Collapsibility and Consistency in Quantified Constraint Satisfaction

Hubie Chen
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
hubes@cs.cornell.edu

Abstract

The concept of *consistency* has pervaded studies of the constraint satisfaction problem. We introduce two concepts, which are inspired by consistency, for the more general framework of the quantified constraint satisfaction problem (QCSP). We use these concepts to derive, in a uniform fashion, proofs of polynomial-time tractability and corresponding algorithms for certain cases of the QCSP where the types of allowed relations are restricted. We not only unify existing tractability results and algorithms, but also identify new classes of tractable QCSPs.

Introduction

The constraint satisfaction problem (CSP) is widely acknowledged as a convenient framework for modelling search problems. An instance of the CSP consists of a set of variables, a domain, and a set of constraints; each constraint consists of a tuple of variables paired with a relation (over the domain) which contains permitted values for the variable tuple. The question is to decide whether or not there is an assignment mapping each variable to a domain element that satisfies all of the constraints. Because the CSP is in general NP-complete, a tremendous amount of research effort has been devoted to identifying solution techniques that are effective in practice, as well as restricted cases of the problem that are polynomial-time tractable.

All of the variables in a CSP can be thought of as being implicitly existentially quantified. A useful generalization of the CSP is the quantified constraint satisfaction problem (QCSP), where variables may be both existentially and universally quantified. Whereas the CSP concerns deciding the existence of a *static* object, a satisfying assignment, the QCSP concerns deciding the existence of a *dynamic* object: a strategy telling how to set the existentially quantified variables in reaction to an arbitrary setting of the universally quantified variables, so that the constraints are satisfied. The generality of the QCSP framework permits the modelling of a variety of artificial intelligence problems that cannot be expressed using the CSP, for instance, problems from the areas of planning, game playing, and non-monotonic reasoning. Of course, the relatively higher expressiveness of the QCSP comes at the price of higher complexity: the QCSP is in general complete for the complexity class PSPACE, which is believed to be much larger than NP.

In contrast to the rich and ever-expanding body of research on the CSP, the study of the QCSP appears to be in its infancy. In light of this age gap, a natural approach to understanding the QCSP is to develop analogs of notions that have been successful in understanding the CSP. *Consistency* is a concept that has pervaded both practical and theoretical investigations of the CSP, and is thus extremely obvious as a candidate notion for import into the QCSP world. From a high-level viewpoint, consistency procedures involve performing computation that is localized, in that a small number of data items are considered at a time. More specifically, consistency procedures allow for the "tightening" of a CSP by the continual consideration of bounded-size sets of variables and elimination of tuples from relations that cannot be used by any satisfying assignment. Although they are simple, such procedures have been used heavily to identify polynomial-time tractable cases of the CSP; as a sampling of work along these lines, we name the papers (Freuder 1978; 1982; 1985; Dechter 1992; van Beek & Dechter 1995; Dechter & van Beek 1996; van Beek & Dechter 1997; Feder & Vardi 1998; Jeavons, Cohen, & Cooper 1998; Dalmau & Pearson 1999; Dalmau, Kolaitis, & Vardi 2002; Bulatov 2002; Bulatov & Jeavons 2003). Indeed, the HORN SATISFIABILITY and 2-SATISFIABILITY problems, two classic tractable cases of the CSP, are solvable by consistency-based algorithms.

In this paper, we introduce two new concepts for use with QCSPs which are inspired by consistency: *collapsibility* and *extended minimality*. A class of QCSP problems has *bounded collapsibility* when, roughly speaking, deciding the truth of an instance can be reduced to deciding the truth of an ensemble of instances, all of which have a bounded number of universally quantified variables and are derived from the original instance by "collapsing" together universally quantified variables. *Extended minimality* extends a notion of consistency called minimality that has been used to study the CSP; it is a "smooth" extension of minimality in that the notions of minimality and extended minimality perfectly coincide for CSPs (equivalently, QCSPs with no universal quantification).

These two new concepts yield a number of ramifications, which we present and discuss in this paper. First, when used together, the concepts provide a uniform proof technique for deriving the tractability of certain cases of the QCSP where the types of allowed relations are restricted. This technique reconciles and reveals common structure among known tractable cases of the QCSP that were originally proved to be tractable using disparate proof techniques (Aspvall, Plass, & Tarjan 1979; Karpinski, Büning, & Schmitt 1987; Börner *et al.* 2003; Chen 2003b). (These cases generalize tractable cases of the CSP solvable by consistency procedures.) Second, any cases of the QCSP to which this proof technique applies, are solvable by (possibly different parameterizations of) a single algorithm which, as with consistency-based algorithms for the CSP, is not only efficient, but conceptually simple. In fact, we obtain the first unified polynomial-time algorithms for QUANTIFIED HORN SATISFIABILITY and QUANTIFIED 2-SATISFIABILITY as a corollary of our results; these two cases of the QCSP have been known to be polynomial-time tractable for many years (Aspvall, Plass, & Tarjan 1979; Karpinski, Büning, & Schmitt 1987). In addition to unifying known QCSP tractability results, we expand the tractability frontier by identifying new, broad classes of tractable QCSPs. For example, we show that a certain large class of *multi-sorted* QCSPs are tractable; tractability results for multi-sorted CSPs (Bulatov & Jeavons 2003) play a major role in the recent, spectacular dichotomy theorem on CSP complexity for a three-element domain (Bulatov 2002), and we expect that multi-sorted QCSPs will similarly be significant in understanding the complexity of the QCSP.

Preliminaries

We use $[n]$ to denote the set containing the first n positive integers, that is, $\{1, \ldots, n\}$. For a function $f : A \to B$ and a subset $A' \subseteq A$, we denote by $f_{|A'}$ the restriction of f to A'. For a set F of functions $f : A \to B$ and a subset $A' \subseteq A$, we denote by $F_{|A'}$ the set of functions $\{f_{|A'} : f \in F\}$.

Quantified constraint satisfaction. A *domain* is a nonempty set of finite size. A *tuple* (over domain D) is an element of D^k for some $k \geq 1$, and is said to have arity k. The ith coordinate of a tuple $\overline{t}$ is denoted by t_i. A *relation* (over domain D) is a subset of D^k for some $k \geq 1$, and is said to have arity k.

A *constraint* is an expression of the form $R(\overline{v})$, where R is a relation and $\overline{v}$ is a tuple of variables such that R and $\overline{v}$ have the same arity. A *constraint network* is a finite set of constraints, all of which have relation over the same domain, and is said to be over the variable set V if all of its constraints have variables from V. Throughout, we let D denote the domain of our constraints and constraint networks.

Definition 1 *A quantified formula is an expression of the form $Q_1 v_1 \ldots Q_n v_n \mathcal{C}$, where each Q_i is a quantifier from the set $\{\forall, \exists\}$, and $\mathcal{C}$ is a constraint network over the variable set $\{v_1, \ldots, v_n\}$.*

We call $Q_1 v_1 \ldots Q_n v_n$ the *quantifier prefix* of the quantified formula $Q_1 v_1 \ldots Q_n v_n \mathcal{C}$. We say that the variable v_i comes (strictly) before the variable v_j if $i \leq j$ ($i < j$). We let V_ϕ, Y_ϕ, and X_ϕ denote the variables, universally quantified variables, and existentially quantified variables of a quantified formula ϕ, respectively; we drop the ϕ subscript when it is understood from the context. (Notice that V_ϕ is the disjoint union of Y_ϕ and X_ϕ). We generally assume that the universally quantified variables (of a quantified formula) are denoted $y_1, \ldots, y_{|Y|}$, where y_i comes strictly before y_j for $i < j$; similarly, we generally assume that the existentially quantified variables (of a quantified formula) are denoted $x_1, \ldots, x_{|X|}$, where x_i comes strictly before x_j for $i < j$. When W is a non-empty subset of the variable set V_ϕ (of a quantified formula ϕ), we use $\mathsf{first}_\phi(W)$ to denote the unique variable in W coming before all of the other variables in W, and we use $\mathsf{last}_\phi(W)$ to denote the unique variable w in W such that all of the other variables in W come before w. For a subset W of the variable set V (of a quantified formula) and a variable v of V, we let $W[\leq v]$ ($W[< v]$) denote the set of variables in W coming (strictly) before v. As an example, consider a quantified formula with quantifier prefix $\forall y_1 \exists x_1 \exists x_2 \forall y_2 \exists x_3$. Relative to this formula, $X[\leq x_3]$ denotes the set $\{x_1, x_2, x_3\}$; $X[< x_3]$ denotes the set $\{x_1, x_2\}$; and $Y[\leq x_3]$ and $Y[< x_3]$ both denote the set $\{y_1, y_2\}$.

Strategies and truth. A constraint $R(v_1, \ldots, v_k)$ is *satisfied* by an assignment f defined on $\{v_1, \ldots, v_k\}$ if $(f(v_1), \ldots, f(v_k)) \in R$. A constraint network $\mathcal{C}$ (over the variable set V) is *satisfied* by an assignment $f : V \to D$ if each constraint C in $\mathcal{C}$ is satisfied by f.

Definition 2 *A* strategy σ *is a sequence of mappings*

$$\{\sigma_i : D^{\mathsf{rank}(\sigma_i)} \to D\}_{i \in [n]}$$

where the ith mapping σ_i is a function over D having rank $\mathsf{rank}(\sigma_i) \geq 0$.

Note that when σ_i is a mapping of a strategy such that $\mathsf{rank}(\sigma_i) = 0$, we consider σ_i to be a constant, that is, an element of D.

A *strategy for the quantified formula ϕ* is a strategy $\sigma_1, \ldots, \sigma_{|X_\phi|}$ where for $i \in [|X_\phi|]$, the mapping σ_i has rank $|Y_\phi[< x_i]|$. An *adversary for the quantified formula ϕ* is a function $\tau : Y_\phi \to D$. When σ is a strategy and τ is an adversary for the quantified formula ϕ, the *outcome* of σ and τ, denoted by $\mathsf{outcome}(\sigma, \tau) : V_\phi \to D$, is the assignment defined by $\mathsf{outcome}(\sigma, \tau)(x_i) = \sigma_i(\tau(y_1), \ldots, \tau(y_{|Y[<x_i]|}))$ for $x_i \in X_\phi$ and $\mathsf{outcome}(\sigma, \tau)(y_i) = \tau(y_i)$ for $y_i \in Y_\phi$.

A strategy σ for the quantified formula ϕ is said to be *winning* if for all adversaries τ for ϕ, the assignment $\mathsf{outcome}(\sigma, \tau) : V_\phi \to D$ satisfies the constraint network $\mathcal{C}$ of ϕ. We consider a quantified formula ϕ to be *true* if there exists a winning strategy for ϕ. (This is one of many equivalent ways to define truth of a quantified formula.)

Problem formulation. This paper focuses on the following restricted version of the QCSP, where all relations must come from a *constraint language*. A *constraint language* is defined to be a set of relations (not necessarily of the same arity), all of which are over the same domain.

Definition 3 *Let Γ be a constraint language. The* $\mathsf{QCSP}(\Gamma)$ *problem is to decide, given as input a quantified formula ϕ with constraints having relations from Γ, whether or not ϕ is true.*

We define the $\mathsf{CSP}(\Gamma)$ problem to be the restriction of the $\mathsf{QCSP}(\Gamma)$ problem to instances where all quantifiers are existential.

We will find it convenient to use an alternative formulation of the notion of a constraint. A *functional constraint* C consists of a set of variables called the *scope* of C, denoted by $\mathsf{scope}(C)$; and, a set of functions from $\mathsf{scope}(C)$ to a domain D, denoted by $\mathsf{funs}(C)$. A functional constraint C is considered to be satisfied by an assignment $f : V \to D$ defined on $\mathsf{scope}(C)$ when $f_{|\mathsf{scope}(C)} \in \mathsf{funs}(C)$. A constraint $R(v_1, \ldots, v_k)$ naturally induces the functional constraint C where $\mathsf{scope}(C) = \{v_1, \ldots, v_k\}$ and $\mathsf{funs}(C)$ contains all functions $f : \{v_1, \ldots, v_k\} \to D$ such that $\{(v_i, f(v_i)) : i \in [k]\} = \{(v_i, t_i) : i \in [k]\}$ for some tuple $\overline{t} \in R$. Note that an assignment $f : V \to D$ satisfies a constraint $R(v_1, \ldots, v_k)$ if and only if it satisfies the functional constraint naturally induced by $R(v_1, \ldots, v_k)$. We use functional constraints and constraints interchangeably in this paper.

Invariance and set functions. A powerful algebraic theory for studying the complexity of $\mathsf{CSP}(\Gamma)$ problems, which can also be applied to study the complexity of $\mathsf{QCSP}(\Gamma)$ problems, was introduced in (P.G.Jeavons, D.A.Cohen, & M.Gyssens 1997; Jeavons 1998). (We refer the reader to those papers for more information.) Two key definitions of this theory to be used throughout this paper are the following. A relation $R \subseteq D^m$ is *invariant* under an operation $\mu : D^k \to D$ if for all tuples $\overline{t_1}, \ldots, \overline{t_k} \in R$, the tuple $(\mu(t_{11}, \ldots, t_{k1}), \ldots, \mu(t_{1m}, \ldots, t_{km}))$ is in R. A constraint language Γ is *invariant* under an operation $\mu : D^k \to D$ if all relations $R \in \Gamma$ are invariant under μ.

We consider a *set function* (on domain D) to be a mapping from $\mathcal{P}(D) \setminus \{\emptyset\}$ to D, where $\mathcal{P}(D)$ denotes the power set of D. A set function $f : \mathcal{P}(D) \setminus \{\emptyset\} \to D$ naturally induces a sequence of functions $f_k : D^k \to D$ defined by $f_k(d_1, \ldots, d_k) = f(\{d_1, \ldots, d_k\})$, for $k \geq 1$. We consider a relation (or constraint language) to be invariant under a set function $f : \mathcal{P}(D) \setminus \{\emptyset\} \to D$ if it is invariant under all of the functions f_k (for $k \geq 1$). We say that a set function $f : \mathcal{P}(D) \setminus \{\emptyset\} \to D$ is an *idempotent set function* if all of the functions f_k (for $k \geq 1$) are idempotent.

Collapsibility

In this section, we develop machinery for demonstrating that a constraint language Γ has *bounded collapsibility*; essentially, this means that an arbitrary instance of $\mathsf{QCSP}(\Gamma)$ can be reduced to an ensemble of instances of $\mathsf{QCSP}(\Gamma)$, all of which have a constant number of universally quantified variables. Bounded collapsibility is computationally a very strong condition, since quantified formulas with a constant number of universally quantified variables are, intuitively, very close to being CSPs. After developing sufficient conditions for bounded collapsibility, we give examples of classes of constraint languages that have bounded collapsibility.

Collapsings of formulas. Define a quantified formula ϕ' to be a *j-collapsing* of a quantified formula ϕ if there exists a subset Y' of Y_ϕ such that $|Y'| = \min(j, |Y_\phi|)$ and ϕ' can be obtained from ϕ by first eliminating, from the quantifier prefix of ϕ, all variables in $Y_\phi \setminus Y'$ except for $\mathsf{first}_\phi(Y_\phi \setminus Y')$; and then replacing, in the constraint network of ϕ, all variables in $Y_\phi \setminus Y'$ with $\mathsf{first}_\phi(Y_\phi \setminus Y')$. For example, the quantified formula

$$\forall y_1 \exists x_1 \forall y_2 \forall y_3 \exists x_2 \{R_1(y_1, x_1), R_2(y_3, x_2), R_3(y_2, y_3, x_2)\}$$

has the following three 1-collapsings:

$$\forall y_1 \exists x_1 \forall y_2 \exists x_2 \{R_1(y_1, x_1), R_2(y_2, x_2), R_3(y_2, y_2, x_2)\}$$

$$\forall y_1 \exists x_1 \forall y_2 \exists x_2 \{R_1(y_1, x_1), R_2(y_1, x_2), R_3(y_2, y_1, x_2)\}$$

$$\forall y_1 \exists x_1 \forall y_3 \exists x_2 \{R_1(y_1, x_1), R_2(y_3, x_2), R_3(y_1, y_3, x_2)\}.$$

We have the following fact concerning the j-collapsings of a true quantified formula.

Proposition 4 *If a quantified formula ϕ is true, then for any $j \geq 1$ and any j-collapsing ϕ' of ϕ, the quantified formula ϕ' is true.*

We will be able to demonstrate classes of formulas for which the converse of Proposition 4 holds; for such formulas, deciding whether or not ϕ is true is equivalent to deciding whether or not all j-collapsings of ϕ are true. The following characterization of when the j-collapsings of a quantified formula are true, will be wieldy. Define a set of functions F of the form $\tau : Y \to D$ to be a *j-adversary set* for Y if there exists a subset $Y' \subseteq Y$ of size $\min(j, |Y|)$ such that F contains exactly those functions τ having the property that $\tau(y_1) = \tau(y_2)$ for all $y_1, y_2 \in Y \setminus Y'$. When ϕ is a quantified formula and F is a set of adversaries $\tau : Y_\phi \to D$, we say that the formula ϕ is F-winnable if there exists a strategy σ for ϕ such that for all $\tau \in F$, the assignment $\mathsf{outcome}(\sigma, \tau)$ satisfies the constraint network of ϕ.

It is straightforward to verify that the j-collapsing of a quantified formula corresponding to a subset Y' is true, if and only if the formula is F-winnable, where F is the j-adversary set corresponding to Y'. Hence, we have the following proposition.

Proposition 5 *Let ϕ be a quantified formula. The j-collapsings ϕ' of ϕ are all true if and only if for each j-adversary set F for Y_ϕ, the formula ϕ is F-winnable.*

Collapsibility of constraint languages. We say that a constraint language Γ is *j-collapsible* if for every quantified formula ϕ that is an instance of $\mathsf{QCSP}(\Gamma)$, the following property holds: if all j-collapsings of ϕ are true, then ϕ is true. We say that a constraint language Γ has *bounded collapsibility* if there exists a $j \geq 1$ such that Γ is j-collapsible. Using the notion of invariance, we will be able to identify classes of constraint languages that have bounded collapsibility. In order to do this, we introduce a notion of composability, with respect to a function μ.

When $\mu : D^k \to D$ is a function, and $F, F_1, \ldots, F_k$ are sets of functions of the form $\tau : \{y_1, \ldots, y_n\} \to D$, we say that F *is μ-composable in one step from* $F_1, \ldots, F_k$ if there exist strategies $\pi^1 = \{\pi^1_i : D^i \to D\}_{i \in [n]}, \ldots, \pi^k = \{\pi^k_i : D^i \to D\}_{i \in [n]}$ such that for all $\tau \in F$, the following two properties hold: (1) for all $i \in [k]$, the mapping $\tau^i : \{y_1, \ldots, y_n\} \to D$ defined by $\tau^i(y_j) = \pi^i_j(\tau(y_1), \ldots, \tau(y_j))$ (for all $j \in [n]$) is contained in F_i, and (2) it holds that $\tau(y_j) = \mu(\tau^1(y_j), \ldots, \tau^k(y_j))$ (for all $j \in [n]$). The key feature of this definition is the following lemma.

Lemma 6 *Let ϕ be a quantified formula with $\{y_1, \ldots, y_n\}$ as its universally quantified variables and with relations invariant under μ, and suppose that F is μ-composable in one step from $F_1, \ldots, F_k$. If ϕ is F_i-winnable for all $i \in [k]$, then ϕ is F-winnable.*

When $\mu : D^k \to D$ is a function, and $F, F_1, \ldots, F_k$ are sets of functions of the form $\tau : \{y_1, \ldots, y_n\} \to D$, we say that F *is μ-composable from* $F_1, \ldots, F_k$ if there exist sets of functions $F_{k+1}, \ldots, F_m$ (containing functions of the form $\tau : \{y_1, \ldots, y_n\} \to D$) such that $F_m = F$ and for all $i \in [m]$ with $i > k$, the set of functions F_i is μ-composable in one step from $F_{j_1}, \ldots, F_{j_k}$ for some $j_1, \ldots, j_k \in [i-1]$. A function $\mu : D^k \to D$ is j-collapsible if for any $n \geq 1$, the set of all functions from $Y = \{y_1, \ldots, y_n\}$ to D can be composed from the j-adversary sets (for Y). By use of Lemma 6, the following theorem can be established.

Theorem 7 *If the operation μ is j-collapsible, then any constraint language Γ invariant under μ is j-collapsible.*

Applications. We can now derive the collapsibility of a variety of constraint languages. In particular, we now give a number of results which demonstrate that constraint languages invariant under an operation of some type are j-collapsible, for some j. These results are all obtained by demonstrating the j-collapsibility of the operations at hand, and then applying Theorem 7.

A *near-unanimity operation* is an operation $\mu : D^k \to D$ of rank $k \geq 3$ where for all tuples $\overline{t} \in D^k$, if there exists $i \in [k]$ and $d \in D$ such that $t_j = d$ for all $j \in [k] \setminus \{i\}$, then $\mu(\overline{t}) = d$. In words, if all but at most one of the arguments to μ agree, then μ returns the "nearly-unanimous" argument.

Theorem 8 *Any constraint language invariant under a near-unanimity operation of rank k is $(k-1)$-collapsible.*[1]

We say that a set function $f : (\mathcal{P}(D) \setminus \{\emptyset\}) \to D$ is a *set function with default* if there exists a non-empty subset $E \subseteq D$ such that the induced mapping $f_E : (\mathcal{P}(D) \setminus \{\emptyset\}) \to D$ defined by $f_E(S) = f(E \cup S)$ is surjective. Set functions with default provide another class of collapsible constraint languages.

[1]By this theorem, it can be deduced that any constraint language invariant under a *dual discriminator operation* (a particular type of rank 3 near-unanimity operation) is 2-collapsible. However, we can strengthen this implication and show that constraint languages invariant under a dual discriminator are 1-collapsible.

Theorem 9 *Any constraint language invariant under a set function with default is* 1*-collapsible.*

A *semilattice operation* is a binary operation $\oplus : D^2 \to D$ that is associative, commutative, and idempotent. Every semilattice operation naturally induces a set function. The semilattice operations can be divided into two classes which yield considerably different behaviors with respect to QCSP complexity. Define a semilattice operation $\oplus$ to be a *semilattice operation with unit* if there exists an element $u \in D$ such that $\oplus(u, d) = \oplus(d, u) = d$ for all $d \in D$, and a *semilattice operation without unit* otherwise. From Theorem 9, we can deduce the following corollary.

Corollary 10 *Any constraint language invariant under a semilattice operation with unit is* 1*-collapsible.*

It was shown in (Chen 2003b) that for any constraint language Γ invariant under a semilattice operation with unit, the problem $\mathsf{QCSP}(\Gamma)$ is tractable. On the other hand, it was demonstrated in (Chen 2003a) that there exist constraint languages Γ invariant under semilattice operations without unit, such that $\mathsf{QCSP}(\Gamma)$ is coNP-hard. By this hardness result and the results in this paper, it follows that demonstrating the j-collapsibility of a semilattice operation without unit is not possible, unless $\mathsf{P} = \mathsf{NP}$. In fact, we can demonstrate the non-j-collapsibility of such operations in a manner that is purely combinatorial, and does not make use of any complexity-theoretic assumptions.

Theorem 11 *For any semilattice operation $\oplus$ without unit, there exists a constraint language invariant under $\oplus$ that is not j-collapsible for any $j \geq 1$.*

Theorem 11 implies that Theorem 9 cannot be improved to address the class of all set functions. Precisely, Theorem 11 implies that for no $j \geq 1$ can it be proved that all set functions are j-collapsible.

Extended Minimality

As mentioned in the introduction, various forms of consistency have been developed for solving the CSP. In this section, we generalize a flavor of consistency called *k-minimality* to the QCSP setting, obtaining a new form of consistency for QCSPs which we call *extended k-minimality*. There are constraint languages Γ for which establishing some level of k-minimality and then checking for an empty constraint, is a decision procedure for $\mathsf{CSP}(\Gamma)$; such constraint languages are said to have *bounded relational width*. We will correspondingly say that a constraint language Γ has *bounded extended width* if establishing some level of extended k-minimality and then checking for an empty constraint, is a decision procedure for $\mathsf{QCSP}(\Gamma)$. Our main result in this section is that, under a mild assumption, bounded relational width implies bounded extended width! With this result in hand, one can take existing results showing constraint languages to have bounded relational width, and immediately infer the constraint languages to have bounded extended width.

Bounded relational width constraint languages are tractable in the CSP setting since any CSP can be converted in polynomial time into one that is k-minimal (for each fixed

k); the corresponding claim that can be made concerning extended k-minimality is that any QCSP *with a constant number of universal quantifiers* can be converted in polynomial time into one that is extended k-minimal (for each fixed k). As we will discuss in the next section, this latter fact gives rise to a two-phase decision procedure for $\mathsf{QCSP}(\Gamma)$ when Γ is both of bounded collapsibility and bounded extended width.

k-minimality. When $\mathcal{C}$ is a constraint network over variable set V and S is a subset of V, we define $\mathcal{C}_{|S}$ to be the constraint network $\{C_{|S} : C \in \mathcal{C}\}$, where $C_{|S}$ is the constraint such that $\mathsf{scope}(C_{|S}) = \mathsf{scope}(C) \cap S$ and $\mathsf{funs}(C_{|S}) = \mathsf{funs}(C)_{|\mathsf{scope}(C)\cap S}$. Intuitively, $\mathcal{C}_{|S}$ is the restriction of the entire constraint network $\mathcal{C}$ to S.

Definition 12 *(Bulatov 2002) A constraint network $\mathcal{C}$ over variable set V is* k-minimal *if the following conditions hold:*

- *for all subsets $S \subseteq V$ with $|S| = k$, there exists a constraint $C \in \mathcal{C}$ such that $S \subseteq \mathsf{scope}(C)$, and*
- *for all constraints $C \in \mathcal{C}$, functions $f \in \mathsf{funs}(C)$, and subsets $S \subseteq \mathsf{scope}(C)$ with $|S| \leq k$, the constraint network $\mathcal{C}_{|S}$ is satisfied by $f_{|S}$.*

Theorem 13 *(Bulatov 2002) For each fixed $k \geq 1$, there is a polynomial-time algorithm that converts an arbitrary constraint network $\mathcal{C}$ into a k-minimal constraint network that is equivalent to $\mathcal{C}$ (that is, has the same satisfying assignments as $\mathcal{C}$).*

There are $\mathsf{CSP}(\Gamma)$ problems solvable in polynomial-time by using the algorithms of Theorem 13. Following (Bulatov 2002), we say that a constraint language Γ has *relational width k* if for every instance $\mathcal{C}$ of $\mathsf{CSP}(\Gamma)$, the constraint network $\mathcal{C}$ is satisfiable if and only if a k-minimal constraint network equivalent to $\mathcal{C}$ has no empty constraint. For any constraint language Γ of relational width k, the problem $\mathsf{CSP}(\Gamma)$ is polynomial-time tractable: to decide satisfiability of an input constraint network, convert it into an equivalent k-minimal constraint network, and then report "satisfiable" if and only if the second constraint network has no empty constraint.

Previous work has identified the following classes of constraint languages having *bounded relational width*, that is, of relational width k for some $k \geq 1$.

Theorem 14 *(Jeavons, Cohen, & Cooper 1998) Any constraint language invariant under a near-unanimity operation of rank k has relational width k.*

Theorem 15 *(Dalmau & Pearson 1999) Any constraint language invariant under a set function has relational width* 1.

Extended k-minimality. We now generalize the condition of k-minimality to one called *extended k-minimality*. The first two conditions in the following definition can be thought of as analogs of the first two conditions of Definition 12, while the third condition has no analog in the CSP setting.

Definition 16 *A quantified formula ϕ with constraint network $\mathcal{C}$ is* extended k-minimal *if the following conditions hold:*

- *for all subsets $S \subseteq X$ with $|S| = k$, there exists a constraint $C \in \mathcal{C}$ such that $S \cup Y \subseteq \mathsf{scope}(C)$,*
- *for all constraints $C \in \mathcal{C}$, functions $f \in \mathsf{funs}(C)$, and subsets $S \subseteq \mathsf{scope}(C)$ with $|S \cap X| \leq k$ and $\mathsf{last}(S) \in X$, the constraint network $\mathcal{C}_{|S}$ is satisfied by $f_{|S}$, and*
- *for all constraints $C \in \mathcal{C}$, functions $f \in \mathsf{funs}(C)$, universally quantified variables $y \in \mathsf{scope}(C) \cap Y$, subsets $S \subseteq \mathsf{scope}(C)[< y]$, and domain elements $d \in D$, the extension of $f_{|S}$ mapping y to d is in $\mathsf{funs}(C)_{|S \cup \{y\}}$.*

Definition 16 is a smooth generalization of Definition 12 in the sense that a CSP is k-minimal if and only if it is extended k-minimal. (This is straightforward to verify.)

We have the following analog (indeed, generalization) of Theorem 13.

Theorem 17 *For fixed $j, k \geq 1$, there is a polynomial-time algorithm that converts a quantified formula ϕ with at most j universally quantified variables into an extended k-minimal quantified formula that is equivalent to ϕ (that is, has the same winning strategies as ϕ).*

We say that a constraint language Γ has *extended width k* if for every instance ϕ of $\mathsf{QCSP}(\Gamma)$, the quantified formula ϕ is true if and only if an extended k-minimal quantified formula ϕ' equivalent to ϕ has no empty constraint; when a constraint language Γ has extended width k for some $k \geq 1$, we say that it has *bounded extended width*. The following is our main result concerning extended k-minimality; it shows that one can automatically infer bounded extended width from bounded relational width.

Theorem 18 *Let μ be an idempotent operation. If every constraint language invariant under μ has relational width k, then every constraint language invariant under μ has extended width k.*

As a testament to the power of Theorem 18, we can immediately deduce the following corollaries from it and Theorems 14 and 15.

Corollary 19 *Any constraint language invariant under a near-unanimity operation of rank k has extended width k.*

Corollary 20 *Any constraint language invariant under an idempotent set function has extended width* 1.

Tractability

We say that a constraint language has *quantified width (j, k)* if it is j-collapsible and has extended width k; a constraint language has *bounded quantified width* if it has quantified width (j, k) for some $j, k \geq 1$ (equivalently, if it is of bounded collapsibility and bounded extended width). By considering Theorems 8, 9 along with Corollaries 19, 20, we can identify the following classes of constraint languages having bounded quantified width.

Theorem 21 *Any constraint language invariant under a near-unanimity operation of rank k has quantified width $(k-1, k)$.*

Theorem 22 *Any constraint language invariant under an idempotent set function with default has quantified width* $(1, 1)$.

For any constraint language with quantified width (j, k), the problem $\mathsf{QCSP}(\Gamma)$ is solvable in polynomial-time via a two-phase decision procedure. Given an input formula ϕ, the procedure is to (1) compute all j-collapsings of ϕ, and (2) for each of these j-collapsings, use the algorithm of Theorem 17 to compute an equivalent formula that is extended k-minimal, and check for an empty constraint. The formula is true if and only if no empty constraint is detected. Note that, for a fixed j, there are polynomially many j-collapsings of an input formula.

Theorem 23 *For any constraint language* Γ *with bounded quantified width, the problem* $\mathsf{QCSP}(\Gamma)$ *is decidable in polynomial-time. Moreover, all such problems* $\mathsf{QCSP}(\Gamma)$ *are decidable by (different* (j, k) *parameterizations of) the same algorithm.*

Theorem 23 allows us to derive, in a uniform fashion, a number of different tractability results.

Corollary 24 *For any constraint language* Γ *invariant under a near-unanimity operation, the problem* $\mathsf{QCSP}(\Gamma)$ *is decidable in polynomial-time.*

We remark that near-unanimity operations were shown to be tractable in (Chen 2003b), and dual discriminator operations were shown to be tractable in (Börner *et al.* 2003).

Corollary 25 *For any constraint language* Γ *invariant under an idempotent set function with default, the problem* $\mathsf{QCSP}(\Gamma)$ *is decidable in polynomial-time.*

The problems $\mathsf{QCSP}(\Gamma)$ identified by Corollary 25 constitute a new tractable subclass of the QCSP that has not been previously observed, although the tractability of semilattices with unit (which is implied by Corollary 25) was demonstrated in (Chen 2003b).

From Corollaries 24 and 25, we can derive the following.

Corollary 26 QUANTIFIED HORN SATISFIABILITY *and* QUANTIFIED 2-SATISFIABILITY *are decidable in polynomial-time.*

We can also show that semilattice operations with unit and near-unanimity operations can be combined to derive tractability results in a multi-sorted setting. The following theorem holds for a multi-sorted analog of bounded quantified width; for its statement, we adopt the terminology of (Bulatov & Jeavons 2003).

Theorem 27 *Suppose that* $\mathcal{A}$ *is a collection of idempotent algebras such that each algebra in* $\mathcal{A}$ *contains either a semilattice operation with unit or a near-unanimity operation. Any constraint language* Γ *invariant under* $\mathcal{A}$ *has bounded quantified width, and hence for such constraint languages* Γ *the problem* $\mathsf{QCSP}(\Gamma)$ *is decidable in polynomial-time.*

Acknowledgements. The author thanks Víctor Dalmau for useful discussions, and Carla Gomes, Riccardo Pucella, Bart Selman, and Vicky Weissman for helpful comments on drafts of this paper.

References

Aspvall, B.; Plass, M. F.; and Tarjan, R. E. 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* 8(3):121–123.

Börner, F.; Bulatov, A.; Krokhin, A.; and Jeavons, P. 2003. Quantified constraints: Algorithms and complexity. In *Computer Science Logic 2003*.

Bulatov, A., and Jeavons, P. 2003. An algebraic approach to multi-sorted constraints. In *CP 2003*.

Bulatov, A. 2002. A dichotomy theorem for constraints on a three-element set. In *Proceedings of 43rd IEEE Symposium on Foundations of Computer Science*, 649–658.

Chen, H. 2003a. Quantified constraint satisfaction and small commutative conservative operations. Cornell technical report.

Chen, H. 2003b. Quantified constraint satisfaction problems: Closure properties, complexity, and proof systems. Cornell technical report.

Dalmau, V., and Pearson, J. 1999. Closure functions and width 1 problems. In *CP 1999*, 159–173.

Dalmau, V.; Kolaitis, P. G.; and Vardi, M. Y. 2002. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Constraint Programming '02*, LNCS.

Dechter, R., and van Beek, P. 1996. Local and global relational consistency. *Theoretical Computer Science*.

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

Feder, T., and Vardi, M. Y. 1998. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.* 28(1):57–104.

Freuder, E. C. 1978. Synthesizing constraint expressions. *Commun. ACM* 21(11):958–966.

Freuder, E. C. 1982. A sufficient condition for backtrack-free search. *J. ACM* 29(1):24–32.

Freuder, E. C. 1985. A sufficient condition for backtrack-bounded search. *J. ACM* 32(4):755–761.

Jeavons, P.; Cohen, D.; and Cooper, M. 1998. Constraints, consistency, and closure. *Articial Intelligence* 101(1-2):251–265.

Jeavons, P. 1998. On the algebraic structure of combinatorial problems. *Theoretical Computer Science* 200:185–204.

Karpinski, M.; Büning, H. K.; and Schmitt, P. H. 1987. On the computational complexity of quantified horn clauses. In *CSL 1987*, 129–137.

P.G.Jeavons; D.A.Cohen; and M.Gyssens. 1997. Closure properties of constraints. *Journal of the ACM* 44:527–548.

van Beek, P., and Dechter, R. 1995. On the minimality and global consistency of row-convex constraint networks. *Journal of the ACM* 42(3):543–561.

van Beek, P., and Dechter, R. 1997. Constraint restrictiveness versus local and global consistency. *Journal of the ACM* 44(4):549 – 566.