Building Tractable Disjunctive Constraints

David Cohen,
Royal Holloway, University of London, UK,
Peter Jeavons
University of Oxford, UK,
Peter Jonsson
Linköpings Universitet, Sweden
and
Manolis Koubarakis
Technical University of Crete

Many combinatorial search problems can be expressed as 'constraint satisfaction problems'. This class of problems is known to be NP-hard in general, but a number of restricted constraint classes have been identified which ensure tractability. This paper presents the first general results on combining tractable constraint classes to obtain larger, more general, tractable classes. We give examples to show that many known examples of tractable constraint classes, from a wide variety of different contexts, can be constructed from simpler tractable classes using a general method. We also construct several new tractable classes that have not previously been identified.

Categories and Subject Descriptors: F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.2.1 [**DISCRETE MATHEMATICS**]: Combinatorics—*Combinatorial algorithms*; I.2.4 [**ARTIFICIAL INTELLIGENCE**]: Knowledge Representation Formalisms and Methods—*relation systems*

General Terms: Algorithms, Theory

An earlier version of some parts of this paper was presented at the International Conference on Constraint Programming in 1997. (See reference [Cohen et al. 1997].)
Name: David Cohen
Address: Dept. of Computer Science, Royal Holloway, University of London, Egham Hill, Egham, Surrey TW20 0EX, UK
Name: Peter Jeavons
Address: Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
Name: Peter Jonsson
Address: Department of Computer and Information Science, Linköpings Universitet, S-581 83 Linköping, SWEDEN
Name: Manolis Koubarakis
Address: Department of Electronic and Computer Engineering, Technical University of Crete, University Campus - Kounoupidiana, 73100 Chania, Crete, GREECE

Additional Key Words and Phrases: constraint satisfaction problem, disjunctive constraints, complexity, independence, NP-completeness, relations

1. INTRODUCTION

Many combinatorial search problems can be expressed as 'constraint satisfaction problems' [Montanari 1974; Mackworth 1977], in which the aim is to find an assignment of values to a given set of variables subject to specified constraints. For example, the standard propositional satisfiability problem [Garey and Johnson 1979] may be viewed as a constraint satisfaction problem where the variables must be assigned Boolean values, and the constraints are specified by clauses.

The general constraint satisfaction problem is known to be NP-hard [Montanari 1974; Mackworth 1977]. However, by imposing restrictions on the constraint interconnections [Dechter and Pearl 1989; Freuder 1985; Gyssens et al. 1994; Montanari 1974], or on the form of the constraints [Cooper et al. 1994; Jeavons et al. 1997; Jeavons and Cooper 1995; Kirousis 1993; Montanari 1974; van Beek and Dechter 1995; van Hentenryck et al. 1992], it is possible to obtain restricted versions of the problem that are tractable.

Now that a number of tractable constraint types have been identified, it is of considerable interest to investigate how these constraint types can be *combined*, to yield more general problem classes that are still tractable. This paper presents the first general results of this kind: we identify conditions under which different tractable constraint classes can be combined, in order to construct larger, more general, tractable constraint classes.

We focus specifically on 'disjunctive constraints', that is, constraints which have the form of the disjunction of two constraints of specified types. We show that whenever we are given tractable constraint types with certain properties, then the class of problems involving all possible disjunctions of constraints of these types is also tractable. This allows new tractable constraint classes to be constructed from simpler tractable classes, and so extends the range of known tractable constraint classes.

We give examples to show that many known examples of tractable disjunctive constraints over both finite and infinite domains can be constructed from simpler classes using these results. In particular, we demonstrate that five out of the six tractable classes of Boolean constraints identified by Schaefer in [Schaefer 1978] can be obtained in this way (see Examples 5, 7 and 9). These include the standard Horn clauses and Krom clauses of propositional logic. Furthermore, we show that similar results hold for the 'max-closed' constraints first identified in [Jeavons and Cooper 1995], the 'connected row-convex' constraints first identified in [Deville et al. 1997] (see also [Jeavons et al. 1998]), the ORD-Horn constraints over temporal intervals described in [Nebel and Burckert 1995], the disjunctive linear constraints over the real numbers described in [Jonsson and Bäckström 1998; Koubarakis 1996], the 'extended Horn clauses' described in [Chandru and Hooker 1991], and the tractable set constraints described in [Drakengren 1997; Drakengren and Jonsson 1998; Drakengren and Jonsson 1997]. In all of these cases our results lead to simplifications of earlier proofs, and in many cases we are able to generalise the earlier results to obtain larger families of tractable constraint classes. We also describe some new tractable classes of constraints that can be derived from the same results.

The paper is organised as follows. In Section 2 we give the basic definitions for the constraint satisfaction problem, and define the notion of a tractable set of constraints. In Section 3 we describe how sets of constraints can be combined to form disjunctive constraints, and identify a number of different conditions that are sufficient to ensure tractability of these disjunctive constraints. In Section 4 we give examples to illustrate how these results can be used to establish the tractability of a wide variety of tractable constraint classes.

2. THE CONSTRAINT SATISFACTION PROBLEM

The 'constraint satisfaction problem' was introduced by Montanari in 1974 [Montanari 1974] and has been widely studied.

Definition 1. An instance of a *constraint satisfaction problem* consists of:

—A finite set of *variables*, V;

—A set of *values*, D, (which may be finite or infinite);

—A finite set of constraints $C = \{c_1, c_2, \ldots, c_q\}$.
Each constraint c_i is a pair (S_i, R_i), where $S_i \subseteq V$ is a set of variables, called the constraint *scope*, and R_i is a set of (total) functions from S_i to D, called the constraint *relation*.

The elements of a constraint relation indicate the allowed combinations of simultaneous values for the variables in the constraint scope. The number of variables in the scope of a constraint will be called the 'arity' of the constraint. In particular, unary constraints specify the allowed values for a single variable, and binary constraints specify the allowed combinations of values for a pair of variables. There is a unique 'empty constraint', $(\emptyset, \emptyset)$, for which the scope and the constraint relation are both empty.

Note that we are representing constraint relations as sets of *functions* rather than the usual representation as sets of *tuples*. These two representations are clearly equivalent, since by fixing an ordering for the variables in the scope of a constraint, we can associate each function with a corresponding tuple of values. However, the use of the functional representation simplifies some of the definitions below.

Example 1. When the set of values D is the set of real numbers $\mathbb{R}$, then a relation on some set of variables, say $\{u, v, w\}$, is a set of total functions from $\{u, v, w\}$ to $\mathbb{R}$. For example, the following is a typical relation

$$\{f : \{u, v, w\} \to \mathbb{R} \mid 3u + 2v + w = 0\}$$

If we fix any ordering on the variables, say (w, u, v), then the same relation can be represented as a set of 3-tuples of real numbers:

$$\{(a, b, c) \in \mathbb{R}^3 \mid a + 3b + 2c = 0\}$$

A *solution* to a constraint satisfaction problem instance is a function, f, from the set of variables of that instance to the set of values, such that for each constraint (S_i, R_i) in C, the restriction of f to S_i, denoted $f|_{S_i}$, is an element of R_i.

In order to simplify the presentation we shall make a number of simplifying assumptions throughout the paper about the way in which constraint satisfaction problem instances are specified. First, we shall assume that we have a fixed countable universe of possible variable names, $U = \{x_1, x_2, \ldots\}$, and that every variable occurring in a problem instance is a member of this set. Furthermore, we shall assume that every variable in a problem instance occurs in the scope of some constraint. This means that for any problem instance, the set of variables, V, does not need to be specified explicitly, but is given by the union of the constraint scopes of that instance. Finally, we shall assume that the set of values, D, for any instance does not need to be specified explicitly, but will be understood from the context. With these assumptions, a constraint satisfaction problem instance can be specified simply by specifying the corresponding set of constraints. Hence, we shall talk about 'solutions to a set of constraints'. The set of all solutions to the set of constraints C will be denoted $\mathrm{Sol}(C)$.

In order to determine the computational complexity of a constraint satisfaction problem we need to specify how the constraints are encoded in finite strings of symbols. We shall assume in all cases that this representation is chosen so that the complexity of determining whether a constraint allows a given assignment of values to the variables in its scope is bounded by a polynomial function of the length of the representation.

Example 2. Constraint relations may be finite or infinite. A constraint with a finite constraint relation can be represented simply by giving an explicit list of all the elements in that relation, but constraints with infinite relations clearly cannot. In both cases it is possible to use a suitable specification language, such as logical formulas, or linear equations. For example, when the set of possible values for the variables is $\{\text{TRUE}, \text{FALSE}\}$, representing the Boolean values 'true' and 'false', then the logical formula '$x_1 \vee x_2 \vee \neg x_3$' can be used to specify the constraint with scope $\{x_1, x_2, x_3\}$ and relation

$$\{f : \{x_1, x_2, x_3\} \to \{\text{TRUE}, \text{FALSE}\} \mid f(x_1) \vee f(x_2) \vee \neg f(x_3) \equiv \text{TRUE}\}.$$

Similarly, when the set of possible values is the real numbers, $\mathbb{R}$, then the equation '$x_1 + 2x_2 = 7$' can be used to specify the constraint with scope $\{x_1, x_2\}$ and relation

$$\{f : \{x_1, x_2\} \to \mathbb{R} \mid f(x_1) + 2f(x_2) = 7\}.$$

Deciding whether or not a given set of constraints has a solution is known to be NP-hard in general [Montanari 1974; Mackworth 1977]. In this paper we shall consider how restricting the allowed constraints affects the complexity of this decision problem. We therefore make the following definition.

Definition 2. For any set of constraints, Γ, $\mathrm{CSP}(\Gamma)$ is defined to be the decision problem with

Instance:. A finite set of constraints $C \subseteq \Gamma$.

Question:. Does C have a solution?

If there is some algorithm which solves every instance in $\mathrm{CSP}(\Gamma)$ in polynomial time, then we shall say that $\mathrm{CSP}(\Gamma)$ is 'tractable', and refer to Γ as a tractable set of constraints.

Example 3. For any set of possible values D, and any pair of variables, x and y, the binary disequality constraint with scope $\{x, y\}$ and set of values D is defined as follows:

$$\neq_D(x,y) \text{ denotes the constraint } (\{x,y\},\ \{f : \{x,y\} \to D \mid f(x) \neq f(y)\}).$$

Since we are assuming that we have a fixed universe of possible variable names, we can consider the set of all possible binary disequality constraints over D, for all possible choices of a pair of variables. This set will be denoted $\Gamma_{\neq_D}$.

For any finite set D, the decision problem CSP($\Gamma_{\neq_D}$) corresponds precisely to the GRAPH $|D|$-COLOURABILITY problem [Garey and Johnson 1979]. This problem is well-known to be tractable when $|D| \leq 2$ and NP-complete when $|D| \geq 3$.

3. TRACTABLE DISJUNCTIVE CONSTRAINTS

The remainder of the paper focuses on the complexity of constraint satisfaction problems involving *disjunctive* constraints.

We first define how individual constraints can be combined disjunctively.

Definition 3. Let $c_1 = (S_1, R_1)$ and $c_2 = (S_2, R_2)$ be two constraints with a common set of possible values D.

The disjunction of c_1 and c_2, denoted $c_1 \vee c_2$, is defined as follows:

$$c_1 \vee c_2 = (S_1 \cup S_2, \{f : (S_1 \cup S_2) \to D \mid (f|_{S_1} \in R_1) \vee (f|_{S_2} \in R_2)\})$$

The idea behind this definition is that an assignment satisfies the disjunction of two constraints if it satisfies either one of them. Note that for any constraint c, the disjunction of c and the empty constraint, $(\emptyset, \emptyset)$, gives c.

Now we define how a set of disjunctive constraints can be obtained from two arbitrary sets of constraints over the same set of possible values.

Definition 4. For any two sets of constraints Γ and Δ, with a common set of possible values, define the set of constraints $\Gamma \overset{\times}{\vee} \Delta$ as follows:

$$\Gamma \overset{\times}{\vee} \Delta = \{c_1 \vee c_2 \mid c_1 \in \Gamma, c_2 \in \Delta\}$$

The set of constraints $\Gamma \overset{\times}{\vee} \Delta$ (read as 'Γ *or-cross* Δ') contains the disjunction of each possible pair of constraints from Γ and Δ.

In many cases of interest Γ and Δ will both contain the empty constraint, $(\emptyset, \emptyset)$, and in these cases $\Gamma \overset{\times}{\vee} \Delta \supseteq \Gamma \cup \Delta$. In most cases of this kind $\Gamma \overset{\times}{\vee} \Delta$ will be much larger than $\Gamma \cup \Delta$, and will therefore allow a much richer class of constraint satisfaction problems to be expressed.

The next example shows that when tractable sets of constraints are combined using the disjunction operation defined in Definition 4 the resulting set of disjunctive constraints may or may not be tractable.

Example 4. Let Λ be the set containing all Boolean constraints which can be specified by a formula of propositional logic consisting of a single literal (where a literal is either a variable or a negated variable).

The set of constraints Λ is clearly tractable, as it is straightforward to verify in linear time whether a collection of simultaneous literals has a solution.

Now consider the set of constraints $\Lambda^{\vee 2} = \Lambda \overset{\times}{\vee} \Lambda$. This set contains all Boolean constraints specified by a disjunction of 2 literals. The problem $\mathrm{CSP}(\Lambda^{\vee 2})$ corresponds to the 2-Satisfiability problem, which is well-known to be tractable [Garey and Johnson 1979].

Finally, consider the set of constraints $\Lambda^{\vee 3} = \Lambda^{\vee 2} \overset{\times}{\vee} \Lambda$. This set contains all Boolean constraints specified by a disjunction of 3 literals. The problem $\mathrm{CSP}(\Lambda^{\vee 3})$ corresponds to the 3-Satisfiability problem, which is well-known to be NP-complete [Garey and Johnson 1979].

In many of the examples below we shall be concerned with constraints that are specified by disjunctions of an arbitrary number of constraints from a given set. To provide a uniform notation for such constraints we make the following definition.

Definition 5. For any set of constraints, Δ, define the set Δ^* as follows:

$$\Delta^* = \bigcup_{i=0}^{\infty} \Delta^{\vee i}$$

where

$$\begin{aligned} \Delta^{\vee 0} &= \{(\emptyset, \emptyset)\} \\ \Delta^{\vee (i+1)} &= \Delta^{\vee i} \overset{\times}{\vee} \Delta \quad \text{for} \quad i = 0, 1, 2, \ldots \end{aligned}$$

The final piece of machinery that we shall need to deal with disjunctive sets of constraints is a uniform way to recover the separate components in the disjunction.

Definition 6. For any disjunctive set of constraints $\Gamma \overset{\times}{\vee} \Delta$, we define two operations $\Pi_1 : \Gamma \overset{\times}{\vee} \Delta \to \Gamma$ and $\Pi_2 : \Gamma \overset{\times}{\vee} \Delta \to \Delta$ such that for any $c \in \Gamma \overset{\times}{\vee} \Delta$, $c = \Pi_1(c) \vee \Pi_2(c)$.

We shall assume that the constraints in $\Gamma \overset{\times}{\vee} \Delta$ are represented in such a way that Π_1 and Π_2 can be computed in linear time.

In the following sections we identify certain conditions on sets of constraints Γ and Δ which are sufficient to ensure that $\Gamma \overset{\times}{\vee} \Delta$ is tractable.

3.1 The guaranteed satisfaction property

The first condition we identify is rather trivial, but it is included here for completeness, and because it is sufficient to show how two of the six tractable classes of Boolean constraints identified by Schaefer in [Schaefer 1978] can be constructed from simpler classes.

Definition 7. A set of constraints, Γ, has the *guaranteed satisfaction property* if every finite $C \subseteq \Gamma$ has a solution.

Theorem 1. *For any sets of constraints* Γ *and* Δ*, if* Γ *has the guaranteed satisfaction property, then* $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta)$ *also has the guaranteed satisfaction property, and is therefore tractable.*

Proof. Every constraint in $\Gamma \overset{\times}{\vee} \Delta$ is of the form $c \vee d$ for some $c \in \Gamma$ and some $d \in \Delta$. Hence if Γ has the guaranteed satisfaction property, then any problem instance in $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta)$ has a solution satisfying the first disjunct of each constraint. $\square$

Example 5. Recall the set of unary Boolean constraints, Λ, defined in Example 4, which contains all constraints specified by a single literal.

Let Γ be the subset of Λ containing only the constraints specified by a single *negative* literal.

It is clear that Γ has the guaranteed satisfaction property, since any problem instance in $\mathrm{CSP}(\Gamma)$ has the solution which assigns the value FALSE to all variables.

Hence, by Theorem 1, $\Gamma \overset{\times}{\vee} \Lambda^*$ has the guaranteed satisfaction property and $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Lambda^*)$ is tractable. This tractable set contains all constraints specified by Boolean clauses containing at least one negative literal. Examples of such clauses include the following:

$$\begin{gathered} x_1 \vee x_2 \vee \neg x_3 \\ \neg x_2 \\ \neg x_1 \vee \neg x_2 \vee \neg x_4 \vee \neg x_5 \end{gathered}$$

The constraint relations defined by conjunctions of clauses of this form are precisely the elements of the first class of tractable Boolean relations identified by Schaefer in [Schaefer 1978] (which he calls '0-valid' relations).

A symmetric argument shows that if Γ contains only the constraints specified by a single positive literal, then $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta^*)$ is again tractable. This tractable set contains all constraints specified by Boolean clauses containing at least one positive literal. The constraint relations defined by conjunctions of clauses of this form are precisely the elements of the second class of tractable Boolean relations identified by Schaefer in [Schaefer 1978] (which he calls '1-valid' relations).

Example 6. Let Γ be the set of all constraints specified by a linear disequality over the real numbers, that is, an expression of the form $\sum a_i x_i \neq b$, where the a_i and b are (real-valued) constants.

It is clear that Γ has the guaranteed satisfaction property, since any problem instance in $\mathrm{CSP}(\Gamma)$ only rules out a finite number of hyperplanes from $\mathbb{R}^n$.

Hence, by Theorem 1, for any set of constraints, Δ, over the real numbers, $\Gamma \overset{\times}{\vee} \Delta^*$ has the guaranteed satisfaction property and $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta^*)$ is tractable.

For example, let Δ be the set containing all the constraints in Γ together with all constraints specified by a single (weak) linear inequality, that is, an expression of the form $\sum a_i x_i \leq b$, where the a_i and b are (real-valued) constants. In this case $\Gamma \overset{\times}{\vee} \Delta^*$ contains constraints such as the following:

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

This example should be compared with the similar, but much more significant, tractable class defined in Example 13 below.

3.2 The independence property

In this section we identify a rather more subtle condition which can be used to construct tractable disjunctive constraints. We first need the following definition:

Definition 8. For any sets of constraints Γ and Δ, define $\mathrm{CSP}_{\Delta \leq k}(\Gamma \cup \Delta)$ to be the subproblem of $\mathrm{CSP}(\Gamma \cup \Delta)$ consisting of all instances containing at most k constraints which are members of Δ.

Using this definition, we now define what it means for one set of constraints to be 'k-independent' with respect to another.

Definition 9. For any sets of constraints Γ and Δ, we say that Δ is *k-independent with respect to* Γ if the following condition holds: any set of constraints C in $\mathrm{CSP}(\Gamma \cup \Delta)$ has a solution provided every subset of C belonging to $\mathrm{CSP}_{\Delta \leq k}(\Gamma \cup \Delta)$ has a solution.

The intuitive meaning of this definition is that the satisfiability of any set of constraints chosen from the set Δ can be determined by considering those constraints k at a time, even in the presence of arbitrary additional constraints from Γ. In the examples below we shall demonstrate that several important constraint types have the 1-independence property.

A more restricted notion of 1-independence has been widely studied in the literature of constraint programming, where it has been called simply 'independence' (see [Lassez and McAloon 1989; Lassez and McAloon 1992; Lassez and McAloon 1991], for example). The earlier property applies to an individual constraint class containing *positive* constraints and *negative* constraints, and has been used in the development of consistency checking algorithms and canonical forms [Lassez and McAloon 1991; Lassez and McAloon 1992]. However, we will show below that the more general notion of 1-independence of one class with respect to another, introduced here, can be used to prove the tractability of a wide variety of *disjunctive* constraint classes for which the earlier notion of independence does not hold.

Consider the algorithm shown in Figure 1 for a function **Ind-Solvable** which determines whether or not a finite set of constraints $C \subseteq \Gamma \overset{\times}{\vee} \Delta$ has a solution.

Ind-Solvable(C: FINITE SUBSET OF $\Gamma \overset{\times}{\vee} \Delta$)
  $S := \emptyset$
  REPEAT
    $X := \{c \in C \mid \mathrm{Sol}(S \cup \Pi_2(c)) = \emptyset\}$
    IF $X = \emptyset$ THEN
      RETURN TRUE
    ELSE
      $C := C \setminus X$
      $S := S \cup \{\Pi_1(x) \mid x \in X\}$
    ENDIF
  UNTIL $\mathrm{Sol}(S) = \emptyset$
  RETURN FALSE

Fig. 1. An algorithm for the function **Ind-Solvable**

The next result shows that **Ind-Solvable** correctly determines whether or not a finite set of constraints chosen from $\Gamma \overset{\times}{\vee} \Delta$ has a solution in all cases where Δ is 1-independent with respect to Γ.

LEMMA 1. *If C is a finite subset of $\Gamma \overset{\times}{\vee} \Delta$, and Δ is 1-independent with respect to Γ, then the function* **Ind-Solvable** *defined in Figure 1 correctly determines whether or not C has a solution.*

PROOF. The algorithm shown in Figure 1 clearly terminates, because C is finite.

Assume that C is a finite subset of $\Gamma \overset{\times}{\vee} \Delta$ for some Γ and Δ such that Δ is 1-independent with respect to Γ. We first prove by induction that after every assignment to S, all of the constraints in S must be satisfied in order to satisfy the original set of constraints C.

This is vacuously true after the first assignment to S, because S is then equal to the empty set.

At each subsequent assignment, S is augmented with the constraints obtained by applying Π_1 to the constraints in X. Now, the elements of X are constraints c of C such that $\Pi_2(c)$ is incompatible with S. Hence the only way such a c can be satisfied together with S is to satisfy the other disjunct of c, that is, the constraint given by $\Pi_1(c)$. Hence, by the inductive hypothesis, each constraint added to S must be satisfied in order to satisfy the original set of constraints C, and the result follows, by induction.

This result establishes that when **Ind-Solvable**(C) returns FALSE, C has no solutions.

Conversely, when **Ind-Solvable**(C) returns TRUE, then we know that X is empty. This implies that for each constraint c in C either $\Pi_1(c)$ belongs to S, or else $\Pi_2(c)$ is compatible with the constraints in S. Now, using the fact that Δ is 1-independent with respect to Γ, we conclude that $S \cup \{\Pi_2(c) \mid c \in C,\ \Pi_1(c) \notin S\}$ has a solution, and hence C has a solution. $\square$

By analysing the complexity of the algorithm in Figure 1 we now establish the following result:

THEOREM 2. *For any two sets of constraints* Γ *and* Δ, *if* $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ *is tractable, and* Δ *is 1-independent with respect to* Γ, *then* $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta)$ *is tractable.*

PROOF. By Lemma 1, it is sufficient to show that the algorithm in Figure 1 runs in polynomial time. We can bound the time complexity of this algorithm as follows.

First note that $|S|$ increases on each iteration of the repeat loop, but $|S|$ is bounded by $|C|$ since each constraint in S arises from an element of C. Hence there can be at most $|C|$ iterations of this loop.

Now let $l(C)$ be the length of the string specifying C. During each iteration of the loop the algorithm determines whether or not there is a solution to $S \cup \Pi_2(c)$ for each c remaining in C. Since this set of constraints is a member of $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$, which is assumed to be tractable, these calculations can each be carried out in polynomial time in the size of their input. Note also that the length of this input is less than or equal to $l(C)$. Hence, the time complexity of each of these calculations is bounded by $p(l(C))$, for some polynomial p.

At the end of each iteration of the loop the algorithm determines whether or not there is a solution to S. Since S is also an element of $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$, and the length of the specification of S is less than or equal to $l(C)$, this calculation can also be carried out in at most $p(l(C))$ time.

Hence, the total time required to complete the algorithm is $O(|C|(|C|+1)p(l(C)))$, which is polynomial in the size of the input. $\square$

Finally, we show that this result can be extended to arbitrary disjunctions of constraints in Δ.

LEMMA 2. *For any set of constraints Δ, if Δ is 1-independent with respect to Γ, then Δ^* is also 1-independent with respect to Γ.*

PROOF. Assume that Δ is 1-independent with respect to Γ and let C be an arbitrary finite subset of $\Gamma \cup \Delta^*$. We need to show that if every subset of C which belongs to $\mathrm{CSP}_{\Delta^* \leq 1}(\Gamma \cup \Delta^*)$ has a solution, then so does C.

Let C' be a maximal subset of C belonging to $\mathrm{CSP}_{\Delta^* \leq 1}(\Gamma \cup \Delta^*)$ and let s be a solution to C'. Since C' is maximal, it contains the set C_Γ, consisting of all the constraints in C which are elements of $\Gamma \setminus \Delta^*$, so s is a solution to C_Γ. If C' also contains a constraint $d \in \Delta^*$, then there must be at least one constraint d' in Δ such that s is a solution to d', by the definition of Δ^*. Hence, we can replace d with a (possibly) more restrictive constraint $d' \in \Delta$, without losing the solution s.

If we carry out this replacement for each C', then we have a set of constraints in $\mathrm{CSP}(\Gamma \cup \Delta)$ such that each subset belonging to $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ has a solution. Now, by the fact that Δ is 1-independent with respect to Γ, it follows that this modified set of constraints has a solution, and hence the original set of constraints C has a solution, which gives the result. □

COROLLARY 1. *For any two sets of constraints Γ and Δ, if $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ is tractable, and Δ is 1-independent with respect to Γ, then $\mathrm{CSP}(\Gamma \between \Delta^*)$ is tractable.*

PROOF. If $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ is tractable, then $\mathrm{CSP}_{\Delta^* \leq 1}(\Gamma \cup \Delta^*)$ is tractable, because each instance of $\mathrm{CSP}_{\Delta^* \leq 1}(\Gamma \cup \Delta^*)$ only contains at most one disjunctive constraint belonging to Δ^*, and each disjunct of this constraint can be considered separately in polynomial time.

Furthermore, if Δ is 1-independent with respect to Γ, then by Lemma 2 Δ^* is also 1-independent with respect to Γ.

Hence, Theorem 2 can be applied to Γ and Δ^*, giving the result. □

Example 7. Recall the set of unary Boolean constraints, Λ, defined in Example 4, which contains all constraints specified by a single literal.

Let Γ be the subset of Λ containing only the constraints specified by a single *positive* literal, and let $\Gamma' = \Gamma \cup \{(\emptyset, \emptyset)\}$. Let Δ be the subset of Λ containing only the constraints specified by a single *negative* literal. Note that the set of constraints Δ^*, constructed according to Definition 5, is equal to the set of constraints specified by arbitrary finite disjunctions of negative literals (including the empty disjunction).

Now it is easily shown that Δ is 1-independent with respect to Γ' (since any collection of positive and negative literals has a solution if and only if all subsets containing at most one negative literal have a solution). Also, $\mathrm{CSP}_{\Delta \leq 1}(\Gamma' \cup \Delta)$ is tractable, since each instance is specified by a conjunction of zero or more positive literals together with at most one negative literal. Hence, by Corollary 1, we conclude that $\Gamma' \between \Delta^*$ is tractable. But $\Gamma' \between \Delta^*$ is the set of constraints specified by a disjunction of literals containing at most one positive literal. Examples of such clauses include the following:

$$x_1 \vee \neg x_2 \vee \neg x_3$$
$$x_2$$
$$\neg x_1 \vee \neg x_2 \vee \neg x_4 \vee \neg x_5$$

It is easy to see that $\mathrm{CSP}(\Gamma' \between \Delta^*)$ corresponds exactly to the HORN-CLAUSE SATISFIABILITY problem [Garey and Johnson 1979]. The constraint relations defined by conjunctions of clauses of this form are precisely the elements of the third class of tractable Boolean relations identified by Schaefer in [Schaefer 1978] (which he calls 'weakly negative' relations).

By a symmetric argument, it follows that the set of constraints specified by a disjunction of literals containing at most one negative literal is also a tractable set of constraints. The constraint relations defined by conjunctions of clauses of this form are precisely the elements of the fourth class of tractable Boolean relations identified by Schaefer in [Schaefer 1978] (which he calls 'weakly positive' relations).

We have shown that 1-independence can be used to establish tractability, and further examples are given in Section 4 below. To conclude this section we show that, if a set of constraints Δ is k-independent with respect to a set Γ, for some value of k larger than one, then this may *not* be sufficient to ensure tractability of $\Gamma \overset{\times}{\vee} \Delta$.

Example 8. In this example we construct two sets of constraints, Γ and Δ, such that $\mathrm{CSP}_{\Delta\leq 2}(\Gamma \cup \Delta)$ is tractable, and Δ is 2-independent with respect to Γ, but $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta)$ is NP-complete.

Recall the set of unary Boolean constraints, Λ, defined in Example 4, which contains all constraints specified by a single literal. Let $\Gamma = \Lambda \overset{\times}{\vee} \Lambda$, and $\Delta = \Lambda$.

With these definitions, $\mathrm{CSP}(\Gamma \cup \Delta)$ is equivalent to the standard 2-SATISFIABILITY problem, and hence $\mathrm{CSP}_{\Delta\leq 2}(\Gamma \cup \Delta)$ is tractable. Furthermore, the 2-SATISFIABILITY problem has the property that 'path-consistency' guarantees global consistency [Jeavons et al. 1998], which implies that any minimal insoluble subset of clauses in an instance of 2-SATISIFIABILITY contains at most two single literals, and hence Δ is 2-independent with respect to Γ,

However, as discussed in Example 4, $\mathrm{CSP}(\Gamma \overset{\times}{\vee} \Delta)$ corresponds to the 3-SATISFIABILITY problem, and is therefore NP-complete.

3.3 The Krom property

In this section we identify a final sufficient condition for constructing tractable disjunctive constraints.

Definition 10. A set of constraints, Δ, has the *Krom property* if it is 2-independent with respect to the empty set.

Note that Δ has the Krom property if and only if for every finite $C \subseteq \Delta$ having no solution, there exists a pair of (not necessarily distinct) constraints $c_i, c_j \in C$ such that $\{c_i, c_j\}$ has no solution.

The name "Krom property" is chosen to emphasise the close connection with Krom clauses (*i.e.*, Boolean clauses of length ≤ 2) [Denenberg and Lewis 1984], which will be demonstrated later.

Consider the algorithm shown in Figure 2, for a function **Krom-Solvable**, which determines whether or not a finite set of constraints $C \subseteq \Delta \overset{\times}{\vee} \Delta$ has a solution.

The next result shows that **Krom-Solvable** correctly determines whether or not a finite set of constraints chosen from $\Delta \overset{\times}{\vee} \Delta$ has a solution in all cases when Δ has the Krom property.

LEMMA 3. *If C is a finite subset of $\Delta \overset{\times}{\vee} \Delta$, and Δ has the Krom property, then the function* **Krom-Solvable** *defined in Figure 2 correctly determines whether or not C has a solution.*

Krom-Solvable(C: FINITE SUBSET OF $\Delta \overset{\times}{\vee} \Delta$)
 $P := \{\Pi_1(c) \mid c \in C\} \cup \{\Pi_2(c) \mid c \in C\}$
 DEFINE A SET OF BOOLEAN VARIABLES $\{q_c \mid c \in P\}$
 $A := \{(\neg q_{c'} \vee \neg q_{c''}) \mid c', c'' \in P$ AND $\mathrm{Sol}(\{c', c''\}) = \emptyset\}$
 $B := \{(q_{c'} \vee q_{c''}) \mid \exists c \in C, c' = \Pi_1(c), c'' = \Pi_2(c)\}$
 IF $A \cup B$ IS SATISFIABLE
  THEN RETURN TRUE
  ELSE RETURN FALSE

Fig. 2. An algorithm for a function **Krom-Solvable**

PROOF. We show that the function **Krom-Solvable** defined in Figure 2 returns TRUE when applied to C if and only if C has a solution.

only-if:. Assume that **Krom-Solvable** returns TRUE. This implies that there exists a satisfying truth assignment, μ, for $A \cup B$. Define the set of constraints $C' \subseteq \Delta$ as follows

$$C' = \{c \mid \mu(q_c) = \text{TRUE}\}.$$

We first show that C' has a solution. Since Δ has the Krom property, C' has no solution only if there exist $c', c'' \in C'$ such that $\{c', c''\}$ has no solution. However, this cannot happen because μ satisfies the formulae in the set A.

Now, let the function f be a solution of C'. For each constraint $c \in C$ we know that at least one of $\Pi_1(c)$ and $\Pi_2(c)$ is a member of C', because μ satisfies the formulae in B. Since f is a solution of C', it follows that f can be extended (if necessary) to a solution of C (by assigning an arbitrary value to any variable not constrained by C').

if:. Assume that C has a solution and let f be any such solution. Define the truth assignment $\mu : \{q_c \mid c \in P\} \to \{\text{TRUE}, \text{FALSE}\}$ as follows:

$$\mu(q_c) = \text{TRUE} \text{ iff } c \text{ is satisfied by } f.$$

We show that μ is a satisfying truth assignment of $A \cup B$ by considering the elements of A and B in turn.

(1) For each formula $(\neg q_{c'} \vee \neg q_{c''}) \in A$, we know that $\{c', c''\}$ has no solutions. Hence it cannot be the case that $\mu(q_{c'}) = \mu(q_{c''}) = \text{TRUE}$, which means that $(\neg q_c \vee \neg q_{c'})$ is satisfied by μ.
(2) For each formula $(q_{c'} \vee q_{c''}) \in B$ we know that there is a constraint $c \in C$ such that $\Pi_1(c) = c'$ and $\Pi_2(c) = c''$. Since f is a solution to C, f satisfies either c, c', or both. Hence, μ assigns TRUE to at least one of $q_{c'}, q_{c''}$, which means that $(q_{c'} \vee q_{c''})$ is satisfied by μ.

It follows that $A \cup B$ is satisfiable, so **Krom-Solvable** returns TRUE.

□

By analysing the complexity of the algorithm in Figure 2 we now establish the following result:

THEOREM 3. *For any set of constraints Δ, if* $\mathrm{CSP}(\Delta)$ *is tractable, and Δ has the Krom property, then* $\mathrm{CSP}(\Delta \overset{\times}{\vee} \Delta)$ *is tractable.*

Proof. By Lemma 3, it is sufficient to show that the algorithm in Figure 2 runs in polynomial time. We can bound the time complexity of this algorithm as follows.

Let C be a finite subset of $\Delta \stackrel{\times}{\vee} \Delta$ and let $l(C)$ be the length of the string specifying C.

Since computing Π_1 and Π_2 takes linear time, the set P can be computed in $O(l(C))$ time. It contains at most $2|C|$ elements.

To compute the set A, the algorithm must determine whether or not there is a solution to $\{c', c''\}$ for every pair of constraints $c', c'' \in P$. Since $\{c', c''\}$ is an instance of CSP(Δ), which is assumed to be tractable, these calculations can each be carried out in polynomial time in the size of their input. Hence the time required to compute the set A is $O(|C|^2 \cdot p(l(C)))$, for some polynomial p. The set A contains at most $|C|(2|C| - 1)$ elements.

The set B can clearly be computed in time $O(|C|l(C))$ and contains $|C|$ elements.

Finally, the algorithm must decide the satisfiability of a set of Krom clauses containing at most $|C|(2|C|-1)+|C|$ elements. By using the linear time algorithm for this problem given in [Aspvall et al. 1979], this step can be carried out in $O(|C|^2)$ time.

Hence, the total time required by the algorithm is $O(|C|^2 \cdot p(l(C)))$, which is polynomial in the size of the input. $\square$

The proof of Theorem 3 uses the well-known result about the tractability of solving Krom clauses. The next example shows that these two results are equivalent.

Example 9. Recall the set of unary Boolean constraints, Λ, defined in Example 4, which contains all constraints specified by a single literal.

It is easily shown that Λ has the Krom property (since any collection of positive and negative literals has a solution unless it contains two different literals involving the same variable). Hence, by Theorem 3, $\Lambda \stackrel{\times}{\vee} \Lambda$ is tractable.

However, the set $\Lambda \stackrel{\times}{\vee} \Lambda$ contains all constraints specified by Boolean clauses containing at most two literals, that is, all Krom clauses.

The constraint relations defined by conjunctions of clauses of this form are precisely the elements of the fifth class of tractable Boolean relations identified by Schaefer in [Schaefer 1978] (which he calls 'bijunctive' relations).

To conclude this section we show that if a set of constraints Δ has a higher level of k-independence with respect to the empty set, then this may *not* be sufficient to ensure tractability of $\Delta \stackrel{\times}{\vee} \Delta$.

Example 10. In this example we construct a tractable set of relations Δ such that Δ is 3-independent with respect to the empty set, but we show that CSP($\Delta \stackrel{\times}{\vee} \Delta$) is NP-complete.

For all possible variables x and y, we define the unary constraints $zero(x)$, $one(x)$ and the binary constraint $\neq_{\mathbb{N}}(x, y)$ as follows:

(1) $zero(x)$ denotes the constraint $(\{x\},\ \{f : \{x\} \rightarrow \mathbb{N} \mid f(x) = 0\})$;
(2) $one(x)$ denotes the constraint $(\{x\},\ \{f : \{x\} \rightarrow \mathbb{N} \mid f(x) = 1\})$;
(3) $\neq_{\mathbb{N}}(x, y)$ denotes the constraint $(\{x, y\},\ \{f : \{x, y\} \rightarrow \mathbb{N} \mid f(x) \neq f(y)\})$.

Define Δ to be the set containing all possible constraints of these three types. (Note that Δ is well-defined since we are assuming that we have a fixed universe of possible variable names.)

Let C be a finite subset of Δ. We will show that if C has no solution, then (at least) one of the following constraint sets is a subset of C, for some variables x and y.

$$\begin{aligned}
\mathbf{S1} &= \{\mathit{zero}(x), \mathit{one}(x)\} \\
\mathbf{S2} &= \{\neq_{\mathbb{N}}(x, x)\} \\
\mathbf{S3} &= \{\mathit{zero}(x), \mathit{zero}(y), \neq_{\mathbb{N}}(x, y)\} \\
\mathbf{S4} &= \{\mathit{one}(x), \mathit{one}(y), \neq_{\mathbb{N}}(x, y)\}
\end{aligned}$$

To establish this fact, assume that none of the above stated sets of constraints are subsets of C. We will show that in this case it is always possible to construct a solution.

Let the set of variables appearing in the constraints of C be $X = \{x_2, \ldots, x_n\}$. Define the function $f : X \to \mathbb{N}$ as follows:

$$f(x_i) = \begin{cases} 0 & \text{if } \mathit{zero}(x_i) \in C; \\ 1 & \text{if } \mathit{one}(x_i) \in C \text{ and } \mathit{zero}(x_i) \notin C; \\ i & \text{otherwise.} \end{cases}$$

For each constraint $c \in C$ we can reason as follows:

—Since $\mathbf{S1} \not\subseteq C$, if c equals $\mathit{zero}(x_i)$ or $\mathit{one}(x_i)$, then f satisfies c.

—Since $\mathbf{S2} \not\subseteq C$, if c is the constraint $\neq_{\mathbb{N}}(x_i, x_j)$, then $i \neq j$.

—Since neither $\mathbf{S3}$ nor $\mathbf{S4}$ is a subset of C, if c is the constraint $\neq_{\mathbb{N}}(x_i, x_j)$ and $i \neq j$, then $f(x_i) \neq f(x_j)$.

Hence, in all cases c is satisfied by f, so f is a solution to C.

Since none of the sets **S1,S2,S3,S4** contains more than 3 elements, we have established that Δ is 3-independent with respect to the empty set, and that Δ is tractable.

To establish the NP-completeness of CSP($\Delta \overset{\times}{\vee} \Delta$), we construct a polynomial time reduction from the NP-complete problem 4-Colourability [Garey and Johnson 1979], as follows.

Let $G = \langle V, E \rangle$ be an arbitrary graph and construct an instance of CSP($\Delta \overset{\times}{\vee} \Delta$) as follows: for each $v \in V$, introduce two variables v' and v'' together with the constraints

—$\mathit{zero}(v') \vee \mathit{one}(v')$; and

—$\mathit{zero}(v'') \vee \mathit{one}(v'')$.

For each edge $(v, w) \in E$, introduce the constraint

—$\neq_{\mathbb{N}}(v', w') \vee \neq_{\mathbb{N}}(v'', w'')$.

This transformation can obviously be carried out in polynomial time and the resulting set of constraints is a subset of $\Delta \overset{\times}{\vee} \Delta$.

To see that the resulting instance has a solution if and only if the graph G is 4-colourable, identify colour 1 with $v' = 0, v'' = 0$, colour 2 with $v' = 0, v'' = 1$, and so on. For every pair of adjacent nodes v, w in G, the constraint imposed on the corresponding variables v', w', v'', w'' ensures that v and w must be assigned different colours.

4. APPLICATIONS

In this section we will use the results established above to demonstrate that many known tractable sets of constraints can be obtained by combining simpler tractable sets of constraints using the disjunction operation defined in Definition 4. We will also describe some new tractable sets of constraints which have not previously been identified.

Example 11. [**Max-closed constraints**]
The class of constraints known as 'max-closed' constraints was introduced in [Jeavons and Cooper 1995] and shown to be tractable. This class of constraints has been used in the analysis and development of a number of industrial scheduling tools [Lesaint et al. 1998; Purvis and Jeavons 1999].

Max-closed constraints are defined in [Jeavons and Cooper 1995] for arbitrary finite sets of values which are totally ordered. This class of constraints includes all of the 'basic constraints' over the natural numbers in the constraint programming language CHIP [van Hentenryck et al. 1992]. The following are examples of max-closed constraints when the set of possible values is any finite set of natural numbers:

$$\begin{gathered}
3x_1 + x_5 + 3x_4 \geq 2x_2 + 10, \\
4x_1 \neq 8, \\
x_1 \in \{1, 2, 3, 5, 7, 11, 13\}, \\
2x_1x_3x_5 \geq 3x_2 + 1, \\
(3x_1 \geq 7) \vee (2x_1 \geq 4) \vee (5x_2 \leq 7).
\end{gathered}$$

In this example we will show that the tractability of max-closed constraints is a simple consequence of Corollary 1. Furthermore, by using Corollary 1 we are able to generalise this result to obtain tractable constraints over infinite sets of values.

Max-closed constraints were originally defined in terms of an algebraic closure property on the constraint relations [Jeavons and Cooper 1995]. However, it is shown in [Jeavons and Cooper 1995] that they can also be characterised as those constraints which can be specified by a conjunction of disjunctions of inequalities of the following form:

$$(x_1 > a_1) \vee (x_2 > a_2) \vee \cdots \vee (x_r > a_r) \vee (x_j < a_j)$$

In this expression the x_i are variables and the a_i are constants.

To apply the results of Section 3.2, let the set of possible values, D, be any totally ordered set. Define Γ to be the set of all constraints specified by a single inequality of the form $x_i < a_i$, for some $a_i \in D$, together with the empty constraint. Define Δ to be the set of all constraints specified by a single inequality of the form $x_i > a_i$, for some $a_i \in D$. Note that the set of constraints Δ^*, constructed as in Definition 5, is equal to the set of constraints specified by arbitrary finite disjunctions of inequalities of the form $x_i > a_i$.

It is easily shown that Δ is 1-independent with respect to Γ. Also, $\mathrm{CSP}_{\Delta_{\leq 1}}(\Gamma \cup \Delta)$ is tractable, since each instance consists of a conjunction of upper bounds for individual variables together with at most one lower bound. Hence, by Corollary 1, $\Gamma \between \Delta^*$ is tractable. By the result quoted above, this establishes that max-closed constraints are tractable.

Unlike the arguments used previously to establish that max-closed constraints are tractable [Jeavons and Cooper 1995; Jeavons et al. 1995], the argument above can still be applied when the set of values D is infinite.

Example 12. **[Connected row-convex constraints]**
The class of binary constraints known as 'connected row-convex' constraints was introduced in [Deville et al. 1997] and shown to be tractable. This class properly includes the 'monotone' relations, identified and shown to be tractable by Montanari in [Montanari 1974].

In this example we will show that the tractability of connected row-convex constraints is a simple consequence of Theorem 3. Furthermore, by using Theorem 3 we are able to generalise this result to obtain tractable constraints over infinite sets of values.

Let the set of possible values D be the ordered set $\{d_1, d_2, \ldots, d_m\}$, where $d_1 < d_2 < \cdots < d_m$. The definition of connected row-convex constraints given in [Deville et al. 1997] uses a standard matrix representation for binary relations: the binary relation R over D is represented by the $m \times m$ 0-1 matrix M, by setting $M_{ij} = 1$ if the relation contains the pair $\langle d_i, d_j \rangle$, and $M_{ij} = 0$ otherwise.

A relation is said to be connected row-convex if the following property holds: the pattern of 1's in the matrix representation (after removing rows and columns containing only 0's) is connected along each row, along each column, and forms a connected 2-dimensional region (where some of the connections may be diagonal).

Here are some examples of connected row-convex relations:

$$\begin{pmatrix} 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&1&0&1&0 \\ 0&1&1&1&1&1&1&0&1&0 \\ 1&1&1&1&1&1&1&0&1&1 \\ 0&1&1&1&1&1&1&0&1&0 \\ 0&0&1&1&1&1&1&0&1&0 \\ 0&0&1&1&1&1&1&0&1&0 \\ 0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \end{pmatrix} \quad \begin{pmatrix} 1&1&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&1&1&1&1 \\ 0&0&0&0&0&1&1&1&1&1 \\ 0&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&1&1&1&0&0 \end{pmatrix} \quad \begin{pmatrix} 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&1&1&1&0 \\ 0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&0&0 \end{pmatrix}$$

An alternative characterisation of this class of constraints, in terms of an algebraic closure property was given in [Jeavons et al. 1998].

Here we obtain another alternative characterisation by noting that the corresponding 0-1 matrices have a very restricted structure. If we eliminate all rows and columns consisting entirely of zeros, and then consider any remaining zero in the matrix, all of the ones in the same row as the chosen zero must lie one side of it (because of the connectedness condition on the row). Similarly, all of the ones in the same column must lie on one side of the chosen zero. Hence there is a complete path of zeros from the chosen zero to the edge of the matrix along both the row and column in one direction. But this means there must be a complete rectangular sub-matrix of zeros extending from the chosen zero to one corner of the matrix (because of the connectedness condition).

This implies that the whole matrix can be obtained as the intersection (conjunction) of 0-1 matrices that contain all ones except for a submatrix of zeros in one corner (simply take one such matrix, obtained as above, for each zero in the matrix to be constructed).

There are four different forms of such matrices, depending on which corner submatrix is zero, and they correspond to constraints expressed by disjunctive expressions of the four following forms:

$$\begin{array}{c}(x_i \geq d_i) \;\vee\; (x_j \geq d_j)\\ (x_i \geq d_i) \;\vee\; (x_j \leq d_j)\\ (x_i \leq d_i) \;\vee\; (x_j \geq d_j)\\ (x_i \leq d_i) \;\vee\; (x_j \leq d_j)\end{array}$$

In these expressions x_i, x_j are variables and d_i, d_j are constants.

Finally, we note that a row or column consisting entirely of zeros corresponds to a constraint of the form $(x_i \leq d_1) \vee (x_i \geq d_2)$ for an appropriate choice of d_1 and d_2.

Hence, any connected row-convex constraint is equivalent to a conjunction of expressions of these forms.

To apply the results of Section 3.3, define Δ to be the set of all unary constraints specified by a single inequality of the form $x_i \leq d_i$ or $x_i \geq d_i$, for some $d_i \in D$.

It is easily shown that Δ has the Krom property and CSP(Δ) is tractable, since each instance consists of a conjunction of upper and lower bounds for individual variables. Hence, by Theorem 3, $\Delta \overset{\times}{\vee} \Delta$ is tractable. By the alternative characterisation described above, this establishes that connected row-convex constraints are tractable.

Unlike the arguments used previously to establish that connected row-convex constraints are tractable [Deville et al. 1997; Jeavons et al. 1998], the argument above can still be applied when the set of values D is infinite.

Example 13. **[Linear Horn constraints]**

The class of constraints over the real numbers known as 'linear Horn' constraints was introduced in [Jonsson and Bäckström 1998; Koubarakis 1996] and shown to be tractable.

A *linear Horn constraint* is specified by a disjunction of weak linear inequalities and linear disequalities where the number of inequalities does not exceed one. The following are examples of linear Horn constraints:

$$\begin{array}{c}3x_1 + x_5 - 3x_4 \leq 10,\\ x_1 + x_3 + x_5 \neq 7,\\ (3x_1 + x_5 - 4x_3 \leq 7) \vee (2x_1 + 3x_2 - 4x_3 \neq 4) \vee (x_2 + x_3 + x_5 \neq 7),\\ (4x_1 + x_3 \neq 3) \vee (5x_2 - 3x_5 + x_4 \neq 6).\end{array}$$

Linear Horn constraints form an important class of linear constraints with explicit connections to temporal reasoning [Jonsson and Bäckström 1998]. In particular, the class of linear Horn constraints properly includes the point algebra of [Vilain et al. 1989], the (quantitative) temporal constraints of [Koubarakis 1992; Koubarakis 1995] and the ORD-Horn constraints of [Nebel and Burckert 1995]. All these classes of temporal constraints can therefore be shown to be tractable using the framework developed here.

Let the set of possible values be the real numbers (or the rationals). Define Γ to be the set of all constraints specified by a single (weak) linear inequality (e.g., $3x_1 + 2x_2 - x_3 \leq 6$), together with the empty constraint. Define Δ to be the set of all constraints specified by a single linear disequality (e.g., $x_1 + 4x_2 + x_3 \neq 0$).

Note that Δ^* is the set of constraints specified by a disjunction of disequalities, and the problem $\text{CSP}(\Gamma \cup \Delta^*)$ corresponds to deciding whether a convex polyhedron, possibly *minus* the union of a finite number of hyperplanes, is the empty set. It was shown in [Lassez and McAloon 1989] that the set $\Gamma \cup \Delta^*$ is independent (using their more restrictive notion of independence referred to in Section 3.2, above), and hence that this problem is tractable.

However, the set of constraints specified by linear Horn constraints corresponds to the much larger set $\Gamma \overset{\times}{\vee} \Delta^*$, and this set is *not* independent in the sense defined in [Lassez and McAloon 1989] (see [Koubarakis 1996]). In order to establish that this larger set of constraints is tractable we shall use the more general notion of 1-independence introduced in this paper.

Consider any set of constraints C in $\text{CSP}(\Gamma \cup \Delta)$. Let $C' = C \cap \Gamma$ (the subset of C which is specified by weak linear inequalities), and let $C'' = C \cap \Delta$ (the subset of C which is specified by linear disequalities). By considering the geometrical interpretation of the constraints as half spaces and excluded hyperplanes in $\mathbb{R}^n$, it is clear that C is consistent if and only if C' is consistent, and, for each $c \in C''$, the set $C' \cup \{c\}$ is consistent (see [Koubarakis 1996]). Hence, Δ is 1-independent with respect to Γ.

Now Lemma 2 and Lemma 1 imply that the function **Ind-Solvable** defined in Figure 1 can be used to determine whether $\text{CSP}(\Gamma \overset{\times}{\vee} \Delta^*)$ has a solution. (In fact, the algorithm in Figure 1 can be seen as a generalisation of the algorithm CONSISTENCY which was developed specifically for this problem in [Koubarakis 1996].)

Finally, to establish tractability, we note that whether a set of inequalities, C', is consistent or not can be decided in polynomial time, using Khachian's linear programming algorithm [Khachian 1979]. Furthermore, for any single disequality constraint, c, we can detect in polynomial time whether $C' \cup \{c\}$ is consistent by simply running Khachian's algorithm to determine whether C' implies the negation of c. Hence, $\text{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ is tractable, so we can apply Corollary 1 and conclude that $\text{CSP}(\Gamma \overset{\times}{\vee} \Delta^*)$ is tractable.

Example 14. [**Extended Horn clauses**]

It was shown by Chandru and Hooker in [Chandru and Hooker 1991] that the class of Horn clauses may be generalised to a much larger class of tractable sets of clauses, which they refer to as 'extended Horn clauses'.

To establish that any extended Horn set of clauses can be solved in polynomial time, Chandru and Hooker give a very indirect argument based on a result from the theory of linear programming [Chandru and Hooker 1991]. In this example we shall establish the tractability of a slightly more general class of constraint sets using a much more direct argument, based on Corollary 1.

In order to define this new class of tractable constraint sets over Boolean variables, we first need to describe how a set of Boolean constraints can be associated with a tree structure.

Let T be a rooted, undirected, tree in which the edges are labelled with propositional literals in such a way that each variable name occurs at most once. Note that T may have an infinite number of edges. If we select an edge of T, then it can be oriented in two different ways: either towards the root, or else away from the root. An edge of T, together with a particular selected orientation, will be called an *arc* of T.

For each possible arc, a, of T we define an associated literal, in the following way. If a is oriented away from the root, then we define the associated literal to be the label of a in T, otherwise, if a is oriented towards the root, then we define the associated literal to be the *negation* of the label of a in T.

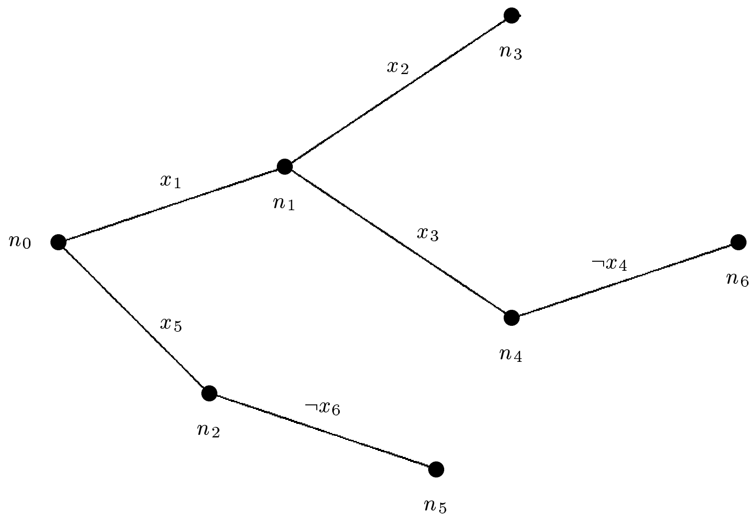

Fig. 3. A labelled tree

For example, let T be the tree shown in Figure 3, with root node n_0. and consider the arc from node n_0 to n_1, denoted $[n_0, n_1]$. The literal associated with $[n_0, n_1]$, according to the definition just given, is x_1. Now consider the arc from node n_5 to n_2, denoted $[n_5, n_2]$. The literal associated with $[n_5, n_2]$ is x_6.

Given any set of arcs, we define an associated clause consisting of the disjunction of the associated literals. For example, the clause associated with the set of arcs $\{[n_0, n_1], [n_5, n_2]\}$ is $x_1 \vee x_6$.

Some sets of arcs have the special property that they form a *path*. For example, in the tree shown in Figure 3, the arcs $[n_0, n_1]$ and $[n_1, n_3]$ form a path of length 2. The arcs $[n_6, n_4], [n_4, n_1], [n_1, n_0]$ and $[n_0, n_2]$ form a path of length 4.

For any rooted, undirected, tree T, we define Γ_T to be the set of all Boolean constraints specified by a clause associated with some *path* in T.

For example, when T is the tree shown in Figure 3, then Γ_T includes the clauses $x_1 \vee x_2$, $x_4 \vee \neg x_3 \vee \neg x_1 \vee x_5$, and $\neg x_2 \vee x_3 \vee \neg x_4$, but does *not* include the clause $x_1 \vee x_6$, or the clause $x_1 \vee \neg x_2$.

We first note that for any tree T, including infinite trees, the set of constraints Γ_T is tractable. This is because any instance of $\mathrm{CSP}(\Gamma_T)$ can be solved by the following polynomial-time algorithm (adapted from [Chandru and Hooker 1991]):

(1) If the instance contains any clauses associated with paths of length one (i.e., unit clauses), then assign values to the corresponding variables to satisfy these

clauses. If any variable receives contradictory assignments then report that the instance is insoluble and stop.

(2) If any variables have been assigned values, then remove all clauses which are satisfied by these assignments, contract the corresponding edges in T (i.e., remove these edges and identify each pair of end points), and return to Step 1.

(3) Report that the problem is soluble. (Since all remaining clauses correspond to paths of length two or more in T, there is guaranteed to be a solution which can be obtained by assigning the values TRUE and FALSE alternately along each branch of T, starting with TRUE.)

However, if we allow disjunctions of constraints in Γ_T, then we no longer have tractability. In fact, we will now show that $\mathrm{CSP}(\Gamma_T \stackrel{\times}{\vee} \Gamma_T)$ can be NP-complete.

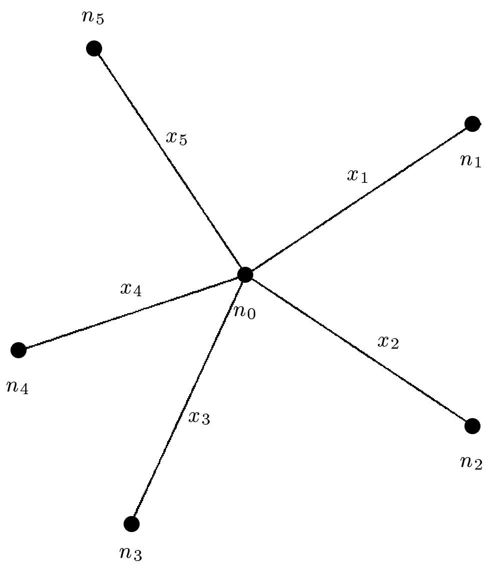

Fig. 4. A labelled star

This intractability result holds even when the trees are restricted to be *stars* (that is, a tree where every edge is incident to the root). Define S_n to be a star with n edges labelled $x_1, x_2, \ldots, x_n$. For example, the star S_5 is shown in Figure 4. Now consider the infinite star, S_∞, with edges labelled $x_1, x_2, \ldots$. The set of constraints Γ_{S_∞} consists of all constraints specified by clauses of the form x_i, $\neg x_i$, or $x_i \vee \neg x_j$, for all choices of i and j. Hence the class of constraints $\Gamma_{S_\infty} \stackrel{\times}{\vee} \Gamma_{S_\infty}$ consists of all constraints specified by disjunctions of these clauses, or, in other words, by clauses of the form $x_i \vee x_j$, $x_i \vee \neg x_j$, $\neg x_i \vee \neg x_j$, $x_i \vee x_j \vee \neg x_k$, $x_i \vee \neg x_j \vee \neg x_k$, or $x_i \vee \neg x_j \vee x_k \vee \neg x_l$, for all possible choices of i, j, k, l. This set of clauses does not fall into any of the 6 tractable classes identified by Schaefer in [Schaefer 1978], and hence the corresponding satisfiability problem, $\mathrm{CSP}(\Gamma_{S_\infty} \stackrel{\times}{\vee} \Gamma_{S_\infty})$, is NP-complete.

In order to obtain *tractable* sets of disjunctive constraints, we now identify a restricted subset of Γ_T which is 1-independent with respect to Γ_T.

For any rooted, undirected, tree T, we define Δ_T to be the set of all Boolean constraints specified by a clause associated with some *path* in T *which ends at the root node of* T.

For example, when T is the tree shown in Figure 3, then Δ_T includes the clauses $\neg x_2 \vee \neg x_1$, $x_4 \vee \neg x_3 \vee \neg x_1$, and $\neg x_5$, but does not include the clause $x_4 \vee \neg x_3$, or the clause $\neg x_5 \vee x_1$. As a second example, when T is a star labelled with positive literals, such as the one shown in Figure 4, then Δ_T consists of all constraints specified by a single negative literal.

Since $\Delta_T \subseteq \Gamma_T$, we can use the same algorithm as before to show that the problem $\mathrm{CSP}_{\Delta_T \leq 1}(\Gamma_T \cup \Delta_T)$ is tractable. To show that Δ_T is 1-independent with respect to Γ_T we note that any minimal incompatible subset of clauses can involve at most one clause from Δ_T. In view of these facts, we can apply Corollary 1 to conclude that $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ is tractable.

Notice that in the special case when T is a star labelled with positive literals the set of constraints $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ corresponds precisely to the set of Horn clauses over the variables labelling T.

In the more general case when T is an arbitrary tree labelled with positive literals, the set of constraints $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ includes the extended Horn set of clauses associated with T, as defined by Chandru and Hooker [Chandru and Hooker 1991]. Furthermore, in the most general case when T is an arbitrary tree labelled with arbitrary literals, the set of constraints $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ includes the *hidden* extended Horn clauses associated with T, as defined by Chandru and Hooker [Chandru and Hooker 1991]. Hence our results are sufficient to show that extended Horn clauses and hidden extended Horn clauses are tractable.

However, we point out that the extended Horn set associated with a tree T, as defined in [Chandru and Hooker 1991], is a little more restricted than the set of constraints $\Gamma_T \overset{\times}{\vee} \Delta_T^*$, and hence our result represents a generalisation of the result in [Chandru and Hooker 1991]. This is because the literals in any clause from an extended Horn set are required to form an 'extended star-chain' pattern in the corresponding tree [Chandru and Hooker 1991]. Such a pattern consists of an arbitrary number of *edge disjoint* paths terminating at the root node, together with at most one other path. The literals of the clauses representing constraints in $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ form similar patterns, but there is no requirement for the paths into the root to be disjoint.

For example, when T is the tree shown in Figure 3, then $\Gamma_T \overset{\times}{\vee} \Delta_T^*$ includes the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee x_6$ (formed from the disjunction of $\neg x_1 \vee \neg x_2$ and $\neg x_1 \vee \neg x_3$ and x_6). However, this clause is not in the (hidden) extended Horn set associated with T.

Example 15. [**Extended Krom clauses**]

The ideas described in Example 14, together with Theorem 3, can be used to identify a new class of tractable Boolean constraint sets which we will call 'extended Krom' sets.

As in Example 14, we associate propositional clauses with sets of arcs in a labelled tree. Let T be a rooted, undirected tree, whose edges are labelled with propositional literals in such a way that each variable name appears at most once. Note that T may have an infinite number of edges. We define Δ_T to be the set of all Boolean constraints specified by a clause associated with some *path* in T which *either starts or ends at the root node of* T.

For example, when T is the tree shown in Figure 3, then Δ_T includes the clauses $\neg x_2 \vee \neg x_1$, $x_1 \vee x_2$, and $\neg x_5$, but does not include the clause $x_4 \vee \neg x_3$, or the clause $\neg x_5 \vee x_1$. As a second example, when T is a star labelled with positive literals, such as the one shown in Figure 4, then Δ_T consists of all constraints specified by a single positive or negative literal.

The algorithm described in Example 14 can be used to show that CSP(Δ_T) is tractable. Furthermore, it is easy to show that any set of clauses chosen from Δ_T is satisfiable, unless it contains the unit clauses x_i and $\neg x_i$ for some i. Hence Δ_T has the Krom property. In view of these facts, we can apply Theorem 3 to conclude that $\Delta_T \overset{\times}{\vee} \Delta_T$ is tractable.

Notice that in the special case when T is a star the set of constraints $\Delta_T \overset{\times}{\vee} \Delta_T$ corresponds precisely to the set of Krom clauses over the variables labelling T.

In the more general case when T is an arbitrary tree, the set of constraints $\Delta_T \overset{\times}{\vee} \Delta_T$ includes a wider variety of clauses. For example, when T is the tree shown in Figure 3, then Δ_T includes the following clauses:

$$x_1 \vee x_2 \vee x_3$$
$$\neg x_2 \vee \neg x_1 \vee x_5 \vee \neg x_6$$
$$\neg x_1 \vee \neg x_5$$

Example 16. [**Tractable set constraints**]

In [Drakengren and Jonsson 1998; Drakengren and Jonsson 1997], Drakengren and Jonsson identify a number of tractable classes of set constraints that are specified by expressions involving set-valued variables and the relation symbols $\subseteq$, disj (denoting disjointness) and $\neq$.

In this example we will show that these tractability results can be obtained as a simple consequence of Theorem 2. In this way we provide a shorter proof and show that these results about set constraints conform to one of the general patterns for tractable disjunctive constraints described in this paper.

We first give the relevant definitions from [Drakengren and Jonsson 1998; Drakengren and Jonsson 1997]. An *atomic set constraint* or *atomic set relation* is an expression of the form $x_i \subseteq x_j$, $x_i \ \mathsf{disj}\ x_j$ or $x_i \neq x_j$. A *disjunctive set constraint* or *disjunctive set relation (DSR)* is a disjunction of atomic constraints. A DSR is called *Horn* if it consists of zero or more disjuncts of the form $x_i \neq x_j$ or $x_i \ \mathsf{disj}\ x_j$ and at most one disjunct of the form $x_i \subseteq x_j$. A DSR is called *2-Horn* if it consists of zero or more disjuncts of the form $x_i \neq x_j$ and at most one disjunct of the form $x_i \subseteq x_j$ or $x_i \ \mathsf{disj}\ x_j$. The following are examples of Horn DSRs.

$$(x_1 \subseteq x_2) \vee (x_3 \neq x_4) \vee (x_4 \neq x_7) \vee (x_5 \ \mathsf{disj}\ x_7)$$
$$(x_1 \subseteq x_2) \vee (x_3 \neq x_4) \vee (x_4 \neq x_7)$$
$$(x_5 \ \mathsf{disj}\ x_7) \vee (x_3 \neq x_4) \vee (x_4 \neq x_7)$$

The latter two examples in this list are also 2-Horn.

An $S_\emptyset$*-interpretation* is a function that maps all set variables in a problem instance to (possibly empty) sets. An S*-interpretation* is a function that maps all set variables to nonempty sets. An atomic constraint $x_1\ R\ x_2$ is satisfied by an $S_\emptyset$-interpretation (S-interpretation), I, if and only if $I(x_1)\ R\ I(x_2)$. A DSR $d = d_1 \vee \cdots \vee d_n$ is *$S_\emptyset$-satisfiable (S-satisfiable)* if there exists an $S_\emptyset$-interpretation (S-interpretation) which satisfies some disjunct d_i of d. A set C of DSRs is *$S_\emptyset$-satisfiable (S-satisfiable)* if there exists an $S_\emptyset$-interpretation (S-interpretation) which satisfies all members of C. Such a satisfying $S_\emptyset$-interpretation (S-interpretation) is called an $S_\emptyset$-model (S-model) of C.

The following decision problems are studied in [Drakengren and Jonsson 1998; Drakengren and Jonsson 1997]:

—HornDSRSat$_\emptyset$:
Instance:. A finite set C of Horn DSRs.
Question:. Does there exist an $S_\emptyset$-model of C?

—HornDSRSat:
Instance:. A finite set C of Horn DSRs.
Question:. Does there exist an S-model of C?

—2HornDSRSat$_\emptyset$:
Instance:. A finite set C of 2-Horn DSRs.
Question:. Does there exist an $S_\emptyset$-model of C?

—2HornDSRSat:
Instance:. A finite set C of 2-Horn DSRs.
Question:. Does there exist an S-model of C?

The problem HornDSRSat$_\emptyset$ is NP-complete [Drakengren and Jonsson 1998].

To show that the problem HornDSRSat can be solved in polynomial time we define Γ to be the set of all constraints specified by expressions of the form $x \subseteq y$ (where x and y are set-valued variables which must be assigned non-empty sets), together with the empty constraint. We define Δ to be the set of all constraints specified by expressions of the form $x \neq y$ or $x \ \mathsf{disj}\ y$ (where x and y are set-variables which must be assigned non-empty sets), together with the empty constraint. Then Δ^* is the set of all constraints specified by arbitrary disjunctions of expressions of the form $x \neq y$ or $x\ \mathsf{disj}\ y$.

In [Drakengren and Jonsson 1997] it is shown that the problem of deciding whether a set of atomic set constraints has an S-model can be solved in polynomial time. The algorithm presented in [Drakengren and Jonsson 1997] represents set constraints by a labeled directed graph and essentially consists of two tests. First, if there is a triple of set variables x, y and z for which the constraints $z \subseteq x, z \subseteq y$ and $x\ \mathsf{disj}\ y$ are implied by the given constraint set, the input set is rejected (because variable z is forced to be equal to $\emptyset$). Secondly, if there is a pair of set variables x and y for which the constraints $x \subseteq y, y \subseteq x$ and $x \neq y$ are implied by the given constraint set, then the input is rejected (because these constraints cannot all be satisfied). If a constraint set passes both of these tests then it is accepted.

The existence of this simple algorithm implies that $\mathrm{CSP}(\Gamma \cup \Delta)$ and hence also $\mathrm{CSP}_{\Delta \leq 1}(\Gamma \cup \Delta)$ can be solved in polynomial time. It also implies that Δ is 1-independent with respect to Γ. Hence, by Corollary 1 we can immediately conclude that $\mathrm{CSP}(\Gamma \curlyvee \Delta^*)$, which corresponds to the decision problem HornDSRSat, can be solved in polynomial time. In fact the algorithm defined in Figure 1 can be seen as a generalisation of the iterative version of Algorithm **Horn-Sat** presented in [Drakengren and Jonsson 1997].

The problem 2HornDSRSat is a subproblem of HornDSRSat so it can also be solved in polynomial time.

To show that the decision problem $2\text{HornDSRSat}_\emptyset$ can be solved in polynomial time, define Γ to be the set of all constraints specified by an expression which is either of the form $x \subseteq y$ or else of the form x disj y (where x and y are set-valued variables which can be assigned arbitrary sets), together with the empty constraint. Define Δ to be the set of all constraints specified by an expression of the form $x \neq y$ (where x and y are set-valued variables which can be assigned arbitrary sets), together with the empty constraint. In this case Δ^* is the set of constraints specified by a disjunction of expressions of the form $x \neq y$.

In [Drakengren and Jonsson 1998] it is shown that the problem of deciding whether a set of atomic set constraints has an $S_\emptyset$-model can be solved in polynomial time. As in the above case, set constraints are represented by a labeled directed graph. The algorithm proceeds in two steps. First, for any triple of set variables x, y and z for which the constraints $z \subseteq x, z \subseteq y$ and x disj y are implied by the given constraint set, variable z is forced to be equal to $\emptyset$. Then the constraints are examined to find out whether a contradictory triple of constraints $x \subseteq y, y \subseteq x$ and $x \neq y$ is implied by the given constraint set. If this is the case, then the set is rejected. Otherwise, it is accepted.

The existence of this simple algorithm implies that $\text{CSP}(\Gamma \cup \Delta)$ and hence also $\text{CSP}_{\Delta_{\leq 1}}(\Gamma \cup \Delta)$ can be solved in polynomial time. It also implies that Δ is 1-independent with respect to Γ. Hence, by Corollary 1 we can immediately conclude that $\text{CSP}(\Gamma \overset{\times}{\vee} \Delta^*)$, which now corresponds to the decision problem $2\text{HornDSRSat}_\emptyset$, can be solved in polynomial time. As in the above case, the algorithm defined in Figure 1 can be seen as a generalisation of the iterative version of Algorithm **2-Horn-Sat** presented in [Drakengren and Jonsson 1998].

Having identified the key properties underlying all of these tractable classes, we are now in a position to identify new tractable classes simply by searching for appropriate sets of tractable constraints which have these properties.

Example 17. **[Disjunctive congruences]**

Constraints in the form of congruence relations over the integers are used in a variety of applications, including the representation of large integers in computer algebra systems [von zur Gathen and Gerhard 1999], and the representation of periodic events in temporal databases [Kabanza et al. 1995; Bertino et al. 1998].

One of the fundamental results of elementary number theory is the Chinese Remainder Theorem [Jackson 1975], which states that a collection of simultaneous linear congruences

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

is solvable if and only if the greatest common divisor of m_i and m_j divides $a_i - a_j$, for all distinct i and j. When this condition holds between a pair of congruences, we shall say that they are *compatible* (Note that compatibility can be decided in polynomial time.)

Using this result, together with the results established earlier in this paper, we will construct a number of new tractable classes of constraints.

Let the set of possible values be the set of integers. Define Γ to be the set of all unary constraints which are specified by a linear congruence of the form $x \equiv a \pmod{m}$, for some natural number a, and some modulus m, together with the empty constraint. For each natural number b, define Γ_b to be the subset of Γ containing all unary constraints which are specified by a congruence of the form $x \equiv b \pmod{m}$, for some m.

A problem instance in CSP(Γ) is specified by a collection of simultaneous congruences. For example, a typical set of constraints in CSP(Γ), involving the variables x_1 and x_2, would be:

$$\begin{aligned} x_1 &\equiv 3 \pmod{5}, \\ x_2 &\equiv 4 \pmod{6}, \\ x_2 &\equiv 2 \pmod{4}. \end{aligned}$$

The Chinese Remainder Theorem implies that any set of constraints in CSP(Γ) has a solution, unless it contains a pair of constraints which are incompatible. In view of this fact, the results obtained in this paper can be used to construct tractable disjunctive constraints in the following three ways:

—For any natural number b, it is clear from the definition of compatibility above that every pair of constraints in Γ_b is compatible. Hence, Γ_b has the guaranteed satisfaction property, and by Theorem 1 we conclude that CSP($\Gamma_b \stackrel{\times}{\vee} \Gamma^*$) is tractable. This means that there is an efficient way to solve any collection of simultaneous disjunctions of congruences which all have the property that at least one disjunct comes from Γ_b. For example, when $b = 2$ we might have the following collection:

$$\begin{aligned} &x_1 \equiv 2 \pmod{4} \ \vee\ x_1 \equiv 2 \pmod{3} \ \vee\ x_2 \equiv 1 \pmod{5}, \\ &x_2 \equiv 2 \pmod{3} \ \vee\ x_3 \equiv 2 \pmod{5}, \\ &x_3 \equiv 2 \pmod{7}, \\ &x_2 \equiv 2 \pmod{7} \ \vee\ x_1 \equiv 2 \pmod{12} \ \vee\ x_3 \equiv 1 \pmod{3} \end{aligned}$$

—For any natural number b, $\mathrm{CSP}_{\Gamma_b \leq 1}(\Gamma \cup \Gamma_b)$ is tractable, because we can determine whether or not any given instance has a solution in polynomial time by examining each pair of constraints to see whether they are compatible. Furthermore, no pair of constraints which are both in Γ_b can be incompatible, so Γ_b is 1-independent with respect to Γ. Hence, by Corollary 1, we conclude that CSP($\Gamma \stackrel{\times}{\vee} \Gamma_b^*$) is also tractable, for any natural number b. This means that there is an efficient way to solve any collection of simultaneous disjunctions of congruences which all have the property that at most one disjunct comes from Γ and the remainder (if any) from Γ_b. For example, when $b = 2$ we might have the following collection of congruences:

$$x_1 \equiv 1 \pmod{4} \ \vee\ x_1 \equiv 2 \pmod{3} \ \vee\ x_2 \equiv 2 \pmod{5},$$

$$x_2 \equiv 2 \pmod 3 \;\vee\; x_3 \equiv 2 \pmod 5,$$
$$x_3 \equiv 5 \pmod 7,$$
$$x_2 \equiv 1 \pmod 7 \;\vee\; x_1 \equiv 2 \pmod{12} \;\vee\; x_3 \equiv 2 \pmod 3$$

—From the observations already made it is clear that the set Γ has the Krom property. Hence, by Theorem 3 we conclude that CSP($\Gamma \between \Gamma$) is tractable. This means that there is an efficient way to solve any collection of simultaneous disjunctions of congruences which all contain at most two disjuncts. For example, we might have the following collection:

$$x_1 \equiv 1 \pmod 4 \;\vee\; x_1 \equiv 0 \pmod 3,$$
$$x_2 \equiv 2 \pmod 3 \;\vee\; x_3 \equiv 2 \pmod 5,$$
$$x_3 \equiv 5 \pmod 7,$$
$$x_2 \equiv 1 \pmod 7 \;\vee\; x_1 \equiv 2 \pmod{12}$$

Note that the new tractable disjunctive constraints sets constructed in these three distinct ways are all incomparable with each other.

5. CONCLUSION

In this paper we have established three sufficient conditions for tractability of disjunctive constraints. We have shown that these conditions account for a wide variety of known tractable constraint classes, over both finite and infinite sets of values, and that they aid the search for new tractable constraint classes. The examples we have given of new tractable classes obtained in this way are as follows:

—A generalisation of max-closed constraints to infinite (ordered) domains (Example 11);

—A generalisation of connected row-convex constraints to infinite (ordered) domains (Example 12);

—A (slight) generalisation of extended Horn Clauses (Example 14);

—The new class of extended Krom clauses (Example 15);

—Three new classes of tractable disjunctive congruences (Example 17).

These results provide the first examples of a constructive approach to obtaining tractable constraints, based on combining known tractable classes.

This new approach to obtaining tractable classes leads to results of great generality, as we have shown in this paper. It raises the possibility that on any given domain there are only a small number of 'basic' tractable constraint types, and all other tractable constraint classes can be built up from these using a small number of standard construction techniques.

REFERENCES

Aspvall, B., Plass, M., and Tarjan, R. 1979. A linear time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters 8*, 121–123.

Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. 1998. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems 23*, 3, 231–285.

CHANDRU, V. AND HOOKER, J. 1991. Extended Horn sets in propositional logic. *Journal of the ACM 38*, 205–221.

COHEN, D., JEAVONS, P., AND KOUBARAKIS, M. 1997. Tractable disjunctive constraints. In *Proceedings 3rd International Conference on Constraint Programming—CP'97 (Linz, October 1997)*, Volume 1330 of *Lecture Notes in Computer Science* (1997), pp. 478–490. Springer-Verlag.

COOPER, M., COHEN, D., AND JEAVONS, P. 1994. Characterising tractable constraints. *Artificial Intelligence 65*, 347–361.

DECHTER, R. AND PEARL, J. 1989. Tree clustering for constraint networks. *Artificial Intelligence 38*, 353–366.

DENENBERG, H. AND LEWIS, H. R. 1984. The complexity of the satisfiability problem for Krom formulas. *Theoretical Computer Science 30*, 319–341.

DEVILLE, Y., BARETTE, O., AND VAN HENTENRYCK, P. 1997. Constraint satisfaction over connected row convex constraints. In *Proceeedings of IJCAI'97* (1997), pp. 405–411.

DRAKENGREN, T. 1997. *Algorithms and Complexity for Temporal and Spatial Formalisms.* Ph. D. thesis, Dept. of Computer and Information Science, Linköping University. Available from http://www.ida.liu.se/ext/pur/tphd/0498.html.

DRAKENGREN, T. AND JONSSON, P. 1997. Qualitative reasoning about sets applied to spatial reasoning. (Paper #7 in [Drakengren 1997]).

DRAKENGREN, T. AND JONSSON, P. 1998. Reasoning about set constraints applied to tractable inference in intuitionistic logic. *Journal of Logic and Computation 8*, 855–875.

FREUDER, E. 1985. A sufficient condition for backtrack-bounded search. *Journal of the ACM 32*, 755–761.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA.

GYSSENS, M., JEAVONS, P., AND COHEN, D. 1994. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence 66*, 1, 57–89.

JACKSON, T. 1975. *Number Theory.* Routledge and Kegan Paul.

JEAVONS, P., COHEN, D., AND COOPER, M. 1998. Constraints, consistency and closure. *Artificial Intelligence 101*, 1-2, 251–265.

JEAVONS, P., COHEN, D., AND GYSSENS, M. 1995. A unifying framework for tractable constraints. In *Proceedings 1st International Conference on Constraint Programming—CP'95 (Cassis, France, September 1995)*, Volume 976 of *Lecture Notes in Computer Science* (1995), pp. 276–291. Springer-Verlag.

JEAVONS, P., COHEN, D., AND GYSSENS, M. 1997. Closure properties of constraints. *Journal of the ACM 44*, 527–548.

JEAVONS, P. AND COOPER, M. 1995. Tractable constraints on ordered domains. *Artificial Intelligence 79*, 2, 327–339.

JONSSON, P. AND BÄCKSTRÖM, C. 1998. A unifying approach to temporal constraint reasoning. *Artificial Intelligence 102*, 143–155.

KABANZA, F., STÉVENNE, J.-M., AND WOLPER, P. 1995. Handling infinite temporal data. *Journal of Computer and System Sciences 51*, 3–17.

KHACHIAN, L. 1979. A polynomial time algorithm for linear programming. *Soviet Math. Dokl. 20*, 191–194.

KIROUSIS, L. 1993. Fast parallel constraint satisfaction. *Artificial Intelligence 64*, 147–160.

KOUBARAKIS, M. 1992. Dense time and temporal constraints with $\neq$. In B. NEBEL, C. RICH, AND W. SWARTOUT Eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)* (San Mateo, CA, 1992), pp. 24–35. Morgan Kaufmann.

KOUBARAKIS, M. 1995. From local to global consistency in temporal constraint networks. In *Proceedings 1st International Conference on Constraint Programming—CP'95 (Cassis, France, September 1995)*, Volume 976 of *Lecture Notes in Computer Science* (1995), pp. 53–69. Springer-Verlag.

KOUBARAKIS, M. 1996. Tractable disjunctions of linear constraints. In *Proceedings 2nd International Conference on Constraint Programming—CP'96 (Boston, August 1996)*, Volume 1118 of *Lecture Notes in Computer Science* (1996), pp. 297–307. Springer-Verlag.

LASSEZ, J.-L. AND MCALOON, K. 1989. A canonical form for generalized linear constraints. In *TAPSOFT '89, Advanced Seminar on Foundations of Innovative Software Development*, Volume 351 of *Lecture Notes in Computer Science* (1989), pp. 19–27. Springer-Verlag.

LASSEZ, J.-L. AND MCALOON, K. 1991. A constraint sequent calculus. In *Constraint Logic Programming, Selected Research*, pp. 33–43. MIT Press.

LASSEZ, J.-L. AND MCALOON, K. 1992. A canonical form for generalized linear constraints. *Journal of Symbolic Computation 13*, 1–24.

LESAINT, D., AZARMI, N., LAITHWAITE, R., AND WALKER, P. 1998. Engineering dynamic scheduler for work manager. *BT Technology Journal 16*, 16–29.

MACKWORTH, A. 1977. Consistency in networks of relations. *Artificial Intelligence 8*, 99–118.

MONTANARI, U. 1974. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences 7*, 95–132.

NEBEL, B. AND BURCKERT, H.-J. 1995. Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. *Journal of the ACM 42*, 43–66.

PURVIS, L. AND JEAVONS, P. 1999. Constraint tractability theory and its application to the product development process for a constraint-based scheduler. In *Proceedings of the First International Conference on the Practical Applications of Constraint Technologies and Logic Programming - PACLP'99* (1999), pp. 63–79.

SCHAEFER, T. 1978. The complexity of satisfiability problems. In *Proceedings 10th ACM Symposium on Theory of Computing (STOC)* (1978), pp. 216–226.

VAN BEEK, P. AND DECHTER, R. 1995. On the minimality and decomposability of row-convex constraint networks. *Journal of the ACM 42*, 543–561.

VAN HENTENRYCK, P., DEVILLE, Y., AND TENG, C.-M. 1992. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence 57*, 291–321.

VILAIN, M., KAUTZ, H., AND VAN BEEK, P. 1989. Constraint propagation algorithms for temporal reasoning: A revised report. In D. WELD AND J. DE KLEER Eds., *Readings in Qualitative Reasoning about Physical Systems*, pp. 373–281. Morgan Kaufmann.

VON ZUR GATHEN, J. AND GERHARD, J. 1999. *Modern Computer Algebra.* Cambridge University Press.