Algebraic Properties of CSP Model Operators*

Y.C. Law and J.H.M. Lee

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR, China
{yclaw,jlee}@cse.cuhk.edu.hk

1 Introduction

The task at hand is to tackle *Constraint Satisfaction Problems* (CSPs) defined in the sense of Mackworth [4]. This paper aims to take a first step towards a CSP-based *module systems* for constraint programming languages and modeling tools. The call for such a system is two-fold. First, most existing constraint programming languages have some sort of module systems, but these systems are designed for the underlying languages. Thus these module systems facilitate the construction of large constraint programs in a particular language, but not of CSP models. Second, a module system designed for CSP models with clear and clean semantics should allow us to reason the properties of CSP models declaratively without actually solving the CSPs. As a first attempt, we introduce six operators for manipulating and transforming CSP models: namely intersection, union, channeling, induction, negation, and complementation. For each operator, we give its syntactic construction rule, define its set-theoretic meaning, and also examine its algebraic properties, all illustrated with examples where appropriate. Our results show that model intersection and union form abelian monoids respectively among others.

The rest of the paper is organized as follows. Section 2 provides the basic definitions relating to CSP models. In Section 3, we examine the definitions and properties of the six operators in details. Section 4 gives further algebraic properties, which allow us to identify possible algebraic structures of the operators. We summarize and shed light on possible future direction of research in Section 5.

2 From Viewpoints to CSP Models

There are usually more than one way of formulating a problem P into a CSP. Central to the formulation process is to determine the variables and the domains (associated sets of possible values) of the variables. Different choices of variables and domains are results of viewing the problem P from different angles/perspectives. We define a *viewpoint* to be a pair (X, D_X), where

* The work described in this paper was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK4183/00E).

$X = \{x_1, \ldots, x_n\}$ is a set of variables, and D_X is a set containing, for every $x \in X$, an associated domain $D_X(x)$ giving the set of possibles values for x.

A viewpoint $V = (X, D_X)$ defines the possible assignments for variables in X. An *assignment* in V (or in $U \subseteq X$) is a pair $\langle x, a\rangle$, which means that variable $x \in X$ (or U) is assigned the value $a \in D_X(x)$. A *compound assignment* in V (or in $U \subseteq X$) is a set of assignments $\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}$, where $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ (or U) and $a_j \in D_X(x_{i_j})$ for each $j \in \{1, \ldots, k\}$. *Note* the requirement that *no* variables may be assigned more than one value in a compound assignment. Given a set of assignments θ, we use the predicate $cmpd(\theta, V)$ to ensure that θ is a compound assignment in V. A *complete assignment* in V is a compound assignment $\{\langle x_1, a_1\rangle, \ldots, \langle x_n, a_k\rangle\}$ for all variables in X.

When formulating a problem P into a CSP, the choice of viewpoints is not arbitrary. Suppose $sol(P)$ is the set of all solutions of P (in whatever notations and formalism). We say that viewpoint V is *proper* for P if and only if we can find a subset S of the set of all possible complete assignments in V so that there is a one-one mapping between S and $sol(P)$. In other words, each solution of P must correspond to a distinct complete assignment in V. We note also that according to our definition, any viewpoint is proper with respect to a problem that has no solutions.

A *constraint* can be considered a predicate that maps to $true$ or $false$. The *signature* $sig(c) \subseteq X$, which is the set of variables involved in c, defines the scope of c. We abuse terminology by saying that the compound assignment $\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}$ also has a signature: $sig(\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}) = \{x_{i_1}, \ldots, x_{i_k}\}$. Given a compound assignment θ such that $sig(c) \subseteq sig(\theta)$, the *application* of θ to c, $c\theta$, is obtained by replacing all variables in c by the corresponding values in θ. If $c\theta$ is $true$, we say θ *satisfies* c, and θ *violates* c otherwise. In addition, the negation $\neg c$ of a constraint c is defined by the fact that $(\neg c)\theta = \neg(c\theta)$ for all compound assignments θ in $X \supseteq sig(c)$. We overload the $\neg$ operator so that it operates on both constraints and boolean expressions.

A *CSP model* M (or simply *model* hereafter) of a problem P is a pair (V, C), where V is a proper viewpoint of P and C is a set of constraints in V for P. Note that, in our definition, we allow two constraints to be on the same set of variables: $c_i, c_j \in C$ and $sig(c_i) = sig(c_j)$. A *solution* of $M = (V, C)$ is a complete assignment θ in V so that $c\theta = true$ for every $c \in C$. Since M is a model of P, the constraints C must be defined in such a way that there is a one-one correspondence between $sol(M)$ and $sol(P)$. Thus, the viewpoint V essentially dictates how the constraints of P are formulated (*modulo* solution equivalence).

3 Operators over CSP Models

We are interested in operators in the space of CSP models. In this section, we introduce several such operators and give the set-theoretic semantics and properties of these operators. In the rest of the presentation, we assume $M_1 = (V_1, C_{X_1})$ and $M_2 = (V_2, C_{X_2})$, $V_1 = (X_1, D_{X_1})$ and $V_2 = (X_2, D_{X_2})$.

Model intersection forms *conjuncted models* by essentially conjoining constraints from constituent models. A solution of a conjuncted model must thus also be a solution of all of its constituent models. More formally, the conjuncted model $M_1 \cap M_2$ is $((X_1 \cup X_2, D_{X_1 \cup X_2}), C_{X_1} \cup C_{X_2})$, where for all $x \in X_1 \cup X_2$,

$$D_{X_1 \cup X_2}(x) = \begin{cases} D_{X_1}(x) & \text{if } x \in X_1 \wedge x \notin X_2 \\ D_{X_2}(x) & \text{if } x \notin X_1 \wedge x \in X_2 \\ D_{X_1}(x) \cap D_{X_2}(x) & \text{otherwise} \end{cases}$$

We overload the $\cap$ operator so that it operates on CSP models as well as sets. A consequence of the definition is that every solution of a conjuncted model must satisfy all constraints in its constituent models.

Model union deals with choices in constraint processing. The result is a *disjuncted model*, which allows solutions of any one of the constituent models to be extended to solutions of the disjuncted model. More formally, the disjuncted model $M_1 \cup M_2$ is $((X_1 \cup X_2, D_{X_1 \cup X_2}), \{c_1 \vee c_2 | c_1 \in C_{X_1} \wedge c_2 \in C_{X_2}\})$, where for all $x \in X_1 \cup X_2$,

$$D_{X_1 \cup X_2}(x) = \begin{cases} D_{X_1}(x) & \text{if } x \in X_1 \wedge x \notin X_2 \\ D_{X_2}(x) & \text{if } x \notin X_1 \wedge x \in X_2 \\ D_{X_1}(x) \cup D_{X_2}(x) & \text{otherwise} \end{cases}$$

We overload the $\cup$ operator so that it operates on CSP models as well as sets. The strength of the combined whole may well be more than the sum of the strength of the individuals. This is the case with the solution set of a disjuncted model with respect to its constituent models.

Cheng *et al.* [1] define a *channeling constraint* c to be a constraint, where $sig(c) \not\subseteq X_1$, $sig(c) \not\subseteq X_2$, and $sig(c) \subseteq X_1 \cup X_2$. We note in the definition that the constraints in the two models are immaterial. Channeling constraints relate actually viewpoints but not models. Suppose there is a set C_c of channeling constraints connecting the viewpoints V_1 and V_2. *Model channeling* combines M_1 and M_2 using C_c to form a *channeled model*, which is $M_1 \cap M_2$ plus the channeling constraints C_c. More formally, the channeled model $M_1 \overset{C_c}{\bowtie} M_2$ is $((X_1 \cup X_2, D_{X_1 \cup X_2}), C_{X_1} \cup C_{X_2} \cup C_c)$, where for all $x \in X_1 \cup X_2$,

$$D_{X_1 \cup X_2}(x) = \begin{cases} D_{X_1}(x) & \text{if } x \in X_1 \wedge x \notin X_2 \\ D_{X_2}(x) & \text{if } x \notin X_1 \wedge x \in X_2 \\ D_{X_1}(x) \cap D_{X_2}(x) & \text{otherwise} \end{cases}$$

Given two models M_1 and M_2. The channeled model $M_1 \overset{C_c}{\bowtie} M_2$ is more constrained than the conjuncted model $M_1 \cap M_2$. A solution of $M_1 \overset{C_c}{\bowtie} M_2$ must satisfy all constraints in M_1 and M_2 plus the channeling constraints C_c.

Model induction [3] is a method for systematically generating a new model from an existing model, using another viewpoint and channeling constraints. We note that a model M_1 contains two types of constraints: the explicit constraints as stated in C_{X_1} and the implicit constraints for enforcing valid variable assignments. Given a set of channeling constraints defining a total and injective function f from the possible assignments in V_1 to those in V_2. The core of model induction is a *meaning-preserving* transformation from constraints in model M_1, both implicit and explicit (C_{X_1}), using f to generate constraints C_{X_2} in viewpoint V_2. Due to space limitation, readers are referred to Law and Lee [3] for the detailed definition of model induction.

Model negation takes a model as input and generates a *negated model* by negating all constraints in the original model. Given a model $M = (V, C)$, the viewpoint of the negated model remains unchanged. For each constraint $c \in C$, the negated constraint $\neg c$ is in the negated model. Thus $\neg M = (V, \{\neg c | c \in C\})$. We overload also the $\neg$ operator so that it operates on CSP models as well as boolean expressions. Since we negate all constraints, solutions of the negated model $\neg M$ consist of all complete assignments that violate all constraints in M. Unfortunately, solutions of $\neg M$ cannot be constructed from solutions of M, but negation does neutralize each other by the fact that $(\neg(\neg c))\theta = \neg((\neg c)\theta) = \neg(\neg(c\theta)) = c\theta$.

Model complementation provides an alternative means to handle negative information. The *complemented model* $\overline{M}$ of a model M contains the same viewpoint as M. The only constraint in $\overline{M}$ is the negation of the conjunction of all constraints in M. Solutions of $\overline{M}$ thus violates at least one constraint in M. More formally, if $M = (V, C)$, then $\overline{M} = (V, \{\neg(\bigwedge C)\})$, where $\neg(\bigwedge C)$ is equivalent to $\bigvee\{\neg c | c \in C\}$. Solutions of M and $\overline{M}$ partition the set of all possible complete assignments for (the viewpoint of) M. By definition, complementation also annihilates the effect of another.

4 Algebraic Structures

In this section, we identify the algebraic structures of some of the introduced operators. In the following, $M = (V, C)$, M_1, and M_2 denote CSP models. $E_\emptyset = ((\emptyset, \emptyset), \emptyset)$ is the *empty CSP*, which consists of no variables and no constraints. $E_\perp = ((\emptyset, \emptyset), \{false\})$ is the *contradictory CSP*, which has also no variables and only the constant $false$ as constraint. The empty CSP is a satisfiable CSP with the *empty assignment* $\emptyset$ as its solution, while the contradictory CSP is unsatisfiable with no solutions. A *monoid* [2] $(G, \odot)$ is a nonempty set G together with a binary operation $\odot$ on G which is associative, and there exists an identity element $e \in G$ such that $a \odot e = e \odot a = a$ for all $a \in G$. A monoid is said to be *abelian* if $\odot$ is commutative. Let $\mathcal{M}$ be the set of all CSP models.

Table 1 summarizes the common algebraic properties of some of the introduced model operators. Except for the distributivity of union over intersection, we skip the proof of the other straightforward properties. As we can see, $(\mathcal{M}, \cap)$ forms an abelian monoid with the empty CSP $E_\emptyset$ as the identity element. Model intersection is also idempotent since $M \cap M = M$. Similarly, $(\mathcal{M}, \cup)$ forms also an abelian monoid with the contradictory CSP $E_\perp$ as the identity element. Besides, taking the union of any model and the empty CSP $E_\emptyset$ vanishes the constraints in the disjuncted model, which has all complete assignments as solutions. Both intersection and union fail to be a group due to the lack of inverse elements.

Table 1. Algebraic Properties of Some Model Operators

• $M_1 \cap M_2 = M_2 \cap M_1$	• $M_1 \cup (M_2 \cap M_3) = (M_1 \cup M_2) \cap (M_1 \cup M_3)$
• $(M_1 \cap M_2) \cap M_2 = M_1 \cap (M_2 \cap M_3)$	
• $M \cap E_{\emptyset} = M$	• $M_1 \overset{C_c}{\bowtie} M_2 = M_2 \overset{C_c}{\bowtie} M_1$
• $M \cap M = M$	• $(M_1 \overset{C_{c_1}}{\bowtie} M_2) \overset{C_{c_2}}{\bowtie} M_3 = M_1 \overset{C_{c_1}}{\bowtie} (M_2 \overset{C_{c_2}}{\bowtie} M_3)$
	• $M_1 \overset{\emptyset}{\bowtie} M_2 = M_1 \cap M_2$
• $M_1 \cup M_2 = M_2 \cup M_1$	
• $(M_1 \cup M_2) \cup M_3 = M_1 \cup (M_2 \cup M_3)$	• $\neg(M_1 \cap M_2) = \neg M_1 \cap \neg M_2$
• $M \cup E_{\perp} = M$	
• $M \cup E_{\emptyset} = (V, \emptyset)$	

5 Concluding Remarks

A good module system should be compositional and be based on a rich algebra of model operators. We introduce six such operators and examine their properties. The work as reported is insufficient to form a practical model algebra, but should serve to shed light on the design of future CSP-based module systems.

We believe that we are the first to propose a systematic study of model operators and their algebraic properties. It is a purpose of the paper to arouse interest in this important new direction of research. There is plenty of scope for future work. First, it would be interesting to look for other useful operators, and even perhaps to refine the definition of the proposed operators. In particular, we focus on satisfiable models, and relatively little is known about the negation and complementation operators. Second, much work is needed to design a practical and yet versatile module system, based on an algebra (even if there is one), in constraint-based interactive problem-solving tools and constraint programming languages. Third, the work suggests the possible notions of "reusable model components" and "model patterns," which can serve as the brick and mortar for and save much effort in the construction of huge and complex CSP models.

References

1. B.M.W. Cheng, K.M.F. Choi, J.H.M. Lee, and J.C.K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *Constraints*, 4(2):167–192, 1999.
2. T.W. Hungerford. *Algebra*. Springer-Verlag, 1974.
3. Y.C. Law and J.H.M. Lee. Model induction: a new source of CSP model redundancy. In *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002.
4. A.K. Mackworth. Consistency in networks of relations. *AI Journal*, 8(1):99–118, 1977.