Enhancing network diagnosis with reflection in Prolog

Anduo Wang[0000−0002−1078−107X]

Temple University, Philadelphia, PA 19122, USA
anduo.wang@gmail.com
https://anduowang.github.io/

1 Introduction

The use of formal methods has helped transform network diagnosis, alongside software-defined networking (SDN) [28] and programmable networks [6], from a black art into a more disciplined practice [22]: Trial-and-error guesswork with distributed protocols has been replaced by systematic software engineering [29], rudimentary troubleshooting methods of the past, such as ping and traceroute, have been supplemented, if not entirely overtaken, by formal verification and synthesis [5,17,18,14,16,30,33,1,9]. A notable example is lightning fast reachability analysis tools deployed in today's hyper-scale datacenters [16,14,24]. Indeed, with carefully crafted network representations and highly effective reasoning engines, formal methods-based diagnosis flourished, thanks to the rise of the clean-slate networks —– over-provisioned datacenters in particular -—- that assume well-defined control software and well-measured workloads.

But networks in the wild remain mundane. Consider the public Internet, enterprise networks, or smart-spaces; they are all evolving with unanticipated requirements while operating continuously and are not stand-alone or well-planned. The clean-slate diagnosis alone is inadequate in this context: A reachability checker outperforms humans by orders of magnitudes in catching an anomaly (property violation) in a network's forwarding states, but it is still up to the human expert to explain how the network procedure (protocols or SDN software) actually produces that anomaly. Similarly, state-of-art tools scale to hyper-scale networks with millions of nodes, but only a human expert can direct the course of diagnosis to changing circumstances, new concerns, and past results — a specific context, so to speak — discriminating between important and less relevant cases so the diagnosis retains a sharp focus. One key missing in tools but achieved by humans is the ability to reflect [26] —– the capability to reason not only about networks but also about the reasoning process itself.

Can we, then, bring in some form of reflection as a means to more effective network diagnosis? Can we go beyond "what properties a network state hold" and reflect on the increasingly complex procedures [7,9,11] that manipulate those states, explaining why and how a certain (unintended) state is encountered [10,7]? While today's diagnosis are performed in isolation and are optimized for fixed metrics such as maximum coverage, can we make a more

active use of a problem's context, by using that extra knowledge to better frame and scope the diagnosis and reflecting on the diagnosis (reasoning) process itself, enable diagnosis formulation and test generation that directly reflect the context [32,4,34]? These are just a few possibilities, of course. This extended abstract presents an initial study of reflective network diagnosis, discussing its effectiveness and feasibility.

2 Ongoing work

We identify two reflective use cases: the first complements current diagnosis that focuses on *what* the network states are (detecting anomalies). We instead reason about *how* and *why* the network arrives at an anomaly by reflecting on the state-manipulating procedures. The second case seeks to go beyond context-free tools developed in isolation. By developing an explicit account of context — such as a motivating concern or a related operation that do not easily refactor into traditional problem formulation — within diagnosis, and seek to bring into the diagnosis a sharper focus.

Non-monotonic diagnosis of network procedures Network verification [9,35,30], the most advanced tool today, reasons about network forwarding states (data), but does not reflect on the procedures — protocols or software (data-manipulation procedures) — that produce the states. Those state manipulation programs in real networks, however, are interesting by themselves. A notable example is the so called oscillation problem of Internet routing [12,13], in which the path computation of a set of uncoordinated routers (e.g., with conflicting path preferences) fails to converge. Figure1 (left) depicts a subtle oscillation scenario reported in [23]. Built atop this example, we propose reflective diagnosis *Case I: Reflect on how a network arrives at certain state, even when the procedure (e.g., protocols, control software) generating the states is non-deterministic or non-terminating.*

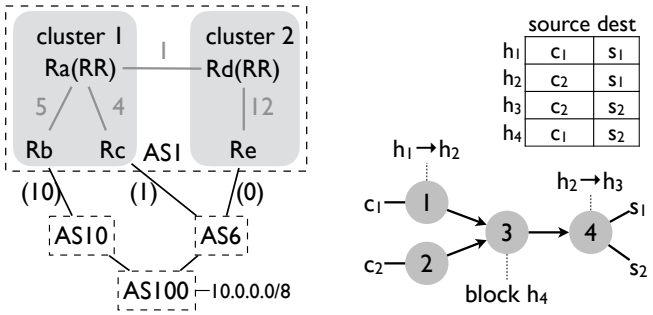

Fig. 1: (left) iBGP oscillation in `AS1`: reflectors `Ra` and `Rd` make route decisions based on IGP path cost (in gray) and MED (in parenthesis). (right) $\mathtt{c}_1$ can send packets to $\mathtt{s}_2$ and bypass the filter at `3`.

Context-aware debugging of network states A second limitation we seek to address is that existing diagnosis solutions often assume a well-defined problem in a vacuum, but networks are "mundane", and real problems almost always arise from a specific context. To see the subtlety of "context", consider the simple network in Figure 1 (right). A network of 4 nodes (routers) is carrying traffic between clients $\mathtt{c}_1, \mathtt{c}_2$ and servers $\mathtt{s}_1, \mathtt{s}_2$ that are divided into 4 flows (group of packets), as shown in the table (so the first class $\mathtt{h}_1$ are those from $\mathtt{c}_1$ (source) to $\mathtt{s}_1$). The arrows denote the next-hop of the forwarding action, and the $\mathtt{h}_\mathtt{x} \rightarrow \mathtt{h}_\mathtt{y}$ label of a node says the header rewrite (of either a packet's source or destination field). So in addition to the usual packet forwarding (to the next-hop), node `1` turns flow $\mathtt{h}_1$ into $\mathtt{h}_2$. Node `3` also drops any packet in $\mathtt{h}_4$, thus acting as a firewall blocking $\mathtt{h}_4$. Suppose the rewrites at $\mathtt{1}, \mathtt{4}$ are recently introduced, the operator wants to know the impact of them on the firewall `3`. Such context — does two seemingly innocent rewrites collectively viloate a firewall? — does not easily translate to well-supported diagnosis problems today. For example, reachability diagnosis typically checks for certain (by default all) packets entering a network at certain entry points. What are the right (relevant) flows and entries to test? More generally, we propose reflective diagnosis *Case II: Raise contexts to first-class objects in the diagnosis process by (as a first step) using contexts natively in the problem specification and directly in test generation, saving the user from manually refactoring them in.*

Implementation We built a prototype of the above reflective cases by enhanced meta-interpreters in prolog [2,3,15]. Our key insight is that reflection [8,26] is a specific form of meta-reasoning: reasoning about another reasoning process which happens to be itself. Reflection thus aligns well with meta-programming, meta-interpreters in particular, where a program treats another program (or itself) as object data, which is well supported in Prolog [20,27,19,21], because it does not distinguish between data and programs like its more conventional counterparts and includes first-class meta-level constructs. In addition, Prolog has a clear procedural reading -—- a sequential execution model — that makes it remarkably suitable for capturing the operational complexities of network procedures (e.g., routing). Based on these observations, we use meta-interpreters as a reflection engine and an injection point for enhancements tailored to the new proposed functions.

All implementations are fully tested on the latest stable version of XSB 5.0 (May 12, 2022) [31], and are publicly available [25]. The # of clauses (in parenthesis) for the two cases is: iBGP (77), data-plane (19); Three enhanced meta-interpreters for iBGP explaination (10), Two data-plane test generations (5 for each). We also implemented a total of `46` utility clauses such as `length/2` (for returning the length of a list) to avoid processing external libraries in the meta-interpreter, and additional facilities like `pretty_print` to display an explanation.

References

1. Abhashkumar, A., Gember-Jacobson, A., Akella, A.: Tiramisu: Fast multilayer network verification. In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20) (2020)
2. Abramson, H., Rogers, M.H. (eds.): Meta-programming in logic programming. MIT Press, Cambridge, MA, USA (1989)
3. Apt, K.R.: From logic programming to Prolog. Prentice-Hall, Inc., USA (1996)
4. Beckett, R., Mahajan, R.: Putting network verification to good use. In: Proceedings of the 18th ACM Workshop on Hot Topics in Networks. HotNets '19, Association for Computing Machinery (2019). https://doi.org/10.1145/3365609.3365866, https://doi.org/10.1145/3365609.3365866
5. Beckett, R., Mahajan, R., Millstein, T., Padhye, J., Walker, D.: Don't mind the gap: Bridging network-wide objectives and device-level configurations. In: Proceedings of the 2016 ACM SIGCOMM Conference. pp. 328–341. SIGCOMM '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2934872.2934909, http://doi.acm.org/10.1145/2934872.2934909
6. Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., Walker, D.: P4: Programming protocol-independent packet processors. SIGCOMM Comput. Commun. Rev. **44**(3), 87–95 (Jul 2014). https://doi.org/10.1145/2656877.2656890, http://doi.acm.org/10.1145/2656877.2656890
7. Brown, M., Fogel, A., Halperin, D., Heorhiadi, V., Mahajan, R., Millstein, T.: Lessons from the evolution of the batfish configuration analysis tool. In: Proceedings of the ACM SIGCOMM 2023 Conference. p. 122–135. ACM SIGCOMM '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3603269.3604866, https://doi.org/10.1145/3603269.3604866
8. Demers, J.M.F.N.: Reflection in logic, functional and object-oriented programming: a short comparative study. In: IJCAI'95, Workshop on Reflection and Metalevel Architectures and their Applications in AI. pp. 29–38 (1995), http://fparreiras/papers/reflectionlogicfuncoocomparative.pdf
9. Fayaz, S.K., Sharma, T., Fogel, A., Mahajan, R., Millstein, T., Sekar, V., Varghese, G.: Efficient network reachability analysis using a succinct control plane representation. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. p. 217–232. OSDI'16, USENIX Association, USA (2016)
10. Fogel, A., Fung, S., Pedrosa, L., Walraed-Sullivan, M., Govindan, R., Mahajan, R., Millstein, T.: A general approach to network configuration analysis. NSDI'15, USENIX Association, USA (2015)
11. Gember-Jacobson, A., Viswanathan, R., Akella, A., Mahajan, R.: Fast control plane analysis using an abstract representation. In: Proceedings of the 2016 ACM SIGCOMM Conference. p. 300–313. SIGCOMM '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2934872.2934876, https://doi.org/10.1145/2934872.2934876
12. Griffin, T.G., Shepherd, F.B., Wilfong, G.: The stable paths problem and interdomain routing. IEEE Trans. on Networking **10**, 232–243 (2002)
13. Griffin, T.G., Wilfong, G.: An analysis of BGP convergence properties. In: SIGCOMM (1999)
14. Guo, D., Chen, S., Gao, K., Xiang, Q., Zhang, Y., Yang, Y.R.: Flash: Fast, consistent data plane verification for large-scale network settings. In: Proceedings of the ACM SIGCOMM 2022 Conference. p. 314–335. SIGCOMM '22, Association for

Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3544216.3544246, https://doi.org/10.1145/3544216.3544246
15. Hill, P.M., Gallagher, J.: Meta-Programming in Logic Programming. In: Handbook of Logic in Artificial Intelligence and Logic Programming: Volume 5: Logic Programming. Oxford University Press (01 1998). https://doi.org/10.1093/oso/9780198537922.003.0010, https://doi.org/10.1093/oso/9780198537922.003.0010
16. Jayaraman, K., Bjørner, N., Padhye, J., Agrawal, A., Bhargava, A., Bissonnette, P.A.C., Foster, S., Helwer, A., Kasten, M., Lee, I., Namdhari, A., Niaz, H., Parkhi, A., Pinnamraju, H., Power, A., Raje, N.M., Sharma, P.: Validating datacenters at scale. In: Proceedings of the ACM Special Interest Group on Data Communication. p. 200–213. SIGCOMM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341302.3342094, https://doi.org/10.1145/3341302.3342094
17. Kazemian, P., Chang, M., Zeng, H., Varghese, G., McKeown, N., Whyte, S.: Real time network policy checking using header space analysis. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. p. 99–112. nsdi'13, USENIX Association, USA (2013)
18. Khurshid, A., Zhou, W., Caesar, M., Godfrey, P.B.: Veriflow: Verifying network-wide invariants in real time. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. p. 49–54. HotSDN '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2342441.2342452, https://doi.org/10.1145/2342441.2342452
19. Kowalski, R.: Logic programming. In: Siekmann, J.H. (ed.) Computational Logic, Handbook of the History of Logic, vol. 9, pp. 523–569. North-Holland (2014). https://doi.org/https://doi.org/10.1016/B978-0-444-51624-4.50012-5, https://www.sciencedirect.com/science/article/pii/B9780444516244500125
20. Körner, P., Leuschel, M., Barbosa, J., Costa, V.S., Dahl, V., Hermenegildo, M.V., Morales, J.F., Wielemaker, J., Diaz, D., Abreu, S., Ciatto, G.: Fifty years of prolog and beyond (2022)
21. Lloyd, J.W.: Foundations of logic programming; (2nd extended ed.). Springer-Verlag, Berlin, Heidelberg (1987)
22. Making Networks Safe and Agile with Formal Methods and Programming Abstractions: Future Directions: http://tinyurl.com/2by799dz/. NSF Workshop on Long-Term Research Directions in Wired Networking, Cornell Tech, Roosevelt Island, New York City, September 15-15, 2023
23. McPherson, D., Gill, V., Walton, D., Retana, A.: Border Gateway Protocol (BGP) persistent route oscillation condition (RFC 3345, 2002)
24. Plotkin, G.D., Bjørner, N., Lopes, N.P., Rybalchenko, A., Varghese, G.: Scaling network verification using symmetry and surgery. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2837614.2837657, https://doi.org/10.1145/2837614.2837657
25. Prolog implementation:: https://github.com/wadaries/reflective-diagnosis-padl25-extended-abstract.git
26. Smith, B.C.: Procedural Reflection in Programming Languages. Ph.D. thesis, Massachusetts Institute of Technology, Laboratory for Computer Science (1982), http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-272.pdf

27. Sterling, L., Shapiro, E.: The art of Prolog (2nd ed.): advanced programming techniques. MIT Press, Cambridge, MA, USA (1994)
28. The Future of Networking, and the Past of Protocols: http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf
29. Vahdat, A., Clark, D., Rexford, J.: A purpose-built global network: Google's move to sdn. Queue **13**(8), 100:100–100:125 (Oct 2015). https://doi.org/10.1145/2838344.2856460, http://doi.acm.org/10.1145/2838344.2856460
30. Xie, G., Zhan, J., Maltz, D., Zhang, H., Greenberg, A., Hjalmtysson, G., Rexford, J.: On static reachability analysis of ip networks. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. (2005). https://doi.org/10.1109/INFCOM.2005.1498492
31. XSB homepage: https://xsb.sourceforge.net/
32. Xu, X., Beckett, R., Jayaraman, K., Mahajan, R., Walker, D.: Test coverage metrics for the network. In: Proceedings of the 2021 ACM SIGCOMM 2021 Conference. p. 775–787. SIGCOMM '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3452296.3472941, https://doi.org/10.1145/3452296.3472941
33. Ye, F., Yu, D., Zhai, E., Liu, H.H., Tian, B., Ye, Q., Wang, C., Wu, X., Guo, T., Jin, C., She, D., Ma, Q., Cheng, B., Xu, H., Zhang, M., Wang, Z., Fonseca, R.: Accuracy, scalability, coverage: A practical configuration verifier on a global wan. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. p. 599–614. SIGCOMM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3387514.3406217, https://doi.org/10.1145/3387514.3406217
34. Zeng, H., Kazemian, P., Varghese, G., McKeown, N.: Automatic test packet generation. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies. p. 241–252. CoNEXT '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2413176.2413205, https://doi.org/10.1145/2413176.2413205
35. Zhang, P., Liu, X., Yang, H., Kang, N., Gu, Z., Li, H.: Apkeep: Realtime verification for real networks. In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20). pp. 241–255. USENIX Association, Santa Clara, CA (Feb 2020), https://www.usenix.org/conference/nsdi20/presentation/zhang-peng