PhD Thesis (Grant at CRIL, Lens-France)

Towards Explanable AI by Reasoning with (Smart) Constraints

Context. Constraint Programming (CP) is a general framework providing efficient models and algorithms for solving combinatorial constrained problems. The research conducted at CRIL (AI Laboratory situated at Lens, France) about CP aims at developping generic approaches (i.e., AI-related) for both CSP/COP (Constraint Satisfaction/Optimization Problem) and SAT (Satisfiability Testing). Useful tools for conducting innovative research at CRIL include state-of-the-art academic solvers (the CSP solver AbsCon and the SAT solver Glucose), the integrated representation format XCSP^3 [1], the modeling Python library PyCSP^3 [3] (and the modeling Java API JvCSP^3), as well as a computing cluster. The thesis is about studying how (new extended forms of) so-called *smart* constraints can be useful for contributing to the challenging issue of explanable AI, as explained below.

Subject Description. Recently, a new kind of generic constraint has been introduced: the smart constraint [4, 5, 2]. A *smart* constraint looks like an hybridization of extensional and intensional constraint representations since a smart constraint is defined from a smart table whose rows contain simple arithmetic constraints. A smart constraint can be seen as a way of compressing constraints using primitives taken from a simple arithmetic language.

Technically, a *smart* constraint c is defined by a *smart table*, denoted by $table(c)$, that contains a set of smart rows. If the set of variables variables involved in c, called scope and denoted by $scp(c)$, is $\langle x_1, \ldots, x_r \rangle$, then a smart row ρ in $table(c)$ is a sequence of column restrictions $\langle \gamma_1, \ldots, \gamma_r \rangle$, where a column restriction γ_i takes one of the following forms:

- $x_i = *$
- x_i <op> a
- $x_i \in S$
- $x_i \notin S$
- x_i <op> x_j
- x_i <op> $x_j + b$

with a, b being constants, S a set of constants, and <op> an operator in $\{<, \leq, \geq, >, =, \neq\}$. Naturally, any ordinary (classical) tuple $(a_1, \ldots, a_r)$ can be re-written as the smart row $\langle x_1 = a_1, \ldots, x_r = a_r \rangle$. The semantics of smart constraints is simple and natural: an ordinary tuple τ is allowed by a smart constraint c iff there exists at least one smart row $\rho \in table(c)$ such that τ satisfies ρ. Note that a column constraint of the form $x_i = *$ is always satisfied. As an illustration, the following set of ordinary tuples $\{(1,2,1), (1,3,1), (2,2,2), (2,3,2), (3,2,3), (3,3,3)\}$ on variables $\langle x, y, z \rangle$ that can take their values in $\{1, 2, 3\}$ can be represented by a smart table containing only one smart row:

x	y	z
$= z$	≥ 2	$*$

As a second illustration, let us consider the global constraint `maximum` that accepts as parameters a sequence of integer variables $X = \langle x_1, x_2, \ldots, x_m \rangle$ as well as an integer variable M, and that enforces $\max(x_1, x_2, \ldots, x_m) = M$. This constraint can be represented compactly as follows:

x_1	x_2	$\ldots$	x_m	M
$*$	$\leq x_1$	$\ldots$	$\leq x_1$	$= x_1$
$\leq x_2$	$*$	$\ldots$	$\leq x_2$	$= x_2$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\leq x_m$	$\leq x_m$	$\ldots$	$*$	$= x_m$

Here, we have a smart table composed of m rows, where each row i captures the fact that $x_i = M \wedge \forall j \neq i, x_j \leq x_i$. As we can observe, a smart constraint offers the user a fine-grained flexible and readable mechanism to define the semantics of a constraint.

This is why we propose to use them for capturing the structural patterns (semantics) behind the proofs of unsatisfiability. Eventually, this will help the user to understand proofs and possibly formalize them compactly. We propose to exploit smart constraints by automatically reformulating (parts of) proofs under the form of smart constraints. Basically, we need first to associate a set of nogoods (instantiations of variables leading to no solution) with each proof of unsatisfiability. These nogoods can be derived during search (by a backtracking algorithm), or extracted from recorded proofs. We can envision a two-steps procedure. The first step involves partitioning the set of nogoods in clusters containing nogoods with related scope (variables). The second step involves synthesizing an equivalent smart table from a given cluster (table). This is best viewed as a set covering problem: each smart tuple is semantically equivalent to a set of (ordinary) tuples; so we aim at minimizing the number of elements of a set of smart tuples covering the given table constraint. Since the set covering problem is NP-complete, we shall follow a heuristic approach, possibly guided by abstract interpretation principles.

Laboratory. The CRIL (Centre de Recherche en Informatique de Lens) is a research lab affiliated with the University of Artois and with the CNRS (as UMR 8188), the French national research centre. The research activities at CRIL are centered around symbolic Artificial Intelligence and its applications. Constraint Programming (CSP, SAT, ...), which is at the core of this proposal, is a topic of high interest at CRIL.

The PhD student will benefit from a full-time grant during a period of three years. The position is based at CRIL (University of Artois, Lens, France). The student will have access to all lab resources, namely, personal computers, cluster computing, library subscription, etc.

Application. The candidate should have an outstanding degree in computer science (a perfect knowledge of programming with a language such as Java and/or Python is required - a good knowledge of Unix/Linux is appreciated), and a solid background in artificial intelligence and algorithms. A first experience in research is highly recommended. Prior knowledge of french is not mandatory. **Between February 15, 2020 and May 1, 2020**, every applicant must submit in a zip file a CV, a copy of his university degrees, a list of publications if any, and a covering letter by email to `{boussemart,lecoutre,piette}@cril.fr` with the following subject: Thesis:Smart. Selected candidates are expected to come for an interview in the lab or by visio-conference. Interviews will be conducted before May 8, 2020.

Supervision. Christophe Lecoutre, Professor in Computer Science, is the author of the constraint solver AbsCon, leader for the project XCSP^3, the co-developer of the modeling Python library PyCSP^3 the author of a book about constraint networks (Wiley, 2009), and co-author of several algorithms and heuristics that have been widely adopted by the CP community (for example, wdeg, last-conflict, STR2 and CT). He is the scientific leader, at CRIL, of the project CPER (Contrat de Plan Etat-région) 'Data' (since 2016), supported by the region "Hauts-de-France", and the scientific leader of the DIM IA (since 2018), one of the four major interest research domains of the university d'artois. Co-supervision: Frédéric Boussemart and Cédric Piette, assistant professors in computer science, are members of CRIL, involved in the XCSP^3 project and authors of many popular CP algorithms.

References

[1] F. Boussemart, C. Lecoutre, G. Audemard, and C. Piette. XCSP3: An integrated format for benchmarking combinatorial constrained problems. Technical Report arXiv:1611.03398, Specifications 3.0.6, CoRR, `http://arxiv.org/abs/1611.03398`, 2016–2020.

[2] B. Le Charlier, M.T. Khong, C. Lecoutre, and Y. Deville. Automatic synthesis of smart table constraints by abstraction of table constraints. In *Proceedings of IJCAI'17*, pages 681–687, 2017.

[3] C. Lecoutre and N. Szczepanski. PyCSP^3: Modeling combinatorial constrained problems in Python. Technical report, CRIL, `https://github.com/xcsp3team/pycsp3`, 2020.

[4] J.-B. Mairy, Y. Deville, and C. Lecoutre. The smart table constraint. In *Proceedings of CPAIOR'15*, pages 271–287, 2015.

[5] H. Verhaeghe, C. Lecoutre, Y. Deville, and P. Schaus. Extending Compact-Table to basic smart tables. In *Proceedings of CP'17*, pages 297–307, 2017.