Constraint Lingo: A program for solving logic puzzles and other tabular constraint problems.

Raphael Finkel, Victor W. Marek and Mirosław Truszczyński

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046, USA

1 Introduction

Constraint Lingo is a high-level language for specifying and solving tabular constraint problems [FMT01]. We show the syntax of this language through examples. Our software translates Constraint Lingo programs into a variety of back-end logic formalisms, including *smodels* [NS00], *dlv* [ELM+98], *ECLiPSe* [WNS97] and *aspps* [ET01]. The associated logic engine then generates a set of answers, each of which our software converts to a human-readable table.

2 Tabular constraint-satisfaction problems

Informally, a tabular constraint-satisfaction problem (tCSP) is one that has a specified number of rows and columns. The values in the table are subject to constraints, both implicit and explicit.

Logic puzzles are good examples of tCSPs, as are some graph problems. For example, we present a simplified version of the "French Phrases, Italian Soda" puzzle (or French puzzle, for short)[1]:

> Claude and five others (three women: Jeanne, Kate, and Liana, and two men: Martin and Robert) sat at a circular table. Each person described a trip to a different place. Each person sipped a different soda. Match each person with his or her seat (numbered one through six [circularly]) and determine the soda that each drank, as well as the place that each plans to visit.
>
> 1. The person who is planning a trip to Quebec, who drank either blueberry or lemon soda, didn't sit in seat number one.
> 2. Robert, who didn't sit next to Kate, sat directly across from the person who drank peach soda.
> 3. The three men are the person who is going to Haiti, the one in seat number three, and Claude's brother.
> 4. The three people who sat in even-numbered seats are Kate, Claude, and a person who didn't drink lemon soda, in some order.

[1] Copyright 1999, Dell Magazines; quoted by permission. We present only four of the nine clues.

A solution has five columns, representing `name`, `gender`, `position`, `soda` and `country` (each with its associated domain). Each row represents a particular combination, that is, a person of some gender sitting in some position, drinking some soda, and planning to visit some country. The implicit constraints include the legitimate values for each column (Haiti is a value that may occur only in the `country` column) and that all columns but `gender` are **key**: all the legitimate values are used exactly once. This solution satisfies all nine clues of the French puzzle:

name	gender	position	soda	country
claude	man	6	tangelo	haiti
jeanne	woman	1	grapefruit	ivory
kate	woman	4	kiwi	tahiti
liana	woman	5	peach	belgium
martin	man	3	lemon	quebec
robert	man	2	blueberry	martinique

3 Representation in Constraint Lingo

We encode the implicit constraints of the French puzzle by the following Constraint Lingo code.

```
CLASS person: claude jeanne kate liana martin robert
PARTITION gender: men women
CLASS position: 1 .. 6 circular
CLASS soda: blueberry lemon peach tangelo kiwi grapefruit
CLASS visits: quebec tahiti haiti martinique belgium ivory
```

These lines declare the names of the five columns, specify the values that may appear in those columns, and indicate whether the columns are key (by the indicator `CLASS`). They also indicate that the `position` column has numeric values to be treated with modular arithmetic.

We encode the explicit constraints involved in the statement of the problem and the four clues by the following Constraint Lingo code.

```
# from the statement of the problem
    AGREE men: martin robert
    AGREE women: jeanne kate liana

# clue 1
    CONFLICT quebec 1
    REQUIRED quebec blueberry OR quebec lemon

# clue 2
    OFFSET !+-1 position: robert kate
    OFFSET +-3 position: robert peach

# clue 3
```

```
    VAR brother
    CONFLICT brother claude
    AGREE men: haiti 3 brother
    CONFLICT haiti 3 brother

# clue 4
    VAR unlemon
    MATCH 2 4 6, kate claude unlemon
    CONFLICT unlemon lemon
```

Comments start with `#` and continue to the end of the line. Clue 1, which says that the person going to Quebec is not sitting in seat 1, becomes a single `CONFLICT` constraint. Both `quebec` and `1` are values in the table; the `CONFLICT` constraint indicates these values must be in distinct rows. A `REQUIRED` constraint indicates that two values appear in the same row. The `OFFSET` constraints in Clue 2 say that Robert and Kate are not in adjacent positions (circularly) and that Robert and the row identified with Peach are 3 positions apart (circularly). Other encodings require a touch of cleverness. Clue 3 talks about Claude's brother. We encode this person's row by a variable `brother` constrained to refer to a man other than Claude. The complex English statements of the French puzzle reduce to a small set of short, clear, constraints.

4 Applying Constraint Lingo to graph problems

Despite a restricted repertoire of operators aimed initially at solving logic problems, Constraint Lingo is sufficient to model such important combinatorial problems as independent sets, graph coloring, and finding Hamiltonian cycles.

An *independent set* in a graph is a set of v vertices no two of which share an edge. The independent-set problem is to find an independent set with at least k vertices. We represent the problem in the following Constraint Lingo program, setting $v = 100$ and $k = 30$, with edges (2, 5) and (54, 97), for concreteness. There are two attributes: a class `vertex`, to represent vertices of the graph (line 1 below) and a partition `status`, to indicate the membership of each vertex in an independent set (line 2). We employ `USED` to constrain the independent set to have at least k elements (line 3). The `REQUIRED` constraints in lines 4 and 5 enforce the independent-set constraint.

```
1  CLASS vertex: 1..100 # v = 100
2  PARTITION status: in out
3  USED 30 <= in # k = 30
4  REQUIRED 2 out OR 5 out # edge (2,5): at least one vertex is out
5  REQUIRED 54 out OR 97 out # edge (54,97): at least one vertex is out
```

5 Translation of Constraint Lingo into a logic formalism

We use Perl scripts to translate Constraint Lingo into logic **formalisms** such as *smodels* [NS00], *dlv* [ELM+98], *ECLiPSe* [WNS97] and *aspps* [ET01] by means of

a **strategy**. The **standard strategy** introduces a cross-class predicate for every pair of columns. The **best-class** strategy chooses a special column and introduces cross-class predicates between it and the other columns. The **row-number** strategy numbers the rows and introduces equality and inequality constraints on row numbers. We have programmed many but not all combinations of formalism and strategy. No logic formalism is uniformly best, although *aspps* is generally fastest for the standard strategy. No strategy is uniformly best, but the best-class strategy is often fastest, especially when the best class is picked by a good heuristic.

The time taken by the translation and post-processing of the logic-engine output is negligible. Most of the 80 puzzles we have programmed are solved in well under a second by any combination of logic engine and translation strategy. We can generate graph-based problems requiring arbitrarily large computation time.

6 Demonstration

Our demonstration will display several puzzles and graph problems. We will examine the logic-formalism code generated by our translator for various formalism-strategy pairs, particularly {standard, *smodels*}, {best-class, *smodels*}, and {row-number, *ECLiPSe*}. All the software we use is publicly available, including our translators; we need only a Unix laptop loaded with our software to demonstrate our work. Our translation software and a set of puzzles can be found at `http://www.cs.uky.edu/ai/cl.html`.

References

[ELM+98] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A KR system `dlv`: Progress report, comparisons and benchmarks. In *Proceeding of the Sixth International Conference on Knowledge Representation and Reasoning (KR '98)*, pages 406–417. Morgan Kaufmann, 1998.

[ET01] D. East and M. Truszczyński. Propositional satisfiability in answer-set programming. In *Proceedings of Joint German/Austrian Conference on Artificial Intelligence, KI'2001*, volume 2174, pages 138–153. Lecture Notes in Artificial Intelligence, Springer Verlag, 2001.

[FMT01] R. Finkel, V. Marek, and M. Truszczyński. Tabular constraint-satisfaction problems and answer-set programming. *AAAI-2001 Spring Symposium Series, Workshop on Answer Set Programming*, 2001.

[NS00] I. Niemelä and P. Simons. Extending the smodels system with cardinality and weight constraints. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 491–521. Kluwer Academic Publishers, 2000.

[WNS97] M. Wallace, S. Novello, and J. Schimpf. ECLiPSe: A platform for constraint logic programming, 1997. `http://www.icparc.ic.ac.uk/eclipse/reports/eclipse.ps.gz`.