Static Analysis Taxonomies

PALANIAPPAN MUTHURAMAN*, University of Paderborn, Germany

Context: Static analysis is a fundamental technique in software engineeering used to analyse program code without executing it. It helps in finding security vulnerabilities, defects in the software system early in the development lifecycle. Despite its effectiveness, it still faces challenges related to scalability, precision impact, performance overhead, computational complexity. Traiditional static analysis techniques may generate false positives, struggle to scale for large code bases, and demand significant computational resources. Optimizing static analysis techniques aims to enhance analysis precision, improve scalability, reduce computational costs, and minimize false positives.

Objective: Our goal is to present a clear overview of the state-of-the-art static analysis optimization techniques and highlight emerging trends in static analysis approaches. This will help us identify where most of the current research efforts are concentrated, and pinpoint areas where future researches should be directed.

Method: We conducted a Systematic Literature Review (SLR) by analyzing 124 research papers published in static analysis, program analysis, and software engineering venues over the past 15 years (January 2009 to October 2024). The primary objective of this review is to gather insights into the problems addressed by these approaches, the fundamental techniques employed, the static analysis sensitivities considered, and the potential for optimization.

Result: TODO: Write the results at the end

Conclusion: TODO: Write what has been done and what is been lacking for the futire research to be taken care of

ACM Reference Format:

Palaniappan Muthuraman. 2025. Static Analysis Taxonomies. 1, 1 (April 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Background

Here comes the background needed for the paper

2 Methodology for the SLR

For this SLR, we followed the guidelines provided by Kitchenham [1]

2.1 Research Questions

RQ1: What are the purpose of these static analysis techniques/optimizations? With this research question, we will survey the various optimization techniques in static analysis.

RQ2: How are the analyses designed and implemented?

In this research question, we conduct a detailed study of the analysis that have been developed. It also includes several sub-questions:

RQ2.1 What fundamental techniques are used for by this static analysis optimization?

RQ2.2 What sensitivity features are applied?

Author's Contact Information: Palaniappan Muthuraman, palaniappan.muthuraman@upb.de, University of Paderborn, Paderborn, Germany.

Manuscript submitted to ACM

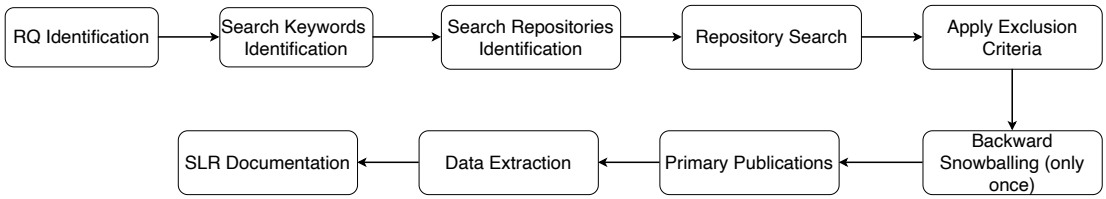

Fig. 1. An overview of the systematic literature review process.

RQ2: Are the research outputs publcly available? We aim to investigate whether the developed tools are open-source or publicly available, reproducible, and easily accessible for use by other practitioners.
RQ3: What challenges remain to be addressed? This question addresses issues that have not yet received significant research attention. It also examines how the focus of the research has evolved over time. Additionally, it helps in identifying the research gaps in the current knowledge base, and aims to understand the emerging trends and shifts in priorities within the field.

2.2 Search Strategy

This section discusses the keywords we used in our search and the datasets employed to find the relevant publications.

2.2.1 Search Keywords. We used the PICOC strategy to develop our search term. Since the **Intervention (I)** and **Comparison (C)** terms were not relevant to our scope, they were left empty.

Each of the terms from **Population (P)**, **Outcome (O)**, and **Context (C)** formed a seperate line in the search string. The final search string was constructed by logically combining these lines, using **AND** to connect different categories (P, O, C), and **OR** within each category for synonyms or related terms. i.e., s =: P **AND** O **AND** C
Table 1 shows the actual keywords we used, which were derived from a manual investigation of relevant publications.

PICOC	Search Terms
P	"control-flow analysis", "data-flow analysis", "static analysis"
O	"accuracy", "efficiency", "memory usage", "overhead", "performance", "precision", "scalability", "speedup"
C	"control-flow analysis", "data-flow analysis", "static analysis"

Table 1. Search Terms

2.2.2 Search Datasets. We used four well-known repositories, namely ACM Digital Library [1], IEEE Xplore Digital Library [2], Springer Link [3], and Google Scholar [4]. Some of these repositories impose restrictions on the amount of search result metadata that can be downloaded. For instance, Google Scholar does not allow frequent search requests from a single device via its API. To overcome this limitation, we used Publish or Perish [5], a tool that helps retrieve academic documents from Google Scholar. Similarly, Springer Link limits metadata downloads to the first 1,000 search results.

[1] http://dl.acm.org/
[2] http://ieeexplore.ieee.org/
[3] http://link.springer.com/
[4] https://scholar.google.com/
[5] https://harzing.com/resources/publish-or-perish

However, our search query yielded approximately 10,000 results on this repository. Manually downloading metadata in batches would have been tedious and time-consuming. To handle this efficiently, we used Python scripts to extract data from Springer Link, IEEE Xplore, and ACM Digital Library.

2.2.3 Exclusion Criteria. The search terms we used were quite broad, resulting in an exhaustive list of publications. Due to this broad scope, many papers in the search results may be irrelevant to our review. To refine our selection and exclude non-relevant papers, we applied specific exclusion criteria, which are detailed below:

1. Given that the majority of scientific publications today are in English, we exclude all non-English papers from our review.
2. Papers under 5 pages in double-column format or under 7 pages in LNCS single-column format were excluded. Additionally, papers exceeding 30 pages in LNCS single-column format were also excluded.
3. If multiple papers described the same or similar approaches, we included only the one with the most comprehensive description. For example, an extended journal paper [3] was selected over its shorter conference version [2].
4. Papers lacking sufficient technical details about their approaches were excluded.
5. Papers that did not focus on optimizing static analysis itself were excluded. For example, papers that use static analysis to optimize the analyzed program were considered out of scope and excluded.
6. Papers that focus on dynamic or hybrid analysis were excluded.

2.2.4 Primary publications selection. Table 2 summarizes the results of the search process. For each paper, we first reviewed the title and keywords to determine its relevance to our use case. If the relevance was unclear, we proceeded to read the abstract. If the abstract was still insufficient to make a decision, we skimmed the paper to assess its suitability. TODO: Usually here we should discuss how many reviewers did read the paper, and how did you overcome the inconsistencies the results? In the end, we got 124 papers in total.

2.2.5 Backward snowballing. To ensure the completeness of our study and to capture relevant works not identified through our initial search terms, we conducted a lightweight backward snowballing process, performed only once. The objective was to identify additional papers cited by our initially selected primary publications that align with the scope of our study. Manual snowballing process it tedious and time consuming, so we developed a series of python scripts to automate and streamline the process. First, We used pdfx [1] to extract the text from the pdf and then retrieved the references from the extracted text. Subsequently, additional scripts were used to filter out papers which fell outside the defined year boundary of our study scope. Furthermore, we used python scripts to extract keywords from the paper, and eliminated those that did not align with the scope of our review. After the automatic filtering stage, we manually reviewed the titles and abstracts of the remaining references. Papers found relevant to the scope of study, and not already included in our primary publications set were added to the final list of primary publications. The relatively high number of additional papers identified through backward snowballing can be attributed to the fine-grained terms used by the original studies, which were not fully covered by our broader search strategy. For instance, many papers contains keywords such as context-sensitivity, context, parallel processing, call graph, call site sensitivity, which, although highly relevant, were not explicitly included in our original search queries. Our initial search term aimed for broader coverage, while snowballing allowed us to capture these more granular studies.

[1] https://github.com/metachris/pdfx

Source	IEEE	ACM	Springer	Google Scholar	Total
Search Results	5561	3195	3980	1000	13736
After Reviewing Title/Keywords	44	136	13	43	317
After Reading Abstracts	117	30	22	2	171
After Skimming	72	22	3	20	117
After Final Discussion					96
Backward snowballing					28
Total					124

Table 2. Summary of the Primary Publications Selection Process

3 Graphs

Understanding how control flows through the program, how data moves between the variables, and how different program components depend on each other is a very crucial thing to know before any analysis.

There are 4 different widely used graph structures for them namely Control Flow Graph (CFG), Value Flow Graph (VFG), Program Dependence Graph (PDG), and System Dependence Graph (SDG)

References

[1] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.

[2] Jingbo Lu. 2020. *Precision-Preserving Acceleration of Object-Sensitive Pointer Analysis with CFL-Reachability.* Ph. D. Dissertation. UNSW Sydney.

[3] Jingbo Lu, Dongjie He, and Jingling Xue. 2021. Eagle: CFL-reachability-based precision-preserving acceleration of object-sensitive pointer analysis with partial context sensitivity. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 4 (2021), 1–46.