Some Research Challenges and Remarks on CP

Nicolas Beldiceanu 1  Pierre Flener 2  Jean-Noël Monette 2  Helmut Simonis 3
 Mines de Nantes, France
 Uppsala University, Sweden
 4C, University College Cork, Ireland

Automata and Learning

Question Is there a declarative way for describing concisely a generator of automata (or MDDs) that have a very regular structure?

Question Is it possible to perform propagation directly on a declarative generator description rather than on the automaton (or MDD)?

Question Are there learning algorithms that learn smallest automaton generators rather than smallest automata?

Beyond Classical Indexicals

Question Can we extend indexicals that manipulate sets of values to indexicals that directly deal with combinatorial objects?

A. Maps of Graph Invariants

Question Is there a way of building in a systematic way this map of graph invariants? This map is an oriented graph \( G \) such that:

- The vertices of \( G \) are inequalities of the form \((g) \) \((c) \) \((expression)\), where \( g \) is a graph characteristic and \( c \) is a comparison operator.
- There is an arc from a vertex \( v \) to a vertex \( w \) of \( G \) that is labelled by \( g \) \(=\) \( c \), where \( g \) is a graph characteristic mentioned in \( v \) but not in \( w \), and \( e \) is an expression replacing \( g \) in \( v \) to obtain \( w \).

B. Making Explicit Extremal Objects Associated to Bounds

Question Is it possible to characterise and describe with an appropriate dedicated language an assignment associated with effectively reaching the bound? The hope is that if one can explicitly describe such an extremal object, then the regret and the corresponding propagator can be obtained mechanically.

C. Propagator on Graphs As Interpreter of Graph Indexicals

Question Is there a concise and compositional way of denoting how to derive graphs from the arguments of a constraint (and from other graphs, since this construction takes sometimes more than one step)?

Verification and Synthesis of Propagators

Question Can we adapt the techniques of formal verification in order to certify (semi-)automatically the correctness and other properties of propagators?

Question Are there higher-level abstractions that recur in the manual synthesis of propagators for some global constraints, so that propagators need not be designed from first principles each time?

Question Is it possible to use inductive or deductive synthesis to generate (semi-)automatically efficient propagators from the declarative description of some global constraints?

Visualisation and Explanation

Question Is there a mechanism that generates a visualiser of a global constraint based on a declarative description of the constraint? Or, at least, can we define a toolkit to visualise classes of global constraints without tedious and error-prone adaptation for each case?

Question Can explanations for global constraints be derived automatically from a declarative description of the constraint or from a high-level description of the propagator? What is required to generate automatically also explanations when synthesising propagators for global constraints? Can the explanations be provided in a form that is understandable without describing details of the propagators?

A Systematic Reconstruction of the Global Constraint Catalog

Redesign from first principles

- Identify core concepts
- Define derived concepts that apply across all core concepts
- Consistent, uniform naming
- Community effort required for adoption

Scalability

Question Are there, for some constraints, greedy propagators that perform a kind of limited propagation? From an industrial perspective, it would also be nice to be able to integrate some side constraints without breaking scalability.

Toward the Development of Sustainable CP Solvers

Good Practice

- Promote source code to be associated with submitted and published papers on algorithms; see for instance P. Prosser. Exact algorithms for maximum clique: A computational study. Technical Report 2012-333, Glasgow University, Scotland, 2012.
- Promote open-source solvers, such as Choco, ECLiPSe, Gecode, JaCoP, and Minion.
- Promote algorithms that can be implemented in more than one solver.
- Promote approaches that see an algorithm as the interpretation of a formula, since formulae can migrate between solvers.
- Promote standards for encoding knowledge that can be shared by different solvers (ideally a solver should be limited to an abstract machine that can interpret various types of combinatorial knowledge).

Interface of CP with Other Computer Science Areas

Actions

- Organise more out-reach meetings with experts of other CS areas.
- Develop (on-line) material explaining CP to CS experts.
- Maintain (together with at least the SAT/SMT and MIP communities) a showcase of significant benchmarks where CP solvers outperform SAT/SMT and MIP solvers, or where CP practitioners challenge SAT/SMT and MIP practitioners to beat CP solvers (and vice-versa).
- Limit the impact of the absence of a standard interface to CP solvers.