Tractable Disjunctions of Linear Constraints: Basic Results and Applications to Temporal Reasoning*

Manolis Koubarakis

Dept. of Electronic and Computer Engineering
Technical University of Crete
University Campus - Kounoupidiana
73100 Chania, Crete
GREECE
Tel: +30 821 37222
Fax: +30 821 37202
E-mail: manolis@ced.tuc.gr

Keywords: linear constraints, variable elimination, interval algebra, ORD-Horn constraints, global consistency

Abstract

We study the problems of deciding consistency and performing variable elimination for disjunctions of linear inequalities and disequations with *at most one* inequality per disjunction. This new class of constraints extends the class of generalized linear constraints originally studied by Lassez and McAloon. We show that deciding consistency of a set of constraints in this class can be done in polynomial time. We also present a variable elimination algorithm which is similar to Fourier's algorithm for linear inequalities. Finally, we use these results to provide new temporal reasoning algorithms for the Ord-Horn subclass of Allen's interval formalism. We also show that there is no low level of local consistency that can guarantee global consistency for the Ord-Horn subclass. This property distinguishes the Ord-Horn subclass from the pointizable subclass (for which strong 5-consistency is sufficient to guarantee global consistency), and the continuous endpoint subclass (for which strong 3-consistency is sufficient to guarantee global consistency).

1 Introduction

Linear constraints over the reals have recently been studied in depth by researchers in constraint logic programming (CLP) and constraint databases (CDB) [13, 16, 21].

*To appear in Theoretical Computer Science. This is an extended version of the paper "Tractable Disjunctions of Linear Constraints" which appears in the Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming (CP96), Cambridge, MA, August 19-22, 1996. Lecture Notes in Computer Science, Vol. 1118, pages 297-307. This research was supported by TMR project CHOROCHRONOS funded by ESPRIT IV.

Two very important operations in CLP and CDB systems are deciding consistency of a set of constraints, and performing variable elimination. Subclasses of linear constraints over the reals have also been considered in temporal reasoning [5, 18, 19, 22, 23, 31, 14, 15]. Important operations in temporal reasoning applications are the following: (i) deciding consistency of a set of binary temporal constraints, (ii) performing variable elimination, and (iii) computing the strongest feasible constraints between every pair of variables.

Disjunctions of linear constraints over the reals are important in many applications [13, 5, 18, 19, 20, 22, 23, 31, 14, 15]. The problem of deciding consistency for an arbitrary set of disjunctions of linear constraints is NP-complete [34]. It is therefore interesting to discover classes of disjunctions of linear constraints for which consistency can be decided in PTIME. In [30], Lassez and McAloon have studied the class of *generalized linear constraints* which includes linear inequalities (e.g., $2x_1 + 3x_2 - 5x_3 \leq 6$) and disjunctions of linear disequations (e.g., $2x_1 + 3x_2 - 4x_3 \neq 4 \ \vee \ x_2 + x_3 + x_5 \neq 7$). Among other things, they have shown that the consistency problem for this class can be solved in PTIME.

[18, 12, 10, 9, 11] have studied the problem of variable elimination for generalized linear constraints. The basic algorithm for variable elimination has been discovered independently in [18] and [10, 9], but [18] has used the result only in the context of temporal constraints. The basic algorithm is essentially an extension of Fourier's elimination algorithm [33] to deal with disjunctions of disequations. If S is a set of constraints, let $|S|$ denote its cardinality. Let $C = I \cup D_n$ be a set of generalized linear constraints, where I is a set of inequalities and D_n is a set of disjunctions of disequations. If we eliminate m variables from C using the basic algorithm proposed by Koubarakis and Imbert then the resulting set contains $O(|I|^{2^m})$ inequalities and $O(|D_n|\,|I|^{2^{m+1}})$ disjunctions of disequations. A lot of these constraints are redundant. Imbert has studied this problem in more detail and presented more advanced algorithms that eliminate redundant constraints [10, 9, 11].

In this paper we go beyond the above work on generalized linear constraints. Our contributions can be summarized as follows:

- We extend the class of generalized linear constraints to include disjunctions with an unlimited number of disequations and *at most one* inequality per disjunction. For example:

$$3x_1 + x_5 - 4x_3 \leq 7 \ \vee \ 2x_1 + 3x_2 - 4x_3 \neq 4 \ \vee \ x_2 + x_3 + x_5 \neq 7$$

  The resulting class will be called the class of *Horn* constraints since there seems to be some analogy with Horn clauses. We show that deciding consistency can still be done in PTIME for this class (Theorem 3.4). This result has also been obtained independently by Jonsson and Bäckström [14, 15]. We also extend the basic variable elimination algorithm of [18, 10, 9] for this new class of constraints.

- We study a special case of Horn constraints, called *Ord-Horn constraints*, originally introduced in [31]. Ord-Horn constraints are important for temporal reasoning based on the *Ord-Horn subclass* of interval relations expressible in Allen's formalism [1, 31]. Our results allow us to develop algorithms for consistency checking and computing the strongest feasible constraints that improve the best known algorithms for this class [31].

  The complexity of the consistency checking algorithm of [31] is $O(n^3)$ where n is the number of variables in the input set. The complexity of our consistency checking algorithm (Theorem 5.3) is $O(\max(n^2, hn))$ where h is the number of constraints $i \; R \; j$ such that $R \in \mathcal{H} \setminus \mathcal{C}$ (the symbol $\mathcal{H}$ denotes the Ord-Horn subclass and $\mathcal{C}$ denotes the continuous endpoint subclass [37]).

The complexity of the algorithm of [31] for computing the strongest feasible relations is $O(n^5)$. The corresponding algorithm that we give (Theorem 5.4) has complexity $O(\max(n^4, hn^3))$ where n and h are as above.

- We develop global consistency algorithms for Horn constraints (Section 6) and constraints in the Ord-Horn subclass (Section 7.1).
- We show that there is no *low* level of local consistency which is sufficient to guarantee global consistency of a set of constraints in the Ord-Horn class (Theorem 7.2). This property distinguishes the Ord-Horn subclass from the pointizable subclass (for which strong 5-consistency is sufficient to guarrantee global consistency), and the continuous endpoint subclass (for which strong 3-consistency is sufficient to guarantee global consistency).

All our results are developed for Horn constraints with variables ranging over the real numbers. The results do not change if we consider rationals as our domain. Since *density* is the important property we rely on, our results do not hold if we consider Horn constraints over the integers. In this case deciding consistency becomes an NP-complete problem as can be easily seen.

The paper is organized as follows. Section 2 introduces the basic concepts needed for the developments of this paper. Section 3 presents the algorithm for deciding consistency of Horn constraints. Section 4 presents an algorithm for variable elimination of Horn constraints. Section 5 presents new algorithms for consistency checking and computing the strongest feasible constraints for the Ord-Horn subclass. Section 6 studies the problem of global consistency for Horn constraints. Section 7 presents our results on the global consistency problem of the Ord-Horn subclass. Finally, Section 8 discusses future research.

2 Preliminaries

We consider the n-dimensional Euclidean space $\mathcal{R}^n$. A *linear constraint* over $\mathcal{R}^n$ is an expression $a_1 x_1 + \cdots + a_n x_n \sim b$ where $a_1, \ldots, a_n, b$ are integers, $x_1, \ldots, x_n$ are variables ranging over the real numbers, and $\sim$ is $\leq, =$ or $\neq$. Depending on what $\sim$ is, we will distinguish linear constraints into *inequalities* (e.g. $2x_1 + 3x_2 - 5x_3 \leq 6$), *equations* (e.g., $2x_1 + 3x_2 - 5x_3 = 6$) and *disequations* (e.g., $2x_1 + 3x_2 - 5x_3 \neq 6$).

Let us now define the class of constraints that we will consider.

Definition 2.1 A *Horn constraint* is a disjunction $d_1 \;\vee\; \cdots \;\vee\; d_n$ where each $d_i, \; i = 1, \ldots, n$ is a weak linear inequality or a linear disequation, and the number of inequalities among $d_1, \ldots, d_n$ does not exceed one. If there are no inequalities then a Horn constraint will be called *negative*. Otherwise it will be called *positive*. Horn constraints of the form $d_1 \;\vee\; \cdots \;\vee\; d_n$ with $n \geq 2$ will be called *disjunctive*.

Example 2.1 The following are examples of Horn constraints:

$$
\begin{gathered}
3x_1 + x_5 - 3x_4 \leq 10, \\
x_1 + x_3 + x_5 \neq 7, \\
3x_1 + x_5 - 4x_3 \leq 7 \;\vee\; 2x_1 + 3x_2 - 4x_3 \neq 4 \;\vee\; x_2 + x_3 + x_5 \neq 7, \\
4x_1 + x_3 \neq 3 \;\vee\; 5x_2 - 3x_5 + x_4 \neq 6
\end{gathered}
$$

The first and the third constraint are positive while the second and the fourth are negative. The third and fourth constraint are disjunctive.

According to the above definition weak inequalities are positive Horn constraints. Sometimes we will find it more convenient to consider inequalities separately from positive disjunctive Horn constraints. If d is a positive disjunctive Horn constraint then

$$d \equiv \neg(E \ \wedge \ i)$$

where E is a conjunction of equations and i is an inequality. We will often use this notation for positive Horn constraints.

Notice that we do not need to introduce strict inequalities in the above definition. A strict inequality like $x_1 + x_2 + x_3 < 5$ can be equivalently written as follows:

$$x_1 + x_2 + x_3 \leq 5, \ x_1 + x_2 + x_3 \neq 5$$

Similarly, the constraint $x_1 + x_2 + x_3 < 5 \vee \phi$ where ϕ is a disjunction of disequations can be equivalently written as the conjunction of the following constraints:

$$x_1 + x_2 + x_3 \leq 5 \ \vee \ \phi, \ x_1 + x_2 + x_3 \neq 5 \ \vee \ \phi$$

A similar observation is made in [31].

Negative Horn constraints have been considered before in [28, 29, 30, 18, 12, 10, 11, 22]. Nebel and Bürckert have studied the class of *Ord-Horn constraints* in the context of qualitative interval reasoning [31]. Ord-Horn constraints form a proper subclass of Horn constraints, and will be considered in detail in Section 5. Horn constraints have also been studied by Jonsson and Bäckström [14, 15] who discovered independently the result discussed in Section 3.

We will now present some standard definitions.

Definition 2.2 Let C be a set of constraints in variables $x_1, \ldots, x_n$. The *solution set* of C, denoted by $Sol(C)$, is:

$$\{(r_1, \ldots, r_n) \in \mathcal{R}^n : \text{ for every } c \in C, \ (r_1, \ldots, r_n) \text{ satisfies } c\}$$

Each member of $Sol(C)$ is called a *solution* of C. A set of constraints is called *consistent* if its solution set is non-empty. We will use the same notation, $Sol(\cdot)$, for the solution set of a single constraint.

Remark 2.1 In the rest of the paper we will usually consider one or more sets of constraints e.g., $C_1, \ldots, C_m$ in variables $x_1, \ldots, x_n$. In this case we will always regard $Sol(C_i)$ a subset of $\mathcal{R}^n$ even though C_i might contain less than n variables. For example, if we happen to deal with

$$C_1 = \{x_1 \leq x_2, \ x_2 \leq x_3, \ x_3 \neq x_4\} \ \text{ and } \ C_2 = \{x_1 \leq x_2, \ x_2 \leq x_3\}$$

we may write $Sol(C_1) \subset Sol(C_2)$ where $Sol(C_1), Sol(C_2) \subset \mathcal{R}^4$. Similarly, we may write $Sol(C_2) \subset Sol(x_1 \leq x_2)$.

We will also use the alternative notation $Sol^*(\cdot)$. If C is a set of constraints, $Sol^*(C)$ will always be regarded a subset of $\mathcal{R}^k$ where k is the number of variables of C (independently of any other constraint set considered at the same time). This notation will come handy in Section 4 where we study variable elimination.

Definition 2.3 Let C_1 and C_2 be sets of constraints in the same set of variables. C_1 will be called *equivalent* to C_2 if $Sol(C_1) = Sol(C_2)$. A constraint c *logically follows* from a set of constraints C, denoted by $C \models c$, if every solution of C satisfies c.

We will now present some concepts of convex geometry [33, 8] that will enable us to study the geometric aspects of the constraints considered. We will take our definitions from [28, 30]. If V is a subspace of the n-dimensional Euclidean space $\mathcal{R}^n$ and p a vector in $\mathcal{R}^n$ then the translation $p + V$ is called an *affine space*. The intersection of all affine spaces that contain a set S is an affine space, called the *affine closure* of S and denoted by $Aff(S)$. If e is a linear equation then the solution set of e is called a *hyperplane*. In $\mathcal{R}^3$ the hyperplanes are the planes. In $\mathcal{R}^2$ the hyperplanes are the straight lines. A hyperplane is an affine space and every affine space is the intersection of a finite number of hyperplanes. If E is a set of equalities then $Sol(E)$ is an affine space. If i is a linear inequality then the solution set of i is called a *half-space*. If I is a finite set of inequalities then $Sol(I)$ is the intersection of a finite number of half-spaces, and is called a *polyhedral* set.

A set $S \subseteq \mathcal{R}^n$ is called *convex* if the line segment joining any pair of points in S is included in S. Affine subspaces of $\mathcal{R}^n$ are convex. Half-spaces are convex. Also, polyhedral sets are convex.

If d is a negative Horn constraint then the solution set of d is $Sol(d) = \mathcal{R}^n \setminus Sol(\neg d)$. The constraint $\neg d$ is a conjunction of equations thus $Sol(\neg d)$ is an affine space. If $\neg d$ is inconsistent then d is equivalent to *true* (e.g., $x \neq 2 \ \vee \ x \neq 3$). In the rest of the paper we will ignore negative Horn constraints that are equivalent to *true*.

If d is a positive disjunctive Horn constraint of the form $\neg(E \ \wedge \ i)$ then $Sol(d) = \mathcal{R}^n \setminus Sol(\neg d)$. The constraint $\neg d$ is a conjunction $E \ \wedge \ i$ where E is a conjunction of equations and i is a strict inequality. If $E \equiv true$ then d is essentially a weak inequality $\neg i$ (e.g., $x \neq x \ \vee \ y \leq 5 \equiv y \leq 5$). If $E \ \wedge \ i$ is inconsistent then its corresponding Horn constraint d is equivalent to *true* (e.g., $x \neq 2 \vee x \leq 3 \equiv true$). If $E \ \wedge \ i$ is consistent and $Sol(E) \subseteq Sol(i)$ then $d \equiv \neg E$, so d is actually a negative Horn constraint (e.g., $x \neq 2 \ \vee \ y \neq 2 \ \vee \ x \geq 3 \equiv x \neq 2 \ \vee \ y \neq 2$). If $E \ \wedge \ i$ is consistent and $Sol(E) \not\subseteq Sol(i)$ then its solution set will be called a *half affine space*. In $\mathcal{R}^3$ the half affine spaces are half-lines or half-planes. For example, $z = 2 \wedge x > 0$ is a half plane. In the rest of the paper we will ignore positive disjunctive Horn constraints equivalent to a weak inequality, a negative Horn constraint or *true*.

3 Deciding Consistency

[28, 30] showed that negative Horn constraints can be treated independently of one another for the purposes of deciding consistency. The following is one of their basic results.

Theorem 3.1 Let $C = I \cup D_n$ be a set of constraints where I is a set of linear inequalities and D_n is a set of negative Horn constraints. Then C is consistent if and only if I is consistent, and for each $d \in D_n$ the set $I \cup \{d\}$ is consistent.

Whether a set of inequalities is consistent or not, can be decided in PTIME using Kachiyan's linear programming algorithm [33]. We can also detect in PTIME whether $I \cup \{d\}$ is consistent by simply running Kachiyan's algorithm $2n$ times to decide whether I implies every equality e in the conjunction of n equalities $\neg d$. In other words, deciding consistency in the presence of negative Horn contraints can be done in PTIME.[1]

Is it possible to extend this result to the case of positive disjunctive Horn constraints? In what follows, we will answer this question affirmatively. Let us start by pointing out that the independence property of negative Horn constraints does *not* carry over to positive ones as the following example demonstrates.

[1]The exact algorithm that Lassez and McAloon give in [28] is different but this is not significant for the purposes of this paper.

Example 3.1 Let $I = \{x \geq 1, x \leq 5, y = 3\}$. The constraint sets

$$I \cup \{\neg(y = 3 \ \wedge \ x > 1)\} \ \text{ and } \ I \cup \{\neg(y = 3 \ \wedge \ x = 1)\}$$

are consistent. But the set $I \cup \{\neg(y = 3 \ \wedge \ x > 1), \ \neg(y = 3 \ \wedge \ x = 1)\}$ is inconsistent.

Fortunately, there is still enough structure available in our problem which we can exploit to come up with a PTIME consistency checking algorithm. Let $C = I \cup D_p \cup D_n$ be a set of constraints where I is a set of inequalities, D_p is a set of positive *disjunctive* Horn constraints, and D_n is a set of negative Horn constraints. Intuitively, the solution set of C is empty only if the polyhedral set defined by I is *covered* by the affine spaces and half affine spaces defined by the negations of Horn constraints.

The algorithm CONSISTENCY shown in Figure 1 proceeds as follows. Initially, we check whether I is consistent. If this is the case, then we proceed to examine whether $Sol(I)$ can be covered by

$$\overline{Sol(D_p \cup D_n)} = \bigcup_{d \in D_p \cup D_n} \overline{Sol(d)} = \bigcup_{d \in D_p \cup D_n} Sol(\neg d).$$

To verify this, we make successive passes over $D_p \cup D_n$. In each pass, we carry out two checks. The *first check* discovers whether there is any positive Horn constraint $d \equiv \neg(E \ \wedge \ i)$ such that $Sol(I)$ is included in the affine space defined by E. If this is the case then d is discarded and I is updated to reflect the part possibly "cut off" by d. The resulting solution set $Sol(I)$ is still a polyhedral set. An inconsistency can arise if $Sol(I)$ is reduced to $\emptyset$ by successive "cuts". In each pass we also check whether there is an affine space (represented by the negation of a negative Horn constraint) which covers $Sol(I)$. In this case there is an inconsistency as well. The algorithm stops when there are no more affine spaces or half affine spaces that pass the two checks. In this case C is consistent.

Let us now prove the correctness of algorithm CONSISTENCY. First, we will need a few technical lemmas. The first two lemmas show that the sets resulting from successive "cuts" inflicted on $Sol(I)$ by positive Horn constraints passing the first check of the algorithm are indeed polyhedral. The lemmas also give a way to compute the inequalities defining these sets.

Lemma 3.1 Let I be a set of inequalities and $\neg(E \ \wedge \ i)$ be a positive disjunctive Horn constraint such that $Sol(I) \subseteq Sol(E)$. Then $Sol(I \wedge \neg(E \wedge i)) = Sol(I \wedge \neg i)$.

Proof: Let $\overline{x} \in Sol(I \ \wedge \ \neg(E \ \wedge \ i))$. Then $\overline{x} \in Sol(I)$ and $\overline{x} \in Sol(\neg(E \ \wedge \ i))$. If $\overline{x} \in Sol(\neg(E \ \wedge \ i))$ then $\overline{x} \in Sol(\neg E)$ or $\overline{x} \in Sol(\neg i)$. But $Sol(I) \cap Sol(\neg E) = \emptyset$ because $Sol(I) \subseteq Sol(E)$. Therefore, $\overline{x} \in Sol(I)$ and $\overline{x} \in Sol(\neg i)$. Equivalently, $\overline{x} \in Sol(I \ \wedge \ \neg i)$. The other direction of the proof is trivial. ■

Lemma 3.2 Let I be a set of inequalities and $d_k \equiv \neg(E_k \ \wedge \ i_k), \ k = 1, \ldots, m$ be a set of positive disjunctive Horn constraints such that $Sol(I) \subseteq Sol(E_1)$ and

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{l} \neg i_k) \subseteq Sol(E_{l+1}) \ \text{ for } \ l = 1, \ldots, m-1.$$

Then

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k) = Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k).$$

Algorithm CONSISTENCY
Input: A set of constraints $C = I \cup D_p \cup D_n$.
Output: "consistent" if C is consistent. Otherwise "inconsistent".
Method:

If I is inconsistent **then return** "inconsistent"

Repeat
  $Done \leftarrow true$
  For each $d \in D_p \cup D_n$ **do**
    If $d \equiv \neg(E \wedge i) \in D_p$ and $Sol(I) \subseteq Sol(E)$ **then**
      $I \leftarrow I \wedge \neg i$
      If I is inconsistent **then return** "inconsistent"
      $Done \leftarrow false$
      Remove d from D_p
    Else If $d \in D_n$ and $Sol(I) \subseteq Sol(\neg d)$ **then**
      Return "inconsistent"
    End If
  End For
Until $Done$

Return "consistent"

Figure 1: Deciding consistency of a set of Horn constraints

Proof: The proof is by induction on m. The base case $m = 1$ follows from Lemma 3.1. For the inductive step, let us assume that the lemma holds for $m - 1$ Horn constraints. Then

$$Sol(I \wedge \bigwedge_{k=1}^{m} d_k) = Sol(I \wedge \bigwedge_{k=1}^{m-1} d_k) \cap Sol(d_m) =$$

$$Sol(I \wedge \bigwedge_{k=1}^{m-1} \neg i_k) \cap Sol(d_m) = Sol((I \wedge \bigwedge_{k=1}^{m-1} \neg i_k) \wedge d_m)$$

using the inductive hypothesis.
The assumptions of this lemma and Lemma 3.1 imply that

$$Sol((I \wedge \bigwedge_{k=1}^{m-1} \neg i_k) \wedge d_m) = Sol(I \wedge \bigwedge_{k=1}^{m} \neg i_k).$$

Thus

$$Sol(I \wedge \bigwedge_{k=1}^{m} d_k) = Sol(I \wedge \bigwedge_{k=1}^{m} \neg i_k).$$

■

The following lemmas show that if there are Horn constraints that do not pass the two checks of algorithm CONSISTENCY then the affine spaces or half affine spaces corresponding to their negations cannot cover the polyhedral set defined by the inequalities.

Lemma 3.3 Let S be a convex set of dimension δ and suppose that $S_1, \ldots S_n$ are convex sets of dimension $\delta_i < \delta,\ i = 1, \ldots, n$. Then $S \not\subseteq \bigcup_{i=1}^{n} S_i$.

Proof: See Lemma 2 of [28]. ■

Lemma 3.4 Let I be a consistent set of inequalities and $d_k \equiv \neg(E_k \ \wedge \ i_k)$, $k = 1, \ldots, m$ be a set of Horn constraints such that $Sol(I) \not\subseteq Sol(E_k)$ and $Sol(I) \cap Sol(E_k) \neq \emptyset$ for all $k = 1, \ldots, m$. Then $Sol(I) \not\subseteq \bigcup_{k=1}^{m} Sol(\neg d_k)$.

Proof: The proof is very similar to the proof of Theorem 1 of [28]. Since $Sol(I) \not\subseteq Sol(E_k)$ then $Aff(Sol(I)) \not\subseteq Sol(E_k)$. This means that $Sol(E_k) \cap Aff(Sol(I))$ is an affine space of strictly lower dimension than $Aff(Sol(I))$. Then $Sol(E_k) \cap Sol(I)$ is of strictly lower dimension than $Sol(I)$ since the dimension of $Sol(I)$ is equal to that of $Aff(Sol(I))$. Thus from Lemma 3.3, $Sol(I) \not\subseteq \bigcup_{k=1}^{m} Sol(E_k)$. Notice now that $Sol(\neg d_k) \subseteq Sol(E_k)$ for all $k = 1, \ldots, m$. Therefore $\bigcup_{k=1}^{m} Sol(\neg d_k) \subseteq \bigcup_{k=1}^{m} Sol(E_k)$. We can now conclude that $Sol(I) \not\subseteq \bigcup_{k=1}^{m} Sol(\neg d_k)$. ■

The following theorems demonstrate that the algorithm Consistency is correct and can be implemented in PTIME.

Theorem 3.2 If algorithm Consistency returns "inconsistent" then its input C is inconsistent.

Proof: If the algorithm returns "inconsistent" in its first line then I, and therefore C, is inconsistent.

If the algorithm returns "inconsistent" in the third if-statement then there are positive Horn constraints $d_k \equiv \neg(E_k \ \wedge \ i_k)$, $k = 1, \ldots, m \leq |D_p|$ such that the assumptions of Lemma 3.2 hold for I and $d_1, \ldots, d_m$. Therefore

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k) = Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) = \emptyset.$$

Consequently, $Sol(C) = \emptyset$ because $Sol(C) \subseteq Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k)$.

If the algorithm returns "inconsistent" in the fourth if-statement then there are positive Horn constraints $d_1, \ldots, d_n \in D_p$ and negative constraint $d_{m+1} \in D_n$ such that the assumptions of Lemma 3.2 hold for I and $d_1, \ldots, d_m$, and

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k) \subseteq Sol(\neg d_{m+1}).$$

But then

$$Sol(C) \subseteq Sol(I \ \wedge \ \bigwedge_{k=1}^{m+1} d_k) = Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k) \cap Sol(d_{m+1}) = \emptyset.$$

■

Theorem 3.3 If algorithm Consistency returns "consistent" then its input C is consistent.

Proof: If the algorithm returns "consistent" then I is consistent. Let $d_1, \ldots, d_m$ be the positive Horn constraints removed from $D_p \cup D_n$ by the algorithm, and $d_{m+1}, \ldots, d_n$ be the remaining Horn constraints. Then

$$Sol(C) = Sol(I \ \wedge \ \bigwedge_{k=1}^{n} d_k) = Sol(I \ \wedge \ \bigwedge_{k=1}^{m} d_k) \setminus \bigcup_{k=m+1}^{n} Sol(\neg d_k) =$$

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) \setminus \bigcup_{k=m+1}^{n} Sol(\neg d_k).$$

Notice that $Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) \neq \emptyset$ otherwise the algorithm outputs "inconsistent" in Step 2. Also, $Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) \not\subseteq Sol(E_k)$ for all $k = m+1, \ldots, n$ otherwise the algorithm would have removed d_k from $D_p \cup D_n$ or have returned "inconsistent".

Without any loss of generality we can also assume that

$$Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) \cap Sol(E_k) \neq \emptyset$$

for all $k = m+1, \ldots, n$ (if this does not hold for constraint d_k, this constraint can be discarded without changing $Sol(C)$). From Lemma 3.4 we can now conclude that $Sol(I \ \wedge \ \bigwedge_{k=1}^{m} \neg i_k) \not\subseteq \bigcup_{k=m+1}^{n} Sol(\neg d_k)$. Therefore $Sol(C) \neq \emptyset$. ■

Theorem 3.4 The algorithm CONSISTENCY can be implemented in PTIME.

Proof: It is not difficult to see that the algorithm can be implemented in PTIME. The consistency of I can be checked in PTIME using Khachiyan's algorithm for linear programming [17, 33]. The test $Sol(I) \subseteq Sol(E)$ can be verified by checking whether every equation e in the conjunction E is implied by I. This can be done in PTIME using Khachiyan's algorithm $2n$ times where n is the number of equations in E. In a similar way one can implement the test $Sol(I) \subseteq Sol(\neg d)$ in PTIME when d is a negative Horn constraint. ■

We have just showed that the consistency of a set of Horn constraints can be determined in PTIME. This is an important result with potential applications in any CLP or CDB system dealing with linear constraints [13, 16, 21]. We will now turn our attention to the problem of eliminating one or more variables from a given set of Horn constraints.

4 Variable Elimination

In this section we study the problem of variable elimination for sets of Horn constraints. The algorithm VARELIMINATION, shown in Figure 2, eliminates a given variable x from a set of Horn constraints C. More variables can be eliminated by successive applications of VARELIMINATION.

The first step of VARELIMINATION rewrites every constraint containing variable x in a convenient form: each constraint involving x is rewritten as $x \neq A \ \vee \ \phi$ or $x \leq U \ \vee \ \phi$ or $L \leq x \ \vee \ \phi$ where U, L, A and ϕ do *not* contain x. Then the algorithm proceeds to eliminate x using a procedure similar to the one presented in [18, 10, 9] for the case when we only have inequalities and negative Horn constraints. Notice that VARELIMINATION does *not* consider inequalities separately from positive disjunctive Horn constraints (as algorithm CONSISTENCY did in Section 3).

The following is an example of the algorithm in operation.

Example 4.1 Let C be the following set of constraints:

$$7 \leq x_1 \ \vee \ x_5 \neq 3 \tag{1}$$

$$x_1 - x_2 \leq 0 \tag{2}$$

$$x_1 \leq 8 \ \vee \ x_1 \neq x_3 \ \vee \ x_1 \neq x_4 \tag{3}$$

$$x_1 \neq x_6 \tag{4}$$

Let us assume that we want to eliminate variable x_1. The rewriting step of Algorithm VARELIMINATION transforms the above set into the following:

$$7 \leq x_1 \ \vee \ x_5 \neq 3 \tag{5}$$

Algorithm VarElimination
Input: A set of Horn constraints C in variables X, and a variable $x \in X$ to be eliminated from C.
Output: A set of Horn constraints C' in variables $X \setminus \{x\}$ such that $Sol^*(C') = Projection_{X \setminus \{x\}}(Sol^*(C))$.
Method:

Step 1:
For each constraint $c \in C$ **do**
  If c contains a disequation involving x **then**
    Rewrite c as $x \neq A \ \vee \ \phi$ such that ϕ and A do not contain x
  ElseIf c contains an inequality involving x **then**
    Rewrite c as $x \leq U \ \vee \ \phi$ or $L \leq x \ \vee \ \phi$ such that U, L and ϕ do not contain x
  EndIf
EndFor

Step 2:
$C' := \emptyset$
For each pair of positive Horn constraints $x \leq U \ \vee \ \phi_1$ and $L \leq x \ \vee \ \phi_2$ **do**
  Add $L \leq U \ \vee \ \phi_1 \ \vee \ \phi_2$ to C'
End For

Step 3:
For each pair of positive Horn constraints $x \leq U \ \vee \ \phi_1$ and $L \leq x \ \vee \ \phi_2$ **do**
  For each Horn constraint $x \neq A \ \vee \ \phi$ **do**
    Add $A \neq L \ \vee \ A \neq U \ \vee \ \phi \ \vee \ \phi_1 \ \vee \ \phi_2$ to C'
  End For
End For

Step 4:
Add each constraint not containing x to C'

Return C'

Figure 2: A variable elimination algorithm

$$x_1 \le x_2 \quad (6)$$
$$x_1 \ne x_4 \ \vee \ x_4 \le 8 \ \vee \ x_4 \ne x_3 \quad (7)$$
$$x_1 \ne x_6 \quad (8)$$

Now constraints 5 and 6 are used by Step 2 to derive the new constraint

$$7 \le x_2 \ \vee \ x_5 \ne 3.$$

Similarly, constraints 5, 6 and 7 are used by Step 3 to derive

$$x_4 \ne 7 \ \vee \ x_4 \ne x_2 \ \vee \ x_4 \le 8 \ \vee \ x_4 \ne x_3 \ \vee \ x_5 \ne 3.$$

Also, constraints 5, 6 and 8 are used by Step 3 to derive

$$x_6 \ne 7 \ \vee \ x_6 \ne x_2 \ \vee \ x_5 \ne 3.$$

Step 4 does not introduce any additional constraints. Thus VarElimination returns the following set:

$$7 \le x_2 \ \vee \ x_5 \ne 3,$$
$$x_4 \le 8 \ \vee \ x_4 \ne 7 \ \vee \ x_4 \ne x_2 \ \vee \ x_4 \ne x_3 \ \vee \ x_5 \ne 3,$$
$$x_6 \ne 7 \ \vee \ x_6 \ne x_2 \ \vee \ x_5 \ne 3$$

Theorem 4.1 The algorithm VarElimination is correct.

Proof: Let the variables of C be $X = \{x, x_2, \ldots, x_n\}$. If $(x^0, x_2^0, \ldots, x_n^0) \in \mathcal{R}^n$ is an element of $Sol^*(C)$ then it can be easily seen that $(x_2^0, \ldots, x_n^0)$ is an element of $Sol^*(C')$.

Conversely, take $(x_2^0, \ldots, x_n^0) \in \mathcal{R}^{n-1} \cap Sol^*(C')$ and consider the set of constraints $C(x, x_2^0, \ldots, x_n^0)$ obtained by substituting x_i^0 for x_i $(i = 2, \ldots, n)$ in C. If $C(x, x_2^0, \ldots, x_n^0)$ is simplified by removing constraints equivalent to true, parts of disjunctions equivalent to false (because disjunctions cannot be equivalent to false), and redundant constraints then

$$C(x, x_2^0, \ldots, x_n^0) = \{l^0 \le x, x \le u^0\} \cup \{x \ne a_k^0, \ k = 1, \ldots, \lambda\}.$$

Let us now assume (by contradiction) that there is no value $x^0 \in \mathcal{R}^n$ such that $(x^0, x_2^0, \ldots, x_n^0) \in Sol^*(C)$. This can happen only under the following cases:

1. $u^0 < l^0$. In this case inequalities $x \le u^0$ and $l^0 \le x$ come from positive Horn constraints rewritten as $x \le u \vee \phi_1$ and $l \le x \vee \phi_2$ in Step 1 of the algorithm. Therefore
$$\phi_1(x_2^0, \ldots, x_n^0) \equiv \phi_2(x_2^0, \ldots, x_n^0) \equiv false$$
otherwise these constraints would have been discarded from the set of constraints $C(x, x_2^0, \ldots, x_n^0)$ during its simplification. But because
$$l \le u \ \vee \ \phi_1 \ \vee \ \phi_2 \in C'$$
and $(x_2^0, \ldots, x_n^0) \in Sol^*(C')$ then $l^0 \le u^0$. Contradiction!

2. $l^0 = u^0 = a_j^0$ for some j such that $1 \le j \le \lambda$. With reasoning similar to the above, we can show that this case is also impossible.

Basic Interval Relation	Symbol	Pictorial Representation	Endpoint Constraints
i before j	b	iiiiiiii	$i^- < j^-,\ i^- < j^+,$
j after i	bi	jjjjjjjj	$i^+ < j^-,\ i^+ < j^+$
i meets j	m	iiiiiiii	$i^- < j^-,\ i^- < j^+,$
j met-by i	mi	jjjjjjjjjj	$i^+ = j^-,\ i^+ < j^+$
i overlaps j	o	iiiiiiii	$i^- < j^-,\ i^- < j^+,$
j overlapped-by i	oi	jjjjjjjjjjj	$i^+ > j^-,\ i^+ < j^+$
i during j	d	iiiiiiii	$i^- > j^-,\ i^- < j^+,$
j includes i	di	jjjjjjjjjjjjjjjjj	$i^+ > j^-,\ i^+ < j^+$
i starts j	s	iiiiiiiii	$i^- = j^-,\ i^- < j^+,$
j started-by i	si	jjjjjjjjjjjjjjj	$i^+ > j^-,\ i^+ < j^+$
i finishes j	f	iiiiiiiiii	$i^- > j^-,\ i^- < j^+,$
j finished-by i	fi	jjjjjjjjjjjjjjjj	$i^+ > j^-,\ i^+ = j^+$
i equals j	eq	iiiiiiiiiii	$i^- = j^-,\ i^- < j^+,$
		jjjjjjjjjjj	$i^+ > j^-,\ i^+ = j^+$

Figure 3: The thirteen basic relations of Allen

Finally, we can conclude that there exists a value $x^0 \in \mathcal{R}$ such that

$$(x, x_2^0, \ldots, x_n^0) \in Sol^*(C).$$

■

Let C be a set of Horn constraints. Eliminating m variables from C with repeated applications of the above algorithm will result in a set with $O(|C|^{2^m})$ Horn constraints. Many of these contraints will be redundant; it is therefore important to extend this work with efficient redundancy elimination algorithms that can be used together with VarElimination.

This section concludes our study of the basic reasoning problems concerning Horn constraints. We will now turn our attention to a subclass of Horn constraints with important applications to temporal reasoning.

5 Reasoning with Ord-Horn Constraints

This section studies a special class of Horn constraints, called Ord-Horn constraints, originally introduced in [31]. This class is important in interval based temporal reasoning [1] as we will immediately show below.

Definition 5.1 An *Ord-Horn constraint* is a Horn constraint $d_1 \lor \cdots \lor d_n$ where each $d_i,\ i = 1, \ldots, n$ is an inequality $x \leq y$ or a disequation $x \neq y$ and x and y are variables ranging over the real numbers.

Example 5.1 The following are examples of Ord-Horn constraints:

$$x_1 \leq x_2,\ x_1 \neq x_3,\ x_1 \neq x_3\ \lor\ x_2 \neq x_4,$$
$$x_1 \leq x_2\ \lor\ x_3 \neq x_4\ \lor\ x_5 \neq x_6$$

The first and the last constraint are positive while the second and the third are negative.

In [1], Allen introduced a calculus for reasoning about intervals in time. An *interval* is an element of the following set $\mathcal{I}$:

$$\mathcal{I} = \{(b, e) :\ b, e \in \mathcal{R} \text{ and } b < e\}$$

If i is an interval variable, i^- and i^+ will denote the endpoints of i. Allen's calculus is based on thirteen mutually exclusive binary relations which can capture all the possible ways two time intervals can be related. These *basic* relations are

$$before,\ meets,\ overlaps,\ during,\ starts,\ finishes,\ equals$$

and their inverses (*equals* is its own inverse). Figure 3 gives a pictorial representation of these relations. For reasons of brevity, we will use the symbols b, m, o, d, s, f and eq to refer to the basic relations in Allen's formalism. The inverse of each relation will be denoted by the name of the relation with the suffix i (for example, the inverse of b will be denoted by bi).

Allen's calculus has received a lot of attention and has been the formalism of choice for representing qualitative interval information. Whenever the interval information to be represented is indefinite, a disjunction of some of the thirteen basic relations can be used to represent what is known. There are 2^{13} such disjunctions representing qualitative relations between two intervals. Each one of these relations will be denoted by the set of its constituent basic relations e.g., $\{b, bi, d, m\}$. The *empty* relation will be denoted by $\emptyset$, and the *universal* relation will be denoted by $\perp$. The set of all 2^{13} relations expressible in Allen's formalism will be denoted by $\mathcal{A}$ [31].

The following definition will be useful below.

Definition 5.2 Let $\mathcal{S}$ be a subset of $\mathcal{A}$. An $\mathcal{S}$-*constraint* is an expression of the form $i\ R\ j$ where i and j are interval variables and $R \in \mathcal{S}$.

Example 5.2 If interval i denotes the time that John reads his morning newspaper and j denotes the time that he has breakfast, and we know that John never reads a newspaper while he is eating, then the $\mathcal{A}$-constraint

$$i\ \{b, bi, m, mi\}\ j$$

characterizes i and j according to the information given.

Definition 5.3 Let C be a set of $\mathcal{S}$-*constraints*. The solution set of C is:

$$Sol(C) = \{((b_1, e_1), \ldots, (b_n, e_n)) \in \mathcal{I}^n :$$

$$\text{for every } c \in C,\ ((b_1, e_1), \ldots, (b_n, e_n)) \text{ satisfies } c\}$$

Unfortunately, all interesting reasoning problems associated with Allen's interval calculus are NP-hard [37] therefore it is interesting to consider subsets of Allen's formalism, in the form of subsets of $\mathcal{A}$, that have better computational properties.[2] Three such subsets of $\mathcal{A}$ have received more attention:

- The set $\mathcal{C}$ which consists of all relations $R \in \mathcal{A}$ which satisfy the following condition. If i and j are intervals, $i\ R\ j$ can be equivalently expressed as a conjunction of inequalities $p_1 < p_2$ or $p_1 \leq p_2$ where p_1 and p_2 are endpoints of i and j.

  The set $\mathcal{C}$ is called the *continuous endpoint subclass* of $\mathcal{A}$ [37].

- The set $\mathcal{P}$ which consists of all interval relations $R \in \mathcal{A}$ which satisfy the following condition. If i and j are intervals, $i\ R\ j$ can be equivalently expressed as a conjunction of inequalities $p_1 < p_2$, $p_1 \leq p_2$ or disequation $p_1 \neq p_2$ where p_1 and p_2 are endpoints of i and j.

  The set $\mathcal{P}$ is called the *pointisable subclass* of $\mathcal{A}$ [37, 36, 35]. Because $\mathcal{C} \subset \mathcal{P}$ the pointisable subclass is more expressive than the continuous endpoint subclass.

[2] At the expense of being less expressive; no free meal here!

- The set $\mathcal{H}$ which consists of all interval relations $R \in \mathcal{A}$ which satisfy the following condition. If i and j are intervals, $i\ R\ j$ can be equivalently expressed as a conjunction of Ord-Horn constraints on the endpoints of i and j. The disjunctive Ord-Horn constraints involved in this equivalence are not arbitrary. There are at most three of them, and each one consists of *two* disjuncts of the form $i^-\ op_1\ j^-$ and $i^+\ op_2\ j^+$ where $op_1,\ op_2$ is $\leq$ or $\neq$.

  The set $\mathcal{H}$ was introduced by Nebel and Bürckert and named the *Ord-Horn subclass* [31]. Because $\mathcal{P} \subset \mathcal{H}$ the Ord-Horn subclass is more expressive than the pointisable subclass. It consists of 868 relations i.e., it covers more than 10% of $\mathcal{A}$.

Example 5.3 The following are $\mathcal{P}$-constraints:

$$i_1\ \{d, o\}\ i_2, \quad i_3\ \{d, s\}\ i_4$$

Their equivalent endpoint constraints are:

$$i_1^- < i_1^+,\ i_2^- < i_2^+,\ i_1^- \neq i_2^-,\ i_1^- < i_2^+,\ i_1^+ > i_2^-,\ i_1^+ < i_2^+$$

$$i_3^- < i_3^+,\ i_4^- < i_4^+,\ i_4^- \leq i_3^-,\ i_3^- < i_4^+,\ i_3^+ > i_4^-,\ i_3^+ < i_4^+$$

The second $\mathcal{P}$-constraint is also a $\mathcal{C}$-constraint while the first one is not. For an enumeration of $\mathcal{C}$ and $\mathcal{P}$, see [36].

Example 5.4 The $\mathcal{A}$-constraint $i\ \{b, bi\}\ j$ is not an $\mathcal{H}$-constraint. The constraint

$$i_1\ \{b, d, di, o, oi, si\}\ i_2$$

is an $\mathcal{H}$-constraint which is not a $\mathcal{P}$-constraint. Its equivalent endpoint constraints are:

$$i_1^- < i_i^+,\ i_2^- < i_2^+,\ i_1^- < i_2^+,\ i_1^+ \neq i_2^-,\ i_1^+ \neq i_2^+,$$
$$i_1^- \neq i_2^- \ \vee\ i_1^+ \geq i_2^+,\quad i_1^- \neq i_2^- \ \vee\ i_1^+ \neq i_2^+$$

An enumeration of $\mathcal{H}$ together with several related C programs has been provided by Nebel and Burckert. See [31] for details.

The following reasoning problems have been studied for the above subclasses [37, 36, 35, 27, 31].

- Given a set C of $\mathcal{S}$-constraints, decide whether C is consistent.
- Given a set C of $\mathcal{S}$-constraints, determine the strongest feasible relation between each pair of interval variables i and j. The *strongest feasible relation* between two interval variables i and j is the smallest set R such that $C \models i\ R\ j$. This is the same as computing the *minimal network* corresponding to the given set of constraints.[3]

In this section we will show how our results can be used to improve the best known algorithms for the above reasoning problems in the case of the Ord-Horn subclass. We start with two theorems from [31].

Theorem 5.1 *Let C be a set of $\mathcal{H}$-constraints. Deciding whether C is consistent can be done in $O(n^3)$ time where n is the number of variables in C.*

[3] We will not define minimal networks formally here. The interested reader can consult [35] (or many other temporal reasoning papers).

Theorem 5.2 *Let C be a set of $\mathcal{H}$-constraints. Computing the feasible relations between all pairs of variables can be done in $O(n^5)$ time where n is the number of variables in C.*

We will now use the results of Section 3 to provide new complexity bounds for the reasoning problems in the above theorems.

Theorem 5.3 *Let C be a set of $\mathcal{H}$-constraints. Let n be the number of variables in C, and h be the number of constraints $(i\ R\ j) \in C$ such that $R \in \mathcal{H} \setminus \mathcal{C}$. Deciding whether C is consistent can be done in $O(\max(n^2, hn))$ time.*

Proof: First we translate C into a set of Ord-Horn constraints C'. Since $|C| = O(n^2)$, this translation can be achieved in $O(n^2)$ time. Let $C' = I \cup D_p \cup D_n$ where I is a set of inequalities, D_p is a set of positive disjunctive Horn constraints and D_n a set of negative Horn constraints. The translation of Nebel and Bürckert shows that C' contains $n_1 = 2n$ point variables and $|D_p \cup D_n| = O(h)$ [31].

We will use algorithm Consistency from Figure 1 to decide C'. In this case Consistency can be made to work in $O(\max({n_1}^2, |D_p \cup D_n|\, n_1))$ time as follows. Checking the consistency of I can be done in $O({n_1}^2)$ time by constructing a directed graph G corresponding to I and examining its strongly connected components [35]. Now notice that the statement If-Else-End-If in algorithm Consistency is executed $O(|D_p \cup D_n|)$ times. Each execution of this statement takes $O(n_1)$ time. Let us see why. If the examined constraint d is in D_p, the test $Sol(I) \subseteq Sol(E)$ amounts to checking whether the single inequality E is implied by I. This can be done in $O(n_1)$ time by examining the strongly connected components of G. Similarly, if d is in D_n, the test $Sol(I) \subseteq Sol(\neg d)$ can be done in $O(n_1)$ time. Therefore deciding whether C' is consistent can be done in $O(\max({n_1}^2, |D_p \cup D_n|\, n_1))$ time. Thus deciding whether C is consistent can be done in $O(\max(n^2, hn))$ time. ■

Theorem 5.4 *Let C be a set of $\mathcal{H}$-constraints. Let n be the number of variables in C, and h be the number of constraints $(i\ R\ j) \in C$ such that $R \in \mathcal{H} \setminus \mathcal{C}$. Computing the feasible relations between all pairs of variables can be done in $O(\max(n^4, hn^3))$ time.*

Proof: As in [31], we will consider all $O(n^2)$ pairs of variables in turn. For each pair we check whether each of the thirteen basic relations is consistent with the given constraints. The basic relations that satisfy this criterion form the strongest feasible relation between the pair. Using the algorithm of Theorem 5.3, each check can be done in $O(\max(n^2, hn))$ time. The bound of the theorem follows immediately. ■

In the worst case the parameter h, as specified in the above theorems, can be $O(n^2)$. In practical applications we expect h to be significantly less than $O(n^2)$ because the constraints in $\mathcal{H} \setminus \mathcal{C}$ do not arise frequently in practice. Thus the above theorems improve the results of [31].

Here we conclude our discussion of algorithms for consistency checking and computing the strongerst feasible constraints for the Ord-Horn subclass. The following section develops a global consistency algorithm for Horn constraints. This algorithm will be used in Section 7 for the study of global consistency in the Ord-Horn class.

6 Global Consistency of Horn Constraints

The main idea behind algorithm VarElimination (Figure 2, Section 4) can be used to develop an algorithm for computing a globally consistent constraint set equivalent to a set of Horn constraints. In a *globally consistent* constraint set all "interesting" constraints are explicitly represented and the projection of the solution set on any subset of the variables can be computed by simply collecting the constraints involving these variables. Thus it is very easy to eliminate variables from a globally consistent set: we simply remove all constraints in which the variables to be eliminated occur.

Another important consequence of global consistency is that a solution can be found by *backtrack-free* search [7]. Enforcing global consistency can take an exponential amount of time in the worst case [6, 3]. As a result it is very important to identify cases in which *local consistency*, which presumably can be enforced in polynomial time, implies global consistency [4].

In [22, 23] we have studied in detail the global consistency problem for a subclass of Horn constraints: inequalities of the form $x_i - x_j \leq r$, and disjunctions of disequations of the form $x_i - x_j \neq r$. We showed that strong 5-consistency is necessary and sufficient for achieving global consistency for sets of such constraints. The same result is true for sets of point-algebra constraints. We also developed $O(n^5)$ global consistency algorithms for these cases. In this section we present an algorithm for global consistency of Horn constraints. In Section 7 we will utilize this algorithm to study global consistency for sets of $\mathcal{H}$-constraints.

The following notation will be useful. Let C be a set of constraints in variables $x_1, \ldots, x_n$. For any i such that $1 \leq i \leq n$, $C(x_1, \ldots, x_i)$ will denote the set of constraints in C involving only variables $x_1, \ldots, x_i$.

The following definition is from [4].

Definition 6.1 Let C be a set of constraints in variables $x_1, \ldots, x_n$ and $1 \leq i \leq n$. C is called *i-consistent* iff for every $i-1$ distinct variables $x_1, \ldots, x_{i-1}$, every valuation $u = \{x_1 \leftarrow x_1^0, \ldots, x_{i-1} \leftarrow x_{i-1}^0\}$ such that u is a solution of $C(x_1, \ldots, x_{i-1})$ and every variable x_i different from $x_1, \ldots, x_{i-1}$, there exists a real number x_i^0 such that u can be extended to a valuation $u' = u \cup \{x_i \leftarrow x_i^0\}$ which is a solution of $C(x_1, \ldots, x_{i-1}, x_i)$. C is called *strong i-consistent* if it is j-consistent for every j, $1 \leq j \leq i$. C is called *globally consistent* iff it is i-consistent for every i, $1 \leq i \leq n$.

Let us present some examples illustrating the above definitions.

Example 6.1 The constraint set $C = \{x_2 - x_1 \leq 5,\ x_1 - x_3 \leq 2,\ x_5 - x_4 \leq 1,\ x_4 - x_6 \leq 3\}$ is 1- and 2-consistent but not 3-consistent. For example, the valuation $v = \{x_2 \leftarrow 10,\ x_3 \leftarrow 2\}$ is a solution of $C(x_2, x_3) = \emptyset$ but it cannot be extended to a valuation which is a solution of C.

We can enforce 3-consistency by adding the constraints $x_2 - x_3 \leq 7$ and $x_5 - x_6 \leq 4$ to C. The resulting set is 3-consistent and also globally consistent.

Example 6.2 The constraint set

$$C = \{x_1 - x_2 \leq 5,\ x_3 - x_1 \leq 7\ \vee\ x_4 \neq 0\}$$

is 1-,2- and 3-consistent but not 4-consistent. For example, the valuation $v = \{x_2 \leftarrow 0,\ x_3 \leftarrow 20,\ x_4 \leftarrow 0\}$ is a solution of $C(x_2, x_3, x_4) = \emptyset$ but it cannot be extended to a valuation which is a solution of C.

We can enforce 4-consistency by adding the constraint $x_3 - x_2 \leq 12\ \vee\ x_4 \neq 0$ to C. The resulting set is 4-consistent and also globally consistent.

Figure 4 presents an algorithm which computes a globally consistent set of Horn constraints equivalent to its input set. GlobalConsistency proceeds by examining all variables one by one, each time generating all constraints that would be generated if the variable was to be eliminated. The generated constraints are then added to the original constraint set and the process continues. The variables

Algorithm GLOBALCONSISTENCY
Input: A set of Horn constraints C in variables $X = \{x_1, \ldots, x_n\}$.
Output: A globally consistent set of Horn constraints equivalent to C.
Method:

```
Temp := ∅;  C_new := C
For i = 1 to n do

   For each constraint c ∈ C_new do
      If c contains a disequation involving x_i then
         Rewrite c as x_i ≠ A ∨ φ such that φ and A do not contain x_i
      ElseIf c contains an inequality involving x_i then
         Rewrite c as x_i ≤ U ∨ φ or L ≤ x_i ∨ φ such that U, L and φ
         do not contain x_i
      EndIf
   EndFor

   For each pair of positive Horn constraints x_i ≤ U ∨ φ_1 and L ≤ x_i ∨ φ_2
   in C_new do
      Add L ≤ U ∨ φ_1 ∨ φ_2 to Temp
   End For

   For each pair of positive Horn constraints x_i ≤ U ∨ φ_1 and L ≤ x_i ∨ φ_2
   in C_new do
      For each Horn constraint x_i ≠ A ∨ φ in C_new do
         Add A ≠ L ∨ A ≠ U ∨ φ ∨ φ_1 ∨ φ_2 to Temp
      End For
   End For

   C_new := C_new ∪ Temp;  Temp := ∅
End For

Return C_new
```

Figure 4: An algorithm for global consistency

are "eliminated" by GLOBALCONSISTENCY in ascending numerical order of their indices.[4]

Let C be a set of Horn constraints with variable set $X = \{x_1, \ldots, x_n\}$, and $S \subseteq X$. Then $\pi_{X \setminus S}(C)$ will denote the set of constraints computed by eliminating all variables of S from C using algorithm VARELIMINATION repeatedly (the variables are eliminated following the numerical order of their indices).

The following lemma and theorem demonstrate the correctness of GLOBALCONSISTENCY.

Lemma 6.1 *After the outer for loop of algorithm* GLOBALCONSISTENCY *is executed for the j-th time ($j = 1, \ldots, n$), C_{new} is a superset of all sets $\pi_{X \setminus S}(C)$ where S is a non-empty subset of $\{x_1, \ldots, x_j\}$.*

Proof: The proof is by induction on j. The base case ($j = 1$) is trivial. For the inductive step, let us assume that the lemma holds for $j = k$. This means that after the outer for loop of GLOBALCONSISTENCY is executed for the k-th time, C_{new} is a superset of all sets $\pi_{X \setminus S}(C)$ where S is a non-empty subset of $\{x_1, \ldots, x_k\}$. When the outer loop is executed for the $(k+1)$-th time, we are computing the set $\pi_{X \setminus \{x_{k+1}\}}(C)$, but also all sets

$$\pi_{X \setminus (S \cup \{x_{k+1}\})}(C)$$

where S is a non-empty subset of $\{x_1, \ldots, x_k\}$. Therefore, after the outer loop has been executed for the $(k+1)$-th time, C_{new} is a superset of all sets $\pi_{X \setminus S}(C)$ where S is a non-empty subset of $\{x_1, \ldots, x_k, x_{k+1}\}$. We have now shown that the lemma holds for $j = k + 1$ as well. ■

Theorem 6.1 *Algorithm* GLOBALCONSISTENCY *is correct. The worst-case complexity of* GLOBALCONSISTENCY *is $O(|C|^{2^n})$ where C is the input set and n is the number of variables in C.*

Proof: Let us assume that GLOBALCONSISTENCY returns a constraint set C_{new} which is not globally consistent (proof by contradiction). This means that there is a k, $1 \leq k \leq n$ such that C_{new} is not k-consistent. In other words, there are variables $x_{i_1}, \ldots, x_{i_{k-1}}, x_{i_k}$ (such that $i_1 < \cdots < i_{k-1} < i_k$) and a valuation

$$u = \{x_{i_1} \leftarrow x^0_{i_1}, \ldots, x_{i_{k-1}} \leftarrow x^0_{i_{k-1}}\}$$

such that u is a solution of $C_{new}(x_{i_1}, \ldots, x_{i_{k-1}})$, but u cannot be extended to a valuation

$$u' = u \cup \{x_{i_k} \leftarrow x^0_{i_k}\}$$

which is a solution of $C_{new}(x_{i_1}, \ldots, x_{i_{k-1}}, x_{i_k})$.

If $S = X \setminus \{x_{i_1}, \ldots, x_{i_{k-1}}\}$ then Lemma 6.1 implies that

$$\pi_{X \setminus S}(C) \subseteq C_{new}.$$

Because u is a solution of $C_{new}(x_{i_1}, \ldots, x_{i_{k-1}})$, we can conclude that u is also a solution of $\pi_{X \setminus S}(C)$. But this means that u can be extended to a solution of C (Theorem 4.1), and also to a solution of C_{new} (because all constraints in C_{new} are implied by C). This is a contradiction to our assumption about u. Thus we can conclude that C_{new} is indeed globally consistent.

[4]This does *not* make GLOBALCONSISTENCY a directional algorithm in the sense of [4]. In our case *all* constraints participate in the elimination operation performed at each step. In a directional algorithm only the constraints with variables following the eliminated one (in the ordering used) participate in an elimination operation.

For the complexity result notice that GLOBALCONSISTENCY examines n variables, and that eliminating n variables repeatedly takes $O(|C|^{2^n})$ time. ■

Algorithm GLOBALCONSISTENCY must be improved if it is to be useful in any practical situation. An obvious improvement is to avoid the generation of trivially redundant constraints.

Definition 6.2 *Let C be a set of constraints and $c \in C$. The constraint c is* trivially redundant *if it is equivalent to $true$ or $false$ or if there is another constraint $c' \in C$ such that $Sol(c') \subseteq Sol(c)$.*

Example 6.3 Let C be the following set of constraints

$$x_1 - x_2 \leq 16,\ x_1 - x_2 \neq 20\ \vee\ x_3 \neq 5,\ x_1 - x_2 \leq 30,\ x_4 \neq 3\ \vee\ x_4 \neq 4.$$

All constraints except the first one are trivially redundant.

We will not deal with any other improvements of algorithm GLOBALCONSISTENCY. The above version will be enough for the development of Section 7.

7 Global Consistency in $\mathcal{H}$

Let us assume that we have a set C of $\mathcal{H}$-constraints. This section presents a simple way to compute a globally consistent set of constraints equivalent to C. First, we translate C into a set of Ord-Horn constraints C'. Then we compute a globally consistent set C'' equivalent to C' using algorithm GLOBALCONSISTENCY of Section 6. Finally, we translate C'' back to a set of *disjunctions* of $\mathcal{H}$-constraints. The resulting set is globally consistent.

We will now specify the two translation functions necessary for the above process to work, and prove the correctness of the process. Then we will utilize this machinery in the study of the following important question. Is there a level of *local* consistency that is necessary and sufficient for achieving global consistency in $\mathcal{H}$?

7.1 An Algorithm for Global Consistency

In the rest of this paper $OrdHornCon$ will denote the set of all Ord-Horn constraints. If $\mathcal{S}$ is a subclass of $\mathcal{A}$ then $Con(\mathcal{S})$ will denote the set of all $\mathcal{S}$-constraints and $DisjCon(\mathcal{S})$ the set of all disjunctions of $\mathcal{S}$-constraints.

In [31, 32], Nebel and Bürckert showed how to translate $\mathcal{H}$-constraints into Ord-Horn constraints. If c is an $\mathcal{H}$-constraint then $\tau(c)$ will denote the set of Ord-Horn constraints equivalent to c according to the translation of Nebel and Bürckert [31, 32].

Let us now extend this translation function to sets of $\mathcal{H}$-constraints:[5]

$$\tau : 2^{Con(\mathcal{H})} \longrightarrow 2^{OrdHornCon}$$

If C is a set of $\mathcal{H}$-constraints in variables $i_1, \ldots, i_n$ then $\tau(C)$ is a set of Ord-Horn constraints in variables $i_1^-, i_1^+, \ldots, i_n^-, i_n^+$. This set is defined as follows:

$$\tau(C) = \bigcup_{c \in C} \tau(c)$$

We will also need another translation function which, with some abuse of notation, we will denote by τ^{-1}.

$$\tau^{-1} : 2^{OrdHornCon} \longrightarrow 2^{DisjCon(\mathcal{H})}$$

[5] As usual 2^S denotes the powerset of set S.

Let C be a set of Ord-Horn constraints in variables $i_1^-, i_1^+, \ldots, i_n^-, i_n^+$ and

$$\{i_1^- < i_1^+, \ldots, i_n^- < i_n^+\} \subseteq C.$$

Then $\tau^{-1}(C)$ is a set of *disjunctions* of $\mathcal{H}$-constraints in variables $i_1, \ldots, i_n$ defined as follows:

$$\tau^{-1}(C) = \{\tau^{-1}(c) : \ c \in C \setminus \{i_1^- < i_1^+, \ldots, i_n^- < i_n^+\}\ \}$$

The disjunction $\tau^{-1}(c)$ is defined as follows:[6]

1. If c is of the form $i_k^- \leq i_l^-$ then $\tau^{-1}(c)$ is $i_k \ \{eq, b, di, o, m, s, si, fi\} \ i_l$.
2. If c is of the form $i_k^- \neq i_l^-$ then $\tau^{-1}(c)$ is $i_k \ \{b, bi, d, di, o, oi, m, mi, f, fi\} \ i_l$.
3. If c is of the form $i_k^- \leq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, d, di, o, oi, m, mi, s, si, f, fi\} \ i_l.$$
4. If c is of the form $i_k^- \neq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, m, s, si, f, fi\} \ i_l.$$
5. If c is of the form $i_k^+ \leq i_l^-$ then $\tau^{-1}(c)$ is $i_k \ \{b, m\} \ i_l$.
6. If c is of the form $i_k^+ \neq i_l^-$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, mi, s, si, f, fi\} \ i_l.$$
7. If c is of the form $i_k^+ \leq i_l^+$ then $\tau^{-1}(c)$ is $i_k \ \{eq, b, d, o, m, s, f, fi\} \ i_l$.
8. If c is of the form $i_k^+ \neq i_l^+$ then $\tau^{-1}(c)$ is $i_k \ \{b, bi, d, di, o, oi, m, mi, s, si\} \ i_l$.
9. If c is of the form $i_k^- \leq i_l^- \ \vee \ i_k^+ \neq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\} \ i_l.$$
10. If c is of the form $i_k^- \geq i_l^- \ \vee \ i_k^+ \neq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, m, mi, s, si, f\} \ i_l.$$
11. If c is of the form $i_k^- \neq i_l^- \ \vee \ i_k^+ \leq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, m, mi, s, f, fi\} \ i_l.$$
12. If c is of the form $i_k^- \neq i_l^- \ \vee \ i_k^+ \geq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{eq, b, bi, d, di, o, oi, m, mi, si, f, fi\} \ i_l.$$
13. If c is of the form $i_k^- \neq i_l^- \ \vee \ i_k^+ \neq i_l^+$ then $\tau^{-1}(c)$ is
$$i_k \ \{b, bi, d, di, o, oi, m, mi, s, si, f, fi\} \ i_l.$$
14. If c does not belong to any of the above categories, then it is a disjunction of the form $\bigvee_{\lambda=1}^{n} c_\lambda$. Then
$$\tau^{-1}(c) = \bigvee_{\lambda=1}^{n} \tau^{-1}(c_\lambda).$$

[6] Cases 1-8 can be verified using the enumeration of $\mathcal{P}$ provided by van Beek and Cohen [36]. Cases 9-13 can be verified using the enumeration of $\mathcal{H}$ provided by Nebel and Bürckert [31, 32].

Lemma 7.1 *Let C be a set of $\mathcal{H}$-constraints in variables $i_1, i_2, \ldots, i_n$. If valuation*

$$\{i_1 \leftarrow a_1, \ldots, i_n \leftarrow a_n\}$$

is a solution of C then valuation $\{i_1^- \leftarrow a_1^-, i_1^+ \leftarrow a_1^+, \ldots, i_n^- \leftarrow a_n^-, i_n^+ \leftarrow a_n^+\}$ is a solution of $\tau(C)$. Conversely, if valuation $\{i_1^- \leftarrow b_1, i_1^+ \leftarrow e_1, \ldots, i_n^- \leftarrow b_n, i_n^+ \leftarrow e_n\}$ is a solution of $\tau(C)$ then $\{i_1 \leftarrow \langle b_1, e_1\rangle, \ldots, i_n \leftarrow \langle b_n, e_n\rangle\}$ is a solution of C.

Proof: The lemma follows immediately from the correctness of the translation of Nebel and Bürckert [31]. ■

Lemma 7.2 *Let C be a set of Ord-Horn constraints in variables $i_1^-, i_1^+, \ldots, i_n^-, i_n^+$ and $\{i_1^- < i_1^+, \ldots, i_n^- < i_n^+\} \subseteq C$. If valuation*

$$\{i_1^- \leftarrow b_1,\ i_1^+ \leftarrow e_1, \ldots, i_n^- \leftarrow b_n,\ i_n^+ \leftarrow e_n\}$$

is a solution of C then $\{i_1 \leftarrow \langle b_1, e_1\rangle, \ldots, i_n \leftarrow \langle b_n, e_n\rangle\}$ is a solution of $\tau^{-1}(C)$. Conversely, if valuation $\{i_1 \leftarrow a_1, \ldots, i_n \leftarrow a_n\}$ is a solution of $\tau^{-1}(C)$ then valuation $\{i_1^- \leftarrow a_1^-,\ i_1^+ \leftarrow a_1^+, \ldots, i_n^- \leftarrow a_n^-,\ i_n^+ \leftarrow a_n^+\}$ is a solution of C.

Proof: We only sketch the proof. Firstly, we establish the truth of the lemma for the case where C consists of non-disjunctive constraints. For this, it is enough to check the correctness of the thirteen base cases in the definition of τ^{-1}. For disjunctive constraints, the proof proceeds by induction. ■

Theorem 7.1 *Let C_1 be a set of $\mathcal{H}$-constraints in n variables $i_1, \ldots, i_n$. Let C_2 be a strong $2m$-consistent set of Ord-Horn constraints equivalent to $\tau(C_1)$ (where $1 \leq m \leq n$). Then $\tau^{-1}(C_2)$ is equivalent to C_1 and strong m-consistent.*

Proof: Let $\{i_1 \leftarrow a_1, \ldots, i_n \leftarrow a_n\}$ be a solution of C_1. Then Lemma 7.1 implies that

$$v = \{i_1^- \leftarrow a_1^-,\ i_1^+ \leftarrow a_1^+, \ldots, i_n^- \leftarrow a_n^-,\ i_n^+ \leftarrow a_n^+\}$$

is a solution of $\tau(C_1)$. Therefore v is also a solution of C_2 given the assumptions of the theorem. Now from Lemma 7.2 we can conclude that $\{i_1 \leftarrow a_1, \ldots, i_n \leftarrow a_n\}$ is a solution of $\tau^{-1}(C_2)$.

The converses of the above implications also hold therefore we can conclude that $\tau^{-1}(C_2)$ is equivalent to C_1.

We now turn to the proof of strong m-consistency. Let us assume by contradiction that $\tau^{-1}(C_2)$ is not strong m-consistent. In other words, there are variables $i_1, \ldots, i_k (1 \leq k \leq m)$ and a valuation

$$v = \{i_1 \leftarrow a_1, \ldots, i_{k-1} \leftarrow a_{k-1}\}$$

such that v is a solution of $\tau^{-1}(C_2)(i_1, \ldots, i_{k-1})$, but there is no value a_k such that the valuation

$$v' = v \cup \{i_k \leftarrow a_k\}$$

is a solution of $\tau^{-1}(C_2)(i_1, \ldots, i_k)$. Let us now consider the valuation

$$v_p = \{i_1^- \leftarrow a_1^-,\ i_1^+ \leftarrow a_1^+, \ldots, i_{k-1}^- \leftarrow a_{k-1}^-,\ i_{k-1}^+ \leftarrow a_{k-1}^+\}.$$

From Lemma 7.2, we conclude that v_p is a solution of $C_2(i_1^-, i_1^+, \ldots, i_{k-1}^-, i_{k-1}^+)$. Because C_2 is strong $2m$-consistent, v_p can be extended to a valuation

$$v_p' = v_p \cup \{i_k^- \leftarrow b, i_k^+ \leftarrow e\}$$

Algorithm $\mathcal{H}$-GlobalConsistency
Input: A set of $\mathcal{H}$-constraints C.
Output: A globally consistent set of disjunctions of $\mathcal{H}$-constraints equivalent to C.
Method:
$C_1 := \tau(C)$
$C_2 :=$GlobalConsistency(C_1)
$C_3 :=$Normalize$(\tau^{-1}(C_2))$
Return C_3

Figure 5: Global consistency of $\mathcal{H}$-constraints

such that v'_p is a solution of $C_2(i_1^-, i_1^+, \ldots, i_k^-, i_k^+)$. From Lemma 7.2, we now have that the valuation

$$\{i_1 \leftarrow a_1, \ldots, i_{k-1} \leftarrow a_{k-1}, i_k \leftarrow \langle b, e\rangle\}$$

is a solution of $\tau^{-1}(C_2)$. This is a contradiction to our assumption. We can therefore conclude that $\tau^{-1}(C_2)$ is indeed strong m-consistent. ■

The following corollary is now obvious.

Corollary 7.1 *Let C_1 be a set of $\mathcal{H}$-constraints in n variables $i_1, \ldots, i_n$. Let C_2 be a globally consistent set of Ord-Horn constraints equivalent to $\tau(C_1)$ (where $1 \leq m \leq n$). Then $\tau^{-1}(C_2)$ is equivalent to C_1 and globally consistent.*

Now that we have the above results, we can develop a global consistency algorithm for $\mathcal{H}$. The algorithm is shown in Figure 5 and its correctness follows immediately from Corollary 7.1. Since $\mathcal{H}$-GlobalConsistency uses GlobalConsistency as a subroutine its worst-case time complexity is also doubly exponential in the number of variables in the input constraint set.

The first two steps of algorithm $\mathcal{H}$-GlobalConsistency are easy to understand. The function Normalize invoked in the third step takes as input a set of $\mathcal{H}$-constraints S and returns a set of $\mathcal{H}$-constraints S' such that S' contains at most one constraint of the form $i\ R\ j$ for every pair of variables i and j. Specifying the function Normalize is straightforward. First, for all pairs of distinct variables i and j, we collect all relations R_{ij} or R_{ji} such that $i\ R_{ij}\ j$ or $j\ R_{ji}\ i$ is in S. Then we compute the intersection R of all R_{ij} and R_{ij}^{-1} and add the constraint $i\ R\ j$ to S. Finally, we add to S' any constraints of S that have not been collected by the above process.

Example 7.1 Let us assume that $\mathcal{H}$-GlobalConsistency is called with the following input:[7]

$$i_1\ \{eq\}\ i_2, \quad i_2\ \{b, bi, d, di, o, oi, m, mi, s, si, f, fi\}\ i_3.$$

After the first step of the algorithm C_1 will contain the following constraints:

$$i_1^- \leq i_1^+,\ i_1^- \neq i_1^+,\ i_2^- \leq i_2^+,\ i_2^- \neq i_2^+,\ i_3^- \leq i_3^+,\ i_3^- \neq i_3^+,$$
$$i_1^- \leq i_2^-,\ i_2^- \leq i_1^-,\ i_1^+ \leq i_2^+,\ i_2^+ \leq i_1^+$$
$$i_2^- \neq i_3^- \ \vee\ i_2^+ \neq i_3^+$$

After the second step of the algorithm, C_2 will contain all constraints of C_1 plus the following:

$$i_1^- \leq i_2^+,\ i_1^- \neq i_2^+,\ i_2^- \leq i_1^+,\ i_2^- \neq i_1^+,$$

[7]It would be much easier to understand this example if you notice that the second constraint says that i_2 is not equal to i_3. This constraint can be expressed in $\mathcal{H}$ but not in $\mathcal{P}$.

$$i_1^- \neq i_3^- \ \vee\ i_1^+ \neq i_3^+$$

After the third step of the algorithm C_3 will contain the following constraints:

$$i_1\ \{eq\}\ i_2,$$

$$i_2\ \{b, bi, d, di, o, oi, m, mi, s, si, f, fi\}\ i_3,$$

$$i_1\ \{b, bi, d, di, o, oi, m, mi, s, si, f, fi\}\ i_3.$$

7.2 From Local to Global Consistency in $\mathcal{H}$

The question that remains now is whether there is a low level of local consistency which can guarantee global consistency for sets of $\mathcal{H}$-constraints. To answer this question, let us consider the following set C of $\mathcal{H}$-constraints:

$$i_1\ \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\}\ i_2,$$

$$i_2\ \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\}\ i_3,$$

$$i_2\ \{eq, f, fi\}\ k_1$$

The set of Ord-Horn constraints C' equivalent to the above set is:[8]

$$i_1^- < i_1^+,\ i_2^- < i_2^+,\ i_3^- < i_3^+,\ k_1^- < k_1^+,$$

$$i_1^- \leq i_2^- \ \vee\ i_1^+ \neq i_2^+,\ \ i_2^- \leq i_3^- \ \vee\ i_2^+ \neq i_3^+,\ \ i_2^+ = k_1^+$$

C' is 1- and 2-consistent but not 3-consistent. To enforce 3-consistency, we should add the following constraints:

$$i_2^- < k_1^+,\ k_1^- < i_2^+$$

The resulting set C_1' is 3-consistent, 4-consistent, but not 5-consistent. For example, the valuation

$$v = \{i_1^- \leftarrow 0,\ i_1^+ \leftarrow 10,\ i_2^- \leftarrow 2,\ k_1^+ \leftarrow 10\}$$

satisfies

$$C_1'(i_1^-, i_1^+, i_2^-, k_1^+) = \{i_2^- < k_1^+\}$$

but it cannot be extended to a valuation that satisfies $C_1'(i_1^-, i_1^+, i_2^-, i_2^+, k_1^+)$. To enforce 5-consistency, we should add the following constraints:

$$i_1^- \leq i_2^- \ \vee\ i_1^+ \neq k_1^+,\ \ i_2^- \leq i_3^- \ \vee\ k_1^+ \neq i_3^+$$

The resulting set C_2' is strong 5-consistent but not 6-consistent. To enforce 6-consistency, we should add the following constraints:

$$i_1^- \leq i_3^- \ \vee\ i_1^+ \neq i_2^+ \ \vee\ i_2^+ \neq i_3^+,\ \ i_1^- \leq i_3^- \ \vee\ i_1^+ \neq k_1^+ \ \vee\ k_1^+ \neq i_3^+$$

The resulting set C_3' is strong 6-consistent but not 7-consistent. To enforce 7-consistency, we should add the following constraints:

$$i_1^- \leq i_3^- \ \vee\ i_1^+ \neq i_2^+ \ \vee\ k_1^+ \neq i_3^+,\ \ i_1^- \leq i_3^- \ \vee\ i_1^+ \neq k_1^+ \ \vee\ i_2^+ \neq i_3^+$$

The resulting set is strong 7-consistent and also globally consistent.

The same reasoning can be applied to the original set of $\mathcal{H}$-constraints C. C is strong 3-consistent but not 4-consistent. For example, the valuation

$$v = \{i_1 \leftarrow \langle 5, 10\rangle,\ i_3 \leftarrow \langle 0, 10\rangle,\ k_1 \leftarrow \langle 0, 10\rangle\}$$

[8] To keep the example shorter we make use of the relation $<$.

satisfies $C(i_1, i_3, k_1) = \{\}$, but it cannot be extended to a valuation which satisfies C. To enforce 4-consistency, we should add the following disjunction to C:

$$i_1 \; \{eq, b, di, o, m, s, si, fi\} \; i_3 \; \vee \; i_1 \; \{b, bi, d, di, o, oi, m, mi, s, si\} \; k_1 \; \vee$$

$$k_1 \; \{b, bi, d, di, o, oi, m, mi, s, si\} \; i_3$$

This disjunction is *necessary* for enforcing 4-consistency in C, and there is no constraint with fewer interval variables that could achieve this. The resulting set is strong 4-consistent and also globally consistent. The added disjunction of $\mathcal{P}$-constraints is $\tau^{-1}(c)$ where c is the Ord-Horn constraint

$$i_1^- \leq i_3^- \;\vee\; i_1^+ \neq k_1^+ \;\vee\; k_1^+ \neq i_3^+$$

shown above (when enforcing 6-consistency of C_2').

Let us now consider the following set F of $\mathcal{H}$-constraints:

$$i_1 \; \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\} \; i_2,$$

$$i_2 \; \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\} \; i_3,$$

$$\vdots$$

$$i_{n-1} \; \{eq, b, bi, d, di, o, oi, m, mi, s, si, fi\} \; i_n,$$

$$i_2 \; \{eq, f, fi\} \; k_2, \; i_3 \; \{eq, f, fi\} \; k_3, \; \cdots, i_{n-1} \; \{eq, f, fi\} \; k_{n-1}$$

$\tau(F)$ is the following set of Ord-Horn constraints:

$$i_1^- < i_1^+, \; i_2^- < i_2^+, \; \cdots, i_n^- < i_n^+,$$

$$k_2^- < k_2^+, \; k_3^- < k_3^+, \; \cdots, k_{n-1}^- < k_{n-1}^+,$$

$$i_1^- \leq i_2^- \;\vee\; i_1^+ \neq i_2^+,$$

$$i_2^- \leq i_3^- \;\vee\; i_2^+ \neq i_3^+,$$

$$\vdots$$

$$i_{n-1}^- \leq i_n^- \;\vee\; i_{n-1}^+ \neq i_n^+,$$

$$i_2^+ = k_2^+, \; i_3^+ = k_3^+, \; \ldots, \; i_{n-1}^+ = k_{n-1}^+$$

Reasoning as above we can see that to enforce global consistency in $\tau(F)$, we should *necessarily* add the following disjunction:

$$i_{n-1}^- \leq i_n^- \;\vee\; i_1^+ \neq k_2^+ \;\vee\; k_2^+ \neq k_3^+ \;\vee\; \cdots \;\vee\; k_{n-1}^+ \neq i_n^+$$

The disjunction of $\mathcal{H}$-constraints corresponding to the above disjunction has n interval variables. If we try to enforce global consistency of F in the standard way (by enforcing 1-consistency, 2-consistency and so on), the above disjunction can only be discovered when enforcing $(n+1)$-consistency. The original set F has $2n-2$ interval variables. The following theorem is now obvious.

Theorem 7.2 *Let n be a natural number and $n \geq 4$. There is a set of $\mathcal{H}$-constraints C in n variables such that to enforce global consistency in C it is necessary to enforce strong $(\lfloor \frac{n+2}{2} \rfloor + 1)$-consistency.*

In other words, it is not possible to find a *low* level of local consistency which can guarantee global consistency in $\mathcal{H}$. This result must be contrasted with the following.

Theorem 7.3 *For the continuous endpoint subclass $\mathcal{C}$, strong 3-consistency is necessary and sufficient for achieving global consistency. For the pointizable subclass $\mathcal{P}$, strong 5-consistency is necessary and sufficient for achieving global consistency.*

Proof: The result for $\mathcal{C}$ is due to [36]. The result for $\mathcal{P}$ follows from [22, 23]. ■

Theorem 7.2 cannot be used to show that there is no PTIME algorithm for enforcing global consistency for sets of $\mathcal{H}$-constraints (the constraint exhibited does *not* have exponential size). This leaves us with an interesting open question.

8 Future Research

In future research we would like to concentrate our effort on the following issues:

- We will implement the algorithms of Section 5 and compare them with the path-consistency algorithms of [31].
- We will apply the algorithm of Section 3 to the problem of deciding the consistency of a set of (arbitrary) disjunctions of linear constraints. Previous work on this problem is reported in [2].
- We will apply the main result of Section 3 to the problem of tractable query answering in indefinite linear constraint databases [21, 24]. Preliminary results on this topic appear in [25, 26].

Acknowledgements

The comments of the reviewers have greatly improved the presentation of this paper (one of them has in fact spotted a small problem with an earlier version of the algorithms VarElimination and GlobalConsistency!).

References

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

[2] H. Beringer and B. de Backer. Satisfiability of Boolean Formulas over Linear Constraints. In *Proceedings of IJCAI-93*, pages 296–301. Morgan Kaufmann, 1993.

[3] M.C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41(1):89–95, 1990.

[4] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–107, 1992.

[5] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. Special Volume on Knowledge Representation.

[6] E. Freuder. Synthesizing Constraint Expressions. *Communications of ACM*, 21(11):958–966, November 1978.

[7] E. Freuder. A Sufficient Condition For Backtrack-Free Search. *Journal of ACM*, 29(1):24–32, 1982.

[8] B. Grunbaum. *Convex Polytopes*. John Wiley and Sons, 1967.

[9] J.-L. Imbert. Variable Elimination for Disequations in Generalized Linear Constraint Systems. *The Computer Journal*, 36(5):473–484, 1993. Special Issue on Variable Elimination.

[10] J.-L. Imbert. Variable Elimination for Generalized Linear Constraints: Particular problems involving disequalities. In *Proceedings of the 10th International Conference on Logic Programming*, pages 499–516. MIT Press, 1993.

[11] J.-L. Imbert. Redundancy, Variable Elimination and Linear Disequations. In *Proceedings of the International Symposium on Logic Programming*, pages 139–153, 1994.

[12] J.-L. Imbert and P. van Hentenryck. On the Handling of Disequations in CLP over Linear Rational Arithmetic. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research*, Logic Programming Series, pages 49–71. MIT Press, 1993.

[13] J. Jaffar and M.J. Maher. Constraint Logic Programming: A Survey. *Journal of Logic Programming*, 19-20:503–581, 1994.

[14] P. Jonsson and C. Bäckström. A linear programming approach to temporal reasoning. In *Proceedings of AAAI-96*, pages 1235–1240. AAAI Press/MIT Press, 1996.

[15] P. Jonsson and C. Bäckström. A unifying approach to temporal constraint reasoning. *Artificial Intelligence*, 102:143–155, 1998.

[16] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[17] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[18] M. Koubarakis. Dense Time and Temporal Constraints with $\neq$. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 24–35. Morgan Kaufmann, San Mateo, CA, October 1992.

[19] M. Koubarakis. Complexity Results for First-Order Theories of Temporal Constraints. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, pages 379–390. Morgan Kaufmann, San Francisco, CA, May 1994.

[20] M. Koubarakis. Database Models for Infinite and Indefinite Temporal Information. *Information Systems*, 19(2):141–173, March 1994.

[21] M. Koubarakis. Foundations of Indefinite Constraint Databases. In A. Borning, editor, *Proceedings of the 2nd International Workshop on the Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 266–280. Springer Verlag, 1994.

[22] M. Koubarakis. From Local to Global Consistency in Temporal Constraint Networks. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, volume 976 of *LNCS*, pages 53–69, Cassis, France, September 1995.

[23] M. Koubarakis. From Local to Global Consistency in Temporal Constraint Networks. *Theoretical Computer Science*, 173:89–112, February 1997. Invited submission to the special issue dedicated to the 1st International Conference on Principles and Practice of Constraint Programming (CP95), Editors: U. Montanari and F. Rossi.

[24] M. Koubarakis. The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases. *Theoretical Computer Science*, 171:25–60, January 1997. Special Issue on Uncertainty in Databases and Deductive Systems, Editor: L.V.S. Lakshmanan.

[25] M. Koubarakis and S. Skiadopoulos. Querying Temporal Constraint Networks in PTIME. In *Proceedings of AAAI-99*, pages 745–750, 1999.

[26] M. Koubarakis and S. Skiadopoulos. Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatio-Temporal Information. In *Spatio-Temporal Database Management (Proceedings of the International Workshop STDBM'99)*, volume 1678 of *LNCS*, pages 204–223. Springer, 1999.

[27] P. Ladkin and R. Maddux. On binary constraint problems. *Journal of the ACM*, 41(3):435–469, May 1994.

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

[29] Jean-Louis Lassez and Ken McAloon. A Canonical Form for Generalized Linear Costraints. In *TAPSOFT '89, Advanced Seminar on Foundations of Innovative Software Development, Lecture Notes in Computer Science 351*, pages 19–27. Springer Verlag, 1989.

[30] Jean-Louis Lassez and Ken McAloon. A Canonical Form for Generalized Linear Constraints. *Journal of Symbolic Computation*, 13(1):1–24, 1992.

[31] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, January 1995.

[32] Bernhard Nebel and Hans-Jürgen Bürckert. C programs used for the analysis of the ORD-Horn class. Available electronically from ftp://ftp.informatik.uni-freiburg.de/~nebel/journals.html.

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

[34] E. Sontag. Real Addition and the Polynomial Time Hierarchy. *Information Processing Letters*, 20:115–120, 1985.

[35] Peter van Beek. Reasoning About Qualitative Temporal Information. *Artificial Intelligence*, 58:297–326, 1992.

[36] Peter van Beek and Robin Cohen. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence*, 6:132–144, 1990.

[37] 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.