A commentary on *"A regular language membership constraint for finite sequences of variables"*

Gilles Pesant
Polytechnique Montréal, Montreal, Canada
gilles.pesant@polymtl.ca

In this short note I recall the developments that led me to propose the `regular` constraint fifteen years ago as yet another screwdriver to put in our CP modeling toolbox. I also discuss some of the related work that followed and reflect back in an attempt to offer some lessons.

1 How it came about

I had been working on various rostering problems for a few years [18, 5, 10]. Many aspects of these problems one could already capture quite well with existing constraints in CP but some prescriptions about how work shifts should be sequenced were cumbersome to express: rules such as

> "any staff member working three consecutive night shifts must then have at least three days off"

or

> "a sequence of day shifts followed by a rest period and then by another sequence of day shifts is allowed, unless the rest period overlaps a Sunday or a Monday".

That led me to publish a paper introducing the `stretch` constraint in order to express restrictions on the length of maximal subsequences of identical values [12]. And then I started work on a `pattern` constraint, looking for a better way to express patterns in shifts. I implemented an *ad hoc* algorithm for ternary patterns and started a working paper that never fully materialized because I eventually moved on to its generalization which became `regular`. In an attempt to jog my memory I actually managed to dig up some old notes, including the plan reproduced at Figure 1: one can see the beginnings of a transition to something more general and more formally grounded, namely regular languages and automata. By the time that idea had matured the submission deadline for the main technical program of CP'03 had passed but there was still time to submit something to satellite workshops. So I decided to present the idea in a paper at the Modelling and Reformulation workshop [13]. And the following year the paper was published at CP'04.

So that retraces my personal path to that constraint, but I believe its time had come and it would have emerged one way or another. Indeed Beldiceanu, Carlsson, and Petit published a paper proposing something quite similar at the very same conference [2]: for constraints whose solutions could be checked by an automaton (possibly augmented with counters), they decomposed the constraint into a conjunction of fixed-arity constraints each representing the application of the transition function to one of the variables. Another connection: the filtering algorithm for `regular` builds a layered graph by unfolding the automaton given as input. That graph is similar in spirit to the one Trick had proposed at CPAIOR'01 to filter `knapsack` constraints [20]: both make solutions to the constraint correspond to paths in the graph. One can view such representations as state-space graphs in a dynamic programming approach — an explicit use of the latter led to a stronger filtering algorithm (achieving domain consistency) for the `stretch` constraint, also published at CP'04 [9].

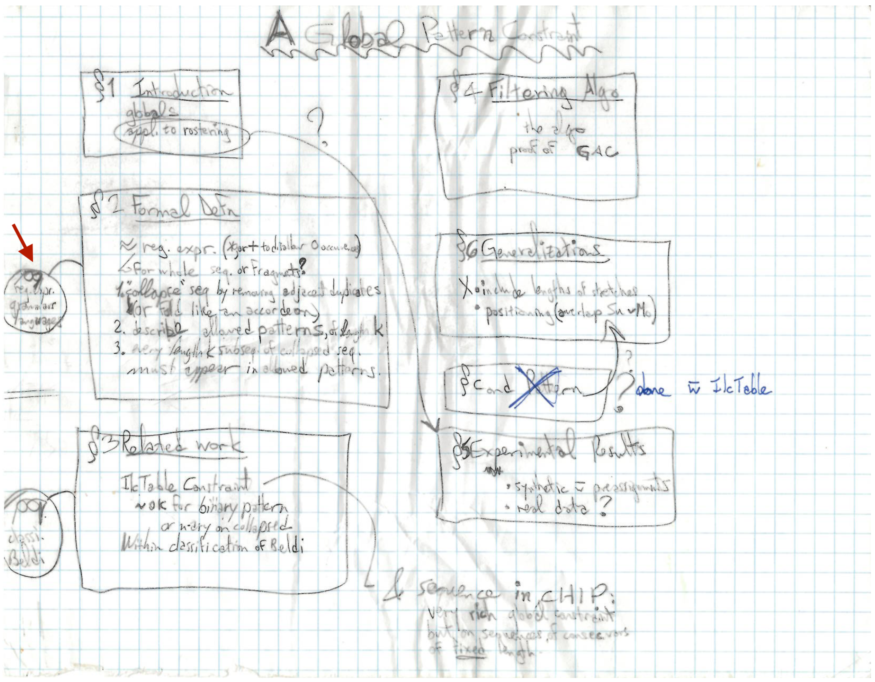
A Global Pattern Constraint
§1 Introduction
globals
appl. to rostering
§2 Formal Defn
reg. expr.
for whole seq. or fragment?
"collapse" seq by removing adjacent duplicates
2. describe allowed patterns, of length k
3. every length-k subseq. of collapsed seq. must appear in allowed patterns.
reg. expr. grammar languages
§3 Related work
IlcTable Constraint
ok for binary pattern
or n-ary on collapsed
Within classification of Beldi
classif. Beldi
§4 Filtering Algo
the algo
proof of GAC
§6 Generalizations
include lengths of stretches
positioning (overlap SA vs Mo)
§C and ~~Xpr~~
done w IlcTable
§5 Experimental Results
synthetic
real data ?
sequence in CHIP:
very rich global constraint
but on sequences of consec vars
of fixed length

Figure 1: My old notes. The red arrow points to the budding realization that I should look at formal languages.

2 Where it went from there

Perhaps because it is expressive, replacing and generalizing several existing constraints, and offers strong filtering, achieving domain consistency whereas some existing constraints did not (e.g. `stretch`), the `regular` constraint has been included in many if not most CP solvers and regularly appears in CP models to solve combinatorial problems from diverse areas. Beldiceanu's catalogue organizes global constraints according to two types of representation, one of them based on automata [1]. Building on `regular`, some including myself took it in several directions.

From the perspective of formal languages, the next natural step was to go from automata to context-free grammars [16, 19]. Also the layered graph built for `regular` provides a compact structured representation of the set of solutions — multi-valued decision diagrams (MDD) offer the same but in an even more general framework [4].

Because it conveniently expresses substructures often found in combinatorial problems, and not just in CP models for them, the `regular` constraint has been reformulated to suit other solving approaches: it was linearized in order to be used by a MIP solver [6] and (indirectly) translated into a set of Boolean clauses in order to be used by a SAT solver [17].

Soft constraints allow one to express preferences or to bend otherwise-hard constraints in an over-constrained problem. One framework for this adds a variable to each such constraint that bounds how much it can be violated by some instantiation of the other variables in its scope, making it possible to filter out domain values leading to a violation level that would be too high. An efficient algorithm to do this was proposed for `regular` [21]. Closely related are *optimization constraints* introduced to express and solve combinatorial optimization problems: in our case one or multiple cost variables were added to the constraint and the original consistency algorithm was extended to perform cost-based filtering [7, 11]. *Lazy clause generation* extracts explanations of domain-value removals by constraints in the form of Boolean clauses that are then fed to a SAT solver. Algorithms that generate such explanations were introduced for MDD constraints, which also apply in our case [8]. *Counting-based search* promotes branching heuristics that rely on the number of solutions to individual constraints in a model. Instead of asking whether a given variable assignment is supported by at least one solution to a constraint, we ask how many such solutions there are, thereby assessing the popularity of that variable assignment with respect to that constraint. Whereas computing the exact answer to this new question becomes much harder for some constraints such as `alldifferent`, it only requires a bit of extra work for `regular` [23]. More recently it was extended to weighted counting so that counting-based search could be used to solve constraint optimization problems [14] and so as to provide finer-grained information to propagate between constraints [15].

3 Some lessons learnt

I think a first lesson is to keep applications in mind. By working to solve real problems one may encounter novel structural properties that deserve being encapsulated in new CP constraints. That was my experience with rostering problems.

But another lesson when proposing a new constraint is to strike the right balance between expressiveness, efficiency, and usefulness for practical problems. For example proposing the `grammar` constraint was a natural and good idea in itself but it hasn't reached the popularity of `regular`, whereas MDDs did. That is all very well in hindsight — I do not suggest it is an easy call to make.

The `regular` constraint has been decomposed, linearized, explained, served warm, nondeterministicated, decorated with costs and counters, and counted over. For some of these it meant replacing the filtering algorithm I proposed by something else. So what is left of the original work? I think that the lasting contribution will be a convenient way to express a combinatorial substructure found in many problems which, once captured, allows us to reason about it in order to solve a problem better. And that is probably also true of CP: what ends up working best under the hood may change but its distinctive force will remain a privileged access to problem structure through the high-level models it uses.

References

[1] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, Mar 2007.

[2] Nicolas Beldiceanu, Mats Carlsson, and Thierry Petit. Deriving filtering algorithms from constraint checkers. In Wallace [22], pages 107–122.

[3] Frédéric Benhamou, editor. *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*. Springer, 2006.

[4] David Bergman, Andre A. Cire, Willem-Jan van Hoeve, and John Hooker. *MDD-Based Constraint Programming*, pages 157–181. Springer International Publishing, Cham, 2016.

[5] Stéphane Bourdais, Philippe Galinier, and Gilles Pesant. HIBISCUS: A constraint programming application to staff scheduling in health care. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2003.

[6] Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Modeling the regular constraint with integer programming. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2007.

[7] Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. Constraint programming based column generation for employee timetabling. In Roman Barták and Michela Milano, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR 2005, Prague, Czech Republic, May 30 - June 1, 2005, Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2005.

[8] Graeme Gange, Peter J. Stuckey, and Radoslaw Szymanek. MDD propagators with explanation. *Constraints*, 16(4):407–429, 2011.

[9] Lars Hellsten, Gilles Pesant, and Peter van Beek. A domain consistency algorithm for the stretch constraint. In Wallace [22], pages 290–304.

[10] Gilbert Laporte and Gilles Pesant. A general multi-shift scheduling system. *JORS*, 55(11):1208–1217, 2004.

[11] Julien Menana and Sophie Demassey. Sequencing and counting with the multicost-regular constraint. In Willem Jan van Hoeve and John N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009, Pittsburgh, PA, USA, May 27-31, 2009, Proceedings*, volume 5547 of *Lecture Notes in Computer Science*, pages 178–192. Springer, 2009.

[12] Gilles Pesant. A filtering algorithm for the stretch constraint. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, pages 183–195. Springer, 2001.

[13] Gilles Pesant. A Regular Language Membership Constraint for Sequences of Variables. In *Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pages 110–119, 2003. https://www-users.cs.york.ac.uk/ frisch/Reformulation/03/proceedings.pdf.

[14] Gilles Pesant. Counting-based search for constraint optimization problems. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3441–3448. AAAI Press, 2016.

[15] Gilles Pesant. From support propagation to belief propagation in constraint programming. *J. Artif. Intell. Res.*, 66:123–150, 2019.

[16] Claude-Guy Quimper and Toby Walsh. Global grammar constraints. In Benhamou [3], pages 751–755.

[17] Claude-Guy Quimper and Toby Walsh. Decompositions of grammar constraints. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1567–1570. AAAI Press, 2008.

[18] Louis-Martin Rousseau, Gilles Pesant, and Michel Gendreau. A general approach to the physician rostering problem. *Annals OR*, 115(1-4):193–205, 2002.

[19] Meinolf Sellmann. The theory of grammar constraints. In Benhamou [3], pages 530–544.

[20] Michael A. Trick. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals OR*, 118(1-4):73–84, 2003.

[21] Willem Jan van Hoeve, Gilles Pesant, and Louis-Martin Rousseau. On global warming: Flow-based soft global constraints. *J. Heuristics*, 12(4-5):347–373, 2006.

[22] Mark Wallace, editor. *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*. Springer, 2004.

[23] Alessandro Zanarini and Gilles Pesant. Solution counting algorithms for constraint-centered search heuristics. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 743–757. Springer, 2007.