Constraint Programming Letters 1 (2007) 3-5 Submitted 5/2007; Published 11/2007

Holy Grail Redux

Eugene C. Freuder E.FREUDER@4C.UCC.IE
Cork Constraint Computation Centre
Computer Science Department
University College Cork
Cork, IRELAND

Editor: Pascal Van Hentenryck

Ten years ago I participated in the ACM Workshop on Strategic Directions for Computing Science and wrote one of the position papers that subsequently appeared in the *ACM Computer Surveys* and in a special issue of the *Constraints* journal. My paper was entitled "In Pursuit of the Holy Grail", and posited that Constraint Programming could bring us nearer the ultimate "declarative" goal of Computer Science: The user states a problem; the computer solves it.

Apparently in the original script for *Monty Python and the Holy Grail*, the Grail is found at Harrods department store in London. There is a lesson here. Accessibility is the way forward for Constraint Programming. In preparing for the CP 2006 panel on the next ten years of Constraint Programming, the first thing I did was to look back over my previous position paper, and "grade" our progress towards the goals I set forth there. The result was disappointing, in large part because those goals already devoted a lot of attention to "accessibility" issues - making it easier to write and to use constraint programs - while the lion's share of research activity in the past ten years has remained focused on faster algorithms.

I conducted a little unscientific experiment. I picked five papers essentially at random from the CP 1995 proceedings and five more from the CP 2004 proceedings, mixed the titles together in a list, and challenged people to identify which papers came from which conference. It is not easy to do. It may be a case of "the grass is greener in the other fellow's yard"; but it seems to me that if these were bioscience conferences, or physics conferences, or even literature conference, that the difference over a ten year period would jump out at the participants.

We need more progress on tools and techniques for:

- Acquisition: Acquiring a complete and correct representation of real problems
- Automation: Automating efficient and effective modeling and solving
- Explanation: Explaining success and failure

And these must be embedded in an Interactive cycle, where Acquisition can inform Automation which leads to Explanation which in turn informs further Acquisition.

This research program is not easy because "ease of use" is not a science. However, that presents us with the exciting opportunity to be the pioneers of a new "usability science"

and then to "engineer usability". Progress here can benefit from more "meta-knowledge", knowledge about our knowledge, which itself might be modelled using constraints. We also should make use of "background knowledge", which is not tied to a particular problem statement.

Usability involves moving more of the burden from the human programmer or end-user to the machine. Artificial Intelligence is a natural ally in that regard. I like to remind people that there is an "AI" in "ConstrAInts". Many fields of AI might profitably be brought to bear, including:

- machine learning
- truth maintenance
- reasoning about uncertainty
- reasoning about preferences
- case-based reasoning
- human interaction
- natural language processing
- knowledge acquisition
- data mining
- automated deduction
- game playing
- software agents
- commonsense reasoning
- cognitive modeling

In moving "beyond algorithmics", there are also, of course, "the usual suspects":

- Engage with industry and government
- Bring CP into undergraduate and professional education
- Attract public attention
- Integrate the CP efforts in diverse disciplines
- Form and fund research networks
- Address the "messiness" of the real world

I would especially encourage our community to find a larger, more ambitious, more "marketable" context in which to embed and exploit our work. Optimisation is one candidate, but a surprisingly "hard sell" in the outside world. Automatic Programming is the "Holy Grail" ideal, but perhaps too ambitious. My current favourite is Decision Support. Constraint Programming can help computers help people make better personal and business decisions.

We should ask ourselves what constraints could do that would capture the imagination - of students, researchers, funding agencies, the public at large. Can we agree on large-scale "Challenges"? Decision support "dashboards" that help us "navigate the knowledge economy"? Software "alter egos" that "represent us" in the electronic world? "Self-improving software" that learns about its applications and its users? The typical reaction at this point is either "we're not ready" or "someone else is doing it". But we need to have big dreams and compete in a worthy game. To rise to such challenges we must work together, form large teams. Ideally we will find government funding for such efforts, but we need not wait on funding to begin.

Science is driven by metaphors. Biologists view molecular biology in computational terms and computer scientists view computation in biological terms. We can look for new

metaphors in the wider scientific literature. Popular science provides easy access, e.g. *The Tipping Point, Blink, The Wisdom of Crowds.* Constraint Programming has made effective use of metaphors in determining *how* to compute, e.g. physical metaphors for search (hill climbing, simulated annealing). We might devote more attention to metaphors for *what* to compute. Typically we take a problem solving approach: satisfaction, optimisation. However, we could devote more attention to Constraint Programming as:

- Simulation/Prediction: A constraint network can be viewed as a large multi-variable equation, where we can set some values (or ranges), have others set for us, and compute the rest. This, especially in the context of uncertainty, can be used for prediction. Qualitative Physics has used constraint processing to power physical models. It is exciting to see the emergence of constraint programming in Systems Biology and Computational Economics.

- Information Gathering: Information gathering is not just indexing. We cannot find all the widgets weighing less than five pounds and bigger than a breadbox by typing 'widgets weight less five pounds size bigger breadbox' into Google. This is a constraint satisfaction problem, made more interesting when preferences are brought into the mix.

- Configuration: If the world of web services and grid computing ever comes of age, the most useful role for constraint satisfaction may not be actually to solve problems, but to find a configuration of software services and computing resources that can be used to solve the problems.

Ultimately what is most important is that we all individually seek new directions for our field. Write a paper for CP 2010 that people will immediately recognize could not possibly have appeared in CP 2000. At the CP 2006 panel I quoted this exchange between Alice and the White Queen from *Alice in Wonderland.*

> Alice laughed: "There's no use trying," she said; "one can't believe impossible things."
>
> "I daresay you haven't had much practice," said the Queen. "When I was younger, I always did it for half an hour a day. Why, sometimes I've believed as many as six impossible things before breakfast."

If you do not want to take advice from the White Queen, consider these observations (from an interview in the *New Scientist*) by Shuji Nakamura, the winner of the 2006 Millennium Technology Prize:

> **Q:** Why did you succeed with the blue LED when others failed?
>
> **A:** I think I succeeded because I challenged the conventional wisdom ... The most important thing in science and technology is to take risks