Position Paper: Declarative Programming in Artificial Intelligence

Siegfried Nijssen

Universiteit Leiden

s.nijssen@liacs.leidenuniv.nl

An old idea in artificial intelligence is to develop software that requires a minimal programming effort to solve complex problems. Arguably, *Constraint Programming* is one of the prominent areas in AI that attempts to implement this vision, with a focus on constraint satisfaction and optimisation problems (CSOPs). By distinguishing modelling from solving, CP has built foundations for programming systems that can be used to solve a wide range of CSOPs. Arguably, a key idea in CP is the modularity of CP systems: not only can many different constraints be used together; also, recent systems such as G12/MiniZinc [8] and Numberjack [5] allow for the use of many different solvers for the same front-end language, where portfolio solvers can automatically choose solvers.

However, CP is not the only area in computer science with the vision to make programming easier. Several other fields of AI also have an interest in *declarative programming* and have been making significant progress in recent years:

Data mining: In data mining, a number of SQL-inspired languages exist as an interface to data mining algorithms, from research-oriented languages [3] to languages that are rapidly gaining ground in the industry [2];

Big data: For processing big data, *reactive programming* systems are emerging as one of the most prominent programming systems; Spark is overtaking Hadoop as the platform of choice for scalable machine learning [6];

Deep learning: Systems such as Tensorflow [1] and Theano [10] provide a declarative framework for specifying optimisation criteria, in combination with symbolic differentation to find solutions;

Probabilistic modeling: *Probabilistic programming* systems [9] are under development for the declarative spefication of distributions; in such systems, programmers can either describe a generative process or impose weights on logic formulas to specify a distribution.

The position advocated here is that CP can learn from these programming paradigms, and these programming paradigms can benefit from CP. To provide a number of questions that could be studied, and which have an interdisciplinary character: can probabilistic programming be combined with stochastic constraint programming [11], probabilistic (concurrent) constraint programming [4] or soft constraints [7]? Would this allow to solve CSPs under uncertainty encoded into probabilistic models or machine learning models? As reactive programming is build on the massive propagation of change, can this propagation be used to solve large CSPs? Can the modular approach advocated by CP, including the use of portfolio solvers, be useful in probabilistic inference? Can concepts of propagation well-known in CP also be relevant to reactive programming? Can symbolic differentation be included in CP systems? Is it possible to build a unifying programming methodology that supports big data, constraints, and uncertainty?

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `http://tensorflow.org/`, software available from tensorflow.org
2. Carasso, D.: Exploring Splunk. CIT, Zagreb, Croatia, Croatia (2012), `http://www.splunk.com/goto/book`
3. Dzeroski, S., Goethals, B., Panov, P. (eds.): Inductive Databases and Constraint-Based Data Mining. Springer (2010)
4. Gupta, V., Jagadeesan, R., Saraswat, V.A.: Probabilistic concurrent constraint programming. In: CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings. pp. 243–257 (1997), `http://dx.doi.org/10.1007/3-540-63141-0_17`
5. Hebrard, E., O'Mahony, E., O'Sullivan, B.: Constraint Programming and Combinatorial Optimisation in Numberjack. In: Lodi, A., Milano, M., Toth, P. (eds.) Proceedings of the 7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR-10). Lecture Notes in Computer Science, vol. 6140, pp. 181–185. Springer-Verlag, Bologna, Italy (May 2010)
6. Meng, X.: MLlib: Scalable machine learning on Spark. In: Spark Workshop April 2014 (2014), `http://stanford.edu/~rezab/sparkworkshop/`
7. Meseguer, P., Rossi, F., Schiex, T.: Soft constraints. In: Handbook of Constraint Programming, pp. 281–328 (2006)
8. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: Bessiere, C. (ed.) Thirteenth International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 4741, pp. 529–543. Springer-Verlag, Providence, RI, USA (Sep 2007)
9. Roy, D., Goodman, N.: Probabilistic programming (2016), `http://probabilistic-programming.org`
10. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688 (May 2016), `http://arxiv.org/abs/1605.02688`
11. Walsh, T.: Stochastic constraint programming. In: van Harmelen, F. (ed.) ECAI. pp. 111–115. IOS Press (2002)