Automated Formulation of Constraint Satisfaction Problems

Mihaela Sabin and **Eugene C. Freuder**

Department of Computer Science
University of New Hampshire
Durham, New Hampshire 03824, USA
mcs,ecf@cs.unh.edu

A wide variety of problems can be represented as constraint satisfaction problems (CSPs), and once so represented can be solved by a variety of effective algorithms. However, as with other powerful, general AI problem solving methods, we must still address the task of moving from a natural statement of the problem to a formulation of the problem as a CSP. This research addresses the task of automating this problem formulation process, using logic puzzles as a testbed. Beyond problem formulation per se, we address the issues of effective problem formulation, i.e. finding formulations that support more efficient solution, as well as incremental problem formulation that supports reasoning from partial information and are congenial to human thought processes.

A CSP is defined by a set of *variables* with their associated *domains of values* and a set of *constraints* which restrict the combinations of values allowed. The example in Fig. 1 shows a logic puzzle text and a corresponding constraint network representation. In CSP terms, associated with the introductory portion of the logic puzzle there are 8 variables, each variable with the same domain of values, the tower positions. All the variables are nodes in the constraint network with the values labeling them. Due to the structure of the problem (no two acrobats, as well as no two items, correspond to the same position in the tower), the CSP variables are partitioned into two *cliques*, `Acrobats` and `Items`, with disequality constraints (#) between every pair of variables in each clique (drawn as edges in the constraint network and marked as `initial` constraints in the figure).

The current implementation of the translation tool handles the translation of the clues into the CSP constraints. The translation scheme recognizes patterns such as *not*, *above* and *below*, that match logic puzzle clues in the input, and applies the translation rules to generate corresponding `clue` constraints. The figure shows the binary constraints as bold continuous lines, drawn as edges and labeled #, $<$ and $>$, and the unary constraints as deleted (hashed) values.

More efficient CSP formulations are possible by exploiting the inherent structure of the problem (e.g. the two cliques in our example) or the semantics of the constraints. The enhanced translation scheme defines specialized consistency functions attached to each constraint. As we build the representation, with each clue parsed, corresponding local consistency can be performed that may add `inferred` constraints, examples of which are shown in the figure as bold, dotted lines. Postponing search as much as possible while locally propagating the available, partial information seems to reflect human problem solving behavior. The acquired reasoning power is encoded in the form of restricted domains and additional constraints which support more efficient problem solving.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant No. IRI-9207633 and Grant No. IRI-9504316 and on work supported by Digital Equipment Corporation. This work profited from discussion with Mohammed H. Sqalli. We thank Nancy Schuster and Dell Magazines for permission to reproduce material from *Dell Logic Puzzles*.

Emie, Jed, Tex, and **Zeke** are 4 acrobats, each wearing one of the 4 items: **chaps, holster, hat,** or **vest,** and being placed in the **1st, 2nd, 3rd,** or **4th** position of the tower they form such that:

1. **Jed** is *not* the **1st**, but is *above* the one with the **hat**.
2. **Zeke** does *not* wear the **holster**.
3. The one in the **vest** is *not* the **1st**.
4. The one in the **chaps** is *below* **Zeke**, but *above* **Tex**.

Logic Puzzle Text

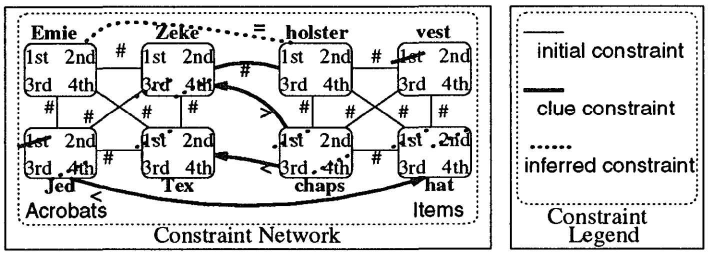

Figure 1. From logic puzzle statement to CSP formulation