Journal of Computer and System Sciences 82 (2016) 347–356

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Conservative constraint satisfaction re-revisited

Andrei A. Bulatov [1]

School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby BC, V5A 1S6, Canada

ARTICLE INFO

Article history:
Received 1 July 2014
Received in revised form 12 December 2014
Accepted 9 July 2015
Available online 17 August 2015

Keywords:
Constraint satisfaction problem
Complexity
Dichotomy
Algebraic approach

ABSTRACT

Conservative constraint satisfaction problems (CSPs) constitute an important particular case of the general CSP, in which the allowed values of each variable can be restricted in an arbitrary way. Problems of this type are well studied for graph homomorphisms. A dichotomy theorem characterizing conservative CSPs solvable in polynomial time and proving that the remaining ones are NP-complete was proved by Bulatov (2003) in [4]. Its proof, however, is quite long and technical. A shorter proof of this result based on the absorbing subuniverses technique was suggested by Barto (2011) in [1]. In this paper we give a short elementary proof of the dichotomy theorem for conservative CSPs.

1. Introduction

In a constraint satisfaction problem (CSP) the aim is to find an assignment of values to a given set of variables, subject to specified constraints. The CSP is known to be NP-complete in general. However, certain restrictions on the form of the allowed constraints can lead to problems solvable in polynomial time. Such restrictions are usually imposed by specifying a constraint language, that is, a set of relations that are allowed to be used as constraints. A principal research direction aims to distinguish those constraint languages that give rise to CSPs solvable in polynomial time from those that do not. The dichotomy conjecture [14] suggests that every constraint language gives rise to a CSP that is either solvable in polynomial time or is NP-complete. The dichotomy conjecture is confirmed in a variety of particular cases [1–5,16,23], but the general problem remains open.

One of the important versions of the CSP is often referred to as the conservative or list CSP. In a CSP of this type the set of values for each individual variable can be restricted arbitrarily. Restrictions of this type can be studied by considering those constraint languages which contain all possible unary constraints; such languages are also called conservative. Conservative CSPs have been intensively studied for languages consisting of only one binary symmetric relation, that is, graphs; in this case CSP is equivalent to the graph homomorphism problem [11–13,16,21].

In [2,4] the dichotomy conjecture was confirmed for conservative CSPs. However, the proof given in [2,4] is quite long and technical, which prompted attempts to find a simpler argument. In [1] Barto gave a simpler proof using the absorbing subuniverses techniques. In the present paper we give another, more elementary, proof that applies the reduction suggested in [22].

As in the majority of dichotomy results the solution algorithm and the proofs heavily use the algebraic approach to the CSP developed in [6,7,20,18]. This approach relates a constraint language to a collection of polymorphisms of the language, that is, operations on the same set that preserves all the relations from the language, and uses polymorphisms of specific types to identify constraint languages solvable in polynomial time. For example, to characterize CSPs on a 2-element set

E-mail address: abulatov@sfu.ca.

[1] This research is supported by an NSERC Discovery Grant.

http://dx.doi.org/10.1016/j.jcss.2015.07.004
0022-0000/

solvable in polynomial time [23] it suffices to consider only 4 types of operations on a 2-element set: constant, semilattice (conjunction and disjunction), majority ($(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$), and affine ($x - y + z$). The same types of operations characterize the complexity of conservative CSPs, except that constant operations cannot be polymorphisms of conservative languages. In a simplified form the main result we prove is

Theorem 1.1. *(See [2,4].) Let Γ be a constraint language on a set A. The conservative CSP using relations from Γ can be solved in polynomial time if for any 2-element subset $\{a, b\} \subseteq A$ there is an operation f on A, a polymorphism of Γ, such that f on $\{a, b\}$ is either a semilattice operation, or a majority operation, or an affine operation. Otherwise this CSP is NP-complete.*

We give a new nearly complete proof of Theorem 1.1. The only statements we reuse in this paper are Proposition 2.2 that we borrow from [2] and the results of Section 4.2.

The problem of verifying the conditions of Theorem 1.1, the so-called Metaproblem, has been considered in [2]. Specifically, if A is fixed then, given a constraint language Γ, the tractability of the conservative CSP using relations from Γ can be checked in polynomial time. If A is not fixed and constraint language is given through its polymorphisms, the Metaproblem is also polynomial time (it also follows from [15]). Finally, the complexity of the Metaproblem is open if the problem is given by a set A and a constraint language Γ on it.

Outline of the proof As well as in [2], the solution algorithm and the proof rely upon a classification of pairs of elements of the base set A with respect to the constraint language Γ into three types, semilattice, majority and affine (see, the next section). Using this classification we define affine-semilattice (as-)components of A as minimal subsets of A that do not separate semilattice and affine pairs.

The main step of the algorithm is a reduction of a CSP instance to a smaller instance, that is, one having smaller domains, sets of possible values for the variables. The base case of this chain of reductions is an instance, in which all the domains have no semilattice pair. In this case the problem can be solved using the algorithm from [9], or the algorithm from [17].

If there are domains in an instance that contain proper as-components, then we use the as-component exclusion reduction from Section 4.1. This reduction relies on the fact, Lemma 3.6, that in this case we either can restrict the problem to certain as-components (thus making it smaller) and find a solution this way, or we can find an as-component in one of the domains such that no solution of the original problem takes a value from it. In the latter case we can remove this component, and so again reduce the problem to a smaller one.

If all domains are as-components themselves, we show that we can employ the rather sophisticated reduction from [22] that reduces the size of the domains containing semilattice pairs, see Section 4.2.

2. Definitions and preliminaries

2.1. Constraint satisfaction problems and algebra

By $[n]$ we denote the set $\{1, \dots, n\}$. Let $A_1, \dots, A_n$ be sets, any element of $A_1 \times \dots \times A_n$ is an (n-ary) tuple. Tuples will be denoted in boldface, say, $\mathbf{a}$, and the ith component of $\mathbf{a}$ will be referred to as $\mathbf{a}[i]$. An n-ary relation over $A_1, \dots, A_n$ is any set of tuples over these sets. For a set $I = \{i_1, \dots, i_k\} \subseteq [n]$, a tuple $\mathbf{a} \in A_1 \times \dots \times A_n$, and a relation $R \subseteq A_1 \times \dots \times A_n$, by $\mathrm{pr}_I \mathbf{a}$ we denote the tuple $(\mathbf{a}[i_1], \dots, \mathbf{a}[i_k])$, the *projection* of $\mathbf{a}$ on I, and $\mathrm{pr}_I R = \{\mathrm{pr}_I \mathbf{b} \mid \mathbf{b} \in R\}$ denotes the projection of R on I. Relation R is said to be a *subdirect product* of $A_1, \dots, A_n$ if $\mathrm{pr}_i R = A_i$ for all $i \in [n]$. Let $I \subseteq [n]$. For $\mathbf{a} \in \mathrm{pr}_I R$ and $\mathbf{b} \in \mathrm{pr}_{[n]-I} R$ by $(\mathbf{a}, \mathbf{b})$ we denote the tuple $\mathbf{c}$ such that $\mathbf{c}[i] = \mathbf{a}[i]$ if $i \in I$ and $\mathbf{c}[i] = \mathbf{b}[i]$ otherwise.

Let $\mathfrak{A}$ be a collection of finite sets (in this paper we assume $\mathfrak{A}$ to be finite as well). A *constraint satisfaction problem* over $\mathfrak{A}$ is a triple $(V, \delta, \mathcal{C})$, where V is a (finite) set of *variables*, δ is a *domain function*, $\delta : V \to \mathfrak{A}$ assigning a domain of values to every variable, and $\mathcal{C}$ is a set of *constraints*. Every constraint is a pair $\langle \mathbf{s}, R \rangle$, where $\mathbf{s} = (v_1, \dots, v_k)$ is a sequence of variables from V (possibly with repetitions) called the *constraint scope*, and R is a relation over $\delta(v_1) \times \dots \times \delta(v_k)$ called the *constraint relation*. A mapping $\varphi : V \to \bigcup \mathfrak{A}$ that maps every variable v to its domain $\delta(v)$ is called a *solution* if for every $\langle \mathbf{s}, R \rangle \in \mathcal{C}$ we have $\varphi(\mathbf{s}) \in R$.

Let $W \subseteq V$. A *partial solution* of $\mathcal{P}$ on W is a mapping $\varphi : W \to \bigcup \mathfrak{A}$ such that for every constraint $\langle \mathbf{s}, R \rangle \in \mathcal{C}$, $\mathbf{s} = (v_1, \dots, v_k)$, we have $\varphi(\mathbf{s}') \in \mathrm{pr}_I R$, where $I = \{i_1, \dots, i_\ell\}$ is the set of indices i_s from $[k]$ such that $v_{i_s} \in W$, and $\mathbf{s}' = (v_{i_1}, \dots, v_{i_\ell})$. The set of all partial solutions on set W is denoted by S_W. Problem $\mathcal{P}$ is said to be *3-minimal* if it contains a constraint $\langle W, S_W \rangle$ for every 3-element $W \subseteq V$, and for any $W_1, W_2 \subseteq V$ such that $|W_1| = |W_2| = 3$ and $|W_1 \cap W_2| = 2$, $\mathrm{pr}_{W_1 \cap W_2} S_W = \mathrm{pr}_{W_1 \cap W_2} S_{W_1} \cap \mathrm{pr}_{W_1 \cap W_2} S_{W_2}$. There are standard polynomial time *propagation* algorithms (see, e.g. [10]) to convert any CSP to an equivalent, that is, having the same solutions, 3-minimal CSP.

An introduction into universal algebra and the algebraic approach to CSP can be found in [8,6,7,2]. Here we only mention several key points. For an algebra $\mathbb{A}$ its universe will be denoted by A. Let $\mathfrak{A}$ be a finite collection of finite similar algebras. For a basic or term operation f of the class $\mathfrak{A}$ by $f^{\mathbb{A}}$, $\mathbb{A} \in \mathfrak{A}$, we denote the interpretation of f in $\mathbb{A}$. Let $\mathbb{A}_1, \dots, \mathbb{A}_k \in \mathfrak{A}$. A relation $R \subseteq A_1 \times \dots \times A_k$ is a *subalgebra* of the direct product $\mathbb{A}_1 \times \dots \times \mathbb{A}_k$, denoted $R \leq \mathbb{A}_1 \times \dots \times \mathbb{A}_k$, if for any basic operation f (say, it is n-ary) of $\mathfrak{A}$ and any $\mathbf{a}_1, \dots, \mathbf{a}_n \in R$ the tuple $f(\mathbf{a}_1, \dots, \mathbf{a}_n) = (f^{\mathbb{A}_1}(\mathbf{a}_1[1], \dots, \mathbf{a}_n[1]), \dots, f^{\mathbb{A}_k}(\mathbf{a}_1[k], \dots, \mathbf{a}_n[k]))$ belongs to R. In this case f is also said to be a *polymorphism* of R.