Proceedings

8th International Workshop, CSLP 2016

October 17, 2016, affiliated with ICLP 2016

Edited by

Henning Christiansen and Verónica Dahl

Preface

Constraints are widely used in linguistics, computer science, and psychology. How they are used, however, varies widely according to the research domain: natural language processing, knowledge representation, cognitive modeling, problem solving mechanisms, etc. Since 2004, the CSLP workshop series has addresses different constraint oriented ways of modelling and treating language, and this time – the 8th CSLP workshop – it is affiliated with the International Conference of Logic Programming, and focuses (non-exclusively) on logic programming based methods.

This workshop addresses the question of how to best integrate constraints from an interdisciplinary perspective and with logic programming as a pivot, in view of powerful and robust human language processing. The topics include, but are not limited to, constraint-based linguistic theories, constraints in human language comprehension and production, Constraint Handling Rules (CHR) and Constraint Handling Rule Grammars (CHRG), context modeling and discourse interpretation, acquisition of constraints, grammar induction through constraints, probabilistic constraint-based reasoning, constraint satisfaction technologies and constraint logic programming – all around the axis of logic programming.

Several years of CSLP workshops (see left menu) aiming at integrating the different approaches on Constraint Solving and Language Processing shed light upon possible common frameworks capable of explaining how constraints play a role in representing, processing and acquiring linguistic information, and this from a formal, technical, and cognitive perspective. Among these frameworks, those that contained logic programming as a main aspect emerged as the most promising ones, e.g.. Constraint Handling Rules (CHR) as an extension to the logic programming language Prolog has added a mechanism for forward-chaining reasoning to complement Prolog's standard backwards chaining, goal-directed style. The combination of the two proved to provide a very powerful reasoning framework that creates an extended potential in applying logic programming for language processing and reasoning. Such results allow us to predict that adding logic programming may quite likely jump-start a whole new area of research that stands to revolutionize and revitalize formal logic approaches to NLP, adding robustness and flexibility to the models that can now be achieved, while elegantly marrying efficiency with direct executability.

Henning Christiansen and Verónica Dahl
Roskilde, Vancouver, October 2016

Organization

Organizers and program chairs

Henning Christiansen, Roskilde University, Denmark
Veròonica Dahl, Simon Fraser University, Canada.

Steering committee

Philippe Blache
Henning Christiansen
Veròonica Dahl
Denys Duchier
Jørgen Villadsen
Yannick Parmentier

Program Committee

Maria Aloni
Leonor Becerra-Bonache
Philippe Blache
Lucas Champollion
Denys Duchier
M. Dolores Jimenez-Lopez
Ruth Kempson
Wolfgang Menzel
Yannick Parmentier
Gerald Penn
Fred Popowich
Patrick Saint-Dizier
Jørgen Villadsen
David Scott Warren

Contents

Processing architectures: Different processing architectures have been proposed integrating several of these features. In particular, the *Good-Enough Theory* [Ferreira and Patson, 2007]; [Traxler, 2014] integrates the fact that sentence interpretation is only done from time to time, **delaying** the integration until **enough information** becomes available. In this case, a complete interpretation is often delayed (or in some cases never done), and replaced by the identification of partial meanings, starting from which a general interpretation can be approximated. The basic principle consists there in finding "*interpretations over small numbers of adjacent words whenever possible*". This framework has been precised by the integration of two different levels of parsing that can be at work: shallow processing with partial interpretation for the average case and deep processing when faced with difficulties [Blache, 2016a]. This new architecture implements the delaying mechanism, opening the way to pattern or global recognition following the principles of the good-enough parsing, integrating a "*whenever possible*" interpretation.

How constraints implement architectures: Constraints offer an appropriate solution for the implementation of all the different requirements of the proposed architecture.

- *Whenever possible interpretation*: In constraint programming, all constraints are potentially active and assessed when their variables are instantiated. Moreover, all constraints in a system are independent from each other, which means that they can be assessed independently, at any time. As a consequence, constraint satisfaction implements implicitly **delayed evaluation**, until variables get values (or more generally until enough information becomes available). When no value is assigned to a variable, the constraint is not fully assessed, but makes it possible to restrict the definition domain (the **search space**) and to maintain the coherence of the system, leading to **approximated** solutions.
- *Good-enough parsing*: Constraints in general, among which unification, makes it possible to work with underspecified structures. More precisely, all structures can be left partially uninstantiated (implementing directly **underspecification**), and progressively completed when necessary. In other words, a same structure, whatever its level of specification, can be used in both types of processing, **shallow** or **deep**.
- *Noisy-channel*: Ill-formed inputs can be parsed thanks to **constraint relaxation**. Constraint **violation**, completed with **weighting** or ranking, offers then the operational framework in the construction of an **optimal** solution. Moreover, the set of violated constraints constitutes a precise description of the source of error.
- *Prediction/activation*: Constraint systems implement **relation networks** within the set of variables, forming **constraint graphs**. The instantiation of a variable makes it possible to **activate** the associated subgraph and their nodes. An activated node corresponds to a predictable object and a set of activated nodes (or variables) implements **category prediction**. In other

words, when enough information becomes available, on top of describing the constrained structure, it becomes possible to predict new objects.
- *Patterns/constructions*: Following the **declarative** characteristics of constraint programming, a structure can be described by a set of constraints. In language processing, complex objects such as constructions can correspond to set of constraints. A construction (or a pattern) is then recognized when the **constraint subsystem** is satisfied.
- *No structure*: Finally, and most importantly, the state of the constraint system after evaluation comes to a precise and in-depth description of the linguistic structure. As a consequence, no specific structure has to be built prior to the interpretation. Constraints constitute then an adequate answer for the implementation of non-modular approaches.

As shown above, the integration of cognitive and computational approaches raises new questions for language processing among which the need to work with partial structures, to interpret objects only when enough information becomes available, to deal with noisy inputs and to implement different level of processing, corresponding to different levels of complexity of the input. These needs form the basis of recent cognitive principles such as "*good-enough parsing*" and "*whenever possible interpretation*". We propose to consider constraints as an adequate and efficient framework for their computational modeling.

References

[Anderson, 1990] Anderson, J. (1990). *The adaptive character of human thought*. Lawrence Erlbaum.

[Blache, 2016a] Blache, P. (2016a). Light-and-deep parsing: a cognitive model of sentence processing. In Poibeau, T. and Villavicencio, A., editors, *Language, Cognition and Computational Models*. Cambridge University Press.

[Blache, 2016b] Blache, P. (2016b). Representing syntax by means of properties: a formal framework for descriptive approaches. *Journal of Language Modelling*.

[Colmerauer, 1986] Colmerauer, A. (1986). Theoretical model of prolog ii. In van Caneghen, M. and Wane, D., editors, *Logic Programming and its Applications*. Ablex Series in Artificial Intelligence.

[Ferreira and Patson, 2007] Ferreira, F. and Patson, N. D. (2007). The 'good enough' approach to language comprehension. *Language and Linguistics Compass*, 1(1).

[Gibson, 1998] Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68:1–76.

[Jaffar and Lassez, 1986] Jaffar, J. and Lassez, J.-L. (1986). Constraint logic programming. Technical report, Department of Computer Science, Monash University, Victoria, Australia.

[Johnson and Charniak, 2004] Johnson, M. and Charniak, E. (2004). A tag-based noisy channel model of speech repairs. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

[Levy, 2008a] Levy, R. (2008a). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

[Levy, 2008b] Levy, R. (2008b). A noisy-channel model of rational human sentence comprehension under uncertain input. *Proceedings of EMNLP*, pages 234–243.

[Levy, 2013] Levy, R. (2013). Memory and surprisal in human sentence comprehension. In van Gompel, R. P. G., editor, *Sentence Processing*, pages 78–114. Psychology Press.

[Prince and Smolensky, 1993] Prince, A. and Smolensky, P. (1993). *Optimality Theory: Constraint Interaction in Generative Grammars*. Technical Report RUCCS TR-2. Rutgers Optimality Archive 537.

[Rommers et al., 2013] Rommers, J., Dijkstra, T., and Bastiaansen, M. (2013). Context-dependent Semantic Processing in the Human Brain: Evidence from Idiom Comprehension. *Journal of Cognitive Neuroscience*, 25(5):762–776.

[Swets et al., 2008] Swets, B., Desmet, T., Clifton, C., and Ferreira, F. (2008). Underspecification of syntactic ambiguities: Evidence from self-paced reading. *Memory and Cognition*, 36(1):201–216.

[Traxler, 2014] Traxler, M. J. (2014). Trends in syntactic parsing: anticipation, bayesian estimation, and good-enough parsing. *Trends in Cognitive Sciences*, 18(11):605–611.

[Vagharchakian et al., 2012] Vagharchakian, L., G., D.-L., Pallier, C., and Dehaene, S. (2012). A temporal bottleneck in the language comprehension network. *Journal of Neuroscience*, 32(26):9089–9102.

[Vespignani et al., 2010] Vespignani, F., Canal, P., Molinaro, N., Fonda, S., and Cacciari, C. (2010). Predictive mechanisms in idiom comprehension. *Journal of Cognitive Neuroscience*, 22(8):1682–1700.

ASP for Abduction in Natural Language Understanding made more efficient using External Propagators

Peter Schüller[1], Carmine Dodaro[2], and Francesco Ricca[2]

[1] Computer Engineering Department, Faculty of Engineering
Marmara University, Turkey
peter.schuller@marmara.edu.tr
[2] Department of Mathematics and Computer Science
University of Calabria, Italy
{dodaro,ricca}@mat.unical.it

Abstract. Answer Set Programming (ASP) is a powerful paradigm for knowledge representation and reasoning. Several tasks in Natural Language Understanding (NLU) have been or have the potential to be modeled in ASP. Among these, abduction under various optimality conditions has been recently implemented in ASP. Experiments revealed that pure ASP is not effective enough because the complete instantiation of some critical constraints is not scalable. The recent extension of ASP solvers with external propagators may provide means for avoiding the instantiation bottleneck, and thus can help to obtain more efficient implementations of abduction in NLU via ASP. We conducted preliminary experiments with ASP solver interfaces for external propagators. The results look promising and provide directions for the development of full-fledged extensions of ASP solvers with non-ground constraints.

Keywords: ASP, Propagators, Abduction, NLU

ASP for Abduction in Natural Language Understanding

Abduction is a popular formalism for NLU, and we here consider a benchmark for abduction under preference relations of cardinality minimality, coherence [7], and Weighted Abduction [6]. For example given the text "Mary lost her father. She is depressed." using appropriate background knowledge and reasoning formalism we can obtain the interpretation of the sentence that Mary is depressed *because* of the death of her father.

Several tasks in Natural Language Understanding (NLU) have been or have the potential to be modeled effectively using logic programming techniques [2, 4, 8]. Answer Set Programming (ASP) [3] is a powerful paradigm for knowledge representation developed in the area of logic programming that is naturally suited as a computational means for the realization of abduction under preferences. Indeed, ASP formulations for the above NLU tasks were described in [8]. However, the prevalent evaluation strategy adopted by state of the art ASP systems, which

is carried out by performing in a row grounding (i.e., variable elimination) and solving (i.e., search for the answer sets of a propositional program), resulted to be not effective in large instances. This is due to the grounding blow-up caused by a small set of constraints.

In this work we study initial experiments on an extension of the ASP solver WASP [1] suitable to overcome this problem. Indeed, WASP has been extended with an API that allows a user to provide the solver with external Python programs extending the main solving procedure. In particular, we experimented with the API features for checking answer sets for constraint violations and adding propositional constraints lazily. In this way we circumvent the critical issue of grounding some constraints of the ASP programs modeling NLU tasks. Our solution works in the presence of optimization including the usage of unsatisfiable-core optimization, which is not possible in the Python API of the CLINGO solver [5].

Experimenting with external propagators

Preliminary Results. Table 1 shows preliminary experiments with the WASP solver on the Bwd-A encoding for first order Horn abduction from [8]. We show accumulated results for 50 natural language understanding instances from [7] for objective functions cardinality minimality, coherence [7], and Weighted Abduction [6]. We compare two evaluation methods: *Constraint* instantiates all constraints during the initial grounding step and sends them to the solver, while *Propagator* omits a significant portion of constraints (those related to transitivity) from the initial grounding and instantiates them lazily in the propagator whenever a transitivity violation is detected in an answer set candidate.

We observe that for all objective functions, there are out-of-memory conditions for 6 instances (maximum memory was 5 GB) while memory is not exhausted with propagators, and average memory usage is significantly lower with propagators (1.7 GB vs. around 150 MB). For cardinality minimality, the average time to find the optimal solution decreases sharply from 76 sec to 8 sec and we find optimal solutions for all instances. For coherence we can solve more instances optimally however the required time increases from 64 sec to 103 sec on average and 4 instances reach the timeout (600 sec). For Weighted Abduction, which represents the most complex optimization criterion, we solve fewer instances (37) compared with using pre-instantiated constraints (44 instances).

Propagators can clearly be used to trade space for time, and in some cases we decrease both space and time usage. For the complex Weighted Abduction objective functions, we can observe in the *Odc* column that many more invalid answer sets (2067) were rejected by the propagators compared with cardinality minimality (70) or coherence (751).

Ongoing and Future Work. Our current prototype implementation only checks when a full answer set candidate has been found, while most violated constraints could also be detected based on a partial interpretations. Thus we are implementing a propagator that can work on partial interpretations. We also plan to experiment with the optimal frequency of propagation, which is known

Objective Function	Method	MO #	TO #	OPT #	T sec	M MB	Odc #
Cardinality Minimality	Constraint	6	0	44	76	1715	0
	Propagator	0	0	50	8	119	70
Coherence	Constraint	6	0	44	64	1723	0
	Propagator	0	4	46	103	131	751
Weighted Abduction	Constraint	6	0	44	66	1731	0
	Propagator	0	13	37	229	141	2067

Table 1. Experimental Results: MO/TO indicates number of instances were memory/time was exhausted, OPT the number of optimally solved instances, T/M indicates average time and memory usage, and *Odc* shows number of times an answer set was invalidated and a new clause was learned, i.e., a constraint was lazily instantiated.

to play a role in similar implementations for robotics planning. Moreover, our prototype is able to learn only a single constraint per invalidated answer set, however one answer set might contain several violations of not instantiated constraints. Adding all these at once might guide the solver much better to find an optimal solution that violates no constraints.

Acknowledgements. This work has been supported by Scientific and Technological Research Council of Turkey (TUBITAK) Grant 114E777 and by MISE under project "PIUCultura", N. F/020016/01-02/X27.

References

1. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: International Conference on Logic Programming and Non-monotonic Reasoning. pp. 40–54 (2015)
2. Balduccini, M., Baral, C., Lierler, Y.: Knowledge representation and question answering. In: Handbook of Knowledge Representation, Foundations of Artificial Intelligence, vol. 3, pp. 779–819. Elsevier (2008)
3. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM 54(12) (2011)
4. Christiansen, H.: Constraint programming for context comprehension. In: Brézillon, P., Gonzalez, A.J. (eds.) Context in Computing - A Cross-Disciplinary Approach for Modeling the Real World, pp. 401–418. Springer (2014)
5. Gebser, M., Kaminski, R., Obermeier, P., Schaub, T.: Ricochet Robots Reloaded: A Case-Study in Multi-shot ASP Solving. In: Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation, pp. 17–32. Springer (2015)
6. Hobbs, J.R., Stickel, M., Martin, P., Edwards, D.: Interpretation as Abduction. Artificial Intelligence 63(1-2), 69–142 (1993)
7. Ng, H.T., Mooney, R.J.: Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In: Knowledge Representation and Reasoning. pp. 499–508 (1992)
8. Schüller, P.: Modeling Variations of First-Order Horn Abduction in Answer Set Programming. Fundamenta Informaticae (2016), to appear, arXiv:1512.08899 [cs.AI]