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

Abstract. A module system is desirable for the construction of large CSP models out of pieces. Such module system, which is long overdue, should have a clear and clean semantics so as to facilitate declarative reasoning. 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 work can shed light on the design of future CSP-based module system.

1 Introduction

The task at hand is to tackle *Constraint Satisfaction Problems* (CSPs) defined in the sense of Mackworth [6]. Much CSP research effort focuses on designing general efficient (systematic or local) search algorithms for solving CSPs, and exploiting domain-specific information to solve particular applications efficiently. A recent important line of research in the community investigates how problem formulation and reformulation affect execution efficiency of constraint-solving algorithms. Freuder [2] lists first problem modeling among the seven most important future directions of constraint research. Sharing the vision of Saraswat [7], we are, in particular, interested in the modeling of large-scale and complex CSPs, and this process should be *compositional*. Similar to the case of programming in the large, there is an urgent need for good ways of assembling large CSP models out of manageable pieces.

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 but not so much related to CSP models. Thus these module systems facilitate the construction of large constraint programs in a particular language, but not of CSP models. As a result, a programmer must have come up with a rather complete picture of the model without modularization support and help, before she can start the programming task. This is in contradiction with conventional programming practice, in which modeling and programming can be carried out concurrently and intermixingly. 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*[1] to be a pair (X, D_X), where $X = \{x_1, \ldots, x_n\}$ 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

[1] Geelan [3] used the notion of viewpoint loosely without actually defining it.

$\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}$ 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).

Suppose M_1 and M_2 are two different models of the same problem P. By definition, there exists a one-one mapping between $sol(M_1)$ and $sol(M_2)$. We say that M_1 and M_2 are *mutually redundant*. As we shall see, it is possible for mutually redundant models M_1 and M_2 to share the same viewpoint. In that special case, it is easy to verify that $sol(M_1) = sol(M_2)$.

3 Operators over CSP Models

We are interested in operators in the space of CSP models. In other words, the operands (termed *constituent models* hereafter) and the results are all CSP models. In this section, we introduce several such operators and give the set-theoretic semantics and properties of these operators. While some form of model intersection and union is available in all constraint programming systems, model channeling and induction generalize the techniques found useful in exploiting model redundancy [1, 5]. The negation and complementation operators, although sounding familiar, offer two ways of handling negative knowledge.

3.1 Model Intersection

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.

Given two models $M_1 = ((X_1, D_{X_1}), C_{X_1})$ and $M_2 = ((X_2, D_{X_2}), C_{X_2})$, the viewpoint $V = (X, D_X)$ of the conjuncted model contains variables from both viewpoints, i.e., $X = X_1 \cup X_2$. If a variable x appears only in either X_1 or X_2, then the domain of x remains the same in D_X; otherwise, $D_X(x)$ is the *intersection* of $D_{X_1}(x)$ and $D_{X_2}(x)$. The constraint set C_X is the union of C_{X_1} and C_{X_2}. More formally, the conjuncted model $M_1 \cap M_2$ is $((X, D_X), C_{X_1} \cup C_{X_2})$, where $X = X_1 \cup X_2$ and for all $x \in X$,

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

We overload the $\cap$ operator so that it operates on CSP models as well as sets.

Example 1. Suppose we have two models $M_1 = ((X_1, D_{X_1}), C_{X_1})$, where

- $X_1 = \{x_1, x_2, x_3\}$ and $D_{X_1}(x_1) = D_{X_1}(x_2) = D_{X_1}(x_3) = \{1, 2, 3\}$,
- $C_{X_1} = \{x_1 \neq x_3\}$,

and $M_2 = ((X_2, D_{X_2}), C_{X_2})$, where

- $X_2 = \{x_2, x_3, x_4\}$ and $D_{X_2}(x_2) = D_{X_2}(x_3) = D_{X_2}(x_4) = \{1, 2\}$,
- $C_{X_2} = \{x_2 = x_3\}$.

Then we have as the conjuncted model $M_1 \cap M_2 = ((X, D_X), C_X)$, where

- $X = \{x_1, \ldots, x_4\}$ and $D_X(x_1) = \{1, 2, 3\}, D_X(x_2) = D_X(x_3) = D_X(x_4) = \{1, 2\}$,
- $C_X = \{x_1 \neq x_3, x_2 = x_3\}$.

□

A consequence of the definition is that every solution of a conjuncted model must satisfy all constraints in its constituent models.

Theorem 1. *If V is the viewpoint of $M_1 \cap M_2$, then $sol(M_1 \cap M_2) = \{\theta_1 \cup \theta_2 | \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V)\}$.*

Proof. Let θ_1 and θ_2 be complete assignments of M_1 and M_2 respectively.

$$\begin{aligned} & \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow\ & \forall c_1 \in C_{X_1} \cdot [c_1\theta_1] \wedge \forall c_2 \in C_{X_2} \cdot [c_2\theta_2] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow\ & \forall c_1 \in C_{X_1} \cdot [c_1(\theta_1 \cup \theta_2)] \wedge \forall c_2 \in C_{X_2} \cdot [c_2(\theta_1 \cup \theta_2)] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow\ & \forall c \in C_X \cdot [c(\theta_1 \cup \theta_2)] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow\ & \theta_1 \cup \theta_2 \in sol(M_1 \cap M_2) \end{aligned}$$

□

Theorem 1 gives a construction of the solution set of $M_1 \cap M_2$ from those of M_1 and M_2. In particular, solutions θ_1 of M_1 and θ_2 of M_2 can be combined to form a solution of $M_1 \cap M_2$ if and only if each shared variable in θ_1 and θ_2 is assigned the same value in θ_1 and θ_2. Otherwise, $\theta_1 \cup \theta_2$ cannot form a compound assignment in the conjuncted viewpoint. This condition is enforced by the *cmpd* predicate.

When two models $M_1 = (V, C_1)$ and $M_2 = (V, C_2)$ share the same viewpoint, $M_1 \cap M_2 = (V, C_1 \cup C_2)$. Furthermore, if M_1 and M_2 are mutually redundant, then $sol(M_1) = sol(M_2) = sol(M_1 \cap M_2)$.

Adding a new constraint to the constraint store, a common operation in constraint programming, can be considered a special application of model intersection, in which M_1 is the CSP already defined in the constraint store and M_2 consists of only the newly added constraint c. The restriction in this case is that the variables in c (the only ones in M_2) must be also in M_1 with the same domains. Our definition of model intersection, however, can handle shared variables in the two models having different domains.

3.2 Model Union

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.

Given two models $M_1 = ((X_1, D_{X_1}), C_{X_1})$ and $M_2 = ((X_2, D_{X_2}), C_{X_2})$, the viewpoint $V = (X, D_X)$ of the disjuncted model contains variables from both viewpoints, i.e., $X = X_1 \cup X_2$. If a variable x appears only in either X_1 or X_2, then the domain of x remains the same in D_X; otherwise, $D_X(x)$ is the *union* of $D_{X_1}(x)$ and $D_{X_2}(x)$.

The constraints in the disjuncted model are constructed in such a way that we can express the idea "either all constraints of C_{X_1} are satisfied or all constraints of C_{X_2} are satisfied," which is equivalent to $\bigwedge C_{X_1} \vee \bigwedge C_{X_2}$. By simple application of the distributive law, we can construct C_X as a set of constraints of the form $c_1 \vee c_2$, for every pair of $c_1 \in C_{X_1}$ and $c_2 \in C_{X_2}$. Such a constraint has $sig(c_1) \cup sig(c_2)$ as signature, and is violated only when both c_1 and c_2 are violated since $(c_1 \vee c_2)\theta = (c_1\theta) \vee (c_2\theta)$. More formally, the disjuncted model $M_1 \cup M_2$ is $((X, D_X), \{c_1 \vee c_2 | c_1 \in C_{X_1} \wedge c_2 \in C_{X_2}\})$, where $X = X_1 \cup X_2$ and for all $x \in X$,

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

We overload the $\cup$ operator so that it operates on CSP models as well as sets. Note that model union results in quadratic expansion of the number of constraints.

Example 2. Consider the same two models M_1 and M_2 in the model intersection example. The disjuncted model $M_1 \cup M_2$ is $((X, D_X), C_X)$, where

- $X = \{x_1, \ldots, x_4\}$, and $D_X(x_1) = D_X(x_2) = D_X(x_3) = \{1, 2, 3\}, D_X(x_4) = \{1, 2\}$,
- $C_X = \{(x_1 \neq x_3 \vee x_2 = x_3)\}$.

□

We can *project* a compound assignment θ in U onto $U' \subseteq U$ using:

$$\pi_{U'}(\theta) = \{\langle x, a\rangle \,|\, x \in U' \wedge \langle x, a\rangle \in \theta\}.$$

Conversely, we can *extend* a compound assignment θ to a set of complete assignments with respect to a viewpoint $V = (X, D_X)$ using:

$$ext(\theta, V) = \{\theta' | cmpd(\theta', V) \wedge sig(\theta') = X \wedge \pi_{sig(\theta)}(\theta') = \theta\}.$$

By overloading the *ext* operator, extension of a set of compound assignments Θ, all of which have the same signature, with respect to a viewpoint V is:

$$ext(\Theta, V) = \bigcup_{\theta \in \Theta} ext(\theta, V).$$

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.

Theorem 2. *If V is the viewpoint of $M_1 \cup M_2$, then $sol(M_1 \cup M_2) \supseteq ext(sol(M_1), V) \cup ext(sol(M_2), V)$.*

Proof. Let θ_1 be a complete assignment for $M_1 \cup M_2$.

$$\begin{aligned}
&\theta_1 \in ext(sol(M_1), V) \\
\Rightarrow\ &\pi_{X_1}(\theta_1) \in sol(M_1) \\
\Rightarrow\ &\forall c_1 \in C_{X_1} \cdot [c_1 \pi_{X_1}(\theta_1)] \\
\Rightarrow\ &\forall c_1 \in C_{X_1} \cdot [c_1 \theta_1] \\
\Rightarrow\ &\forall c_1 \in C_{X_1}, c_2 \in C_{X_2} \cdot [c_1 \theta_1 \vee c_2 \theta_1] \\
\Rightarrow\ &\forall c \in C_X \cdot [c \theta_1] \\
\Rightarrow\ &\theta_1 \in sol(M_1 \cup M_2)
\end{aligned}$$

Similarly, we can prove that $\theta_2 \in ext(sol(M_2), V) \Rightarrow \theta_2 \in sol(M_1 \cup M_2)$. Therefore, $\theta \in ext(sol(M_1), V) \cup ext(sol(M_2), V) \Rightarrow \theta \in sol(M_1 \cup M_2)$. □

When a compound assignment satisfies all the constraints of one constituent model, we can extend it to a solution of the disjuncted model. There are more solutions for the disjuncted model than those obtained using such extensions. The additional solutions originate from the possibility of enlarging the domains of shared variables. These extra domain values, which are not present in one of the constituent models, might well become part of a new solution to the constituent model, which can then be extended to a solution of the disjuncted model. Therefore, the disjuncted model can have a higher number of solutions than the solutions of the constituent models combined. This phenomenon is demonstrated in our example: $\theta = \{\langle x_1, 3\rangle, \langle x_2, 3\rangle, \langle x_3, 3\rangle, \langle x_4, 1\rangle\}$ is a solution of $M_1 \cup M_2$ but $\pi_{X_1}(\theta) = \{\langle x_1, 3\rangle, \langle x_2, 3\rangle, \langle x_3, 3\rangle\}$ and $\pi_{X_2}(\theta) = \{\langle x_2, 3\rangle, \langle x_3, 3\rangle, \langle x_4, 1\rangle\}$ are not solutions of M_1 and M_2 respectively. This is a result of the extra value 3 with respect to the domain of x_2 and x_3 in (X_2, D_{X_2}). The compound assignment $\{\langle x_2, 3\rangle, \langle x_3, 3\rangle\}$, which is not allowed in M_2, satisfies $x_2 = x_3$ and hence also $x_1 \neq x_3 \vee x_2 = x_3$ in $M_1 \cup M_2$.

In Theorem 2, equality occurs when the domains of every shared variable are the same in both viewpoints of the constituent models. Note that this is trivially fulfilled when there are no shared variables.

Theorem 3. *Let X_s be the set of shared variables of M_1 and M_2. If $D_{X_1}(x) = D_{X_2}(x)$ for all $x \in X_s$, then $sol(M_1 \cup M_2) = ext(sol(M_1), V) \cup ext(sol(M_2), V)$, where V is the viewpoint of $M_1 \cup M_2$.*

Proof. Let θ be a complete assignment in viewpoint V.

$$
\begin{aligned}
&\theta \in sol(M_1 \cup M_2) \\
\Leftrightarrow\ &\forall c \in C_X \cdot [c\theta] \\
\Leftrightarrow\ &\forall c_1 \in C_{X_1}, c_2 \in C_{X_2} \cdot [c_1\theta \vee c_2\theta] \\
\Leftrightarrow\ &\forall c_1 \in C_{X_1} \cdot [c_1\theta] \vee \forall c_2 \in C_{X_2} \cdot [c_2\theta] \\
\Leftrightarrow\ &(\forall c_1 \in C_{X_1} \cdot [c_1\theta] \wedge cmpd(\pi_{X_1}(\theta), V_1)) \vee (\forall c_2 \in C_{X_2} \cdot [c_2\theta] \wedge cmpd(\pi_{X_2}(\theta), V_2)) \\
\Leftrightarrow\ &\pi_{X_1}(\theta) \in sol(M_1) \vee \pi_{X_2}(\theta) \in sol(M_2) \\
\Leftrightarrow\ &\theta \in ext(sol(M_1), V) \cup ext(sol(M_2), V)
\end{aligned}
$$

□

$M \cup M \neq M$ in general, but Theorem 3 ensures that $M \cup M$ and M have the same solution set.

Corollary 1. $sol(M \cup M) = sol(M)$

Model union, as defined, again generalizes the disjunction facilities provided in most constraint programming systems, which allow programmers to form only disjunctive constraints. This works at the constraint level while our operator works at the model level, the granularity of which can be controlled easily and accordingly.

3.3 Model Channeling

Given two models $M_1 = (V_1, C_{X_1})$ and $M_2 = (V_2, C_{X_2})$, where $V_1 = (X_1, D_{X_1})$ and $V_2 = (X_2, D_{X_2})$, 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 the viewpoints of the models. In other words, channeling constraints set forth a relationship between the possible complete assignments of the two viewpoints.

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 $((X, D_X), C_{X_1} \cup C_{X_2} \cup C_c)$, where $X = X_1 \cup X_2$ and for all $x \in X$,

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

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.

Theorem 4. *If V is the viewpoint of $M_1 \overset{C_c}{\bowtie} M_2$, then $sol(M_1 \overset{C_c}{\bowtie} M_2) = \{\theta_1 \cup \theta_2 | \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V) \wedge \forall c \in C_c \cdot [c(\theta_1 \cup \theta_2)\}$.*

Proof. Similar to the proof of Theorem 1. □

The redundant modeling approach by Cheng *et al.* [1] is a special case of model channeling.

Theorem 5. *Suppose M_1 and M_2 are mutually redundant models. If C_c is a set of channeling constraints enforcing the one-one mapping between $sol(M_1)$ and $sol(M_2)$, then M_1, M_2, and $M_1 \overset{C_c}{\bowtie} M_2$ are mutually redundant to one another.*

Proof. By definition, any solution θ of $M_1 \overset{C_c}{\bowtie} M_2$ can be projected to V_1 and V_2 to form solutions of M_1 and M_2 respectively. Suppose θ_1 is a solution of M_1. Then there must be θ_2 that can be mapped to θ_1 by C_c (and *vice versa*) and is a solution of M_2. In addition, $cmpd(\theta_1 \cup \theta_2, V)$, where V is the viewpoint of $M = M_1 \overset{C_c}{\bowtie} M_2$, must hold since $\theta_1 \cup \theta_2$ satisfies C_c and shared variables in θ_1 and θ_2 must share the same assignments. Therefore, there is a one-one mapping between $sol(M_1)$ and $sol(M)$. Similarly, we can show the same between $sol(M_2)$ and $sol(M)$. □

3.4 Model Induction

Model induction [5] is a method for systematically generating a new model from an existing model, using another viewpoint and channeling constraints. The resulting model is called an *induced model*. The core of model induction is a meaning-preserving transformation for constraints, both implicit and explicit, from one model to constraints in another viewpoint.

Not all arbitrary sets of channeling constraints can be used in model induction. Given viewpoints $V_1 = (X, D_X)$ and $V_2 = (Y, D_Y)$. A necessary condition is that the set of channeling constraints between V_1 and V_2 must collectively define a *total* and *injective* function f from the possible assignments in V_1 to those in V_2: $f : \{\langle x, a\rangle \mid x \in X \wedge a \in D_X(x)\} \to \{\langle y, b\rangle \mid y \in Y \wedge b \in D_Y(y)\}$. In other words, f maps every assignment in V_1 to a unique assignment in V_2.

Description of model induction assumes constraints to be represented extensionally. We define $rel(c)$ to be the *relation* of a constraint c, and $rel(c)$ is stored explicitly as a set of incompatible assignments in $sig(c)$. Suppose $sig(c) = \{x_{i_1}, \ldots, x_{i_k}\}$, an *incompatible assignment* $\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}$ for c is a compound assignment in $sig(c)$ that makes c false. Otherwise, it is a *compatible assignment*. The incompatible assignment has the logical meaning that $\neg((x_{i_1} = a_1) \wedge \ldots \wedge (x_{i_k} = a_k))$. Hence, a constraint check amounts to a set membership check against $rel(c)$: $c\theta \Leftrightarrow \pi_{sig(c)}(\theta) \notin rel(c)$ for all valid θ.

Suppose we have a model $M = ((X, D_X), C_X)$, a viewpoint (Y, D_Y), and a set of channeling constraints defining a total and injective function f from the possible assignments in (X, D_X) to those in (Y, D_Y). We note that a CSP M contains two types of constraints: the explicit constraints as stated in C_X and the implicit constraints on variable assignments. The latter type of constraints can be further broken down into the restriction that (1) each variable must be assigned a value from its associated domain and (2) each variable cannot be assigned more than one value from its domain. The idea of model induction is to transform the constraints in model M, both implicit and explicit, using f to constraints C_Y in viewpoint (Y, D_Y), yielding the induced model $i(f, M) = ((Y, D_Y), C_Y)$.

- **Stated Constraints.** The first type of constraints to transform is the constraints stated in C_X. Recall that a constraint c consists of a signature and a relation, which is simply a set of incompatible assignments for c. We apply f on the assignments in each incompatible assignment of all constraints in C_X, and collect the transformed incompatible assignments in a set S_Y:

  $$S_Y = \{\theta | \theta = \{f(\langle x_{i_1}, a_1\rangle), \ldots, f(\langle x_{i_k}, a_k\rangle)\} \wedge c \in C \wedge \\ \{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\} \in rel(c) \wedge cmpd(\theta, (Y, D_Y))\}.$$

  It is indeed possible for θ not being a compound assignment with, say, $f(\langle x_{i_u}, a_u\rangle)$ and $f(\langle x_{i_v}, a_v\rangle)$ being $\langle y, b_u\rangle$ and $\langle y, b_v\rangle$, where $y \in Y$ and $b_u \neq b_v$. Since we are transforming incompatible assignments from (X, D_X), the information conveyed in θ, including the restriction that the variable y cannot be assigned values b_u and b_v simultaneously, is correct. In fact, this information is already satisfied implicitly in viewpoint (Y, D_Y) so that we can ignore/discard θ.

  Example 3. Suppose we have a constraint $c \in C_X$ with $\{\langle x_1, 1\rangle, \langle x_2, 2\rangle\} \in rel(c)$. If $f(\langle x_1, 1\rangle) = \langle y_2, 1\rangle$ and $f(\langle x_2, 2\rangle) = \langle y_1, 1\rangle$, then $\{f(\langle x_1, 1\rangle), f(\langle x_2, 2\rangle)\} = \{\langle y_1, 1\rangle, \langle y_2, 1\rangle\} \in S_Y$. If $\{\langle x_1, 1\rangle, \langle x_2, 1\rangle\} \in rel(c)$ and $f(\langle x_2, 1\rangle) = \langle y_2, 2\rangle$, then $\{f(\langle x_1, 1\rangle), f(\langle x_2, 1\rangle)\} = \{\langle y_2, 1\rangle, \langle y_2, 2\rangle\}$ is not a compound assignment and is not in S_Y. □

- **No-Double-Assignment Constraints.** Implicit in a CSP formulation, each variable should be assigned exactly one value. Part of this restriction can be translated to the requirement that no variables can be assigned two values from its domain at the same time. This corresponds to a set of (invalid) incompatible assignments of the form $\{\langle x, a\rangle, \langle x, b\rangle\}$ for all $x \in X$ and all $a, b \in D_X(x)$, which is satisfied implicitly and not represented in M. Their transformed counterparts, however, are needed in (Y, D_Y). We apply f on all these assignment sets, and collect the transformed incompatible assignments in a set N_Y:

  $$N_Y = \bigcup_{x \in X} \{\{f(\langle x, a\rangle), f(\langle x, b\rangle)\} \mid a, b \in D_X(x) \wedge a \neq b \wedge \\ cmpd(\{f(\langle x, a\rangle), f(\langle x, b\rangle)\}, (Y, D_Y))\}$$

  Example 4. Suppose $D_X(x_1) = \{2, 3\}$, $f(\langle x_1, 2\rangle) = \langle y_1, 2\rangle$, and $f(\langle x_1, 3\rangle) = \langle y_2, 1\rangle$. Then, for variable x_1, we have $\{f(\langle x_1, 2\rangle), f(\langle x_1, 3\rangle)\} = \{\langle y_1, 2\rangle, \langle y_2, 1\rangle\} \in N_Y$. □

– **At-Least-One-Assignment Constraints.** The other part of the implicit variable constraint in M can be translated to the requirement that each variable must be assigned at least a value from its domain. This corresponds to the constraints $\bigvee_{b \in D_X(x)} x = b$ for all $x \in X$, which are satisfied implicitly and not represented in M. The other problem is that this unary constraint does not have any incompatible assignments. For each variable $x \in X$, we first apply f to every possible assignment of x. Suppose $D_X(x) = \{b_1, \ldots, b_r\}, f(\langle x, b_1 \rangle) = \langle y_{k_1}, v_1 \rangle, \ldots, f(\langle x, b_r \rangle) = \langle y_{k_r}, v_r \rangle$. These assignments form the compatible assignments of a constraint in $\{y_{k_1}, \ldots, y_{k_r}\}$. Using the closed world assumption, we compute the incompatible assignments by collecting all compound assignments θ with signature $\{y_{k_1}, \ldots, y_{k_r}\}$ such that every individual assignment in θ is not equal to $\langle y_{k_j}, v_j \rangle$ for all $j \in \{1, \ldots, r\}$:

$$\begin{aligned} A_Y = \textstyle\bigcup_{x \in X} \{\theta \mid & D_X(x) = \{b_1, \ldots, b_r\} \wedge \\ & \forall j \in \{1, \ldots, r\} \cdot [f(\langle x, b_j \rangle) = \langle y_{k_j}, v_j \rangle] \wedge \\ & \theta = \{\langle y_{w_1}, a_1 \rangle, \ldots, \langle y_{w_s}, a_s \rangle\} \wedge cmpd(\theta, (Y, D_Y)) \wedge \\ & sig(\theta) = \{y_{k_1}, \ldots, y_{k_r}\} \wedge \\ & \forall i \in \{1, \ldots, s\}, \forall j \in \{1, \ldots, r\} \cdot [\langle y_{w_i}, a_i \rangle \neq \langle y_{k_j}, v_j \rangle]\} \end{aligned}$$

Example 5. Suppose $D_X(x_1) = D_Y(y_1) = D_Y(y_2) = \{1, 2, 3\}, f(\langle x_1, 1 \rangle) = \langle y_1, 1 \rangle, f(\langle x_1, 2 \rangle) = \langle y_1, 2 \rangle$, and $f(\langle x_1, 3 \rangle) = \langle y_2, 1 \rangle$. Then, for variable x_1, applying f to the assignments $\langle x_1, 1 \rangle, \langle x_1, 2 \rangle$, and $\langle x_1, 3 \rangle$ suggests that the incompatible assignments are among y_1 and y_2. All compound assignments $\{\langle y_1, a \rangle, \langle y_2, b \rangle\}$ where $a \neq 1, a \neq 2$, and $b \neq 1$ are the incompatible assignments in A_Y. Hence, we have $\{\langle y_1, 3 \rangle, \langle y_2, 2 \rangle\}$ and $\{\langle y_1, 3 \rangle, \langle y_2, 3 \rangle\} \in A_Y$. □

We note that the incompatible assignments in a constraint $c \in C_X$ may be transformed to contribute to the incompatible assignments of more than one constraint in (Y, D_Y). Thus $S_Y \cup N_Y \cup A_Y$ consists of all the induced incompatible assignments with different signatures in (Y, D_Y). The next step is to extract incompatible assignments with the same signature from $S_Y \cup N_Y \cup A_Y$ and group them into a constraint in (Y, D_Y). Thus,

$$C_Y = \{c \mid sig(c) \subseteq Y \wedge rel(c) = \sigma_{sig(c)}(S_Y \cup N_Y \cup A_Y) \neq \emptyset\},$$

where $\sigma_U(\Theta) = \{\theta \mid \theta \in \Theta \wedge sig(\theta) = U\}$.

Due to page limitation, readers are referred to Law and Lee [5] for a detailed example of model induction using the N-queens problem. The following theorem and corollaries, stated without proof by Law and Lee [5], gives an important consequence of model induction: the transformation of incompatible assignments is meaning-preserving.

Theorem 6. *If $M = (V_1, C_1)$ is a model for problem P, and V_2 is a proper viewpoint of P, then M and $i(f, M)$ are mutually redundant models for all total and injective functions f (defined by channeling constraints connecting V_1 and V_2) mapping from possible assignments in V_1 to those in V_2.*

Proof. Suppose there is a set of channeling constraints defining a total and injective function f such that M and $i(f, M)$ are not mutually redundant. By definition, there is no one-one mapping between $sol(M)$ and $sol(i(f, M))$ and thus also between $sol(P)$ and $sol(i(f, M))$ (since M is a model for P). Thus no subset S of the set of all possible complete assignments in V_2 can have a one-one mapping with $sol(i(f, M))$. Consequently, V_2 is not a proper viewpoint of P; hence a contradiction. □

Corollary 2. *If $M_1 = (V_1, C_1)$ and $M_2 = (V_2, C_1)$ are mutually redundant models of P, and f is a total and injective function mapping from possible assignments in V_1 to those in V_2, then $sol(M_2) = sol(i(f, M_1))$.*

Proof. By Theorem 6, M_1 and $i(f, M_1)$ are mutually redundant. So are M_1, M_2, and $i(f, M_1)$. By definition, there must be a one-one mapping between $sol(M_2)$ and $sol(i(f, M_1))$. Since M_2 and $i(f, M_1)$ share the same viewpoint V_2, the only possible one-one mapping is the identity mapping for V_2 to be proper. Thus $sol(M_2) = sol(i(f, M_1))$. □

Corollary 3. *If $M = (V_1, C_1)$ is a model for problem P, V_2 is a proper viewpoint of P, and f is a total and bijective function (i.e., f^{-1} exists) mapping from possible assignments in V_1 to those in V_2, then $sol(i(f^{-1}, i(f, M))) = sol(M)$.*

Proof. By Theorem 6, M and $i(f, M)$ are mutually redundant. Similarly, $i(f, M)$ and $i(f^{-1}, i(f, M))$ are mutually redundant. M and $i(f^{-1}, i(f, M))$ share the same viewpoint. By Corollary 2, $sol(i(f^{-1}, i(f, M))) = sol(M)$. □

3.5 Model Negation

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.

Example 6. Consider the model M_1 in the model intersection example. The negated model $\neg M_1$ has the same viewpoint as M_1 and a single constraint $\neg(x_1 \neq x_3)$, which is equivalent to $x_1 = x_3$. □

Unfortunately, solutions of $\neg M$ cannot be constructed from solutions of M, but negation does neutralize each other by the fact that $(\neg(\neg c))\theta = \neg((\neg c)\theta) = \neg(\neg(c\theta)) = c\theta$.

Theorem 7. *$\neg(\neg M) = M$ for all models M.*

3.6 Model Complementation

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 $\bigvee\{\neg c | c \in C\}$.

Example 7. Suppose we have a model $M = ((\{x_1, x_2, x_3\}, D_X), \{x_1 \neq x_3, x_2 = x_3\})$, where $D_X(x_1) = D_X(x_2) = D_X(x_3) = \{1, 2, 3\}$. Then the complemented model $\overline{M}$ has exactly the same viewpoint as M and contain only the constraint $(x_1 = x_3 \vee x_2 \neq x_3)$. □

Solutions of M and $\overline{M}$ partition the set of all possible complete assignments for (the viewpoint of) M.

Theorem 8. *If $M = ((X, D_X), C)$, then $\theta \in sol(M) \Leftrightarrow \theta \notin sol(\overline{M})$ for all θ with $X = sig(\theta)$.*

Proof. For all θ with $sig(\theta) = X$,

$$\begin{aligned} \theta \in sol(M) \Leftrightarrow \forall c \in C \cdot [c\theta] \Leftrightarrow \forall c \in C \cdot [\neg\neg(c\theta)] \\ \Leftrightarrow \forall c \in C \cdot [\neg((\neg c)\theta)] \Leftrightarrow \neg\exists c \in C \cdot [(\neg c)\theta)] \Leftrightarrow \theta \notin sol(\overline{M}) \end{aligned}$$

□

Whenever we compute $\overline{\overline{M}}$ from $M = (V, C)$, we apply the simplification

$$\begin{aligned} (\neg(\textstyle\bigvee_{c \in C} \neg c))\theta = \neg((\textstyle\bigvee_{c \in C} \neg c)\theta) = \neg(\textstyle\bigvee_{c \in C} (\neg c)\theta) \\ = \neg(\textstyle\bigvee_{c \in C} \neg(c\theta)) = (\textstyle\bigwedge_{c \in C} \neg\neg(c\theta)) = \textstyle\bigwedge_{c \in C} c\theta. \end{aligned}$$

so that $\overline{\overline{M}}$ contains the constraints $\{c | c \in C\}$, which is the same as in M, instead of the constraint $\neg\bigvee\{\neg c | c \in C\}$. By definition, complementation annihilates the effect of another.

Theorem 9. *$\overline{\overline{M}} = M$ for all models M.*

4 Algebraic Structures

In this section, we give further properties and 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* [4] $(G, \odot)$ 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

• $M_1 \cap M_2 = M_2 \cap M_1$	• $M_1 \cup (M_2 \cap M_3) = (M_1 \cup M_2) \cap (M_1 \cup M_3)$
• $(M_1 \cap M_2) \cap M_2 = M_1 \cap (M_2 \cap M_3)$	
• $M \cap E_\emptyset = M$	• $M_1 \overset{C_c}{\bowtie} M_2 = M_2 \overset{C_c}{\bowtie} M_1$
• $M \cap M = M$	• $(M_1 \overset{C_{c1}}{\bowtie} M_2) \overset{C_{c2}}{\bowtie} M_3 = M_1 \overset{C_{c1}}{\bowtie} (M_2 \overset{C_{c2}}{\bowtie} M_3)$
	• $M_1 \overset{\emptyset}{\bowtie} M_2 = M_1 \cap M_2$
• $M_1 \cup M_2 = M_2 \cup M_1$	
• $(M_1 \cup M_2) \cup M_3 = M_1 \cup (M_2 \cup M_3)$	• $\neg(M_1 \cap M_2) = \neg M_1 \cap \neg M_2$
• $M \cup E_\perp = M$	
• $M \cup E_\emptyset = (V, \emptyset)$	

Model union distributes over intersection. We provide a sketch of the proof.

Theorem 10. *$M_1 \cup (M_2 \cap M_3) = (M_1 \cup M_2) \cap (M_1 \cup M_3)$.*

Proof. Let C_1, C_2, C_3 be the constraints of M_1, M_2, and M_3 respectively. The set of constraints of $(M_1 \cup M_2) \cap (M_1 \cup M_3)$ is:

$$\begin{aligned} &\{c_1 \vee c_2 | c_1 \in C_1 \wedge c_2 \in C_2\} \cup \{c_1 \vee c_3 | c_1 \in C_1 \wedge c_3 \in C_3\} \\ = &\{c_1 \vee c' | c_1 \in C_1 \wedge c' \in C_2 \cup C_3\}, \end{aligned}$$

which is equal to the constraints of $M_1 \cup (M_2 \cap M_3)$. □

However, intersection does not distribute over union. Consider M_1, M_2, and M_3 with constraints $\{x_1 \neq x_3\}, \{x_2 = x_3\}$, and $\{x_3 < x_4\}$ respectively. $M_1 \cap (M_2 \cup M_3)$ contains constraints $\{x_1 \neq x_3, x_2 = x_3 \vee x_3 < x_4\}$, while $(M_1 \cap M_2) \cup (M_1 \cap M_3)$ contains constraints $\{x_1 \neq x_3, x_1 \neq x_3 \vee x_3 < x_4, x_2 = x_3 \vee x_1 \neq x_3, x_2 = x_3 \vee x_3 < x_4\}$. Their solutions, however, are the same because the extra constraints $x_1 \neq x_3 \vee x_3 < x_4$ and $x_2 = x_3 \vee x_1 \neq x_3$ are subsumed by the constraint $x_1 \neq x_3$.

Inheriting from intersection, model channeling is commutative and associative. There can be no restrictions on the assignments between the two viewpoints of the constituent models. In such case, the set C_c of channeling constraints is empty. Model channeling degenerates to model intersection. There is a special rule for model negation and intersection. Negation and intersection interact in a counterintuitive fashion and defy the well-known de Morgan's Law in Boolean algebra.

Suppose $M_1 = (V_1, C_1)$ and $M_2 = (V_2, C_2)$ are mutually redundant models with different viewpoints handcrafted by human modeler. We also have a set C_c of channeling constraints defining a total and injective function f from possible assignments of V_1 to those of V_2. Model induction essentially translates constraint information expressed in V_1 to V_2 via channeling constraints f. The transformed constraints express in V_2 the constraint information of the problem as viewed from V_1. These transformed constraints are likely different from constraints expressed directly using V_2 by the human modeler. Therefore, $i(f, M_1)$ and M_2 are redundant and yet supplementary to each other.

Model channeling and intersection give various possibilities to combine M_1, M_2, and models induced from the two models. Law and Lee [5] suggest that, assuming that f^{-1} *exists*, the following three classes of model combinations can enhance constraint propagation utilizing redundant information in the constituent models.

- $i(f, M_1) \cap M_2$ and $M_1 \cap i(f^{-1}, M_2)$
- $M_1 \overset{C_c}{\bowtie} i(f, M_1)$ and $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$
- $(i(f, M_1) \cap M_2) \overset{C_c}{\bowtie} i(f^{-1}, i(f, M_1) \cap M_2)$ and
  $(i(f^{-1}, M_2) \cap M_1) \overset{C_c}{\bowtie} i(f, i(f^{-1}, M_2) \cap M_1)$

In the first class, we use model intersection to obtain redundancy information. Model intersection is "additive" in that it merges constraints to form stronger constraints, which is the source of increased constraint propagation. In the second class, we explore redundancy information using model channeling. Model channeling is "collaborative" in nature. It allows the constituent models to perform constraint propagation on its own, and yet communicate their results (variable instantiation and domain pruning) to the other constituent models to possibly initiate further constraint propagation. Furthermore, model channeling allows constraint propagation to explore different variable spaces (viewpoints). The third class makes use of both model intersection and channeling. It enjoys the best of both worlds. Each of the constituent of the channeled model is a conjuncted model, encompassing strengthened constraints obtained from model intersection. The conjuncted models are then connected via model channeling to take advantage of the sharing of pruning information and constraint propagation in different viewpoints.

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. This formal study also helps advancing our understanding of model redundancy, and when redundant information can arise and is useful. 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.

Acknowledgements

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).

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. E.C. Freuder. In pursuit of the holy grail. *CONSTRAINTS*, 2:57–62, 1997.
3. P.A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 31–35, 1992.
4. T.W. Hungerford. *Algebra.* Springer-Verlag, 1974.
5. Y.C. Law and J.H.M. Lee. Model induction: a new source of CSP model redundancy. In *(to appear) Proceedings of the 18th National Conference on Artificial Intelligence*, 2002.
6. A.K. Mackworth. Consistency in networks of relations. *AI Journal*, 8(1):99–118, 1977.
7. V. Saraswat. Compositional computing. *CONSTRAINTS*, 2:95–97, 1997.