# Model Induction: a New Source of CSP Model Redundancy

**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**

Based on the notions of viewpoints, models, and channeling constraints, the paper introduces model induction, a systematic transformation of constraints in an existing model to constraints in another viewpoint. Meant to be a general CSP model operator, model induction is useful in generating redundant models, which can be further induced or combined with the original model or other mutually redundant models. We propose three ways of combining redundant models using model induction, model channeling, and model intersection. Experimental results on the Langford's problem confirm that our proposed combined models exhibit improvements in efficiency and robustness over the original single models.

**Keywords**: CSP, Problem Formulation, Model Redundancy

## Introduction

The task at hand is to tackle *Constraint Satisfaction Problems* (CSPs) (Mackworth 1977), which are, in general, NP-hard. 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 (1997) lists problem modeling among the seven most important future directions of constraint research.

Selecting the most appropriate formulation or model for a problem is difficult in general. In fact, no objective and general notions of the "best" formulation exist to date. Different formulations of a problem do not compete. Cheng *et al.* (1999) introduce channeling constraints and present how these constraints can be used to connect mutually redundant CSP models to enhance constraint propagation in tree search. In this paper, we give another use of channeling constraints, namely to use them in generating additional model of a CSP through a process called *model induction*. Before we can present model induction, we define CSP models formally based on the notion of CSP viewpoints. We differentiate two different kinds of mutually redundant models, those sharing and those not sharing the same viewpoint. We give three ways of combining the newly generated induced model with their redundant counterparts. To demonstrate the feasibility of our proposal, we compare our combined models against (1) the original models and (2) the redundant models proposed by Cheng *et al.* (1999) using the Langford's problem, which can be formulated as a Permutation CSP. We demonstrate improvement in terms of execution efficiency, number of failures, and robustness.



## 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 function that maps each $x \in X$ to its 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 $j \in \{i_1, \ldots, i_k\}$. Note the requirement that no variables can be assigned more than one value in a compound assignment. 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[2] between $S$

[1] Geelan (1992) used the notion of viewpoint loosely and informally without actually defining it.

[2] A problem can contain identical objects, in which case we have to apply artificial markings to the objects to help differentiating these objects before we can model the problem as a CSP. We assume that the problem objects are already appropriately marked, if necessary, before we talk about such a mapping.

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 $c$ in $V$ has two attributes. The *signature* $sig(c) \subseteq X$ defines the scope of $c$, while $rel(c)$ is the *relation* of $c$. For the purpose of this paper, we assume that $rel(c)$ 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*. It has the logical meaning that $\neg((x_{i_1} = a_1) \wedge \ldots \wedge (x_{i_k} = a_k))$. Therefore a constraint check amounts to a set membership check against $rel(c)$. We abuse terminology by saying that the incompatible assignment $\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}$ also has a signature: $sig(\{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\}) = \{x_{i_1}, \ldots, x_{i_k}\}$.

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 $sig(c_i) = sig(c_j)$. We can *project* a compound assignment $\theta$ in $U$ onto $U' \subseteq U$ using:

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

A *solution* of $M = (V, C)$ is a complete assignment $\theta$ in $V$ so that $\pi_{sig(c)}(\theta) \notin rel(c)$ for each $c \in C$. 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* models. 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 $sol(M_1) = sol(M_2)$.

## Model Induction

In this section, we introduce *model induction*: 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. In the following, we describe channeling constraints, construction and properties of induced models, and a detailed example.

### Channeling Constraints

Given two models $M_1 = ((X, D_X), C_X)$ and $M_2 = ((Y, D_Y), C_Y)$. Cheng *et al.* (1999) define a *channeling constraint* $c$ to be a constraint, where $sig(c) \nsubseteq X$, $sig(c) \nsubseteq Y$, $sig(c) \subseteq X \cup Y$, and $c \notin C_X \cup C_Y$. Thus, $c$ relates $M_1$ and $M_2$ by limiting the combination of values that their variables can take. Cheng *et al.* show how a collection of channeling constraints can be used to connect two mutually redundant models of the same problem to form a combined model, which exhibits increased constraint propagation and thus improved efficiency.

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 assignments of the two viewpoints. 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 $V_2$:

$$\begin{aligned} f \,:\, &\{\langle x, a\rangle \mid x \in X \wedge a \in D_X(x)\} \\ &\rightarrow \{\langle y, b\rangle \mid y \in Y \wedge b \in D_Y(y)\} \end{aligned}$$

In other words, $f$ maps every assignment in $V_1$ to a unique assignment in $V_2$.

### Induced Models

Given 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* $i(f, M) = ((Y, D_Y), C_Y)$. We show further that if $M$ is a model for problem $P$ and $(Y, D_Y)$ is also a proper viewpoint of $P$, then $M$ and $i(f, M)$ are mutually redundant.

- **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 assignments of all constraints in $C_X$, and collect the transformed assignments in a set $S_Y$:

$$\begin{aligned} S_Y = \{&\{f(\langle x_{i_1}, a_1\rangle), \ldots, f(\langle x_{i_k}, a_k\rangle)\} \mid \\ &c \in C \wedge \{\langle x_{i_1}, a_1\rangle, \ldots, \langle x_{i_k}, a_k\rangle\} \in rel(c) \wedge \\ &cmpd(\{f(\langle x_{i_1}, a_1\rangle), \ldots, f(\langle x_{i_k}, a_k\rangle)\}, (Y, D_Y))\} \end{aligned}$$

where the predicate $cmpd$ ensures that the set of assignments $\theta = \{f(\langle x_{i_1}, a_1\rangle), \ldots, f(\langle x_{i_k}, a_k\rangle)\}$ forms a compound assignment in $(Y, D_Y)$. It is indeed possible for $\theta$ 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 $\theta$, 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 $\theta$.

- **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 assignments in a set $N_Y$:

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

- **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 $\bigvee_{b \in D_X(x_i)} x_i = b$ for all $x_i \in X$, 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 $x_i \in X$, we first apply $f$ to every possible assignment of $x_i$. Suppose $D_X(x_i) = \{b_1, \ldots, b_r\}, f(\langle x_i, b_1\rangle) = \langle y_{k_1}, v_1\rangle, \ldots, f(\langle x_i, b_r\rangle) = \langle y_{k_r}, v_r\rangle$. 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 $\theta$ with signature $\{y_{k_1}, \ldots, y_{k_r}\}$ such that every individual assignment in $\theta$ is not equal to $\langle y_{k_j}, v_j\rangle$ for all $j \in \{1, \ldots, r\}$:

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

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 limitation of space, we state without proof an important consequence of model induction: the transformation of incompatible assignments is meaning-preserving. In other words, if $M$ is a model for problem $P$ and the viewpoint of the induced model is proper with respect to $P$, then the induced model is also a model for $P$.

**Theorem 1** *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 $V_2$.*

**Corollary 1** *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 $sol(M_2) = sol(i(f, M_1))$.*

**Corollary 2** *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 $sol(i(f^{-1}, i(f, M))) = sol(M)$.*

## Example

We illustrate the construction of induced model using the simple 4-queens problem, which is to place four queens on a $4 \times 4$ chessboard in such a way that no two queens can attack each other.

We give a textbook model $M = ((X, D_X), C_X)$ of the 4-queens problem. We use four variables $X = \{x_1, x_2, x_3, x_4\}$ and their associated domain function $D_X$. Each $x_i$ denotes the column position of the queen on row $i$ and $D_X(x_i) = \{1, 2, 3, 4\}$ for $i \in \{1, 2, 3, 4\}$. The constraints $C_X$ enforce that no two queens can be on the same:

- column: $x_i \neq x_j$ for all $1 \leq i < j \leq 4$, and
- diagonal: $|x_i - x_j| \neq j - i$ for all $1 \leq i < j \leq 4$.

Next, we consider a 0-1 viewpoint $(Z, D_Z)$ with sixteen variables $Z = \{z_{ij} \mid i, j \in \{1, 2, 3, 4\}\}$ and associated domain function $D_Z$. The assignment $\langle z_{ij}, 1\rangle$ denotes the fact that square position $(i, j)$ (row $i$ and column $j$) contains a queen; and $\langle z_{ij}, 0\rangle$ denotes otherwise. Therefore, $D_Z(z_{ij}) = \{0, 1\}$ for all $i, j \in \{1, 2, 3, 4\}$. The set of channeling constraints $x_i = j \Leftrightarrow z_{ij} = 1$ for all $i, j = 1, \ldots, 4$ defines the total and injective function

$$g(\langle x_i, j\rangle) = \langle z_{ij}, 1\rangle \text{ for all } i, j \in \{1, 2, 3, 4\}.$$

We first transform the stated constraints in $C_X$. The incompatible assignments for the diagonal constraints in $M$ have the form $\{\langle x_i, k\rangle, \langle x_j, k \pm (i - j)\rangle\}$ for all $i, j, k \in \{1, 2, 3, 4\}, i < j$, and $1 \leq k \pm (i - j) \leq 4$. Hence, the induced incompatible assignments are $\{\langle z_{ik}, 1\rangle, \langle z_{j,k \pm (i-j)}, 1\rangle\}$. For example, the constraints $|x_1 - x_2| \neq 2 - 1$ generates the incompatible assignments $\{\langle z_{11}, 1\rangle, \langle z_{22}, 1\rangle\}$, $\{\langle z_{12}, 1\rangle, \langle z_{23}, 1\rangle\}$, $\{\langle z_{13}, 1\rangle, \langle z_{24}, 1\rangle\}$, $\{\langle z_{12}, 1\rangle, \langle z_{21}, 1\rangle\}$, $\{\langle z_{13}, 1\rangle, \langle z_{22}, 1\rangle\}$, $\{\langle z_{14}, 1\rangle, \langle z_{23}, 1\rangle\}$ for inclusion in $S_Z$. The column constraints can be transformed similarly.

The No-Double-Assignments constraints for $(Z, D_Z)$ include incompatible assignments transformed from the implicit constraints that each $x_i \in X$ cannot be assigned two different values. Thus:

$$\begin{aligned} N_Z &= \bigcup_{x_i \in X} \{\{g(\langle x_i, j_1\rangle), g(\langle x_i, j_2\rangle)\} \mid j_1, j_2 \in \{1, \ldots, 4\} \\ &\quad \wedge j_1 < j_2\} \\ &= \{\{\langle z_{ij_1}, 1\rangle, \langle z_{ij_2}, 1\rangle\} \mid i, j_1, j_2 \in \{1, \ldots, 4\} \wedge j_1 < j_2\} \end{aligned}$$

For example, the implicit requirement for $x_1 \in X$ will generate the following incompatible assignments $\{\langle z_{11}, 1\rangle, \langle z_{12}, 1\rangle\}$, $\{\langle z_{11}, 1\rangle, \langle z_{13}, 1\rangle\}$, $\{\langle z_{11}, 1\rangle, \langle z_{14}, 1\rangle\}$, $\{\langle z_{12}, 1\rangle, \langle z_{13}, 1\rangle\}$, $\{\langle z_{12}, 1\rangle, \langle z_{14}, 1\rangle\}$, $\{\langle z_{13}, 1\rangle, \langle z_{14}, 1\rangle\}$ to ensure that no more than one queen will be placed on row $i$ of the chessboard.

Last but not least, we need to take care of the At-Least-One-Assignment constraints $A_Z$, which are obtained from the implicit constraints "each $x_i \in X$ must be assigned at least one value" in $M$. Applying $g$ to the assignments $\langle x_i, 1\rangle, \ldots, \langle x_i, 4\rangle$ for each $x_i \in X$ suggests that the incompatible assignments in $(Z, D_Z)$ are among variables $z_{i1}, \ldots, z_{i4}$. The incompatible assignments are those $\{\langle z_{i1}, q_1\rangle, \ldots, \langle z_{i4}, q_4\rangle\}$ such that $q_1 \neq 1, \ldots, q_4 \neq 1$. Since the domain of all variables $z_{ij}$ is only $\{0, 1\}$, $\{\langle z_{i1}, 0\rangle, \ldots, \langle z_{i4}, 0\rangle\}$ is the only incompatible assignment needed. Thus:

$$\begin{aligned} A_Z = \{&\{\langle z_{11}, 0\rangle, \langle z_{12}, 0\rangle, \langle z_{13}, 0\rangle, \langle z_{14}, 0\rangle\}, \\ &\{\langle z_{21}, 0\rangle, \langle z_{22}, 0\rangle, \langle z_{23}, 0\rangle, \langle z_{24}, 0\rangle\}, \\ &\{\langle z_{31}, 0\rangle, \langle z_{32}, 0\rangle, \langle z_{33}, 0\rangle, \langle z_{34}, 0\rangle\}, \\ &\{\langle z_{41}, 0\rangle, \langle z_{42}, 0\rangle, \langle z_{43}, 0\rangle, \langle z_{44}, 0\rangle\}\}. \end{aligned}$$

The intuitive meaning of these incompatible assignments is that there cannot be no queens in row $i$ of the chess board.

The induced model $i(g, M) = ((Z, D_Z), C_Z)$ can be formed by extracting and grouping incompatible assignments of the same signatures to form constraints in $C_Z$. By Theorem 1, $i(g, M)$ and $M$ are mutually redundant, and are both models of the 4-queens problem.

## Exploiting Redundancy from Model Induction

Although model induction is an interesting and general model operator in its own right, we leave the study of its algebraic properties and interaction with other model operators as a topic of another paper. In this section, we focus our interest on induced models which are mutually redundant to their original models. We propose three ways of combining the redundant models so as to utilize the redundant information in enhancing constraint propagation.

### Combining Redundant Models

Given a problem $P$, $M_1 = ((X_1, D_{X_1}), C_{X_1})$ and $M_2 = ((X_2, D_{X_2}), C_{X_2})$ are mutually redundant models of $P$ with different viewpoints. Suppose there is a set $C_c$ of channeling constraints connecting variables in $X_1$ and $X_2$. Following the redundant modeling approach (Cheng *et al.* 1999), we can form a combined model $M = M_1 \overset{C_c}{\bowtie} M_2 = (V, C)$, where

- $X = X_1 \cup X_2$ and $V = (X, D_X)$,
- for all $x \in X$,

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

- $C = \{c \mid c' \in C_{X_1} \cup C_{X_2} \cup C_c\}$ with $sig(c) = sig(c')$ and $rel(c) = \{\theta \mid \theta \in c' \wedge cmpd(\theta, V)\}$.

The following theorems about the *model channeling* operation are straightforward to verify.

**Theorem 2** $M_1 \overset{C_c}{\bowtie} M_2 = M_2 \overset{C_c}{\bowtie} M_1$

**Theorem 3** *$M_1$, $M_2$, and $M$ are mutually redundant to one another.*

When we are given two mutually redundant models that share the same viewpoint, the situation is simpler. We propose *model intersection* as a means to combine the models into one. Suppose $M_1 = (V, C_1)$ and $M_2 = (V, C_2)$. We can form $M = M_1 \cap M_2 = (V, C_1 \cup C_2)$. Again, we have the following theorems for model intersection.

**Theorem 4** $M_1 \cap M_2 = M_2 \cap M_1$.

**Theorem 5** $sol(M_1) = sol(M_2) = sol(M)$.

When intersecting two models, we have the option of *merging constraints* with the same signature into one constraint by taking the union of the constraints' sets of incompatible assignments. For example, suppose $sig(c_1) = sig(c_2)$, we can construct a merged constraint $c'$ to replace $c_1$ and $c_2$ such that $sig(c') = sig(c_1) = sig(c_2)$ and $rel(c') = rel(c_1) \cup rel(c_2)$. The resultant constraint $c'$, having a *more* global view on the variables in $sig(c')$, can potentially provide more constraint propagation than the individual constraints $c_1$ and $c_2$ when used separately. Note that constraint merging is applicable, not just in the context of model intersection, whenever we have more than one individual constraint with the same signature in a CSP.

### Three New Forms of Model Redundancy

A viewpoint can greatly influence how a human modeler looks at a problem. Each viewpoint provides a distinct perspective emphasizing perhaps a specific aspect of the problem. Therefore, the modeler will likely express individual constraints differently under different viewpoints, although the constraints under each viewpoint should collectively give the same solutions to the problem being modeled. In particular, a constraint expressed for one viewpoint might not even have an (explicit) counterpart in the other viewpoint, and *vice versa*.

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 complementary to each other.

Model channeling and intersection give various possibilities to combine $M_1$, $M_2$, and models induced from the two models. Model channeling is "collaborative" in nature. It allows the sub-models to perform constraint propagation on its own, and yet communicate their results (variable instantiation and domain pruning) to the other sub-models to possibly initiate further constraint propagation. Furthermore, model channeling allows constraint propagation to explore different variable spaces (viewpoints). Model intersection is "additive" in that it merges constraints to form stronger constraints, which is the source of increased constraint propagation.

Assuming $f^{-1} exists$, we propose three classes of interesting combined 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 $(i(f^{-1}, M_2) \cap M_1) \overset{C_c}{\bowtie} (i(f, i(f^{-1}, M_2) \cap M_1))$

We note that $f^{-1}$ always exists for *Permutation CSPs* (Geelen 1992; Smith 2000; 2001). In a Permutation CSP $((X, D_X), C)$, we always have $D_X(x_i) = D_X(x_j)$ for all $x_i, x_j \in X$, and $|D_X(x_i)| = |X|$. In addition, any solution $\{\langle x_1, k_1\rangle, \ldots, \langle x_n, k_n\rangle\}$ of a Permutation CSP must have the property that $k_i \neq k_j \Leftrightarrow i \neq j$.

### Example

We give an example application where such application of model induction and model redundancy are possible. The Langford's problem, listed as "prob024" in CSPLib (Gent & Walsh 1999), can be modeled as a Permutation CSP having all the desired properties for experimenting with model induction and redundant modeling.

In the Langford's problem, there is an $m \times n$-digit sequence which includes the digits 1 to $n$, with each digit occurs $m$ times. There is one digit between any consecutive pair of digit 1, two digits between any consecutive pair of digit 2, ..., $n$ digits between any consecutive pair of digit $n$. The Langford's problem, denoted as $(m, n)$ problem, is to find such a sequence (or all sequences).

Smith (2000) suggests two ways to model the the Langford's problem a CSP. We use the $(3, 9)$ instance to illustrate the two models. In the first model $M_1$, we use 27 variables $X = \{x_0, \ldots, x_{26}\}$, which we can think of as $1_1, 1_2, 1_3, 2_1, \ldots, 9_2, 9_3$. Here, $1_1$ represents the first digit 1 in the sequence, $1_2$ represents the second digit 1, and so on. The domain of these variables are the values that represent the positions of a digit in the sequence. We use $\{0, \ldots, 26\}$ to represent the domain. Hence, we have the viewpoint $V_1 = (X, D_X)$, where $D_X(x_i) = \{0, \ldots, 26\}$ for $i \in \{0, \ldots, 26\}$. Due to space limitation, we skip the description of constraints.

In the second model $M_2$, we again use 27 variables $Y = \{y_0, \ldots, y_{26}\}$ to represent each position in the sequence. Their domains are $\{0, \ldots, 26\}$, whose elements correspond to the digits $1_1, 1_2, 1_3, 2_1, \ldots, 9_2, 9_3$. Hence, we have the viewpoint $V_2 = (Y, D_Y)$, where $D_Y(y_i) = \{0, \ldots, 26\}$ for $i \in \{0, \ldots, 26\}$.

We can write the channeling constraints $C_c$ connecting $V_1$ and $V_2$ as $x_i = j \Leftrightarrow y_j = i$ for all $i, j = 0, \ldots, 26$. These constraints define a total and bijective function $f$ where $f(\langle x_i, j\rangle) = \langle y_j, i\rangle$ for all valid $i, j$. With $M_1$, $M_2$, and $f$, we can construct the three proposed classes of combined models. We note that, for the special case of the Langford's problem:

- $i(f, M_1 \cap i(f^{-1}, M_2)) = i(f, M_1) \cap M_2$ and
- $i(f^{-1}, i(f, M_1) \cap M_2) = M_1 \cap i(f^{-1}, M_2)$.

It is thus only necessary to consider

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

for the third class of combined models.

## Experiments

To verify the feasibility and efficiency of our proposal, we realize and evaluate the various models for the $(3, 9)$ and $(3, 10)$ instances of the Langford's problem using ILOG Solver 4.4 (1999) and running on a Sun Ultra 1/170 workstation with 512M of memory. We use the $\mathrm{IlcTableConstraint}$ function (ILOG 1999) to create constraints from sets of incompatible assignments. Full arc consistency is enforced in constraint propagation. Variables are chosen using the smallest-domain-first variable-ordering heuristic. Constraints are merged whenever possible, so that every constraint in a model has a different signature.

Table 1 shows our comparison results. Column 1 gives the models. In models with more than one viewpoint, it suffices to search/label variables of either viewpoint, although one may choose to search on both. In column 2, we give also the search variables. Columns 3 and 4 report the execution results of solving for only the first solution of the $(3, 9)$ and $(3, 10)$ instances respectively, while columns 5 and 6 report the results of solving for all solutions. Each cell contains both the number of fails and CPU time in sec (in bracket) of an execution. A cell labeled with "-" means that execution does not terminate within 20 minutes of CPU time. We also highlight in bold the best result of each column.

Each row corresponds to a particular model. We divide the models into five groups, the first two of which are used as control in the experiment. The first group consists of individual models, while the second group consists of combined models constructed using the redundant modeling approach (Cheng *et al.* 1999). The remaining groups correspond to our three proposed classes of combined models.

In analyzing the results, attention is sought not just on the CPU time, but also on the number of fails. In fact, the latter is more important and accurate as a measure of the robustness of a model. Combined models are bigger in size, and higher execution overhead is expected. The idea of combining redundant models is to spend more time in constraint propagation in the hope that the extra effort can result in substantial pruning of the search space. A model that gives more pruning has a higher possibility in solving problems that

| Models | Search Variables | First Solution | | All Solutions | |
|---|---|---|---|---|---|
| | | $(3,9)$ | $(3,10)$ | $(3,9)$ | $(3,10)$ |
| $M_1$ | $X$ | 192 (5.46) | 569 (19.02) | 938 (25.54) | 3114 (101.83) |
| $i(f, M_1)$ | $Y$ | - | - | - | - |
| $M_2$ | $Y$ | - | - | - | - |
| $i(f^{-1}, M_2)$ | $X$ | - | - | - | - |
| $M_1 \overset{C_c}{\bowtie} M_2$ | $X$ | 63 (6.49) | 193 (18.80) | 310 (25.73) | 1109 (98.24) |
| $M_1 \overset{C_c}{\bowtie} M_2$ | $Y$ | 44 (4.44) | 20 (4.25) | 322 (25.28) | 980 (88.05) |
| $M_1 \overset{C_c}{\bowtie} M_2$ | $X \cup Y$ | 39 (4.93) | 104 (13.47) | 225 (20.21) | 697 (71.26) |
| $M_1 \cap i(f^{-1}, M_2)$ | $X$ | 105 (4.87) | 313 (14.34) | 524 (**18.31**) | 1650 (71.46) |
| $i(f, M_1) \cap M_2$ | $Y$ | - | - | - | - |
| $M_1 \overset{C_c}{\bowtie} i(f, M_1)$ | $X$ | 73 (6.36) | 207 (19.98) | 401 (29.88) | 1221 (112.14) |
| $M_1 \overset{C_c}{\bowtie} i(f, M_1)$ | $Y$ | 47 (**4.19**) | 21 (**3.21**) | 338 (27.91) | 1021 (96.85) |
| $M_1 \overset{C_c}{\bowtie} i(f, M_1)$ | $X \cup Y$ | 42 (4.69) | 113 (13.58) | 239 (21.52) | 730 (76.76) |
| $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$ | $Y$ | 589 (54.60) | 475 (51.73) | 1462 (126.45) | 5547 (578.26) |
| $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$ | $X$ | 115 (14.49) | 340 (48.73) | 561 (62.51) | 1812 (249.37) |
| $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$ | $X \cup Y$ | 113 (14.23) | 338 (48.42) | 550 (61.79) | 1788 (246.65) |
| $(M_1 \cap i(f^{-1}, M_2)) \overset{C_c}{\bowtie} (i(f, M_1) \cap M_2)$ | $X$ | 38 (6.53) | 132 (17.80) | 195 (22.49) | 745 (84.91) |
| $(M_1 \cap i(f^{-1}, M_2)) \overset{C_c}{\bowtie} (i(f, M_1) \cap M_2)$ | $Y$ | 31 (5.15) | **9** (5.12) | 217 (22.27) | 665 (78.57) |
| $(M_1 \cap i(f^{-1}, M_2)) \overset{C_c}{\bowtie} (i(f, M_1) \cap M_2)$ | $X \cup Y$ | **29** (5.72) | 83 (14.18) | **156** (19.04) | **514** (**65.74**) |

Table 1: Comparison Results Using the Langford's Problem

are otherwise computationally infeasible when expressed in weaker models.

The first and fifth groups of models represent the two ends of a spectrum, which indicates the amount of model redundancy utilized in the models. The single models in the first group use no redundancy, and thus performs the worst in terms of the number of fails. Their execution times are not among the worst since these models are the smallest in size, incurring the least execution overhead in constraint propagation. Note also that model $M_2$ is a poor model. Any model involving $M_2$ as a base model is bound to perform poorly, both in terms of CPU time and number of fails. In the following, we focus on only models using $M_1$ as a base model.

The second group makes use of only model channeling, which helps $M_1$ and $M_2$ share pruning and variable instantiation information. Constraint propagation also takes place in both viewpoints. Another advantage of this approach is that constraints in $M_1$ and $M_2$, constructed under different viewpoints, are complementary to each other. These characteristics are the source of increased constraint propagation, and thus drastic cut in the number of fails as compared to the models in the first group.

The third group of models uses only one viewpoint, but model intersection combines the constraints from the two models to form stronger constraints, thus entailing again more constraint propagation. We note, however, that the reduction in the number of fails is not as substantial as the case in the second group of models.

The fourth group of models employs both model induction and model channeling. The models inherit the good characteristics of model channeling, except that the constraints in both models are essentially from $M_1$. These models are deprived of the chance to share constraint information from $M_2$. Therefore, the performance of the fourth group is consistently and slightly worse than that of the second group.

The model in the fifth group enjoys the best of both worlds. Each of the sub-models is a combined model, encompassing strengthened constraints obtained from model intersection. The combined models are then connected via model channeling to take advantage of the sharing of pruning information and constraint propagation in different viewpoints. That explains why models in this group always give the lowest number of fails in all benchmarks. Their timings, although not the fastest, are also respectable compared to the fastest time of the respective benchmark, although these models are the largest in size.

## Related Work

Rossi *et al.* (1990) propose a new definition of equivalence of CSPs, based on the concept of mutual reducibility. They believe that it is reasonable to consider two CSPs equivalent if it is possible to obtain the solution of one CSP from that of another, and vice versa. Geelen (1992) introduces two improved problem-independent value and variable ordering heuristics for solving CSPs. He also introduces a "dual-viewpoint" approach for Permutation CSPs. This approach allows suitable extensions to many heuristics including those introduced in his paper. Jourdan (1995) works on multiple modeling, in which models representing different but redundant views of the same problem are synchronized using the communication mechanisms of constraint logic programming and concurrent constraint languages. Weigel *et al.* (1998) introduces an algorithm to transform a CSP into its boolean form which is then used to find its reformulations. Reformulations differ with each other only in redundant constraints, and one can allow pruning in some situations which is not possible in other. Cheng *et al.* (1999) formally introduces redundant modeling. Two models of the same problem are combined together using channeling constraints. They show increased constraint propagation and efficiency by using this approach. Smith (2000; 2001) introduces the idea of minimal dual models for Permutation CSPs. It is similar to redundant modeling but the constraints in the second model is dropped. She shows that for the Langford's problem, the amount of propagation of the minimal dual model is equal to that of redundant modeling. However, it is not clear whether the same result can be transferred to Permutation CSPs in general. Walsh (2001) conducts an extensive theoretical and empirical study on using different models and combined models using channeling constraints. Smith and Walsh's works concentrate on the effect of different levels of constraint propagation on the constraints to ensure a permutation in Permutation CSPs.

## Conclusion

Model induction gives a systematic way of generating alternate model in a different viewpoint from an existing model. Hand-crafting CSP model is an unamiable task performed daily by human modelers, who should find model induction a useful tool. An interesting application of model induction is to generate redundant models, which can be combined using model channeling and intersection. Benchmark results using the Langford's problem confirm that the proposed combined models are robust and efficient, both in terms of CPU time and number of fails.

Model redundancy is a relatively new concept. We take the work reported in this paper as a means to open up new possibilities to study, understand, and apply model redundancy in constraint satisfaction. There is plenty of scope for future work. First, model induction is applicable to general CSPs, although our empirical results are developed for only the Langford's problem. It will be interesting to check if the same techniques can be applied/generalized to other, not necessarily binary and/or Permutation, CSPs to obtain useful redundancy information. Second, we conjecture that our approach, as in the case of redundant modeling, is useful mainly for tight and highly connected CSPs. This property can be verified, perhaps, with the help of randomly generated CSPs. Third, model induction is defined in terms of extensional representation of constraints. There is no reason why we have to solve the resultant CSPs also in the extensional form. It would be worthwhile to study how the intensional (symbolic) representation of a constraint can be learned from its extensional counterpart. Fourth, it is also important to characterize the extra amount of constraint propagation provided by combining mutually redundant models.

## Acknowledgements

We had fruitful discussions about minimal dual models, channeling constraints, and proper viewpoints with Barbara Smith. We thank also the anonymous referees for their constructive comments which help improve the quality of the paper. 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

Cheng, B. M. W.; Choi, K. M. F.; Lee, J. H. M.; and Wu, J. C. K. 1999. Increasing constraint propagation by redundant modeling: an experience report. *Constraints* 4(2):167–192.

Freuder, E. 1997. In pursuit of the holy grail. *CONSTRAINTS* 2:57–62.

Geelen, P. A. 1992. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings of the 10th European Conference on Artificial Intelligence*, 31–35.

Gent, I., and Walsh, T. 1999. CSPLib: A benchmark library for constraints. In *Proceedings of Principles and Practice of Constraint Programming (CP)*, 480–481. Available at http://www-users.cs.york.ac.uk/~tw/csplib/.

ILOG. 1999. *ILOG Solver 4.4 Reference Manual*.

Jourdan, J. 1995. *Concurrent constraint multiple models in CLP and CC languages: Toward a programming methodology by modelling*. Ph.D. Dissertation, Denis Diderot University, Paris VII.

Mackworth, A. 1977. Consistency in networks of relations. *AI Journal* 8(1):99–118.

Rossi, F.; Petrie, C.; and Dhar, V. 1990. On the equivalence of constraint satisfaction problems. In *Proceedings of the 9th European Conference on Artificial Intelligence*, 550–556.

Smith, B. M. 2000. Modelling a permutation problem. Research Report 2000.18, School of Computer Studies, University of Leeds.

Smith, B. M. 2001. Dual models in permutation problems. In *Proceedings of Principles and Practice of Constraint Programming (CP)*, 615–619.

Walsh, T. 2001. Permutation problems and channelling constraints. In *Proceedings of Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, 377–391.

Weigel, R., and Bliek, C. 1998. On reformulation of constraint satisfaction problems. In *Proceedings of 13th European Conference on Artificial Intelligence*, 254–258.