Constructing Constraints

Peter Jeavons

Department of Computer Science, Royal Holloway, University of London, UK
e-mail: `p.jeavons@dcs.rhbnc.ac.uk`

Abstract. It is well-known that there is a trade-off between the expressive power of a constraint language and the tractability of the problems it can express. But how can you determine the expressive power of a given constraint language, and how can you tell if problems expressed in that language are tractable? In this paper we discuss some general approaches to these questions.
We show that for languages over a finite domain the concept of an 'indicator problem' gives a universal construction for any constraint within the expressive power of a language. We also discuss the fact that all known tractable languages over finite domains are characterised by the presence of a particular solution to a corresponding indicator problem, and raise the question of whether this is a universal property of tractable languages.

1 What is a constraint?

A constraint is a way of specifying that a certain relationship must hold between the values taken by certain variables. There are very few "textbook" definitions of this concept (because there are very few textbooks), but the definition given in [22] is that a constraint is a set of *labelings*, and a labeling is a set of variable-value pairs. For example, the constraint that says that the variables X and Y must be assigned different values from the set $\{R, G, B\}$ would be expressed by the following set of labelings

$$\{\{(X,R),(Y,G)\},\{(X,R),(Y,B)\},\{(X,G),(Y,R)\},\\ \{(X,G),(Y,B)\},\{(X,B),(Y,R)\},\{(X,B),(Y,G)\}\}.$$

This interpretation of the notion of constraint is convenient for some types of analysis, but for our purposes, it is important to separate out more clearly two aspects of a constraint which are rather mixed together in this definition. These two aspects are the *relation* which must hold between the values, and the particular variables over which that relation must hold. We therefore prefer to use the following definition of constraints (which is similar to the definition used by many authors, see, for example, [1, 4]).

Definition 1. A constraint C is a pair (s, R), where s is a tuple of variables of length m, called the *constraint scope*, and R is a relation of arity m, called the *constraint relation.*

Note that a relation of arity m is simply a subset of the set of all m-tuples of elements from some set, say D, called the *domain*. In this paper, tuples will be written in the form $\langle d_1, d_2, \ldots, d_m \rangle$, and the set of all tuples of length m will be denoted D^m. The tuples in the constraint relation R indicate the allowed combinations of simultaneous values for the corresponding variables in the scope s. The length of these tuples, m, 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.

Using this definition, the constraint that says that the variables X and Y must be assigned different values from the set $\{R, G, B\}$ would be expressed by the pair

$$(\langle X, Y \rangle,\ \{\langle R, G \rangle, \langle R, B \rangle, \langle G, R \rangle, \langle G, B \rangle, \langle B, R \rangle, \langle B, G \rangle\}).$$

Alternatively, we might specify the constraint relation implicitly, and write this constraint as

$$(\langle X, Y \rangle,\ \{\langle c_1, c_2 \rangle \mid c_1, c_2 \in \{R, G, B\}, c_1 \neq c_2\}).$$

By using implicit specifications of relations in this way we can easily define constraints over both finite and infinite sets.

Satisfying a constraint means choosing a tuple of values for the variables in the scope that is a member of the constraint relation.

A constraint satisfaction problem, $\mathbf{P}$, is given by a set of variables, V, a domain of possible values, D, and a set of constraints. A solution to $\mathbf{P}$ is a mapping from V to D whose restriction to each individual constraint scope satisfies that constraint.

2 What is a constraint language?

Separating out the notions of the constraint *scope* and the constraint *relation* by using the definition above allows us to study each of these aspects independently. For example, the *scopes* of the constraints determine the structure of the constraint satisfaction problem, or, in other words, the way that the constraints overlap. The structure associated with a given problem is often described as the "constraint graph", or "constraint hypergraph", associated with a given problem. By imposing restrictions on the possible scopes of the constraints we can define classes of problems with restricted structures. There are many results in the literature about the tractability of constraint satisfaction problems that have certain forms of restricted structure, such as a tree-structure [6, 20], or some generalisation of a tree-structure [3, 7, 9, 20].

On the other hand, the constraint *relations* determine the kinds of constraints that are involved in our constraint satisfaction problem. By imposing restrictions on the possible constraint relations we are allowed to use we can define the notion of a "constraint language". For example, if we take all relations that can be specified by linear equations over the real numbers, then we have the language of linear equations. If we take just binary relations that can be specified by a disequality, as in the examples above, then we have the language of graph-colouring.

Definition 2. A *constraint language* is a set, $\mathcal{L}$, of relations.

For any constraint language $\mathcal{L}$, the class of all constraint satisfaction problems in which all the constraint relations are elements of $\mathcal{L}$ will be denoted CSP($\mathcal{L}$).

Example 1. Let $\mathcal{L}$ be the set containing the single binary relation $R_{\leq 1}$ over the real numbers, $\mathcal{R}$, defined as follows:

$$R_{\leq 1} = \{\langle a, b\rangle \mid a, b \in \mathcal{R}, a - b \leq 1\}$$

One element of CSP($\mathcal{L}$) is the constraint satisfaction problem **P**, which has 4 constraints, $\{C_1, C_2, C_3, C_4\}$, defined as follows:

- $C_1 = (\langle v_1, v_2\rangle, R_{\leq 1})$;
- $C_2 = (\langle v_2, v_3\rangle, R_{\leq 1})$;
- $C_3 = (\langle v_3, v_2\rangle, R_{\leq 1})$;
- $C_4 = (\langle v_3, v_4\rangle, R_{\leq 1})$.

The structure of this problem is illustrated in Figure 1.

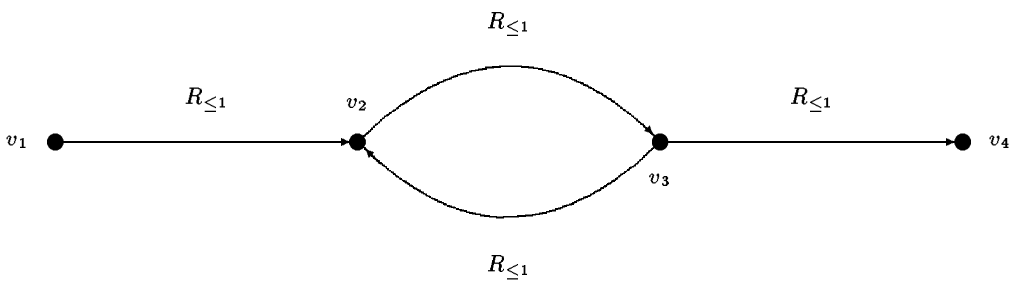

Fig. 1. The CSP defined in Example 1

3 What is expressive power?

Some constraint languages are more powerful than others because they allow us to express a larger collection of problems. For example, if we are dealing with applications involving real-valued variables, then it is possible to express more numerical relationships using arbitrary polynomial equations than if we were restricted to using just linear equations. Similarly, if we are dealing with applications involving Boolean variables, then it is possible to express more logical relationships using ternary clauses than if we were restricted to using just binary clauses.

Of course, the penalty for increased expressive power is generally an increase in computational complexity. For example, the satisfiability problem with ternary clauses is NP-complete [8], whereas the satisfiability problem involving only binary clauses can be solved in polynomial time [8]. More generally, the finite constraint satisfaction problem with arbitrary constraints is known to be NP-complete [19], whereas many families of restricted constraints have been identified which give rise to tractable problem classes [2, 13, 16, 15, 18, 20, 23, 24],

For any given application, it would be very useful to be able to select a constraint language which has sufficient expressive power to express the desired constraints, but is sufficiently restrictive to allow an efficient solution technique. However, it is not immediately clear how to determine what can be expressed in a given language, or whether a language is tractable. These are the questions that we address in this paper.

First, we need to define exactly what it means to "express" a constraint using a constraint language. To clarify this idea, note that in any constraint satisfaction problem some of the required relationships between variables are given explicitly in the constraints, whilst others generally arise implicitly from interactions of different constraints. For any problem in CSP($\mathcal{L}$), the explicit constraint relations must be elements of $\mathcal{L}$, but there may be implicit restrictions on some subsets of the variables for which the corresponding relations are not elements of $\mathcal{L}$, as the next example indicates.

Example 2. Reconsider the constraint satisfaction problem **P**, defined in Example 1.

Note that there is no explicit constraint on the pair $\langle v_1, v_3\rangle$. However, it is clear that the possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $R_{\leq 2} = \{\langle a, b\rangle \mid a, b \in \mathcal{R}, a - b \leq 2\}$.

Similarly, the pairs of values which can be taken by the pair of variables $\langle v_1, v_4\rangle$ are precisely the elements of the relation $R_{\leq 3} = \{\langle a, b\rangle \mid a, b \in \mathcal{R}, a - b \leq 3\}$.

Finally, note that there are two constraints on the pair of variables $\langle v_2, v_3\rangle$. The possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $R'_{\leq 1} = \{\langle a, b\rangle \mid a, b \in \mathcal{R}, -1 \leq (a - b) \leq 1\}$.

We now define exactly what it means to say that a constraint relation can be expressed in a constraint language.

Definition 3. A relation R can be *expressed* in a constraint language $\mathcal{L}$ if there exists a problem **P** in CSP($\mathcal{L}$), and a list, s, of variables, such that the solutions to **P** when restricted to s give precisely the tuples of R.

Example 3. Reconsider the language $\mathcal{L}$ containing the single binary relation $R_{\leq 1}$, defined in Example 1.

It was shown in Example 2 that the relations $R_{\leq 2}, R_{\leq 3}$ and $R'_{\leq 1}$ can all be expressed in this language.

For any constraint language $\mathcal{L}$, the set of *all* relations which can be expressed in $\mathcal{L}$ will be called the *expressive power* of $\mathcal{L}$, and will be denoted $\mathcal{L}^+$.

4 Simple constructions for binary relations

Note that for any binary relation R which belongs to a language $\mathcal{L}$, the relation

$$R^{\cup} = \{\langle a, b\rangle \mid \langle b, a\rangle \in R\},$$

which is called the "converse" of R, can also be expressed in $\mathcal{L}$.

The next example indicates some standard ways to obtain new binary relations which can be expressed using a given set of binary relations.

Example 4. Let $\mathcal{L}$ be a constraint language, and let R_1 and R_2 be any two binary relations in $\mathcal{L}$ (not necessarily distinct).

The class of problems CSP($\mathcal{L}$) contains the problem $\mathbf{P}_{\text{series}}$ with just two constraints C_1 and C_2 where $C_1 = (\langle v_1, v_2\rangle, R_1)$ and $C_2 = (\langle v_2, v_3\rangle, R_2)$.

The possible solutions to $\mathbf{P}_{\text{series}}$ on the variables v_1 and v_3 are exactly the elements of the following relation:

$$\{\langle c_1, c_2\rangle \mid \exists x, \langle c_1, x\rangle \in R_1 \text{ and } \langle x, c_2\rangle \in R_2\}$$

which is called the "composition" of R_1 and R_2 and denoted $R_1; R_2$. Hence, $R_1; R_2$ can be expressed in the language $\mathcal{L}$.

Furthermore, the class of problems CSP($\mathcal{L}$) contains the problem $\mathbf{P}_{\text{parallel}}$ with just two constraints C'_1 and C'_2 where $C'_1 = (\langle v_1, v_2\rangle, R_1)$ and $C'_2 = (\langle v_1, v_2\rangle, R_2)$.

The possible solutions to $\mathbf{P}_{\text{parallel}}$ on the variables v_1 and v_2 are exactly the elements of the relation $R_1 \cap R_2$, the intersection of R_1 and R_2. Hence, $R_1 \cap R_2$ can be expressed in the language $\mathcal{L}$.

We have now seen three ways to express new binary relations using given binary relations: converse, composition and intersection. By repeatedly applying these three operations to the relations of $\mathcal{L}$ we can obtain a large number of new relations, in general.

Definition 4. For any constraint language $\mathcal{L}$, the set of relations that can be obtained from the elements of $\mathcal{L}$ using some sequence of converse, composition and intersection operations will be denoted $\widehat{\mathcal{L}}$.

Example 5. Reconsider the language $\mathcal{L}$ defined in Example 1, which contains the single binary relation $R_{\leq 1}$ over the real numbers defined as follows:

$$R_{\leq 1} = \{\langle a, b\rangle \mid a, b \in \mathcal{R}, a - b \leq 1\}$$

By generalising the constructions given in Example 2, it can be shown that $\widehat{\mathcal{L}}$ contains all relations of the form

$$\{\langle a, b\rangle \mid a, b \in \mathcal{R},\ -n_1 \leq (a-b) \leq n_2\}$$

where n_1 and n_2 are arbitrary positive integers, or ∞.

Hence all of these relations can be expressed in the language $\mathcal{L}$.

At first sight, it might appear that the language $\widehat{\mathcal{L}}$ will contain *all* the binary relations that can be expressed in the language $\mathcal{L}$. It is therefore natural to ask whether there are any languages $\mathcal{L}$ in which we can express any binary relations which are *not* in $\widehat{\mathcal{L}}$.

To answer this question, we note that the definition of $\widehat{\mathcal{L}}$ allows us to express a relation using an arbitrary sequence of converse, composition and intersection operations. The latter two operations correspond to "series" and "parallel" constructions in the constraint graph of the corresponding constraint satisfaction problem. However, it is well-known in graph theory that not every graph can be obtained by a sequence of series and parallel constructions [5]. This suggests that it may be possible to express more constraints by considering constraint satisfaction problems whose constraint graphs cannot be constructed in this way.

The next example shows that for some languages $\mathcal{L}$ it is indeed possible to express relations that are not contained in $\widehat{\mathcal{L}}$.

Example 6. Let $\mathcal{L}$ be the set containing the single binary disequality relation, $\neq_D$, over the set $D = \{R, G, B\}$, defined as follows:

$$\neq_D = \{\langle a, b\rangle \mid a, b \in D, a \neq b\}$$

Note that

$$\neq_D^{\cup} = \neq_D$$
$$\neq_D ; \neq_D = D^2.$$

Hence, the language $\widehat{\mathcal{L}}$ contains just two relations, $\neq_D$ and D^2.

One element of CSP($\mathcal{L}$) is the constraint satisfaction problem $\mathbf{P}$, which has 5 constraints, $\{C_1, C_2, C_3, C_4, C_5\}$, defined as follows:

- $C_1 = (\langle v_1, v_2\rangle, \neq_D)$;
- $C_2 = (\langle v_1, v_3\rangle, \neq_D)$;
- $C_3 = (\langle v_2, v_3\rangle, \neq_D)$;
- $C_4 = (\langle v_2, v_4\rangle, \neq_D)$.
- $C_5 = (\langle v_3, v_4\rangle, \neq_D)$.

The structure of this problem is illustrated in Figure 2. Note that there is no explicit constraint on the pair $\langle v_1, v_4\rangle$. However, by considering all solutions to $\mathbf{P}$, it can be shown that the value taken by the variable v_1 is always equal to the value taken by the variable v_4. Hence, the possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $\{\langle a, b\rangle \mid a, b \in \{R, G, B\}, a = b\}$, and this relation is *not* an element of $\widehat{\mathcal{L}}$.

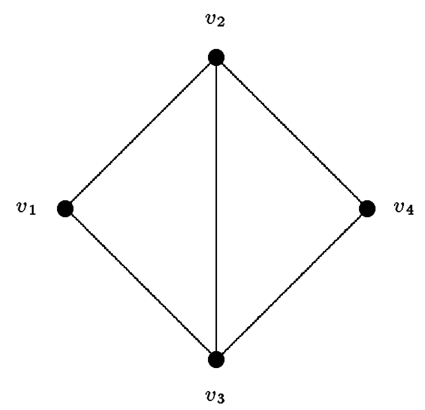

Fig. 2. The constraint graph of the CSP defined in Example 6

Example 7. Let $\mathcal{L}$ be the set containing the single binary relation, $R_\times$, over the set $D = \{0, 1, 2\}$, defined as follows:

$$R_\times = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle\}$$

Note that

$$\begin{aligned} R_\times^{\cup} &= R_\times \\ R_\times ; R_\times &= D^2. \end{aligned}$$

Hence, the language $\widehat{\mathcal{L}}$ contains just two relations, $R_\times$ and D^2.

One element of $\mathrm{CSP}(\mathcal{L})$ is the constraint satisfaction problem $\mathbf{P}$, which has 5 constraints, $\{C_1, C_2, C_3, C_4, C_5\}$, defined as follows:

- $C_1 = (\langle v_1, v_2\rangle, R_\times)$;
- $C_2 = (\langle v_1, v_3\rangle, R_\times)$;
- $C_3 = (\langle v_2, v_3\rangle, R_\times)$;
- $C_4 = (\langle v_2, v_4\rangle, R_\times)$.
- $C_5 = (\langle v_3, v_4\rangle, R_\times)$.

Note that there is no explicit constraint on the pair $\langle v_1, v_4\rangle$. However, by considering all solutions to $\mathbf{P}$, it can be shown that the possible pairs of values which can be taken by this pair of variables are precisely the elements of the relation $R'_\times$, which is defined as follows

$$\{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle\}$$

and this relation is *not* an element of $\widehat{\mathcal{L}}$.

Examples 6 and 7 show that for some languages $\mathcal{L}$, the language $\widehat{\mathcal{L}}$ does not contain all of the binary relations which can be expressed in $\mathcal{L}$, or in other words $\widehat{L} \neq \mathcal{L}^+$. In fact, this is a well-known result in the literature on algebraic properties of binary relations [17]. Hence, in order to calculate the true expressive power of a constraint language, we need to consider more general ways of constructing constraints, even when we consider only binary constraints.

Furthermore, the definitions of the converse and composition operations given above are specific to binary relations, so we also need a more general approach to determine the expressive power of languages containing non-binary relations.

In the next section, we show that for any language of relations over a finite set, there is a single, universal, family of constructions which can be used to obtain all of the relations that can be expressed in that language.

5 A universal construction

Definition 3 states that a relation R can be expressed in a language $\mathcal{L}$ if there is some problem in CSP($\mathcal{L}$) which imposes that relation on some of its variables. We now show that for any language over a finite domain, and any relation, it is only necessary to consider one particular form of problem. In other words, for finite domains, there is a "universal construction" which can be used to express any relation that it is possible to express in a given language.

Definition 5. Let $\mathcal{L}$ be a set of relations over a finite set D.

For any natural number $m \geq 1$, the *indicator problem* of order m for $\mathcal{L}$ is defined to be the constraint satisfaction problem $\mathbf{IP}(\mathcal{L}, m)$ with set of variables D^m, domain of values D, and set of constraints $\{C_1, C_2, \ldots, C_q\}$, where $q = \sum_{R \in \mathcal{L}} |R|^m$, and the constraints C_i are defined as follows. For each $R \in \mathcal{L}$, and for each sequence $t_1, t_2, \ldots, t_m$ of tuples from R, there is a constraint $C_i = (s_i, R)$ with $s_i = (v_1, v_2, \ldots, v_n)$, where n is the arity of R and $v_j = \langle t_1[j], t_2[j], \ldots, t_m[j] \rangle$ for $j = 1$ to n.

Note that for any set of relations $\mathcal{L}$ over a set D, $\mathbf{IP}(\mathcal{L}, m)$ has $|D|^m$ variables, and each variable corresponds to an m-tuple over D. Some examples of indicator problems are given below, and more examples can be found in [14].

Theorem 6. *Let $\mathcal{L}$ be a set of relations over a finite set D, let $R = \{t_1, t_2, \ldots, t_m\}$ be any relation over D, and let n be the arity of R.*

The relation R can be expressed in the language $\mathcal{L}$ if and only if R is equal to the solutions to $\mathbf{IP}(\mathcal{L}, m)$ restricted to the variables $v_1, v_2, \ldots, v_n$, where $v_j = \langle t_1[j], t_2[j], \ldots, t_m[j] \rangle$ for $j = 1$ to n.

Proof. If R is equal to the solutions to $\mathbf{IP}(\mathcal{L}, m)$ restricted to some list of variables, then R can obviously be expressed in $\mathcal{L}$, by Definition 3.

To prove the converse, consider the relation R' whose tuples are given by the solutions to $\mathbf{IP}(\mathcal{L}, m)$ restricted to $v_1, v_2, \ldots, v_n$.

By the definition of the indicator problem, we have $R \subseteq R'$. Hence, if $R' \neq R$, then there must be at least one solution, s, to $\mathbf{IP}(\mathcal{L}, m)$ whose restriction to $v_1, v_2, \ldots, v_n$ is not contained in R.

However, it is shown in [15] that the solutions to $\mathbf{IP}(\mathcal{L}, m)$ are *closure operations*[1] on $\mathcal{L}$, and that every relation in $\mathcal{L}^+$ is closed under all of these operations.

Hence, if $R' \neq R$, then R cannot be expressed in $\mathcal{L}$, which gives the result.

Example 8. Reconsider the relation $R_\times$ over $D = \{0, 1, 2\}$, defined in Example 7.

The indicator problem for $\{R_\times\}$ of order 1, $\mathbf{IP}(\{R_\times\}, 1)$, has 3 variables and 6 constraints. The set of variables is

$$\{\langle 0\rangle, \langle 1\rangle, \langle 2\rangle\},$$

and the set of constraints is

$$\begin{aligned}
\{\ & ((\langle 0\rangle, \langle 0\rangle), R_\times),\\
& ((\langle 0\rangle, \langle 1\rangle), R_\times),\\
& ((\langle 1\rangle, \langle 0\rangle), R_\times),\\
& ((\langle 1\rangle, \langle 2\rangle), R_\times),\\
& ((\langle 2\rangle, \langle 1\rangle), R_\times),\\
& ((\langle 2\rangle, \langle 2\rangle), R_\times)\ \}.
\end{aligned}$$

This problem has has 6 solutions, which may be expressed in tabular form as follows:

	Variables		
	$\langle 0\rangle$	$\langle 1\rangle$	$\langle 2\rangle$
Solution 1	0	0	0
Solution 2	0	1	0
Solution 3	0	1	2
Solution 4	2	1	0
Solution 5	2	1	2
Solution 6	2	2	2

Note that the restriction of this set of solutions to any sequence of variables gives more than one tuple, so by Theorem 6, the language $\{R_\times\}$ cannot express any relation containing exactly one tuple.

Example 9. Reconsider the relation $R_\times$ over $D = \{0, 1, 2\}$, defined in Example 7.

The indicator problem for $\{R_\times\}$ of order 2, $\mathbf{IP}(\{R_\times\}, 2)$, has 9 variables and 36 constraints. The set of variables is

$$\{\langle 0,0\rangle, \langle 0,1\rangle, \langle 0,2\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle, \langle 2,0\rangle, \langle 2,1\rangle, \langle 2,2\rangle\},$$

[1] Sometimes called *polymorphisms* [10]

and the set of constraints is

$$\{\ (((\langle 0,0\rangle,\langle 0,0\rangle),R_\times),\ (((\langle 0,0\rangle,\langle 0,1\rangle),R_\times),\ (((\langle 0,0\rangle,\langle 1,0\rangle),R_\times),\ (((\langle 0,0\rangle,\langle 1,1\rangle),R_\times),$$
$$(((\langle 0,1\rangle,\langle 0,0\rangle),R_\times),\ (((\langle 0,1\rangle,\langle 0,2\rangle),R_\times),\ (((\langle 0,1\rangle,\langle 1,0\rangle),R_\times),\ (((\langle 0,1\rangle,\langle 1,2\rangle),R_\times),$$
$$(((\langle 0,2\rangle,\langle 0,1\rangle),R_\times),\ (((\langle 0,2\rangle,\langle 0,2\rangle),R_\times),\ (((\langle 0,2\rangle,\langle 1,1\rangle),R_\times),\ (((\langle 0,2\rangle,\langle 1,2\rangle),R_\times),$$
$$(((\langle 1,0\rangle,\langle 0,0\rangle),R_\times),\ (((\langle 1,0\rangle,\langle 0,1\rangle),R_\times),\ (((\langle 1,0\rangle,\langle 2,0\rangle),R_\times),\ (((\langle 1,0\rangle,\langle 2,1\rangle),R_\times),$$
$$(((\langle 1,1\rangle,\langle 0,0\rangle),R_\times),\ (((\langle 1,1\rangle,\langle 0,2\rangle),R_\times),\ (((\langle 1,1\rangle,\langle 2,0\rangle),R_\times),\ (((\langle 1,1\rangle,\langle 2,2\rangle),R_\times),$$
$$(((\langle 1,2\rangle,\langle 0,1\rangle),R_\times),\ (((\langle 1,2\rangle,\langle 0,2\rangle),R_\times),\ (((\langle 1,2\rangle,\langle 2,1\rangle),R_\times),\ (((\langle 1,2\rangle,\langle 2,2\rangle),R_\times),$$
$$(((\langle 2,0\rangle,\langle 1,0\rangle),R_\times),\ (((\langle 2,0\rangle,\langle 1,1\rangle),R_\times),\ (((\langle 2,0\rangle,\langle 2,0\rangle),R_\times),\ (((\langle 2,0\rangle,\langle 2,1\rangle),R_\times),$$
$$(((\langle 2,1\rangle,\langle 1,0\rangle),R_\times),\ (((\langle 2,1\rangle,\langle 1,2\rangle),R_\times),\ (((\langle 2,1\rangle,\langle 2,0\rangle),R_\times),\ (((\langle 2,1\rangle,\langle 2,2\rangle),R_\times),$$
$$(((\langle 2,2\rangle,\langle 1,1\rangle),R_\times),\ (((\langle 2,2\rangle,\langle 1,2\rangle),R_\times),\ (((\langle 2,2\rangle,\langle 2,1\rangle),R_\times),\ (((\langle 2,2\rangle,\langle 2,2\rangle),R_\times)\ \}.$$

This problem has 32 solutions, which may be expressed in tabular form as follows:

	Variables								
	$\langle 0,0\rangle$	$\langle 0,1\rangle$	$\langle 0,2\rangle$	$\langle 1,0\rangle$	$\langle 1,1\rangle$	$\langle 1,2\rangle$	$\langle 2,0\rangle$	$\langle 2,1\rangle$	$\langle 2,2\rangle$
Solution 1	0	0	0	0	0	0	0	0	0
Solution 2	0	0	0	0	0	0	0	1	0
Solution 3	0	0	0	0	0	1	0	0	0
Solution 4	0	0	0	0	1	0	0	0	0
Solution 5	0	0	0	0	1	0	0	1	0
Solution 6	0	0	0	0	1	1	0	0	0
Solution 7	0	0	0	1	0	0	0	0	0
Solution 8	0	0	0	1	0	1	0	0	0
Solution 9	0	0	0	1	1	0	0	0	0
Solution 10	0	0	0	1	1	1	0	0	0
Solution 11	0	0	0	1	1	1	2	2	2
Solution 12	0	1	0	0	0	0	0	0	0
Solution 13	0	1	0	0	0	0	0	1	0
Solution 14	0	1	0	0	1	0	0	0	0
Solution 15	0	1	0	0	1	0	0	1	0
Solution 16	0	1	2	0	1	2	0	1	2
Solution 17	2	1	0	2	1	0	2	1	0
Solution 18	2	1	2	2	1	2	2	1	2
Solution 19	2	1	2	2	1	2	2	2	2
Solution 20	2	1	2	2	2	2	2	1	2
Solution 21	2	1	2	2	2	2	2	2	2
Solution 22	2	2	2	1	1	1	0	0	0
Solution 23	2	2	2	1	1	1	2	2	2
Solution 24	2	2	2	1	1	2	2	2	2
Solution 25	2	2	2	1	2	1	2	2	2
Solution 26	2	2	2	1	2	2	2	2	2
Solution 27	2	2	2	2	1	1	2	2	2
Solution 28	2	2	2	2	1	2	2	1	2
Solution 29	2	2	2	2	1	2	2	2	2
Solution 30	2	2	2	2	2	1	2	2	2
Solution 31	2	2	2	2	2	2	2	1	2
Solution 32	2	2	2	2	2	2	2	2	2

Note that the restriction of this set of solutions to the pair of (identical) variables $\langle 0,2\rangle$ and $\langle 0,2\rangle$ gives the tuples $\langle 0,0\rangle$ and $\langle 2,2\rangle$. Hence, by Theorem 6, the relation $\{\langle 0,0\rangle,\langle 2,2\rangle\}$ can be expressed in the language $\{R_\times\}$.

Conversely, note that the restriction of this set of solutions to the variables $\langle 0,0\rangle$ and $\langle 0,1\rangle$ gives the tuples $\langle 0,0\rangle,\langle 0,1\rangle,\langle 2,1\rangle$ and $\langle 2,2\rangle$. Hence, by Theorem 6, the relation $\{\langle 0,0\rangle,\langle 0,1\rangle\}$ *cannot* be expressed in the language $\{R_\times\}$.

Theorem 6 states that in order to determine whether a relation R can be expressed in a language $\mathcal{L}$, it is sufficient to consider the indicator problem $\mathbf{IP}(\mathcal{L},m)$, where $m=|R|$. However, the size of these indicator problems grows rapidly with m. In some cases it is not necessary to build the indicator problem $\mathbf{IP}(\mathcal{L},m)$ because the relation can be expressed using an indicator problem $\mathbf{IP}(\mathcal{L},m')$, where $m'<m$, as the next example illustrates.

Example 10. Reconsider the relation $R_\times$ over $D=\{0,1,2\}$, defined in Example 7.

The indicator problem for $\{R_\times\}$ of order 2, $\mathbf{IP}(\{R_\times\},2)$, has 32 solutions, as shown in Example 9.

If we restrict the solutions to $\mathbf{IP}(\{R_\times\},2)$ to the variables $\langle 0,1\rangle$ and $\langle 1,1\rangle$, then we get the relation $R'_\times$, defined in Example 7, which contains 7 tuples. Hence, we have shown that $R'_\times$ can be expressed in the language $\{R_\times\}$ without building the indicator problem of order 7.

If we restrict the solutions to $\mathbf{IP}(\{R_\times\},2)$ to the variables $\langle 0,0\rangle$ and $\langle 0,1\rangle$, then we get the relation $R''_\times$ containing 4 tuples, defined as follows

$$R''_\times=\{\langle 0,0\rangle,\langle 0,1\rangle,\langle 2,1\rangle,\langle 2,2\rangle\}$$

Hence we have shown that $R''_\times$ can be expressed in the language $\{R_\times\}$ without building the indicator problem of order 4. (Note that $R''_\times$ is *not* symmetric, hence it also illustrates the fact that a language containing symmetric binary relations can express non-symmetric relations, in some cases).

6 Expressive power and complexity

In this section, we shall consider how the choice of constraint language, $\mathcal{L}$, affects the complexity of deciding whether the corresponding class of constraint satisfaction problems, CSP($\mathcal{L}$), has a solution. We shall therefore regard CSP($\mathcal{L}$) as a class of decision problems in which the question to be decided in each problem instance is the existence of a solution.

If there exists an algorithm which decides every instance in CSP($\mathcal{L}$) in polynomial time, then we shall say that $\mathcal{L}$ is a *tractable* constraint language. On the other hand, if CSP($\mathcal{L}$) is NP-complete, then we shall say that $\mathcal{L}$ is NP-complete.

Example 11. The binary inequality relation over a set D, denoted $\neq_D$, is defined as

$$\neq_D=\{\langle d_1,d_2\rangle\in D^2\mid d_1\neq d_2\}.$$

Note that CSP($\{\neq_D\}$) corresponds precisely to the Graph $|D|$-Colorability problem [8]. This problem is tractable when $|D|\leq 2$ and NP-complete when $|D|\geq 3$. Hence, the language $\{\neq_D\}$ is tractable when $|D|\leq 2$, and NP-complete when $|D|\geq 3$.

The next result shows that if we can express a relation in a language, then we might as well add it to the language, because it will not change the complexity of the class of problems we are dealing with.

Proposition 7. *For any constraint language $\mathcal{L}$ and any relation R which can be expressed in $\mathcal{L}$,* CSP($\mathcal{L} \cup \{R\}$) *is reducible to* CSP($\mathcal{L}$) *in linear time and logarithmic space.*

Proof. Since R can be expressed in L we know that there is some problem $\mathbf{P}_R$ in CSP($\mathcal{L}$) for which the values of the solutions restricted to some list of variables, s, are precisely the elements of R.

Let $\mathbf{P}$ be any constraint satisfaction problem in CSP($\mathcal{L} \cup R$). We can reduce $\mathbf{P}$ to a constraint satisfaction problem in CSP($\mathcal{L}$) simply by examining each constraint C in turn, and whenever the constraint relation of C is R, then replacing C by (a disjoint copy of) the problem $\mathbf{P}_R$ described above, and then replacing each variable in s with the corresponding member of the scope of C.

Corollary 8. *For any constraint language $\mathcal{L}$, and any finite constraint language $\mathcal{L}_0$, if $\mathcal{L}_0 \subseteq \mathcal{L}^+$, then* CSP($\mathcal{L}_0$) *is reducible to* CSP($\mathcal{L}$) *in logarithmic space.*

Corollary 9. *Any constraint language $\mathcal{L}$ which can express all of the relations in some finite NP-complete language $\mathcal{L}_0$ is NP-complete.*

By using Corollary 9, together with Theorem 6, we can show that many languages are NP-complete by simply showing that they can express some known NP-complete language.

Example 12. The smallest known NP-complete language is the language containing the single relation $T = \{\langle 1,0,0\rangle, \langle 0,1,0\rangle, \langle 0,0,1\rangle\}$. The associated class of constraint satisfaction problems, CSP($\{T\}$), corresponds precisely to the ONE-IN-THREE SATISFIABILITY problem [21], which is NP-complete.

Hence, for any language $\mathcal{L}$, and any two domain elements d_0, d_1, if the solutions to $\mathbf{IP}(\mathcal{L}, 3)$, restricted to the variables $\langle d_1, d_0, d_0\rangle$, $\langle d_0, d_1, d_0\rangle$ and $\langle d_0, d_0, d_1\rangle$ is equal to $\{\langle d_1, d_0, d_0\rangle, \langle d_0, d_1, d_0\rangle, \langle d_0, d_0, d_1\rangle\}$, then $\mathcal{L}$ is NP-complete.

In particular, if $\mathbf{IP}(\mathcal{L}, 3)$ has only 3 solutions then $\mathcal{L}$ is NP-complete.

Note that this provides a purely mechanical procedure to establish NP-completeness of a constraint language, without having to design any specific reductions, or invent any new constructions.

Example 13. Reconsider the relation $R_\times$ over $D = \{0,1,2\}$, defined in Example 7.

The language $\{R_\times\}$ is clearly tractable, because any problem in CSP($\{R_\times\}$) has the trivial solution in which every variable takes the value 0.

However, if we consider the language $\mathcal{L}_0 = \{R_\times, R_0\}$, where $R_0 = \{\langle 0,1,2\rangle\}$ then we find that the indicator problem for $\mathcal{L}_0$ of order 3, $\mathbf{IP}(\mathcal{L}_0, 3)$, with 27 variables and 217 constraints, has only 3 solutions. Hence, $\mathcal{L}_0$ is NP-complete.

We can also conclude from Corollary 9 that if $\mathcal{L}$ is a tractable language, then it must be impossible to express in $\mathcal{L}$ any finite NP-complete language (assuming that P is not equal to NP). In view of Theorem 6, this means that tractable languages must have extra solutions to their indicator problems, in addition to the standard solutions that are present in all cases. The presence of these additional solutions provides an 'indicator' of tractability, and it is this which gives rise to the name 'indicator problem'[2].

Example 14. Let $\mathcal{L}$ be any Boolean constraint language (i.e. a set of relations over the domain $\{0, 1\}$).

In this case CSP($\mathcal{L}$) corresponds exactly to the GENERALISED SATISFIABILITY problem [8], for which all possible tractable constraint languages are known, and are fully described in [21]. The tractable languages fall into just 6 distinct classes, which are defined as follows:

Class 0a All relations in the language contain the tuple $\langle 0, 0, \ldots, 0\rangle$.
Class 0b All relations contain the tuple $\langle 1, 1, \ldots, 1\rangle$.
Class Ia All relations can be defined using Horn clauses.
Class Ib All relations can be defined using anti-Horn clauses[3].
Class II All relations can be defined using clauses with at most 2 literals.
Class III All relations can be defined using linear equations over the integers modulo 2.

If $\mathcal{L}$ does not fall into one of these 6 classes, then CSP($\mathcal{L}$) is NP-complete [21].

The indicator problem for $\mathcal{L}$ of order 3, $\mathbf{IP}(\mathcal{L}, 3)$, has 8 variables, corresponding to the 8 possible Boolean sequences of length 3. It has 256 possible solutions, corresponding to the 256 possible assignments of Boolean values to these 8 variables.

Amongst these 256 possible solutions, we can identify 6 distinguished assignments as shown in the following table. It can be shown [11] that the language $\mathcal{L}$ falls into one of the 6 tractable classes described above if and only if $\mathbf{IP}(\mathcal{L}, 3)$ has the corresponding solution, as shown in this table.

	Variables							
	$\langle 0,0,0\rangle$	$\langle 0,0,1\rangle$	$\langle 0,1,0\rangle$	$\langle 0,1,1\rangle$	$\langle 1,0,0\rangle$	$\langle 1,0,1\rangle$	$\langle 1,1,0\rangle$	$\langle 1,1,1\rangle$
Class 0a - Constant 0	0	0	0	0	0	0	0	0
Class 0b - Constant 1	1	1	1	1	1	1	1	1
Class Ia - Horn	0	0	0	0	0	0	1	1
Class Ib - Anti-Horn	0	0	1	1	1	1	1	1
Class II - 2-Sat	0	0	0	1	0	1	1	1
Class III - Linear	0	1	1	0	1	0	0	1

[2] Solutions to indicator problems can indicate other properties as well as tractability. For example, whether a certain level of local consistency is sufficient to ensure global consistency [12].

[3] An *anti-Horn* clause is a disjunction of literals, with at most one negative literal.

For larger finite domains it is still the case that all known maximal tractable constraint languages are characterised by the presence of a single additional solution to the indicator problem of order 3 [15]. This solution can be viewed as the 'signature' of that tractable language.

It is currently an open question whether all possible tractable languages are characterised by a single signature in this way. If this were true, it would mean that there are only a finite number of maximal tractable languages over any finite domain. However, with the current state of knowledge, it appears to be possible that there are tractable languages characterised only by the presence of more than one additional solution to the indicator problem of order 3, or by the presence of certain solutions to indicator problems of higher order, although none have so far been identified.

7 Conclusions and open problems

In this paper we have examined the notion of constraint languages, and the expressive power of constraint languages. We have shown that calculating the expressive power of a given language is not a trivial task, but for all languages over finite domains there is a universal construction involving the indicator problem which provides a complete algorithmic solution.

We have also examined how the complexity of a language can be related to the expressive power, and hence shown that the indicator problem can also be used to determine whether a language is NP-complete or tractable.

These investigations raise the following open questions:

1. Is there a more efficient universal construction to determine the expressive power of languages over finite domains?
2. Is there a universal construction for any language over an infinite domain?
3. Is every maximal tractable language over a finite domain characterised by a 'signature' solution to the indicator problem of order 3?
4. Is every constraint language either tractable or NP-complete?

Acknowledgements

This research was supported by EPSRC research grant GR/L09936. I am grateful to Dave Cohen and Victor Dalmau for many helpful discussions.

References

1. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimisation. *Journal of the ACM*, 44:201–236, 1997.
2. M.C. Cooper, D.A. Cohen, and P.G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.
3. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

4. R. Dechter and P. van Beek. Local and global relational consistency. *Theoretical Computer Science*, 173(1):283–308, 1997.
5. R.J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10:303–318, 1965.
6. E.C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982.
7. E.C. Freuder. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 32:755–761, 1985.
8. M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA., 1979.
9. M. Gyssens, P.G. Jeavons, and D.A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66(1):57–89, 1994.
10. T. Ihringer and R. Pöschel. Collapsing clones. *Acta Sci. Math. (Szeged)*, 58:99–113, 1993.
11. P.G. Jeavons and D.A. Cohen. An algebraic characterization of tractable constraints. In *Computing and Combinatorics. First International Conference COCOON'95 (Xi'an,China,August 1995)*, volume 959 of *Lecture Notes in Computer Science*, pages 633–642. Springer-Verlag, 1995.
12. P.G. Jeavons, D.A. Cohen, and M.C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.
13. P.G. Jeavons, D.A. Cohen, and M. Gyssens. 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*, pages 276–291. Springer-Verlag, 1995.
14. P.G. Jeavons, D.A. Cohen, and M. Gyssens. A test for tractability. In *Proceedings 2nd International Conference on Constraint Programming—CP'96 (Boston, August 1996)*, volume 1118 of *Lecture Notes in Computer Science*, pages 267–281. Springer-Verlag, 1996.
15. P.G. Jeavons, D.A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.
16. P.G. Jeavons and M.C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.
17. B. Jónsson. The theory of binary relations. In *Algebraic Logic (Budapest, Hungary 1988)*, volume 54 of *Colloq. Math. Soc. Janos Bolyai*, pages 245–292. North-Holland, 1991.
18. L. Kirousis. Fast parallel constraint satisfaction. *Artificial Intelligence*, 64:147–160, 1993.
19. A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.
20. U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.
21. T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.
22. E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, London, 1993.
23. P. van Beek and R. Dechter. On the minimality and decomposability of row-convex constraint networks. *Journal of the ACM*, 42:543–561, 1995.
24. P. van Hentenryck, Y. Deville, and C-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.