On Programming with Logic Rules and Everything Else

Yanhong A. Liu
Computer Science Department, Stony Brook University
liu@cs.stonybrook.edu

September 14, 2022

Abstract

Logic rules are powerful for expressing complex reasoning and analysis problems. At the same time, they are inconvenient or impossible to use for many other aspects of practical applications. Integrating logic rules in a language with sets and functions, and furthermore with updates to objects, has been a subject of significant study. What's lacking is a language that integrates all constructs seamlessly.

This paper gives an overview of such a language, Alda, especially including how declarations can be used to support more powerful rules for knowledge representation and reasoning, and how methods and systems for efficient implementations of rules can used to build an integrated implementation.

1 Alda: A powerful high-level language

Alda [LST+22] supports all of logic rules, sets, functions, updates, and objects as seamlessly integrated built-ins, including concurrent and distributed processes. The key idea is to support predicates in rules as set-valued variables that can be used and updated directly, and support queries using rules as either explicit or implicit automatic calls to an inference function.

Alda has a formal semantics, is implemented in a prototype compiler that builds on an object-oriented language (DistAlgo [LS09, Dis] extending Python [Pyt22]) and an efficient logic rule system (XSB [SW12, SWS+22]), and has been used successfully on benchmarks and problems from a wide variety of application domains. The current implementation supports Datalog rules extended with unrestricted negation and computes well-founded semantics [VRS91] using XSB, but more general forms of rules and queries can be compiled to XSB rules and queries in a similar fashion.

2 Declarations for more powerful logic rules

For advanced knowledge representation and reasoning, Alda is designed to support declarations for predicates, even though this is not yet implemented and typical defaults are assumed. The declarations can express scopes and types of predicates as usual but, more fundamentally, different underlying assumptions about the predicates, as in founded semantics and constraint semantics [LS20]. The key idea is that each predicate can be declared certain, complete, or closed, or otherwise (e.g., being complete means all rules with the predicate in the conclusion have been given); then the same inference using least fixed point computation and constraint solving yield different desired semantics: well-founded, stable models, etc., and all possible combinations for different predicates.

Furthermore, to support easy use of different desired semantics, especially with modular use of rules, the knowledge units in DA-logic [LS21] can be mapped to rule sets in modules in Alda.

3 Efficient implementations

For efficient inference and queries, Alda is designed to allow any methods and systems to be used, so long as they provide a function for taking a set of rules, facts, and queries, and returning the results of the queries. This is the current implementation of inference using XSB, through an external interface (with data passing via files and invocation via command lines), and the performance is already generally good for our benchmarks.

For complex practical applications and for implementation details to be hidden completely from programmers, efficient implementations with performance guarantees are highly desired, as studied previously for Datalog [LS09, TL10, TL11]. Additionally, Alda also supports direct updates to predicates that automatically trigger calls to the inference function to maintain dependent predicates so as to preserve the declarative semantics of rules. For efficiency, this requires use of incremental query evaluation [RL08, LBSL16] including for circular dependencies [SR03].

References

[Dis] DistAlgo. `distalgo.cs.stonybrook.edu`. Accessed September 14, 2022.

[LBSL16] Yanhong A. Liu, Jon Brandvein, Scott D. Stoller, and Bo Lin. Demand-driven incremental object queries. In *Proceedings of the 18th Interna-*

tional Symposium on Principles and Practice of Declarative Programming, pages 228–241. ACM Press, 2016.

[LS09] Yanhong A. Liu and Scott D. Stoller. From Datalog rules to efficient programs with time and space guarantees. *ACM Transactions on Programming Languages and Systems*, 31(6):1–38, 2009.

[LS20] Yanhong A. Liu and Scott D. Stoller. Founded semantics and constraint semantics of logic rules. *Journal of Logic and Computation*, 30(8):1609–1638, Dec. 2020. Also `http://arxiv.org/abs/1606.06269`.

[LS21] Yanhong A. Liu and Scott D. Stoller. Knowledge of uncertain worlds: Programming with logical constraints. *Journal of Logic and Computation*, 31(1):193–212, Jan. 2021. Also `https://arxiv.org/abs/1910.10346`.

[LST+22] Yanhong A. Liu, Scott D. Stoller, Yi Tong, Bo Lin, and K. Tuncay Tekle. Programming with rules and everything else, seamlessly. *Computing Research Repository*, arXiv:2205.15204 [cs.PL], May 2022. `http://arxiv.org/abs/2205.15204`.

[Pyt22] Python Software Foundation. Python. `http://python.org/`, Accessed September 14, 2022.

[RL08] Tom Rothamel and Yanhong A. Liu. Generating incremental implementations of object-set queries. In *Proceedings of the 7th International Conference on Generative Programming and Component Engineering*, pages 55–66. ACM Press, 2008.

[SR03] Diptikalyan Saha and C. R. Ramakrishnan. Incremental evaluation of tabled logic programs. In *Proceedings of the 19th International Conference on Logic Programming*, pages 392–406. Springer, 2003.

[SW12] Terrance Swift and David S Warren. XSB: Extending Prolog with tabled logic programming. *Theory and Practice of Logic Programming*, 12(1-2):157–187, 2012.

[SWS+22] Theresa Swift, David S. Warren, Konstantinos Sagonas, Juliana Freire, Prasad Rao, Baoqiu Cui, Ernie Johnson, Luis de Castro, Rui F. Marques, Diptikalyan Saha, Steve Dawson, and Michael Kifer. *The XSB System Version 5.0,x*, May 2022. `http://xsb.sourceforge.net`. Latest release May 12, 2022.

[TL10] K. Tuncay Tekle and Yanhong A. Liu. Precise complexity analysis for efficient Datalog queries. In *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*, pages 35–44, 2010.

[TL11] K. Tuncay Tekle and Yanhong A. Liu. More efficient Datalog queries: Subsumptive tabling beats magic sets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 661–672, 2011.

[VRS91] Allen Van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.