allowing inequations and disallowing disjunctions of inequations compromises the ability to answer certain queries. Although the emphasis of this paper is rather theoretical, we plan to investigate the performance of our algorithms in a real temporal reasoning system based on the temporal database model of [Kou92b]. Among other things, this will allow us to determine cases in which the first canonical form is more appropriate than the second one and vice versa.

This work can be extended in several ways. We are currently considering other special cases of temporal constraints with the intention to determine the exact complexity of variable elimination for these cases. We are also investigating the behaviour of our algorithms in cases where constraints are added to the system incrementally.

It would be interesting to study variable elimination in temporal models involving both intervals and points. Currently, we can solve this problem by translating interval constraints to endpoint constraints and then performing variable elimination. The challenge would be to devise algorithms which avoid this translation.

Finally, our framework must be extended to deal with other kinds of non-binary temporal constraints. For example, "the duration of interval I exceeds the duration of interval J" which is a constraint on four points (the endpoints of I and J).

Acknowledgements

Most of this work was performed while the author was on leave from the Dept. of Computer Science, University of Toronto and was visiting the Dept. of Electrical and Computer Engineering, National Technical University of Athens and the Institute of Computer Science, Foundation of Research and Technology - Hellas. Financial support was gratefully received from the Institute of Computer Science, Foundation of Research and Technology - Hellas and the Dept. of Computer Science, University of Toronto. The following people provided important help during different stages of this work: Foto Afrati, Costas Courcoubetis, Jean-Louis Lassez, Itay Meiri, Dimitris Plexousakis, Timos Sellis, Thodoros Topaloglou, Peter van Beek and Yannis Vassiliou. In addition, one of the referees provided comments that allowed us to improve our presentation.

References

[All83] J.F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[DM87] T. Dean and D.V. McDermott. Temporal Data Base Management. *Artificial Intelligence*, 32:1–55, 1987.

[DMP91] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. Special Volume on Knowledge Representation.

[DP88] Rina Dechter and Judea Pearl. Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[KL91] H. Kautz and P. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *Proceedings of AAAI-91*, pages 241–246, 1991.

[Kou92a] Manolis Koubarakis. Dense Time and Temporal Constraints with $\neq$. Long version of this paper. Forthcoming., 1992.

[Kou92b] Manolis Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. Forthcoming., 1992.

[LHM89] Jean-Louis Lassez, Tien Huynh, and Ken McAloon. Simplification and Elimination of Redundant Linear Arithmetic Costraints. In *Proceedings of NACLP-89*, pages 37–51. MIT Press, 1989.

[LM88] Peter Ladkin and Roger Maddux. On Binary Constraint Networks. Technical Report KES.U.88.8, Kestrel Institute Technical Report, 1988.

[LM89] Jean-Louis Lassez and Ken McAloon. A Canonical Form for Generalized Linear Costraints. Technical Report RC15004 (#67009), IBM Research Division, T.J. Watson Research Center, 1989.

[Mac77] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[MBJK90] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: A Language for Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[Mei91] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. Technical Report R-160, Cognitive Systems Laboratory, University of California, Los Angeles, 1991.

[Mon74] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.

[Rei80] Raymond Reiter. Equality and Domain Closure in First Order Databases. *Journal of the ACM*, 27(2):235–249, 1980.

[Sch86] A. Schrijver, editor. *Theory of Integer and Linear Programming*. Wiley, 1986.

[SG88] Yoav Shoham and Nita Goyal. Temporal Reasoning in Artificial Intelligence. In Strobe, Howard and AAAI, editor, *Exploring Artificial Intelligence*, pages 419–439. Morgan Kaufmann, 1988.

[Sno90] R. Snodgrass. Temporal Databases: Status and Research Directions. *Sigmod Record*, 19(4):83–89, 1990.

[vB90a] Peter van Beek. Exact and Approximate Reasoning About Qualitative Temporal Relations. Technical Report TR 90-29, Department of Computing Science, University of Alberta, August 1990.

[vB90b] Peter van Beek. Reasoning About Qualitative Temporal Information. In *Proceedings of AAAI-90*, pages 728–734, 1990.

[VKvB89] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint Propagation Algorithms for Temporal Reasoning: a Revised Report. In D.S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, 1989.

A PROOFS

Lemma 4.1: When m variables are eliminated we are left with $v(C) - m$ variables. Therefore lemma 3.2 implies $|C_i| \leq 2(v(C_i) - m)^2$.

For the second inequality, let C_i^l (resp. C_n^l) be the set of inequalities (resp. disjunctions of inequations) produced by *Variable_Elimination* when the l-th variable is eliminated ($1 \leq l \leq m$). When the l-th variable x is eliminated there are at most $2\binom{v(C_i) - l}{2}$ pairs of constraints $x_k - x \leq r_k,\ x - x_j \leq r_j$ in C_i^l. Therefore $\left|C_n^1\right| \leq |C_n|\,(v(C_i) - 1)^2$, $\left|C_n^2\right| \leq \left|C_n^1\right|(v(C_i) - 2)^2 \leq |C_n|\,(v(C_i)-1)^2(v(C_i)-2)^2$ and so on. Finally, $|C_n^m| \leq |C_n| \prod_{l=1}^{m}(v(C_i)-l)^2 \leq |C_n|\,(v(C_i)-1))^{2m}$.

Theorem 4.3: Assume C and $C' = C_e' \cup C_i' \cup C_n'$ are the input and output sets and m is the number of eliminated variables. Notice that every step of *Variable_Elimination* takes time polynomial in the size of the constraint sets generated in this particular step.

Lemmas 3.2 and 4.1 give $|C_e'| \leq |C_e|$, $|C_i'| \leq 8v(C_i)^2 \leq 2\,\|C\|^2$ and $|C_n'| \leq \|C_n\|\,(2\,\|C_i\|)^{2m} \leq 2^{2m}\,\|C\|^{2m+1}$. The theorem now follows because $\|C_e'\| = |C_e'|$, $\|C_i'\| = |C_i'|$ and $\|C_n'\| \leq |C_n'|\,v(C_n')$.

Theorem 4.4: Given m, k such that $k \geq 4m$, the set C in variables $L_1, \ldots, L_m$, $U_1^1, \ldots, U_m^1$, $U_1^2, \ldots, U_m^2$, $X_1, \ldots, X_m$, $Y_1, \ldots, Y_m$ is $C = \bigcup_{1 \leq i \leq m} C_i \cup C_n$ where $C_i = \{L_i \leq X_i,\ X_i \leq U_i^1,\ X_i \leq U_i^2\}$ and $C_n = \{\bigvee_{i=1}^m Y_i \neq X_i\}$. Let us now eliminate variables $X_1, \ldots, X_m$ to arrive at $C' = \bigcup_{1 \leq i \leq m} C_i' \cup C_n'$ where $C_i' = \{L_i \leq U_i^1,\ L_i \leq U_i^2\}$ and $C_n = \{\bigvee_{i=1}^m Y_i \neq L_i\ \vee\ Y_i \neq U_i^{j_i} :\ j_i = 1, 2\}$. The first canonical form of C' is $C'' = \bigcup_{1 \leq i \leq m} C_i' \cup C_n''$ where

$$C_n'' = \{\bigvee_{i=1}^{m} Y_i \neq L_i\ \vee\ U_i^{j_i} \neq L_i :\ j_i = 1, 2\}$$

if the order of the variables is the one given above. Obviously C' (and C'') contains 2^m disjunctions of inequations and $2m$ inequalities. Now assume that there exists a set of temporal constraints A such that $A \neq C'$, $Sol(C') = Sol(A)$ and $|A| < 2^m + 2m$ (proof by contradiction). If A' is the first canonical form of A then $|A'| \leq |A|$. But equivalent sets of constraints have identical canonical forms therefore $A' = C''$ thus $|A| \geq 2^m + 2m$ which contradicts the assumption $|A| < 2^m + 2m$.

Lemma 5.2: Every constraint in C_n^2 is of the form $x_i \neq x_l\ \vee\ x_m \neq x_l$ and has been generated by considering inequalities $x_l \leq x_k$ and $x_k \leq x_m$ of C_i' and the inequation $x_i \neq x_k$ of C_n (for some fixed i, k, l, m). We now have the following three cases to consider:

1. $z = x_l,\ y = x_i$.
   Let us consider any pair of inequalities $x_a \leq x_l,\ x_l \leq x_b$ from C_i' and the disjunction of inequations $x_l \neq x_i \vee x_m \neq x_i$. These constraints imply $c'' :\ x_a \neq x_i\ \vee x_b \neq x_i\ \vee x_m \neq x_i$. But notice that $x_a \leq x_k \in C_i'$. Thus *Decompose* considers $x_a \leq x_k,\ x_k \leq x_m$ and $x_k \neq x_i$ and adds the constraint $c' :\ x_a \neq x_i\ \vee\ x_m \neq x_i$ to C_n^2. Obviously, $c' \models c''$.

2. $z = x_i,\ y = x_l$.
   Let us consider any pair of inequalities $x_a \leq x_i,\ x_i \leq x_b$ from C_i and the disjunction $x_i \neq x_l \ \vee\ x_m \neq x_l$. These constraints imply $c'' :\ x_a \neq x_l \ \vee\ x_b \neq x_l \ \vee\ x_m \neq x_l$. But if we consider $x_a \leq x_i,\ x_i \leq x_b$ and $x_i \neq x_k$, we can see that $x_k \neq x_a \ \vee\ x_b \neq x_a \in C_n^1$ and x_k is marked. Therefore *Decompose* considers $x_l \leq x_k,\ x_k \leq x_m$ and $x_k \neq x_a \ \vee\ x_b \neq x_a$ and adds $c' : x_a \neq x_l \ \vee\ x_m \neq x_a \ \vee\ x_b \neq x_a$ to C_n^3. Obviously, $c' \models c''$.

3. $z = x_m,\ y = x_l$.
   Let us consider any pair of inequalities $x_a \leq x_m,\ x_m \leq x_b$ from C_i and the disjunction $x_m \neq x_l \ \vee\ x_i \neq x_l$. These constraints imply $c'' :\ x_a \neq x_l \ \vee\ x_b \neq x_l \ \vee\ x_i \neq x_l$. But notice that $x_k \leq x_b \in C_i'$. Therefore *Decompose* considers $x_l \leq x_k,\ x_k \leq x_b$ and $x_k \neq x_i$ and adds $c' : x_l \neq x_i \ \vee\ x_b \neq x_i$ to C_n^2. Obviously, $c' \models c''$.

Lemma 5.4: There can be at most $\binom{v(C_e)}{2}$ constraints in C_e'. Thus $|C_e'| \leq \frac{1}{2}(v(C_e)^2 - v(C_e))$. Similarly, $|C_i'| \leq \frac{1}{2}(v(C_i)^2 - v(C_i))$.

For every variable x there are at most $\frac{1}{4}(v(C_i)-1)^2$ pairs of constraints $y \leq x,\ x \leq z$. Therefore $\left|C_n^1\right| \leq \frac{1}{4}(v(C_i)-1)^2\,|C_n|,\ \left|C_n^2\right| \leq \frac{1}{4}(v(C_i)-1)^2\,|C_n|$ and $\left|C_n^3\right| \leq \frac{1}{16}(v(C_i)-1)^4\,|C_n|$. Finally,

$$|C_n'| \leq \frac{1}{16}(v(C_i)-1)^4\,|C_n|\,.$$

Let us now turn to the function *Implied_Constraints*. Let k denote the maximum number of equations of C_e which involve the same parameter variable. Obviously $k \leq |C_e|$. Notice now that every constraint in C_n' involves at most 5 variables. From each such constraint, at most $(k+1)^5$ constraints can be generated in C_{new}. Similarly, from each constraint in C_i' (which involves at most 2 variables), at most $(k+1)^2$ constraints can be generated in C_{new}.

Therefore $|C_{in}| \leq |C_e|^2\,|C_i'| + |C_e|^5\,|C_n'| \leq |C_e|^5\, v(C_i)^4\,|C_n|$.

Theorem 5.1: For every constraint set C and set of variables $X = \{x_1, \ldots, x_n\}$, the set of constraints in C which involve only variables from X will be denoted by $C(X)$ or $C(x_1, \ldots, x_n)$.

Initially we will show that $C_i' \cup C_n'$ is decomposable. $C_i' \cup C_n'$ is 1-decomposable since it is consistent. We will now prove that it is decomposable for each λ such that $2 \leq \lambda \leq v(C_i' \cup C_n')$. Let us assume that there is a $\lambda,\ 2 \leq \lambda \leq v(C_i' \cup C_n')$ such that $C_i' \cup C_n'$ is not λ-decomposable (proof by contradiction). In other words, there exists a valuation $u = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0\}$ and variable x_λ such that u satisfies $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1})$ but there is no value x_λ^0 such that $u' = u \cup \{x_\lambda \leftarrow x_\lambda^0\}$ satisfies $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$.

Let us now observe that *Deductive_Closure* returns a decomposable constraint set C_i'. Therefore $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$ must contain at least one constraint from C_n. Let us apply valuation u to $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$ to obtain $C_i' \cup C_n'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$. But now lemma 3.1 implies that there exists a disjunction of inequations $c_0 \in C_n'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$ such that $C_i'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda) \cup \{c_0\}$ is inconsistent. Let c_0 be $x_\lambda \neq x_\rho^0$ and $C_i'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$ be $x_\mu^0 \leq x_\lambda,\ x_\lambda \leq x_\nu^0$ where $1 \leq \rho, \mu, \nu \leq \lambda - 1$.

Since we have an inconsistency $x_\mu^0 = x_\nu^0 = x_\rho^0$ must hold. Let us assume that $c : x_\lambda \neq x_\rho \ \vee\ \phi$ is the constraint of $C_n'(x_1, \ldots, x_\lambda)$ which results in c_0 after u is applied (ϕ does not contain x_λ).Now we have to consider the following cases:

1. $c \in C_n$ or $c \in C_n^1$ and x_λ has been marked. Then *Decompose* considers $x_\mu \leq x_\lambda, x_\lambda \leq x_v$ and $x_l \neq x_\rho \ \vee\ \phi$ and generates $c' :\ x_\mu \neq x_\rho \ \vee\ x_\nu \neq x_\rho \ \vee\ \phi$ in C_n'. Let us now note that

u satisfies c' but falsifies ϕ. Thus $x^0_\mu \neq x^0_\rho \ \lor \ x^0_\nu \neq x^0_\rho$ must hold and therefore we have a contradiction with $x^0_\mu = x^0_\nu = x^0_\rho$. Thus the original assumption is false.

2. $c \in C^1_n$ and x_λ is not marked. The proof for this case is similar to the proof for the following case. We only use lemma 5.1 instead of lemma 5.2.

3. $c \in C^2_n$ and c is $x_\lambda \neq x_\rho \lor x_\sigma \neq x_\rho$ where $1 \leq \sigma \leq \lambda - 1$. In this case, lemma 5.2 tells us that there is a constraint $c' \in C^2_n \cup C^3_n$ such that $c' \models x_\mu \neq x_\rho \ \lor \ x_\nu \neq x_\rho \ \lor \ x_\sigma \neq x_\rho$. Obviously c' involves only variables from $\{x_1, \ldots, x_{\lambda-1}\}$ and it is satisfied by u. Thus u must also satisfy c'. But observe that the form of c_0 implies $x^0_\sigma = x^0_\rho$. Therefore $x^0_\mu \neq x^0_\rho \ \lor \ x^0_\nu \neq x^0_\rho$ which contradicts our assumption.

4. $c \in C^3_n$.
   The proof is similar to the one given for the previous case. We now have to use lemma 5.3.

It is not difficult to see now that $C'_e \cup C_{in}$ is decomposable.

Finally, let us calculate the time complexity of *Decompose*. The calls to *Deductive_Closure* take $O(v(C_i)^3)$ time. The rest of the computation depends only on the sizes of the constraint sets generated. Therefore lemma 5.4 implies that the algorithm takes $O(|C_e|^5 \, v(C_i)^4 \, |C_n|)$ time.