Constraints: An International Journal, 2, 57–61 (1997)

In Pursuit of the Holy Grail

EUGENE C. FREUDER ecf@cs.unh.edu
Constraint Computation Center, Department of Computer Science, University of New Hampshire, Durham, NH 03824 USA

Abstract. Constraint programming brings us closer to true declarative programming. Considerable progress has been made in this field; exciting challenges remain.

Keywords: constraint programming, constraint satisfaction, declarative programming

1. Introduction

Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.

In a recent article in *Byte*, Dick Pountain observes:

> Were you to ask me which programming paradigm is likely to gain most in commercial significance over the next 5 years I'd have to pick Constraint Logic Programming (CLP), even though it's perhaps currently one of the least known and understood. That's because CLP has the power to tackle those difficult combinatorial problems encountered for instance in job scheduling, timetabling, and routing which stretch conventional programming techniques beyond their breaking point. Though CLP is still the subject of intensive research, it's already being used by large corporations such as manufacturers Michelin and Dassault, the French railway authority SNCF, airlines Swissair, SAS and Cathay Pacific, and Hong Kong International Terminals, the world's largest privately-owned container terminal.

From a less commercial point of view, one of the most exciting features of this field is its interdisciplinary nature. Relevant disciplines include:

- Artificial Intelligence
- Combinatorial Algorithms
- Computational Logic
- Concurrent Computation
- Databases
- Discrete Mathematics

- Neural Networks
- Operations Research
- Programming Languages
- Symbolic Computation

I will briefly review some of the advantages of constraint programming, then describe some of the emerging infrastructure in this burgeoning field, and finally suggest several directions for further progress.

2. Advantages

- Declarative. In constraint programming we state the problem requirements; we do not need to specify how to meet these requirements. Constraints do not require us to envision how the information is to be used.
- Efficient. Of course, a declarative specification will only be useful for automated problem solving if the internal machinery can successfully process the declarative description to solve the problem. Constraint programming has made progress in several directions here.
  - Theory. Tractable problem classes have been identified.
  - Inference. Inference methods mitigate the problems of combinatorial search. Specialized methods take advantage of problem structure.
  - Synthesis. Progress has been made in synthesizing specialized programs for specific problem domains.
- Natural. All of this would be irrelevant were not constraints a natural medium for people to express problems in many fields. Important examples include:
  - Design and configuration
  - Graphics, visualization, interfaces
  - Hardware verification and software engineering
  - Human-computer interaction and decision support
  - Molecular biology
  - Real-time systems
  - Robotics, machine vision and computational linguistics
  - Scheduling, planning, resource allocation
  - Temporal and spatial reasoning
  - Transportation
  - Qualitative and diagnostic reasoning

3. Infrastructure

Several new forums have emerged to bring together the many disciplines that contribute to constraint programming.

- Conference. An annual International Conference on Principles and Practice of Constraint Programming was established in 1995. The proceedings are published in the Springer LNCS series. Its antecedents include:
  - Workshop on Constraint Logic Programming at Alton Jones in 1988
  - IJCAI-89 Workshop on Constraint Processing
  - AAAI-90 Workshop on Constraint Directed Reasoning
  - AAAI 1991 Spring Symposium on Constraint-Based Reasoning
  - Workshops on Constraint Logic Programming in Marseille in 1991, 1992 and 1993
  - Workshops on Principles and Practice of Constraint Programming in 1993 and 1994
  - International Conference on Constraints in Computational Logics in 1994
- Journal. A new Kluwer journal, *Constraints* recently began publication. Special issues are already underway on Constraints and Databases, Graphics and Visualization, Interval Constraints, Spatial and Temporal Reasoning.
- Internet. A newsgroup, comp.constraints serves the constraints community. There is a Constraints Archive divided between two sites:

  http://www.cirl.uoregon.edu/constraints/ http://www.cs.unh.edu/ccc/archive/.

  These contain many useful materials and pointers, including a FAQ and a directory of people interested in constraints. A mailing list, csp-list@cert.fr, is devoted to constraint satisfaction problems.

4. Directions

There are many ongoing challenges. Many mirror those faced by other representation and reasoning paradigms; we confront them here anew. One of the challenges, and opportunities, here is to integrate the contributions of the many fields that have something to say about constraint satisfaction.

- Modeling. It is one thing to say "all one has to do is express the problem constraints". It is another to express them in a manner which permits efficient solution. This raises issues of:
  - Knowledge acquisition. We want to extract the relevant constraints, and automate the translation of constraints expressed in the user's language into a form appropriate for problem solving.

  - Effective representation. There can be many ways of representing a problem with constraints. In particular, redundant constraints may hinder, or dramatically help, the solution process.
  - Debugging. We need help in correcting improperly or incompletely specified constraints.
  - Reformulation. It may be useful to reformulate a problem during solution, e.g. by abstraction or decomposition.

- Interaction. We need to interact with the problem solving process for many reasons. For example:

  - Symbolic spreadsheet. The powerful spreadsheet metaphor extends to symbolic constraints.
  - Optimization. What are the criteria for choosing a "best" solution, or an "anytime" solution?
  - Cooperative problem solving. We can work with the computer to both formulate and solve our problems.
  - Dynamic problems. Can we react to changes?
  - Real-time problems. Can we do so in real time?

- Explanation. Once we have a solution can we explain it? This can be useful in tutorial or cooperative situations. It is particularly important when a complete solution is not possible, and we need to understand why, and how we might modify the constraints. Assigning "blame" for failure is difficult in constraint satisfaction.

- Knowledge. In achieving these objectives, a key resource, from my symbolic artificial intelligence perspective, is the exploitation of various forms of knowledge.

  - Domain specific knowledge. Can we build a taxonomy of problem classes with an associated taxonomy of specialized constraints and solution methods?
  - Compiled knowledge. How is implicit knowledge most effectively rendered explicit?
  - Meta knowledge. How can knowledge about our knowledge facilitate its effective use?
  - Uncertainty. How do we cope with imprecise, uncertain or noisy information?

- Agents. There are natural links between agents and constraints, including:

  - Negotiation. Constraint satisfaction is a natural medium for negotiation among network agents.
  - Cooperation. Distributed resources can cooperate to share knowledge and solve problems.

  - Representation. Agents that know our own constraints can best represent us as electronic assistants.

- Evaluation. How do we know which methods are best for which problems? Can we move beyond an understanding of random problems to an understanding of structural features that usefully characterize real problems? Ultimately we want to be able to say: "Describe your problem to me, and I will tell you what methods to use to solve it".

- Synthesis. Of course, that is really only penultimately what we want to be able to say. In pursuit of our Holy Grail, we really want our constraint programming environment itself to evaluate a problem's characteristics and synthesize an optimal solution method for it.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant No. IRI-9504316.