Debug Slicing of Logic Programs

László Harmath, Gyöngyi Szilágyi and Tibor Gyimóthy

Slicing methods are widely used for the debugging, testing and maintance of imperative programs. Intuitively, a slice should contain all those parts of a program that may affect the variables in a set V at a program point p. Slicing algorithms can be classified according to whether they only use statically available information (*static slicing*), or compute those statements which influence the value of a variable occurrence for a specific program input (*dynamic slice*). Dynamic slicing methods are more appropriate for debugging than static ones as during debugging we generally investigate the program behaviour under a specific test case. The main advantage of using a dynamic slice during debugging is that many statements can be ignored in the process of bug localization.

Different dynamic slicing methods have been introduced for debugging imperative programs [3]. Most of these methods are based on a dependence graph which contains the explicit control dependences and data dependences of the program. In [1, 2] a slicing method was introduced for logic programs, and this method being used to improve the efficiency of the Shapiro's algorithmic debugging algorithm. The slice presented in [1] contains those parts of a program that actually have an influence on the value of an argument of a predicate. This type of slice (called *data flow slice*) is safe if the structure of the proof tree for a goal is not changed.

However, during debugging we also need to identify those predicates that actually did not affect an argument in a predicate but could have affected it had they been evaluated differently. We can say that these predicates are in the *Potentially Dependent Predicate Set*. We note that the different evaluation of the predicates in this set could change the structure of the success branch of the proof tree.

In this paper we introduce a new type of slicing called *Debug slicing* for Prolog programs without side effects. A Debug slice of an Augmented Proof Tree includes all those predicates that may affect the value of an argument in any success branch's predicate. So this slice is very suitable for debugging. The Debug slice is the set of predicates which contains the predicates of the success branches of the SLD-tree, the Potentially Dependent Predicates and their data dependences.

This slicing method has been integrated into an interactive algorithmic debugging tool to reduce the number of questions to the user during a debugging session [2]. The size of the debug slice is larger than the size of the data flow slice, however data flow slice is not safe for debugging. On the other hand the Debug slice contains all parts of the program that may be responsible for the incorrect behaviour at a selected argument position.

References

[1] T. Gyimóthy and J. Paakki: Static Slicing of Logic Programs.*In Proceedings of Second International Workshop on Automated and Algorithmic Debugging (AADEBUG'95)*, pages 85-105, Saint Malo, France, May 1995.

[2] G. Kókai, L. Harmath, T. Gyimóthy: Algorithmic Debugging and Testing of Prolog Programs. *In Proceedings of the 14th International Conference on Logic Programming (ICLP'97). Eighth Workshop on Logic Programming Environments*, pages 14-21, Leuven Belgium, July 1997.

[3] F. Tip: A survey of Program Slicing Techniques. *Journal of Programming Languages*, Vol.3., No.3, pages 121-189, September, 1995.