Interpretable Network Synthesis via Localized Specifications

Haoxian Chen
ShanghaiTech University
China

ABSTRACT

Network synthesis simplifies network management by automatically generating distributed configurations that fulfill high-level intents. However, typical network synthesizers operate as monolithic algorithms, obscuring the internal workings of the synthesis process and showing no clear connection between the generated configurations and the global intents. Given the critical role of networks as infrastructure, it is crucial for network operators to understand what the synthesizer generates in order to establish trust in these automatic tools. To address this challenge, we propose using sub-specifications localized to each component in the network topology to enhance the interpretability of network synthesis. These sub-specifications provide insights into the workings of synthesizers by connecting each component's functionalities with the global configuration intents. In this extended abstract, we discuss the challenges and opportunities in enhancing the interpretability of network synthesis and call for collective effort in the networking community to work towards this vision.

CCS CONCEPTS

• **Networks** → **Formal specifications**; **Network manageability**.

KEYWORDS

Network Synthesis, Explanability, Modular Reasoning

ACM Reference Format:
Haoxian Chen. 2024. Interpretable Network Synthesis via Localized Specifications. In *SIGCOMM Workshop on Formal Methods Aided Network Operation (FMANO '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3672199.3673889

1 INTRODUCTION

Managing networks via manual configurations has been notoriously difficult and error-prone. Intent-Based Networking (IBN) is a new paradigm that addresses this issue by automatically generating distributed network configurations that enforce high-level intents. Intents provide an intuitive and convenient interface for network management and enable automatic verification [2–4, 10, 13, 15, 16] and synthesis [1, 9, 12, 14] of network configurations, greatly improving the correctness and robustness of networks.

Despite these exciting advancements, the trust and interpretability issues of network synthesis have received limited scrutiny. Practitioners are often concerned that the output of a synthesizer is too complex to understand [9, 11]. Indeed, network configurations are large in scale, and involve complex interactions between protocols and network dynamics, making it challenging for network operators to comprehend how the synthesizer fulfills their intents.

Despite these exciting advancements, the trust and interpretability of network synthesis have received limited scrutiny. Practitioners often express concerns that the output of a synthesizer is too complex to understand [9, 11], hindering wider adoption in practice. Network configurations are indeed large in scale and involve complex interactions between protocols and network dynamics, making it challenging for network operators to comprehend how the synthesizer fulfills their intents.

Although synthesis outputs are typically formally verified against user-specified intents, they are not without potential pitfalls. The complex nature of network systems and their protocols means that both verifiers and synthesizers can contain bugs [7, 8]. Additionally, the intent specification itself can sometimes be ambiguous [8, 12]. Tools [7, 8] have been introduced to proactively identify verifier bugs, but synthesis still operates in a black-box mode.

NetComplete [9] enhances interpretability by using templates familiar to users. However, it requires users to have a comprehensive understanding of the original configurations and to identify which parts need changes to support new policies. Notably, none of these tools addresses the fundamental interpretability problem of network synthesis output, leaving a gap in making synthesized configurations easily understandable and trustworthy for network operators.

In contrast to the end-to-end specification and synthesis flow, when network operators manually configure the network, they typically configure network devices one by one. This process implicitly decouples the global policy into local functionalities, with operators reasoning about how these local functionalities connect to the global policies.

Research on modular network verification [3, 13] formalizes this idea into a semi-automatic process, where network operators explicitly specify local invariants for different network components. This modularization, akin to modularized reasoning in software, significantly enhances verification efficiency and enables easy bug localization in case of verification failure. These work shed light on the possibilities of decomposing global intent specifications into localized specifications for individual network devices or components.

Interpretability to foster trust. Inspired by these ideas, we aim to utilize sub-specifications to foster trust in network synthesis. Given a high-level network intent and the synthesized network configuration, our goal is to generate sub-specifications for each configuration component. These sub-specifications collectively establish the validity of the global specifications. Similar to function comments that improve software readability, sub-specifications

FMANO '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0714-8/24/08.
https://doi.org/10.1145/3672199.3673889

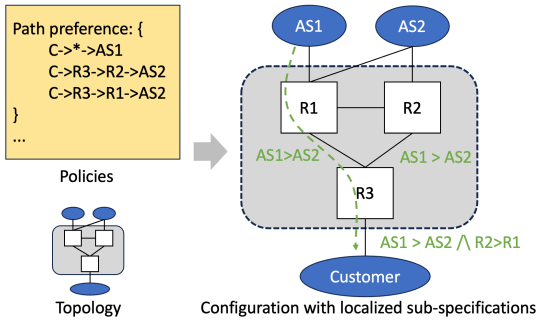

Figure 1: Sub-specifications show how the synthesized configurations can enforce the path preferences.

establish connections between each part of the network configurations and the global intents, revealing insights on why the generated configurations can fulfill the global intents.

In addition to building trust, sub-specifications can provide richer information for network operators to interact with a network synthesizer. They can identify under-specified parts of the network or intents, thereby building accurate and comprehensive specifications in an interactive manner.

Challenges. Generating and validating sub-specifications is inherently difficult [3, 11, 13]. Additionally, real-world networks serve many purposes simultaneously, requiring numerous specifications to fully articulate their intents, and some of them may conflict. For example, availability specifications that mandate maintaining connections during link failures may clash with path preferences that prevent routers in maintenance states from carrying traffic [12]. Interpretable feedback can help network operators identify such contentions and resolve them by modifying the intent specifications. Furthermore, the scale and complexity of network configurations add another layer of difficulty. Even with localized specifications, the sheer volume of configurations can be overwhelming. Maintaining similarity in sub-specification structures across devices may mitigate this issue [1].

Opportunities. Firstly, network topology provides a natural basic structure for sub-specifications. Indeed, many network synthesis tools [5, 6, 9, 12] have already employed a modularized design, where configurations for each protocol and device are generated in an organized structure. Exposing the internal reasoning processes of these tools as sub-specifications can be a natural starting point for interpretable network synthesis. Furthermore, explainable program synthesis [11, 17] has shown success in improving the success rate and usability of software program synthesis. The underlying principles and techniques from this research can be applied to benefit network synthesizer design.

2 MOTIVATING EXAMPLES

To see how localized sub-specifications can assist network synthesis interpretability, consider the network in Figure 1, which connects a customer Autonomous System (AS) with two provider ASes. The synthesis task is to generate BGP configurations such that customer traffic is routed through AS1, and then AS2 via R2, and AS2 via R1, in that order. A synthesizer is then given two inputs: (1) network intents that specify the preferred routing order for customer traffic, as shown in the upper left of Figure 1. and (2) the network topology depicted in the lower left.

Understanding synthesizer output. The synthesized configurations are presented with sub-specifications annotated with each component of the network topology. These sub-specifications illustrate what each router in the network is doing in order to fulfill the specified global path preferences. Specifically, all desired paths are accepted by all routers along the routes. The path for one particular route is highlighted by the dashed arrow in Figure 1, while preferences among alternative paths are maintained along the route, as indicated by annotations on the network edges.

In this example, sub-specifications connect the policies of each individual router with the global policies. Our thesis is that by inspecting these sub-specifications, network operators can gain insight into how the synthesizer works and thus can better validate that the synthesis output indeed fulfills their specified intents.

Identify and correct misspecifications. Another use of localized sub-specifications is to identify and correct misspecifications. Suppose the network operator adds another policy to prevent transit traffic between AS1 and AS2 ($\neg(AS1 \rightarrow * \rightarrow AS2)$). However, they mistakenly specify a policy that prevents traffic from the customer to AS1 ($\neg(C \rightarrow * \rightarrow AS2)$). By inspecting the generated sub-specification at the edge from R3 to the customer, one spots that routes from AS1 are dropped, which is inconsistent with the original intent to prevent transit traffic. Thus, the operator can realize the misspecification and correct it accordingly.

3 CONCLUSION

This abstract highlights the importance and benefits of interpretability in network synthesis to foster trust. Going forward, it is essential for the networking research community to investigate technical advancements that enable more interpretable designs. A promising direction is to embrace a modularized approach and expose internal reasoning to support better interpretability and user interaction. By enhancing interpretability, we can bridge the gap between complex automated systems and human operators, leading to more robust and trustworthy network infrastructures.

This work does not raise any ethical issues.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their insightful feedback. This work is supported by the ShanghaiTech Startup Fund.

REFERENCES

[1] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. Aed: Incrementally synthesizing policy-compliant and manageable configurations. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pages 482–495, 2020.

[2] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. Tiramisu: Fast multilayer network verification. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 201–219, 2020.

[3] Timothy Alberdingk Thijm, Ryan Beckett, Aarti Gupta, and David Walker. Modular control plane verification via temporal invariants. *Proceedings of the ACM on Programming Languages*, 7(PLDI):50–75, 2023.

[4] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. A general approach to network configuration verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 155–168, 2017.

[5] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitendra Padhye, and David Walker. Don't mind the gap: Bridging network-wide objectives and device-level configurations. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 328–341, 2016.

[6] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitendra Padhye, and David Walker. Network configuration synthesis with abstract topologies. In *Proceedings of the 38th ACM SIGPLAN conference on programming language design and implementation*, pages 437–451, 2017.

[7] Rüdiger Birkner, Tobias Brodmann, Petar Tsankov, Laurent Vanbever, and Martin Vechev. Metha: Network verifiers need to be correct too! In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 99–113, 2021.

[8] Matt Brown, Ari Fogel, Daniel Halperin, Victor Heorhiadi, Ratul Mahajan, and Todd Millstein. Lessons from the evolution of the batfish configuration analysis tool. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 122–135, 2023.

[9] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin Vechev. {NetComplete}: Practical {Network-Wide} configuration synthesis with autocompletion. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 579–594, 2018.

[10] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. A general approach to network configuration analysis. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 469–483, 2015.

[11] Amirmohammad Nazari, Yifei Huang, Roopsha Samanta, Arjun Radhakrishna, and Mukund Raghothaman. Explainable program synthesis by localizing specifications. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA2):2171–2195, 2023.

[12] Sivaramakrishnan Ramanathan, Ying Zhang, Mohab Gawish, Yogesh Mundada, Zhaodong Wang, Sangki Yun, Eric Lippert, Walid Taha, Minlan Yu, and Jelena Mirkovic. Practical intent-driven routing configuration synthesis. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 629–644, 2023.

[13] Alan Tang, Ryan Beckett, Steven Benaloh, Karthick Jayaraman, Tejas Patil, Todd Millstein, and George Varghese. Lightyear: Using modularity to scale bgp control plane verification. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 94–107, 2023.

[14] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, et al. Safely and automatically updating in-network acl configurations with intent language. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 214–226. 2019.

[15] Fangdan Ye, Da Yu, Ennan Zhai, Hongqiang Harry Liu, Bingchuan Tian, Qiaobo Ye, Chunsheng Wang, Xin Wu, Tianchen Guo, Cheng Jin, et al. 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*, pages 599–614, 2020.

[16] Peng Zhang, Dan Wang, and Aaron Gember-Jacobson. Symbolic router execution. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 336–349, 2022.

[17] Tianyi Zhang, Zhiyang Chen, Yuanli Zhu, Priyan Vaithilingam, Xinyu Wang, and Elena L Glassman. Interpretable program synthesis. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2021.