

# What's Behind Program Analysis with Polymorphism

Xin Li and Mizuhito Ogawa

Japan Advanced Institute of Science and Technology, Nomi, Japan

Polymorphism is an important programming language concept. However, language features of polymorphism, such as late binding, overloading, etc., also bring difficulties to program analysis. Taking points-to analysis [1] as a typical example, in contrast to classic program analysis that assume an existing inter-procedural control flow (aka call graph) of the program, points-to analysis and call graph construction are mutually dependent on each other. We are aware that, although the problem has been studied for over two decades, there exists no analysis framework that characterizes its nature.

In this work, we investigate analysis framework for points-to analysis or the kind with involving polymorphism. Our proposal is based on the extension of weighted pushdown systems [3] and its model checking problems, as well as its equivalent counterpart abstract grammar problem [2] that is known to encode a large scope of data flow analysis. Our observation is based on the fact that when and which program transfer functions in the universe would play a part in program analysis obeys to conditions imposed by polymorphism.

On the one hand, we present an analysis framework that characterizes the impact of polymorphism on control flow analysis. We extend weighted pushdown systems to conditional weighted pushdown systems, by further specifying conditions under which a pushdown transition rule can be applied, and show that model checking problems on conditional weighted pushdown systems can be reduced to those on weighted pushdown systems.

On the other hand, we present an analysis framework that further characterizes the impact of polymorphism on data flow analysis. We lift abstract grammar problem, and discuss algorithms for solving those problems with an eye on the efficiency and space tradeoff. As a typical instance, we apply our theoretical framework to the design of context-sensitive points-to analysis and conduct preliminary empirical studies. Experimental results show that our proposal effectively guided an efficient and correct analysis design.

## References

1. O. Lhoták and L. Hendren. Context-sensitive points-to analysis: is it worth it? In *CC'06: Proceedings of the 15th International Conference on Compiler Construction*, volume 3923 of *LNCS*, pages 47–64, Vienna, Mar. 2006. Springer.
2. U. Möncke and R. Wilhelm. Grammar flow analysis. In *Proceedings on Attribute Grammars, Applications and Systems*, pages 151–186, London, UK, 1991. Springer.
3. T. Reps, S. Schwoon, S. Jha, and D. Melski. Weighted pushdown systems and their application to interprocedural dataflow analysis. *Sci. Comput. Program.*, 58(1-2):206–263, 2005.