Towards the Systematic Generation of Channelling Constraints

Student: Bernadette Martínez-Hernández
Supervisor: Alan M. Frisch

Artificial Intelligence Group, Dept. of Computer Science, Univ. of York, York, UK

Abstract. The automatic modelling tool Conjure generates CSP models from problem specifications. The generated models may contain multiple alternative (redundant) representations of the same variable. However, the combined models generated by Conjure do not include the channelling constraints needed to maintain the consistency between the alternative representations. In this paper we show that by using only the tools already provided by the Conjure system we can generate correct channelling constraints.

1 Introduction

In this paper we introduce a method to systematically generate channelling constraints for the automatic modelling tool Conjure. The basic concepts related to modelling and channelling are presented below.

Solving a problem using CSP technology requires mapping its informal description (often in natural language) into a formal description in a particular formal system adequate for constraint solving. This process is called *modelling*. In many cases the conceptual gap between the informal problem description and the constraint program is still large.

Modelling is a hard task. We may find that a problem is easily modelled with a variable whose domain is unsupported by current solvers, for example the Sonet problem [1] that requires assigning nodes to a group of rings. This problem is easily modelled with a variable whose domain is integrated by multisets of sets of an integer range. Modellers must use variables with supported domains to represent and implement variables with unsupported domains. Following the example, we can use a two-dimensional matrix of integer variables to represent the multiset of Sonet. To ensure the soundness of this transformation we have to add a collection of `alldifferent` constraints, one for each represented set of nodes. The addition of constraints is a common procedure in this kind of transformation.

In an attempt to reduce the burden of hand-crafted modelling, several systems have been created to automate some of the modelling decisions. One of these systems is Conjure [2], a system that automatically refines problem specifications into CSP models. Variables in the specifications may have domains currently unsupported by solvers. Also, the elements of the domains may be arbitrarily compound, for example, sets of sets, sets of sets of sets. Conjure uses

a set of *refinement rules* to compositionally transform the variables (and constraints) into their representations that can be implemented in current solvers. We briefly describe Conjure in Section 2.

Modellers often come up with several alternative (redundant) representations of the same variable. In Section 3 we discuss the definition of representation and give some examples of alternative representations of the same variable. Good modellers know that combining alternative representations in the same model can improve propagation, among other benefits. To maintain the consistency between these simultaneous alternatives we need to add *channelling constraints* [3] to the model. Section 4 discuses channelling constraints (also called channels) between alternative representations.

Conjure produces several different alternative representations of the same variable. It can also produce these alternatives simultaneously in the same model. Currently, Conjure does not automatically produce the channelling constraints to maintain the consistency between the alternative representations. We show in Section 5 that we can generate those channels systematically. More importantly we can use Conjure for the generation, providing that Conjure ensures the input of a refinement rule is correctly represented by the output.

2 Refinement and Conjure

Specifications for Conjure are given in Essence version one (EV1), a specification language that allows variables to have an associated domain whose elements are either integers, Booleans, sets, multisets, functions, relations, partitions or tuples (among others). Unlike other languages like *F* [4], Essence allows domains to be nested to arbitrary depth, for example sets of integers, sets of sets of integers, and so forth. The current implemented version of Conjure refines only variables whose domains are (arbitrarily nested) sets or multisets. It is planed to extend this implementation to support all the range of domains of Essence. A full description of the Essence language and the performance of Conjure is given in `http://www.cs.york.ac.uk/aig/constraints/AutoModel/`

There are often several refinement rules that can be applied to an expression. Hence, if a variable appears more than once in the constraints of an Essence specification, Conjure may generate models including several simultaneous alternative constructions related to this variable. In this cases Conjure annotates the model specifying the relation between the specification variable and its various related variables in the generated model.

3 Representations

The refinement rules of Conjure return correct representations of their given inputs. Before giving a definition of correct representation we must first introduce the *CSP instances*. A CSP instance is a concept similar to the the *model* used in the model algebra by Law et al [5].

Definition 1 *R=(V, C) is a* ***CSP instance*** *where V is a pair consisting of a set of the variables and their domains (viewpoint) and C is a (possibly empty) set of the constraints in P. For each constraint in C, its variables (and their domains) must be included in V.*

Example of CSP instance: The CSP instance **S1** (*set*) consists of a single variable S whose domain consists of all sets of size n whose elements are drawn from the range $a..b$; where a, b and n are integer numbers such that $a \leq b$ and $n > 0$. The set of constraints in **S1** is empty. This definition of **S1** and S is used throughout this document.

The following definition of representation is similar to the *variable representation* discussed by Jefferson and Frisch [6], and it is strongly related to the preservation of solutions.

Definition 2 *The tuple* $\langle R, R', \psi \rangle$ *is* ***a representation*** *of* R *under* R' *and* ψ *(*R' ***represents*** R *via* ψ*) if* $R' = (V', C')$ *and* $R = (V, C)$ *are CSP instances and* ψ *is a partial function from the total assignments of the variables in* V' *into the total assignments of the variables in* V *such that:*

- *For each total assignment* x' *of the variables in* V'*,* x' *satisfies the constraints in* C' ***if and only if*** $\psi(x')$ *is defined and satisfies the constraints in* C*,*
- *For each assignment* x *of the variables in* V *satisfying the constraints in* C*, exists an assignment* x' *of the variables in* V' *such that* $\psi(x') = x$*.*

Example of representation: The CSP instance **E1** (*explicit set*) is formed by an array of one dimension of integer variables $VaE1S$ indexed by $1..n$. Each element of the array has an integer domain $a..b$. The CSP instance includes constraints that ensure all the elements of the array $VaE1S$ are different (*allDifferent*). Let ψ_E be a function from the assignments of $VaE1S$ into the assignments of S. The application of ψ_E is defined only for arrays whose values are all different. The function ψ_E returns a set containing the values present in the array. Notice that ψ_E fulfills all the requisites of Definition 2, hence $\langle$**S1**, **E1**, $\psi_E\rangle$ is a representation of **S1** under **E1** and ψ_E.

Definition 3 R' ***is equivalent to*** R *via* ψ ***if and only if*** R' *represents* R *via* ψ *and* R *represents* R *via* ψ^{-1}*.*

Examples of equivalent representation: Any CSP instance R is equivalent to itself via the identity function. An example more related to sets is the CSP instance **O1** (*occurrence set*) that contains an array of one dimension of Boolean variables $VaO1S$ indexed by the integer range $a..b$. The constraint $sum(VaO1S) = n$ is the only element of the set of constraints. The function ψ_O from the assignments of $VaO1S$ to the assignments of S is defined only for the assignments satisfying $sum(VaO1S) = n$. ψ_O returns a set with the indexes of $VaO1S$ where the Boolean value **True** is assigned. It is not difficult to conclude that **O1** represents **S1** via ψ_O and **S1** represents **O1** via ψ_O^{-1}. Hence **O1** is equivalent to **S1** via ψ_O.

To comply with an strategy that is representation (and CSP instance) based we say that a specification variable forms the CSP instance consisting only of the variable and the empty set of constraints.

4 Alternative Representations and Channels

The channelling annotations are information tags used by Conjure to indicate the relation between the variables of a generated CSP instance and the variables of an input CSP instance. Following the examples of the previous section, when Conjure generates the representation **E1** of **S1** it uses the channelling annotation `represent` S `by` *expset*(*VaE1S*) to indicate that **E1** is the explicit representation of **S1**. If Conjure generates **O1** as a representation of **S1** it attaches the channelling annotation `represent` S `by` *occset*(*VaO1S*) to indicate **O1** is the explicit representation of **O1**.

Alternative representations of the same variable remain independent until we add channelling constraints to maintain the consistency between them. We present here a definition of channelling constraint similar to the one used by Jefferson [6].

Definition 4 *Let $S_1 = \langle R, R_1, \psi_1 \rangle$ and $S_2 = \langle R, R_2, \psi_2 \rangle$ be CSP instance representations of R. Let $vars(R_1)$ and $vars(R_2)$ be disjoint sets of variables. A set of constraints C_h are **channelling constraints between** S_1 **and** S_2 if:*

- *For each total assignment x_1 (of the variables in R_1) satisfying the constraints in R_1 there is at least one total assignments x_2 (of the variables in R_2) where the composed assignment $x_1 \cup x_2$ satisfies the constraints in C_h. Similarly for each satisfying x_2 there must be an assignment x_1 such that the composed assignment $x_1 \cup x_2$ satisfies the constraints in C_h.*
- *For all total assignments x_1 and x_2 where the composed assignment $x_1 \cup x_2$ satisfies the constraints in C_h, $\psi_1(x_1)$ and $\psi_2(x_2)$ are either both undefined or take the same value.*

Let us show the channels between the previous CSP instances. The CSP instances **S1** and **E1** are representations of **S1**. Notice that **S1** is a representation of **S1** via the identity function, named here ψ_{id}. The following constraint is a channelling constraint between **S1** and **E1**.

$$[\mathbf{CH}_{S1E1}] \qquad \forall_{j \in a..b}.\ (j \in S \Leftrightarrow \exists_{i \in 1..n}.\ VaE1S[i] = j)$$

It is clear that for every assignment of S there is at least one assignment of $VaE1S$ such that $S \cup VaE1S$ satisfies $\mathbf{CH}_{S1E1}$ and $VaE1S$ satisfies the constraints of **E1**. This is also satisfied the other way around. Furthermore, the definition of ψ_{id} and ψ_E ensure that $\psi(S)$ and $\psi_E(VaE1S)$ are either equal or both undefined when $S \cup VaE1S$ satisfies $\mathbf{CH}_{S1E1}$.

Similarly, the following constraint is a channelling constraint between **S1** and **O1**.

$$[\mathbf{CH}_{S1O1}] \qquad \forall_{j \in a..b}.\ (j \in S \Leftrightarrow VaO1S[j])$$

Suppose that CONJURE produces a model with both **E1** and **O1** as representations of **S1**. It is not very difficult to see the following channelling constraint to connect the alternative representations **E1** and **O1** is correct according to Definition 4.

$$[\mathbf{CH}_{E1O1}] \qquad \forall_{j \in a..b}.\ (VaO1S[j] \Leftrightarrow \exists_{i \in 1..n}.\ VaE1S[i] = j)$$

We show below that a CSP instance formed by two alternative representations of an initial CSP instance and their channelling constraints is also a representation the initial CSP instance.

Theorem 1 *Let $\langle R, R_1, \psi_1 \rangle$ and $\langle R, R_2, \psi_2 \rangle$ be variable disjoint representations of R and C_h a channelling constraint between R_1 and R_2. We can compose a function ψ from ψ_1 or ψ_2 such that $\langle R, R_1 \cup R_2 \cup C_h, \psi \rangle$ is a representation of R.*

Proof. Let ψ be the extended version of either ψ_1 or ψ_2 such that it can be applied to any total assignment x that instantiate the variables of $R_1 \cup R_2$. Let ψ return exactly the same values only when given an assignment that satisfies the constraints in C_h and be undefined in any other case. Therefore $R_1 \cup R_2 \cup C_h$ represents R via ψ. □

Notice that $\langle \mathbf{S1}, \mathbf{E1} \cup \mathbf{O1} \cup \mathbf{CH}_{E1O1}, \psi'_O \rangle$[1] is a representation of **S1**.

5 Systematic Generation of Channelling Constraints

The examples of the previous section show examples of channelling constraints generated with the information provided in the annotations. The generation of channelling constraints based on the annotations becomes challenging for human modellers when the input variable(s) have a deeply nested domain.

Let P be a specification refined by CONJURE into P' where the variable X in P has two representations X_1 and X_2 in P'. Suppose we can use the annotations to force CONJURE to produce only certain representations (with the same variable names and domain) for certain variables. Let Y be a new variable with exactly the same domain of X. The constraint $X = Y$ is obviously a channelling constraint between X and Y. We can then *re-refine* $X = Y$ forcing the X to refine into X_1 and the Y to refine into X_2. Such refinement produces a representation of the channelling constraints between X_1 and X_2. Hence, we can compose a representation of X, $\langle X, X_1 \cup X_2 \cup \rho_{X_1,X_2}(X = X), \psi_{X_1} \rangle$, where

[1] ψ'_O is the extended version of ψ_O.

$\rho_{X_1,X_2}(X = X)$ is the conditioned refinement of $X = Y$. We call this algorithm of generation the *post-processing algorithm*.

The following theorem states the correctness of the algorithm. The proof is left out for reasons of space.

Theorem 2 *If the rules of* Conjure *produce representations of their inputs then the* ***post-processing algorithms*** *generates correct models.*

6 Conclusion and Future Work

Based on generalisations over CSP instances we reduce the problem of the automatic generation of channelling constraints to the problem of proving that the refinement rules of Conjure generate only representations of its inputs.

It is important to notice that the post processing algorithm can be generalised to produce the channels between three or more representations of the same specification variable. We may even modify the strategy of generation and produce only some channelling constraints instead of all the multiple channels between these representations.

The work is far from being complete. The conditions and performance of the generation of channels for non-redundant (partial) representations are not included yet. Also, correct channels may need postprocessing for a better reading of the user and/or to generate efficient code for a CSP solver. What is more, due to the nature of the refinement process we may generate several valid alternative channelling constraints for the same kind of variables. This alternative generation increases the complexity of the model selection task.

References

1. A.M. Frisch, B. Hnich, I.M.B.M.S.T.W.: Transforming and refining abstract constraint specifications. In: CSCLP04: Joint Annual Workshop of ERCIM/Cologne on Constraint Solving and Constraint Logic Programming. (2004)
2. Frisch, A.M., Jefferson, C., Martínez-Hernández, B., Miguel, I.: The rules of constraint modelling. In: Nineteenth Int. Joint Conf. on Artificial Intelligence. (2005)
3. Cheng, B.M.W., Choi, K.M.F., Lee, J.H.M., Wu, J.C.K.: Increasing constraint propagation by redundant modeling: An experience report. Constraints **4** (1999) 167–192
4. Hnich, B.: Function Variables for Constraint Programming. PhD thesis, Computer Science Division, Department of Information Science, Uppsala University (2003)
5. Law, Y.C., Lee, J.H.M.: Algebraic properties of CSP model operators. In: Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems: Towards Systematisation and Automation (Held in Conjunction with CP-2002). (2002) 57–71 Shorter paper in CP-2002.
6. Jefferson, C., Frisch, A.M. Available from `http://www.cs.york.ac.uk/aig/constraints/AutoModel/` (2005)