Quantified Constraint Satisfaction and Bounded Treewidth

Hubie Chen[1]

Abstract. Because the constraint satisfaction problem (CSP) is in general intractable, restricted cases of the CSP that are polynomial-time tractable have been heavily sought after. One class of restrictions that has been studied are variable-based restrictions, which concern the interaction among variables. In this paper, we consider the quantified constraint satisfaction problem (QCSP), a framework more general than the CSP. We present a QCSP tractability result arising from variable-based restrictions by giving a polynomial time algorithm for certain classes of QCSP instances having bounded treewidth.

1 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 and a set of constraints, where each constraint consists of a tuple of variables paired with a relation containing permitted values for the variable tuple; the question is to decide whether or not there is an assignment to all of the variables satisfying all of the constraints. The CSP is in general NP-complete; consequently, researchers have devoted much effort to identifying restricted cases of the CSP that are polynomial-time tractable. By and large, the restrictions that have been identified and studied can be placed into one of two categories: *variable-based* restrictions, where the interaction among variables is restricted; and *constraint-based* restrictions, where the types of relations that may appear in constraints is restricted.

All of the variables in a CSP instance can be thought of as being implicitly existentially quantified. The quantified constraint satisfaction problem (QCSP) is a useful generalization of the CSP in which both universal and existential quantification of variables is permitted. 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. The higher expressiveness of the QCSP, however, 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 light of the utility of the QCSP framework, it is natural to consider whether or not any of the restrictions which ensure tractability for the CSP also imply tractability for the QCSP.[2] There have been some results along these lines for constraint-based restrictions: Aspvall, Plass, and Tarjan showed that QUANTIFIED 2-SATISFIABILITY is tractable [1], and Karpinski, Büning, and Schmitt showed that QUANTIFIED HORN SATISFIABILITY is tractable [10]. Constraint-based restrictions for CSPs over domains of size greater than two have been investigated by Börner, Bulatov, Krokhin, and Jeavons [3] and Chen [4]; they have provided some general criteria for tractability. On the other hand, there has not yet been, to the best of our knowledge, any detailed study of variable-based restrictions in the QCSP setting. In this paper, we initiate work in this direction by presenting a polynomial time algorithm for certain classes of QCSP instances having bounded treewidth.

Perspective on restrictions. To offer some perspective on the two forms of restrictions that we have mentioned, it is useful to consider a formulation of the CSP that is equivalent to the standard one, namely, the *relational homomorphism problem*. This problem is to decide, given as input a pair of relational structures $(\mathbf{A}, \mathbf{B})$, whether or not there is a relational homomorphism from $\mathbf{A}$ to $\mathbf{B}$. In this formulation, each relation of $\mathbf{A}$ contains tuples of variables that are constrained together, and the corresponding relation of $\mathbf{B}$ contains the allowable value tuples for those variable tuples. When the CSP is cast in this way, the two forms of restrictions can transparently be seen as two ends of a wide spectrum. Variable-based restrictions are those that arise from considering a restricted class of *left-hand side* structures $\mathbf{A}$, without any restriction on the *right-hand side* structures $\mathbf{B}$; constraint-based restrictions are those that arise from considering a restricted class of right-hand side structures $\mathbf{B}$, without any restriction on the left-hand side structures $\mathbf{A}$.

Bounded treewidth. The restriction of *bounded treewidth* has played a major role in the study of variable-based restrictions on the CSP. Bounded treewidth is a graph-theoretic condition that concerns the hypergraph having as vertices the variables of a CSP instance, and having as edges the sets of variables that are constrained together. Dechter and Pearl [6] and Freuder [8] showed that CSP instances of bounded treewidth are tractable. An alternative proof of this tractability result, which made use of expressibility in fragments of first order logic with finitely many variables, was given by Kolaitis and Vardi [11]. Later, Dalmau et al. [5], making use of connections among expressibility in Datalog, combinatorial pebble games, and consistency that were developed by Kolaitis and Vardi [12], gave another explanation for the tractability of bounded treewidth; they also showed that a natural expansion of the class of instances having bounded treewidth are tractable. The optimality of this result was recently demonstrated by Grohe [9], who proved that for CSP instances of bounded arity, any variable-based restriction that is tractable must

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853, USA. E-mail: hubes@cs.cornell.edu.

[2] Note that because the QCSP is a generalization of the CSP, negative results on CSP complexity transfer over to QCSP complexity: if a restricted version of the CSP is intractable, the analogous restricted version of the QCSP will also be intractable.

fall into the natural expansion studied by Dalmau et al. [5], under a standard complexity-theoretic assumption from parameterized complexity. In view of this progression of tractability results and the complementary result of Grohe, the restriction of bounded treewidth is an important and natural starting point for investigating the potential of variable-based restrictions on the QCSP.

Our contribution. This paper presents a polynomial time algorithm for QCSP instances having bounded treewidth, a constant number of quantifier alternations, and a domain of constant size. Note that the latter two restrictions are not very strong. Without the restriction of bounded treewidth, such QCSP instances are intractable: instances over a domain of size two having a constant number of quantifier alternations, are complete for the various levels of the polynomial hierarchy, a hierarchy of complexity classes which contains and generalizes NP and coNP.

It has been pointed out by Feder and Kolaitis [7] that the polynomial time tractability of such instances follows from Courcelle's theorem. In this paper, we give a relatively simple algorithm for these instances; our presentation is self-contained. The algorithm that we present makes use of novel techniques that are suited to the quantified setting. In particular, we introduce a generalization of the standard notion of constraint that we call the *choice constraint*. We demonstrate that the QCSP instances of interest can be converted into instances having choice constraints, and that in these instances, variables can be eliminated one by one without a super-polynomial increase in the size of the problem representation.

2 Preliminaries

Functions. When A and B are sets, we let $[A \rightarrow B]$ denote the set containing all functions from A to B. When f is a function from set A to set B and $A' \subseteq A$, we let $f|_{A'}$ denote the restriction of f to A'.

Quantified constraint satisfaction. A *domain* is a non-empty set. A *constraint* C (over variable set V and domain D) consists of a *scope*, denoted by $\mathsf{scope}(C)$, which is a subset of the variable set V; and, a set of functions $\mathsf{funs}(C)$, which is a subset of $[\mathsf{scope}(C) \rightarrow D]$. Such a constraint C is *satisfied* by a mapping $f : V \rightarrow D$ if $f|_{\mathsf{scope}(C)} \in \mathsf{funs}(C)$. A *constraint network* $\mathcal{C}$ is a finite set of constraints, all of which are over the same variable set V and domain D; it is satisfied by a mapping $f : V \rightarrow D$ if f satisfies all constraints $C \in \mathcal{C}$.

A *quantified formula* is an expression of the form $\phi = \forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{C}$, where each of the Y_i (and each of the X_i) are sets of variables; and, $\mathcal{C}$ is a constraint network containing constraints over variable set $Y_1 \cup X_1 \cup \ldots \cup Y_t \cup X_t$ and domain D. Note that, for ease of notation, we consider only quantified formulas where the outermost quantifier is universal, and the innermost quantifier is existential. We say that ϕ has $2t - 1$ quantifier alternations. For such a quantified formula ϕ, we let Y denote the set $\cup_{i=1}^{t} Y_i$, we let X denote the set $\cup_{i=1}^{t} X_i$, and we let V denote the set $X \cup Y$. A *strategy* for the quantified formula ϕ is a set of mappings $\{\sigma_x\}_{x \in X}$, where each variable $x \in X_i$ has associated with it a mapping $\sigma_x : [(Y_1 \cup \ldots \cup Y_i) \rightarrow D] \rightarrow D$. A strategy is a *winning strategy* if for all *adversary* functions $\tau : Y \rightarrow D$, the function $o : V \rightarrow D$ such that

- $o(y) = \tau(y)$ for all $y \in Y$, and
- $o(x) = \sigma_x(\tau|_{Y_1 \cup \ldots \cup Y_i})$ for $x \in X_i$

satisfies the constraint network $\mathcal{C}$. A quantified formula ϕ is true if there is a winning strategy for ϕ. The quantified constraint satisfaction problem is to decide, given as input a quantified formula, whether or not it is true.

Strings. When A is a set, we let A^* denote the set of all finite length strings over A, including the empty string ϵ; and, we let $A^{=i}$ denote the set of all length i strings over A. When $w \in A^*$ is a string of length k and $1 \leq i \leq k$, we use $w(i)$ to denote the ith character of w, and $w(\leq i)$ to denote the prefix of w of length i (that is, the string $w(1) \ldots w(i)$).

Trees. A *tree* is a pair (T, t) which has associated with it an *index set* A and an *object set* S, where T is a non-empty prefix-closed subset of A^*, and t is a mapping from

$$\mathsf{leaves}(T) \stackrel{\text{def}}{=} \{w \in T : \forall a \in A, wa \notin T\}$$

to the object set S. The elements of T are called the *nodes* of the tree (T, t); a node $w \in T$ is said to have depth $|w|$. A string $w' \in T$ is the *child* of a node $w \in T$ if there exists $a \in A$ such that $w' = wa$. When $w \in T$ is a node, we denote the set containing all children of w by $\mathsf{children}(T, w)$. When $w \in T$ is a node, we define $(T, t)[w]$ to be the subtree of (T, t) with w as the root node; formally, $(T, t)[w]$ is defined to be the tree (T', t') having the same index set and object set as (T, t), and where $T' = \{w' : ww' \in T\}$ and $t'(w') = t(ww')$.

We define equivalence of trees inductively, as follows. Two trees (T_1, t_1), (T_2, t_2) with index sets A_1, A_2 (respectively) and the same object set are considered to be equivalent if:

- $\mathsf{leaves}(T_1) = \mathsf{leaves}(T_2) = \{\epsilon\}$ and $t_1(\epsilon) = t_2(\epsilon)$, or
- $\mathsf{leaves}(T_1) \neq \{\epsilon\}$, $\mathsf{leaves}(T_2) \neq \{\epsilon\}$, and there exists a bijection $\beta : \mathsf{children}(T_1, \epsilon) \rightarrow \mathsf{children}(T_2, \epsilon)$ such that for all $c \in \mathsf{children}(T_1, \epsilon)$, the tree $(T_1, t_1)[c]$ is equivalent to the tree $(T_2, t_2)[\beta(c)]$.

Note that the children of a node are "unordered"; this is evidenced by the fact that an arbitrary bijection β is permitted in the given definition of equivalence.

Treewidth. For our purposes in this paper, a *free tree* is defined to be an acyclic, undirected graph. A vertex of a free tree is a *leaf* if it is of degree one or zero, that is, if it is part of at most one edge. We define a *tree decomposition* of a constraint network $\mathcal{C}$ over variable set V to be a free tree F such that:

- every vertex w of F has an associated label L_w that is a non-empty subset of V,
- for every constraint $C \in \mathcal{C}$, there is a vertex w of F such that $\mathsf{scope}(C) \subseteq L_w$, and
- for every variable $v \in V$, the set W of nodes w such that $v \in L_w$, is a sub-free tree of F.

The *width* of a tree decomposition is the maximum cardinality of a label in F minus 1. The *treewidth* of a constraint network $\mathcal{C}$ is the smallest integer e such that $\mathcal{C}$ has a tree decomposition of width e.

Statement of result. Having defined the basic terminology of this paper, we can formally state our main result, which is proved in the sections that follow.

Theorem 1 *Let $l, d, e \geq 1$ be any fixed constants. The following problem has a polynomial time algorithm:* Given a quantified formula ϕ with at most l quantifier alternations, over a domain of size at most d, and having a constraint network of treewidth at most e, decide whether or not ϕ is true.

3 Choice constraints

A constraint is typically defined to be a set of variables paired with a set of allowed values for those variables. Our algorithm makes use of a more general form of constraint, called the *choice constraint*, that is designed for the setting of quantified constraint satisfaction. Choice constraints allow us to define the notion of a *choice quantified formula*, a new form of quantified formula which is strictly more general than the notion of quantified formula we have already defined. We first give the intuition behind choice constraints, and then give the associated formal definitions.

A quantified formula $\forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{C}$ (with "standard" constraints) can be viewed as a game between two players, a *universal player* and an *existential player* with $2t$ stages. In the first stage, the universal player sets the variables in Y_1; in the second stage, the existential player sets the variables in X_1, and so forth; in the $(2i-1)$th stage, the universal player sets the variables in Y_i, and in the $2i$th stage, the existential player sets the variables in X_i. The existential player wins the game if the resulting assignment to the variables satisfies all of the constraints in $\mathcal{C}$; the formula is true if the existential player can always win.

A choice constraint that is part of a quantified formula with quantifier prefix $\forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t$ is a tree whose leaves are all of depth $2t$. Each leaf (of the tree of a choice constraint) contains a "standard" constraint. A quantified formula with choice constraints can also be viewed as a game between a universal player and an existential player with $2t$ stages, but in addition to setting the variables of the formula, the players of the game also traverse the trees of each choice constraint. In the first stage, the universal player sets the variables in Y_1, and in addition, for each of the trees, chooses a child of the root node. in the second stage, the existential player sets the variables in X_1, and in addition, for each of the trees, chooses a child of the node chosen by the universal player. The players continue setting variables and choosing children of the nodes chosen by the previous player. At the end of the game, one leaf has been chosen on each of the trees, and an assignment to all of the variables has been defined. The existential player wins the game if the resulting assignment satisfies, for all of the trees, the standard constraint associated with the chosen leaf of the tree.

In the remainder of this section, we make precise the intuitive ideas just discussed, and also define a join operation for choice constraints. Formally, a *choice constraint* H (over variable set V and domain D) consists of a *scope*, denoted by $\mathsf{scope}(H)$, which is a subset of the variable set V; and, a tree (T_H, t_H) with object set $[\mathsf{scope}(H) \to D]$ and index set denoted by A_H. For each of the choice constraints H that we consider, all leaves of the tree (T_H, t_H) will have the same depth.

Choice quantified formulas. A *choice quantified formula* is an expression of the form $\phi = \forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{H}$, where $\mathcal{H}$ is a set of choice constraints over variable set $Y_1 \cup X_1 \cup \ldots \cup Y_t \cup X_t$ and domain D, each of which has leaves of depth $2t$. As with quantified formulas, we say that ϕ has $2t-1$ quantifier alternations, we let Y denote the set $\cup_{i=1}^{t} Y_i$, X denote the set $\cup_{i=1}^{t} X_i$, and V denote the set $X \cup Y$. Note that we define the treewidth of the constraint network $\mathcal{H}$ in a manner identical to the definition of the treewidth for a constraint network $\mathcal{C}$.

A *strategy* for the choice quantified formula ϕ is a pair of mapping sets $(\{\sigma_x\}_{x \in X}, \{\sigma_{H,i}\}_{H \in \mathcal{H}, i \in \{1,\ldots,t\}})$, where:

- each variable $x \in X_i$ has associated with it a mapping

$$\sigma_x : [(Y_1 \cup \ldots \cup Y_i) \to D] \times \prod_{H \in \mathcal{H}} A_H^{=i} \to D$$

and

- for every choice constraint $H \in \mathcal{H}$ and integer $i \in \{1, \ldots, t\}$, there is a mapping

$$\sigma_{H,i} : [(Y_1 \cup \ldots \cup Y_i) \to D] \times \prod_{H \in \mathcal{H}} A_H^{=i} \to A_H.$$

The mappings σ_x for $x \in X_i$ tell how to set the variables in X_i in the $2i$th stage of the game, given all of the previous moves of the universal player. Likewise, the mappings $\sigma_{H,i}$ tell how to choose, in the choice constraint H, a child of the node previously chosen by the universal player (given all of the previous moves of the universal player).

An *adversary* for the choice quantified formula ϕ is a pair $(\tau : Y \to D, \{\tau_H\}_{H \in \mathcal{H}})$ where $\{\tau_H\}_{H \in \mathcal{H}}$ is a set containing a string $\tau_H \in A_H^{=t}$ for each $H \in \mathcal{H}$.

An adversary $(\tau : Y \to D, \{\tau_H\}_{H \in \mathcal{H}})$ is *appropriate* relative to the choice quantified formula ϕ and the strategy $(\{\sigma_x\}_{x \in X}, \{\sigma_{H,i}\}_{H \in \mathcal{H}, i \in \{1,\ldots,t\}})$ when for all $H \in \mathcal{H}$ and for all $j \in \{1, \ldots, t\}$, the following property holds: if

$$\tau_H(1)\sigma_{H,1}(\tau|_{Y_1}, \prod_{H \in \mathcal{H}} \tau_H(\leq 1)) \ldots$$

$$\tau_H(j-1)\sigma_{H,j-1}(\tau|_{Y_1 \cup \ldots \cup Y_{j-1}}, \prod_{H \in \mathcal{H}} \tau_H(\leq j-1))$$

is in T_H, then

$$\tau_H(1)\sigma_{H,1}(\tau|_{Y_1}, \prod_{H \in \mathcal{H}} \tau_H(\leq 1)) \ldots$$

$$\tau_H(j-1)\sigma_{H,j-1}(\tau|_{Y_1 \cup \ldots \cup Y_{j-1}}, \prod_{H \in \mathcal{H}} \tau_H(\leq j-1))\tau_H(j)$$

is also in T_H. Intuitively, an adversary is appropriate relative to a strategy if (for all choice constraints) it properly picks a child of the node picked by the strategy, under the assumption that the strategy picked a node on the tree.

A strategy $\Sigma = (\{\sigma_x\}_{x \in X}, \{\sigma_{H,i}\}_{H \in \mathcal{H}, i \in \{1,\ldots,t\}})$ for the choice quantified formula ϕ is *winning* if for all adversaries $(\tau : Y \to D, \{\tau_H\}_{H \in \mathcal{H}})$ appropriate relative to ϕ and Σ, it holds that for each choice constraint $H \in \mathcal{H}$,

- the string

$$o_H = \tau_H(1)\sigma_{H,1}(\tau|_{Y_1}, \prod_{H \in \mathcal{H}} \tau_H(\leq 1)) \ldots$$

$$\tau_H(t)\sigma_{H,t}(\tau|_{Y_1 \cup \ldots \cup Y_t}, \prod_{H \in \mathcal{H}} \tau_H(\leq t))$$

is in T_H, and

- $o|_{\mathsf{scope}(H)} \in t_H(o_H)$, where $o : V \to D$ is defined as
  - $o(y) = \tau(y)$ for all $y \in Y$, and
  - $o(x) = \sigma_x(\tau|_{Y_1 \cup \ldots \cup Y_i}, \prod_{H \in \mathcal{H}} \tau_H(\leq i))$ for $x \in X_i$.

The first condition given is a form of appropriateness for the strategy Σ; it requires that a winning strategy "stay on the tree" of each choice constraint. The function $o : V \to D$ of the second condition gives the final assignment to all of the variables that has been defined throughout the play of the game; the second condition requires that, for every choice constraint H, the function o satisfies the label on the leaf o_H chosen by the two players. As with quantified formulas, we consider a choice quantified formula ϕ to be true if there is a winning strategy for ϕ.

Joins. We define the join of two choice constraints $H_1, H_2 \in \mathcal{H}$ in a choice quantified formula $\forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{H}$ to be the choice constraint H defined by

$$\begin{aligned}
\mathsf{scope}(H) &= \mathsf{scope}(H_1) \cup \mathsf{scope}(H_2) \\
A_H &= A_{H_1} \times A_{H_2} \\
T_H &= \{w_1 \times w_2 : w_1 \in T_{H_1}, w_2 \in T_{H_2}, \\
&\quad\ |w_1| = |w_2|\} \\
t_H(w_1 \times w_2) &= \{f \in [\mathsf{scope}(H) \to D] : \\
&\quad\ f|_{\mathsf{scope}(H_1)} \in t_{H_1}(w_1), \\
&\quad\ f|_{\mathsf{scope}(H_2)} \in t_{H_2}(w_2)\}
\end{aligned}$$

where the product $w_1 \times w_2$ of two strings w_1, w_2 each of length k is defined as $(w_1(1), w_2(1)) \ldots (w_1(k), w_2(k)))$. It is straightforward to verify that if two choice constraints are replaced by their join in a choice quantified formula, the new choice quantified formula is equivalent to the original in the sense that it has exactly the same winning strategies.

4 Algorithm

In this section, we give a polynomial time algorithm for deciding the truth of quantified formulas where the number of quantifier alternations, the size of the domain, and the treewidth of the constraint network are all bounded above by constants. Let ϕ be the input quantified formula, and let the free tree F with vertex set W be a tree decomposition of the constraint network of ϕ with width obeying the treewidth bound. Note that (for any fixed e) a tree decomposition of treewidth e, when it exists, can be computed in polynomial time (see for instance [2]).

Import. The first step of the algorithm is to import the input quantified formula $\phi = \forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{C}$ into the framework of choice constraints by converting ϕ to a choice quantified formula. This is done by converting each constraint $C \in \mathcal{C}$ into a choice constraint H defined by:

$$\begin{aligned}
\mathsf{scope}(H) &= \mathsf{scope}(C) \\
A_H &= \{a\} \\
T_H &= \{\epsilon, a, \ldots, a^{2t}\} \\
t_H(a^{2t}) &= \mathsf{funs}(C)
\end{aligned}$$

Let ϕ' denote the choice quantified formula $\phi = \forall Y_1 \exists X_1 \ldots \forall Y_t \exists X_t \mathcal{H}$, where $\mathcal{H}$ contains all choice constraints H obtained in the above manner. It is straightforward to verify that ϕ is true if and only if ϕ' is true. This is because the choice constraints created act just like standard constraints: each non-leaf node of the tree has exactly one child, so the two players do not have any choice but to ultimately choose the single leaf.

Before describing the main step of the algorithm, *variable elimination*, we discuss a sub-routine, *trimming*, that will be invoked by this main step.

Trimming. Because the treewidth of the constraint network we are considering is bounded above by a constant, the size of the scope of each choice constraint is bounded above by a constant. For a fixed scope S, the number of different leaf types, that is, the size of the set $[S \to D]$, is also bounded above by a constant, as the domain size $|D|$ is bounded above by a constant. Observe that in the tree (T_H, t_H) of any choice constraint H, if the subtrees below two nodes w_1, w_2 at the same depth and sharing the same parent are equivalent (that is, if $(T_H, t_H)[w_1]$ and $(T_H, t_H)[w_2]$ are equivalent trees), then one of the trees $(T_H, t_H)[w_i]$ can be cut out from (T_H, t_H). This cutting preserves the truth of the formula; intuitively, this is because if a player has moved to the common parent of w_1 and w_2, moving to w_1 and moving to w_2 are equivalent for the next player (in terms of winning the game). Based on this observation, it is natural to define a normal form for choice constraints: we say that a choice constraint H is in *normal form* if for no two distinct nodes w_1, w_2 sharing the same parent is it the case that $(T_H, t_H)[w_1]$ and $(T_H, t_H)[w_2]$ are equivalent.

Given all of the above facts, along with the fact that the number of quantifier alternations is bounded, it is simple to prove that (for a fixed scope S) the number of choice constraints (with scope S) in normal form that can appear in formulas obeying the assumed constant bounds, is itself a constant. The proof is by induction on the height of the tree of the choice constraints: for each height, the number of trees (in normal form) having the height is bounded above by a constant. The base case (height zero) follows from the fact that the size of the set $[S \to D]$ is bounded above by a constant. The induction case follows from the fact that a tree of height h consists of a root node with children that are trees of height $h - 1$; because we may restrict attention to trees that are in normal form, each tree of height h is determined by a subset of the set of all trees of height $h - 1$.

We call the process of converting a choice constraint into one that is in normal form *trimming*, as this process involves cutting out "repeated" branches of choice constraint trees. We note that in both places where trimming is invoked by variable elimination, it can be verified that the trees to be trimmed will be of constant size, and hence can be trimmed in constant time.

We now describe the main step of the algorithm, which eliminates one variable of the choice quantified formula in a way that preserves the truth of the formula. This step is repeatedly applied to the input formula until all variables have been eliminated.

Variable elimination. We pick a variable to eliminate in the following way. We assume that for every leaf w of the free tree F, there is a variable v in L_w not contained in $L_{w'}$ for any vertex $w' \in W \setminus \{w\}$ (if not, for any w' adjacent to the leaf w, we have $L_w \subseteq L_{w'}$ and w can be eliminated from F to obtain a smaller tree decomposition). By assumption, then, we can pick a leaf w of F and a variable $v \in L_w$ not appearing in the labels of any other node. We eliminate the variable v, as follows.

We join together all choice constraints H such that v is contained in the scope of H, by continually performing the following until there is (at most) one choice constraint H with scope containing v: take any pair of constraints with scope containing v, and replace the pair with the join of the pair, trimming the resulting join if necessary, Notice that each of these "join and trim" steps can be carried out in constant time; by the definition of treewidth and the choice of variable v, all of the intermediate constraints generated have scope contained in the set L_w, and hence bounded above by a constant. The untrimmed joins are hence joins of choice constraints having constant size, and so have constant size themselves. Notice also that, by our choice of v, after this joining has been completed, F remains a tree decomposition for the constraint network.

Now we may assume that there is one choice constraint H containing v. Suppose that v is an existentially quantified variable contained in X_i. We replace H with the choice constraint H' defined as follows:

$$\begin{aligned}
\mathsf{scope}(H') &= \mathsf{scope}(H) \setminus \{v\} \\
A_{H'} &= A_H \cup (A_H \times D) \\
T_{H'} &= \{w[2i/d] : w \in T_H, d \in D\} \\
t_{H'}(w[2i/d]) &= \{f \in [\mathsf{scope}(H') \to D] : \\
&\qquad f[v \to d] \in \mathsf{scope}(H)\}
\end{aligned}$$

Here, $w[k/d]$ denotes the string equal to w except where the kth character $w(k)$ of w (if it exists) is replaced with $(w(k), d)$, that is, the string $w(1) \ldots w(k-1)(w(k), d)w(k+1) \ldots w(|w|)$. And, $f[v \to d]$ denotes the extension of f mapping v to d. Observe that the choice constraint H' is defined in such a way that the variable v does not appear in the scope of H', but rather, the variable v is (implicitly) set by the existential player simply by the choice of branch in the $2i$th stage of the game.

In the case that v is a universally quantified variable contained in Y_i, we replace H with a choice constraint H' defined analogously:

$$\begin{aligned}
\mathsf{scope}(H') &= \mathsf{scope}(H) \setminus \{v\} \\
A_{H'} &= A_H \cup (A_H \times D) \\
T_{H'} &= \{w[2i-1/d] : w \in T_H, d \in D\} \\
t_{H'}(w[2i-1/d]) &= \{f \in [\mathsf{scope}(H') \to D] : \\
&\qquad f[v \to d] \in \mathsf{scope}(H)\}
\end{aligned}$$

In both cases, after the choice constraint H is replaced with the choice constraint H', trimming is applied to H'. The result is a formula which can be verified to be true if and only if the original was true, and has one less variable than the original. We indicate how this can be proved formally. Suppose that v is an existentially quantified variable contained in X_i, and that H' is the choice constraint obtained from H as described above–where the $2i$th level of the choice constraint tree is expanded. (In the case that v is a universally quantified variable, the proof idea is similar.) Let $\mathcal{H}_1$ denote the original constraint network (containing H), and let $\mathcal{H}_2$ denote the new constraint network (containing H'). Suppose that

$$(\{\sigma_x\}_{x \in X}, \{\sigma_{H_1,i}\}_{H_1 \in \mathcal{H}_1, j \in \{1,\ldots,t\}})$$

is a winning strategy for the original formula. Define $\rho_{H',i}$ so that $\rho_{H',i}(a) = (\sigma_{H,i}(a), \sigma_v(a))$ for all $a \in [(Y_1 \cup \ldots \cup Y_i) \to D] \times \prod_{H \in \mathcal{H}_1} A_H^{=i}$; define $\rho_{H',j}$ to be equal to $\sigma_{H',j}$ for all $j \in \{1, \ldots, t\} \setminus \{i\}$; and, define $\rho_{H_2,j}$ to be equal to $\sigma_{H_2,j}$ for all $H_2 \in \mathcal{H}_2 \setminus \{H'\}$ and $j \in \{1, \ldots, t\}$. It can be verified that

$$(\{\sigma_x\}_{x \in X \setminus \{v\}}, \{\rho_{H_2,j}\}_{H_2 \in \mathcal{H}_2, j \in \{1,\ldots,t\}})$$

is a winning strategy for the new formula.

Now let

$$(\{\rho_x\}_{x \in X \setminus \{v\}}, \{\rho_{H_2,j}\}_{H_2 \in \mathcal{H}_2, j \in \{1,\ldots,t\}})$$

be a winning strategy for the new formula. Define σ_v so that $\sigma_v(a) = \pi_2(\rho_{H',i}(a))$ for all $a \in [(Y_1 \cup \ldots \cup Y_i) \to D] \times \prod_{H \in \mathcal{H}_2} A_H^{=i}$ (where π_k projects a pair onto its kth coordinate), and define σ_x to be equal to ρ_x for all $x \in X \setminus \{v\}$. Define $\sigma_{H,i}$ so that $\sigma_{H,i}(a) = \pi_1(\rho_{H',i}(a))$ for all $a \in [(Y_1 \cup \ldots \cup Y_i) \to D] \times \prod_{H \in \mathcal{H}_2} A_H^{=i}$; define $\sigma_{H,j}$ to be equal to $\rho_{H,j}$ for all $j \in \{1, \ldots, t\} \setminus \{i\}$; and, define $\sigma_{H_1,j}$ to be equal to $\rho_{H_1,j}$ for all $H_1 \in \mathcal{H}_1 \setminus \{H\}$ and $j \in \{1, \ldots, t\}$. It can be verified that

$$(\{\sigma_x\}_{x \in X}, \{\sigma_{H_1,i}\}_{H_1 \in \mathcal{H}_1, j \in \{1,\ldots,t\}})$$

is a winning strategy for the original formula.

Acknowledgements. The author thanks Víctor Dalmau for helpful discussions, and Riccardo Pucella for comments on a draft of this paper.

REFERENCES

[1] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan, 'A linear-time algorithm for testing the truth of certain quantified boolean formulas', *Information Processing Letters*, **8**(3), 121–123, (1979).
[2] Hans L. Bodlaender, 'A linear time algorithm for finding tree-decompositions of small treewidth', *SIAM Journal on Computing*, **25**, 1305–1317, (1996).
[3] F. Börner, A. Bulatov, A. Krokhin, and P. Jeavons, 'Quantified constraints: Algorithms and complexity', in *Computer Science Logic 2003*, (2003).
[4] Hubie Chen, 'Collapsibility and consistency in quantified constraint satisfaction', in *AAAI*, (2004).
[5] Victor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi, 'Constraint satisfaction, bounded treewidth, and finite-variable logics', in *Constraint Programming '02*, LNCS, (2002).
[6] Rina Dechter and Judea Pearl, 'Tree clustering for constraint networks', *Artificial Intelligence*, 353–366, (1989).
[7] Tomás Feder and Phokion Kolaitis, 'Closures and dichotomies for quantified constraints'. Manuscript.
[8] Eugene Freuder, 'Complexity of k-tree structured constraint satisfaction problems', in *AAAI-90*, (1990).
[9] Martin Grohe, 'The complexity of homomorphism and constraint satisfaction problems seen from the other side', in *FOCS 2003*, pp. 552–561, (2003).
[10] Marek Karpinski, Hans Kleine Büning, and Peter H. Schmitt, 'On the computational complexity of quantified horn clauses', in *CSL 1987*, pp. 129–137, (1987).
[11] P. Kolaitis and M. Vardi, 'Conjunctive-query containment and constraint satisfaction', *Journal of Computer and System Sciences*, 302–332, (2000).
[12] P. Kolaitis and M. Vardi, 'A game-theoretic approach to constraint satisfaction', in *AAAI-00*, pp. 175–181, (2000).