CMPUT 416 - Project proposal by Bjorn Prollius, Levi Stevenson, Favour Akinloye

Proposal

SPDS stands for “Synchronized PushDown Systems”, which is an abstraction that combines context-sensitivity, flow-sensitivity, and field-sensitivity efficiently using pushdown systems. One of the problems in static analysis is that while it's possible to create highly precise analysis combining flow-, context-, and field-sensitivity; they tend not to scale well as the programs grow in size and complexity. Field sensitivity in particular can pose an issue in code that contains cyclic field references. This is especially prevalent in programs utilizing recursive code.

To combat this issue, data-flow algorithms often use what's called k-limiting to limit the access paths to length k, which comes at the cost of limiting the precision of the analysis. SPDS however, takes a different approach to the problem. Within SPDS, two separate pushdown systems are utilized. One which is flow and context-sensitive, and another that is flow and field-sensitive. These two pushdown systems are then synchronized to provide us with data-flow results which are decidable, context and field sensitive. This allows us to avoid the issue of k-limiting that causes such issues in recursive programs by synchronizing the two pushdown systems. By doing this we are able to maintain accuracy without the extensive over-approximation caused by k-limiting in our analyses. In fact, SPDS scaling is similar to the k-limiting of k=1, with the precision of k= ∞. According to a complexity analysis provided in the paper, “Context-, Flow-, and Field-Sensitive Data-Flow Analysis using Synchronized Pushdown Systems” [Späth et al, 2019] SPDS changes the complexity from $|F|^{3k}$ to $|S||F|^2$ which vastly improves the scalability and runtime of the program at desirable levels of precision.

As none of us have yet worked with SCALA directly, our first steps will involve expanding our familiarity of the language and identifying some of the key syntactic differences between it and JAVA. Additionally, while we have a working understanding of the abstract implementation of SPDS, we need to have a more granular understanding of the finer details associated with SPDS in order to help us resolve any bugs that we may encounter over the course of this project.

Motivation

We chose this topic because it gives us an opportunity to do some hands-on work with static analysis tools without the overhead of having to create a project from scratch. This process of porting SPDS to SWAN will require us to shift SPDS ino Scala from its native java implementation and coding. A full porting of SPDS will require us to fully comprehend SPDS to make sure that functionality of the current state of the project is carried over to SWAN, allowing for future updating of SPDS within the SWAN framework to improve current state issues in the project such as silent failing and dataflow gaps.

Related Work

https://karimali.ca/resources/papers/swan.pdf
https://johspaeth.github.io/publications/boomerangPDS.pdf

https://docs.scala-lang.org/overviews/collections-2.13/conversions-between-java-and-scala-collections.html

https://docs.oracle.com/javase/8/docs/api/

https://docs.scala-lang.org/api/all.html

CMPUT 416 - Project proposal by Bjorn Prollius, Levi Stevenson, Favour Akinloye

Experimental Evaluation

We will evaluate our work by running our version of the SPDS imported to SWAN against the current version of SPDS which is implemented in Java. The comparison of the 2 outputs will allow us to check for any issues in our code that may have developed during the course of porting the project as well as to determine if any of the java-specific issues have been solved by implementing SPDS in Scala. We will also do isolation testing of the project on our local branch of the project to verify the functionality of the port. We will be porting SPDS method by method to ensure that

Expectations

We expect that the SPDS port of SWAN will offer all of the current performance aspects of the java based implementation but with the improvements offered by the more robust Scala language framework, allowing the processing of the analysis to be more efficient and to allow greater and more tailored future development while in the SWAN program framework.

Detailed Timeline

We expect to finish a direct if unoptimized port by the end of the project deadline. We will reassess our timeline every two weeks to address any unforeseen issues in the ongoing work of this project.

EoW Date	
2/11/2023	Solid grasp on SPDS math
9/11/2023	Begin Scala port, ~33% of codebase converted
16/11/2023	Continue Scala port, ~66% of raw code converted
23/11/2023	Initial working Scala port
30/11/2023	Bugfixes and feature improvement
7/12/2023	Bugfixes and feature improvement

Citations

Späth, J., Ali, K., & Bodden, E. (2019). Context-, Flow-, and Field-Sensitive Data-Flow Analysis using Synchronized Pushdown Systems. Proceedings of the ACM on Programming Languages, 3(POPL). https://doi-org.login.ezproxy.library.ualberta.ca/10.1145/3290361