13th International Conference on Applied Informatics
Eszterházy Károly Catholic University
Eger, Hungary, February 19–21, 2026

Program patterns in P4 programs*

Zsófia Laczkó, Máté Tejfel

Eötvös Loránd University, ELTE, Budapest
laczkozsofi@student.elte.hu
matej@inf.elte.hu

Abstract

The Programming Protocol-independent Packet Processors (P4) [2] is an open-source domain-specific programming language, which is primary developed to describe algorithms that determine the forwarding of network data packets. With this language we can specify how different network devices (switches, network interface cards (NICs), routers, etc.) process the packets.

In this research we defined and analysed positive (recommended) and negative (to be avoided) programming patterns in P4 codes. We analysed these templates by static methodology. The goal of this exploration was to reduce the number of errors in P4 codes and to demonstrate which recurring solutions support or hinder the correct and effective processing of data packets. Several articles address debugging in P4 programs, but mostly using dynamic methods such as assertions [3]. Birnfeld et al. [1] analyse P4 programs following an approach similar to the one we used. They examine problems like our negative patterns, however, they use data flow analysis specifically. Although [4] provides tools for analysing a different programming language (Python), the description of its static analysis methodology served as a basis for our approach. We applied this in the examination of patterns.

As first part of the study we investigated the positive and negative templates. As the second part, we implemented an analyser program for these. The positive patterns include the recommended coding styles, best practices and solutions. Our analyser verifies whether we have followed some predefined coding conventions, thereby it informs the developers of the quality of their code. As for the negative

*This research was supported by the project no. TKP2021-NVA-29, which has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

patterns, we examined coding strategies that seem to work, but potentially contain errors or unexpected behaviour. This category includes examples that indicate defects only in specific edge cases. Additionally, it covers coding samples that do not directly cause errors but appear unnecessary in the code, for instance an unreachable statement.

As an example, we examined the status (valid or invalid) of headers during the execution of the P4 code, and whether the valid status of a header really guarantees that all the fields have been assigned values. Our analysis also attends to what happens with a header during parsing and deparsing, as well as whether we set its status directly by the `setValid` or `setInvalid` methods. As a negative pattern, we investigated cases where a header is unnecessarily marked as valid or when we want to deparse an invalid header. These would be redundant statements in the code.

During the examination of patterns we used the specification of the programming language [6], as well as we experimented with P4C, which is the official reference compiler for P4 language. We collected possibly incorrect codes, and then we analysed in which cases problems are reported during compilation, either in the form of an error or a warning. Accordingly, in this work, we focused on the specific scenarios for which the compiler does not alert.

P4Query [5] is a static analyser tool for P4 which can map a P4 program code onto a graph applying the Gremlin graph description language [7]. With the resulting internal graph representation we are able to process the code from different perspectives. It makes possible to examine for instance a syntax tree, a symbol table or a control flow graph.

We implemented a new analyser module in P4Query based on the collected patterns. It is able to recognise certain positive or negative templates, as well as generate a transparent report of the results. During the research we utilized the analyses that are already in the P4Query tool, and we created the new module built on them. Primarily, we examined the syntax tree and the control flow graph built from the source code. Based on the results of the analysis, we provide a feedback number to developers that reflects the quality of their code. It is calculated by assigning negative scores to each found undesirable patterns and positive scores to recommended ones. Different templates contribute to the total score with different weights.

This research contributes to the prediction of frequent potential errors through the static analysis of P4 code, so we can increase the number of issues that can be identified without running the program. Moreover, by uncovering positive patterns we can help improve the quality of P4 codes. In addition, we can support work of developers by emphasising well-functioning strategies.

References

[1] K. Birnfeld, D. C. da Silva, W. Cordeiro, B. B. N. de França: *P4 Switch Code Data Flow Analysis: Towards Stronger Verification of Forwarding Plane Software*, in: NOMS 2020

- 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–8, DOI: 10.1109/NOMS47738.2020.9110307.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker: *P4: programming protocol-independent packet processors*, SIGCOMM Comput. Commun. Rev. 44.3 (July 2014), pp. 87–95, ISSN: 0146-4833, DOI: 10.1145/2656877.2656890.

[3] L. Freire, M. Neves, L. Leal, K. Levchenko, A. Schaeffer-Filho, M. Barcellos: *Uncovering Bugs in P4 Programs with Assertion-based Verification*, in: Proceedings of the Symposium on SDN Research, SOSR '18, Los Angeles, CA, USA: Association for Computing Machinery, 2018, ISBN: 9781450356640, DOI: 10.1145/3185467.3185499.

[4] H. Gulabovska, Z. Porkoláb: *Survey on Static Analysis Tools of Python Programs*, in: SQAMIA, 2019, URL: https://api.semanticscholar.org/CorpusID:209149918.

[5] D. Lukács, G. Tóth, M. Tejfel: *P4Query: Static analyser framework for P4*, Annales Mathematicae et Informaticae 57 (2023), pp. 49–64, DOI: 10.33039/ami.2023.03.002.

[6] P4 Language Consortium: *P4 Specifications*, URL: https://p4.org/specifications/ (visited on 12/30/2025).

[7] M. A. Rodriguez: *The Gremlin graph traversal machine and language (invited talk)*, in: Proceedings of the 15th Symposium on Database Programming Languages, DBPL 2015, Pittsburgh, PA, USA: Association for Computing Machinery, 2015, pp. 1–10, ISBN: 9781450339025, DOI: 10.1145/2815072.2815073.