Doctoral Dissertation

Defeasible Logic Programming:
Language Definition, Operational Semantics, and Parallelism

Alejandro Javier García

Departamento de Ciencias de la Computación
Universidad Nacional del Sur – Bahía Blanca, Argentina
http://cs.uns.edu.ar/~ajg

This thesis defines *Defeasible Logic Programming* and provides a concrete specification of this new language through its operational semantics. Defeasible Logic Programming, or *DeLP* for short, has been defined based on the Logic Programming paradigm and considering features of recent developments in the area of Defeasible Argumentation. DeLP relates and improves many aspects of the areas of Logic Programming, Defeasible Argumentation, Intelligent Agents, and Parallel Logic Programming.

The language of DeLP considers two kinds of program rules, *defeasible rules*, used for representing weak or tentative information, like $\sim flies(X) \prec mammal(X)$, "a mammal does not fly", and *strict rules* for representing strict (sound) knowledge, like $mammal(X) \leftarrow dog(X)$, "a dog is a mammal". Syntactically, the symbol "$\prec$" is all that distinguishes a defeasible rule from a strict one. Pragmatically, a defeasible rule is used to represent defeasible knowledge, i. e., tentative information that may be used if nothing could be posed against it. *Defeasible Rules* will add a new representational capability for expressing a weaker link between the head and the body of a rule.

In DeLP, an argumentation formalism is used for deciding between contradictory goals through a dialectical analysis. Intuitively, an argument is a minimal set of rules used to derive a conclusion. A query q will succeed when there is an *argument* $\mathcal{A}$ for q that is *warranted*. The warrant procedure involves looking for *counterarguments* that could be *defeaters* for $\mathcal{A}$. Nute in [16] remarks *"An inference is defeasible if it can be blocked or defeated in some way"*. Weak rules provide the locus where the blocking or defeating might occur. Our defeaters will take the form of arguments, therefore defeaters for the defeaters may exist.

In DeLP,a query q will succeed if a supporting argument $\mathcal{A}$ for q is not defeated. In order to establish whether $\mathcal{A}$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\mathcal{A}$ are considered, i. e., counter-arguments that for some criterion, are preferred to $\mathcal{A}$. Since counter-arguments are arguments, there may exist defeaters for them, and so on. This prompts a complete *dialectical analysis*. This analysis will impose certain constrains for averting problematic situations that may arise during the argumentation process, leading for example to an infinite sequence of defeaters. Thus, DeLP can manage defeasible reasoning, allowing the representation of defeasible and non-defeasible knowledge.

The language of DeLP has been defined as an extension of the latest developments in Logic Programming [14, 3, 20, 9], and considering new features of recent approaches in defeasible argumentation [17, 1, 15, 2, 4, 18]. The defeasible argumentation formalism used for the inference engine of the language has also been extended, considering new restrictions over argumentation lines, and defining a belief operator for warranted conclusions.

Another contribution of this thesis is the study of different sources of parallelism in Defeasible Logic Programming. Implicitly exploitable parallelism for Logic Programming has received ample attention in the last years [11, 12, 13, 10]. Defeasible Logic Programming could take

full advantage of several kinds of parallel evaluation to improve the computational response of its proof procedure. Since DeLP is an extension of Logic Programming, the different types of parallelism studied for Logic Programming can be applied. We also propose new sources of parallelism that can be implicitly exploited in the defeasible argumentation formalism used by DeLP. Both the argumentation process and the dialectical analysis benefit from exploiting these sources of parallelism.

An extension of Defeasible Logic Programming was also developed in this thesis. The syntax of the language was extended allowing *default negation* over literals in the body of defeasible rules. Thus, extended DeLP programs may use two kinds of negation: *strong negation* "$\sim$" for representing contradictory knowledge; and, *default negation*, "*not*" for representing incomplete information. For instance, the following rule uses both types of negation:

$$\sim cross\_railway\_tracks \prec not \sim train\_is\_coming.$$

expressing that: *"in general, do not cross railway tracks if it cannot be proven that no train is coming"*. The defeasible argumentation formalism was extended accordingly in order to handle default negation properly.

Defeasible Logic Programming can be used for representing knowledge and for providing an inference engine in many applications. Applications that deal with incomplete and contradictory information can be easily modeled using DeLP programs. The defeasible argumentation basis of DeLP allows the building of applications for dynamic domains, where information may change. In the thesis, a concrete application of DeLP for building deliberative Intelligent agents was developed. This application consist of a multi-agent system for the stock market domain, where several agents can be programmed for monitoring the stock market and performing actions based on the retrieved information. The agents have a Reasoning Module, based on DeLP, capable of formulating arguments and counterarguments in order to decide whether to perform an action or not. For instance, an agent can be programmed to alert the user when it is the right moment for buying some stock.

Future work includes two main areas. On one hand, we will continue the basic research on the DeLP proof procedure and in the area of defeasible argumentation. On the other hand, new applications for DeLP will be considered.

.

References

[1] Grigoris Antoniou, Michael J. Maher, and David Billington. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, 42:47–57, 2000.

[2] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[3] Yannis Dimopoulos and Antonis Kakas. Logic programming without negation as failure. In *Proceedings of 5th. International Symposium on Logic Programming*, pages 369–384, Cambridge, MA, 1995. MIT Press.

[4] Phan M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[5] Alejandro J. García. *Defeasible Logic Programming: Definition, Operational Semantics and Parallelism*. PhD thesis, Computer Science Department, Universidad Nacional del Sur, Bahía Blanca, Argentina, December 2000.

[6] Alejandro J. García, Devender Gollapally, Paul Tarau, and Guillermo Simari. Deliberative stock market agents using jinni and defeasible logic programming. In *Proceedings of ESAW'00 Engineering Societies in the Agents' World, Workshop of ECAI 2000*, August 2000.

[7] Alejandro J. García and Guillermo R. Simari. Parallel defeasible argumentation. *Journal of Computer Science and Technology Special Issue: Artificial Intelligence and Evolutive Computation. http://journal.info.unlp.edu.ar/*, 1(2):45–57, 1999.

[8] Alejandro J. García and Guillermo R. Simari. Strong and default negation in defeasible logic programming. In *Proc. Fourth Dutch-German Workshop on Nonmonotonic Reasoning Techniques and Their Applications, DGNMR'99*, March 1999.

[9] Michel Gelfond and Tran Cao Son. Reasoning with prioritized defaults. In *Lecture Notes in Artificial Intelligence 1471, Selected Papers from the Workshop on Logic Programming and Knowledge Representation*, pages 164–223, 1997.

[10] Steve Gregory. *Parallel Logic Programming in PARLOG. The language and its implementation.* Addison-Wesley, 1987.

[11] Gopal Gupta. *Multiprocessor Execution of Logic Programs.* Kluwer Academic Publishers, 1994.

[12] Gopal Gupta, Khayri, A.M. Ali, Manuel Hermenegildo, and Mats Carlsson. Parallel execution of prolog programs: A survey. Technical report, Department of Computer Science, New Mexico State University, 1994. http://www.cs.nmsu.edu/lldap/pub_para/survey.html.

[13] M. Hermenegildo. *An Abstract Machine Based Execution Model for Computer Architecture Design and Efficient Implementation of Logic Programs in Parallel.* PhD thesis, Dept. of Electrical and Computer Engineering (Dept. of Computer Science TR-86-20), University of Texas at Austin, Austin, Texas 78712, August 1986.

[14] Vladimir Lifschitz. Foundations of logic programs. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 69–128. CSLI Pub., 1996.

[15] Ronald P. Loui. et al. Progress on Room 5: A Testbed for Public Interactive Semi-Formal Legal Argumentation. In *Proc. of the 6th. International Conference on Artifcial Intelligence and Law*, July 1997.

[16] D. Nute. Defeasible logic. In D.M. Gabbay, C.J. Hogger, and J.A.Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol 3*, pages 355–395. Oxford University Press, 1994.

[17] John Pollock. Implementing defeasible reasoing. *workshop on Computation Dialectics*, 1996.

[18] Henry Prakken and Giovanni Sartor. Argument-based logic programming with defeasible priorities. *J. of Applied Non-classical Logics*, 7(25-75), 1997.

[19] Guillermo R. Simari and Ronald P. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53:125–157, 1992.

[20] Francesca Toni and A. C. Kakas. Computing the acceptability semantics. In *Proceedings of the 3rd. International Workshop on Logic Programming and Non-monotonic reasoning*, pages 401–415, Lexington,USA, June 1995. Springer Verlag.