Adding Weights to Dynamic Pushdown Networks

Extended Abstract

Alexander Wenner

Institut für Informatik, Fachbereich Mathematik und Informatik
Westfälische Wilhelms-Universität Münster
alexander.wenner@uni-muenster.de

The interest in writing parallel programs has increased in recent years, especially with the emergence of programming languages with native support for parallelism and the increased availability of parallel hardware. However parallel programming is notoriously difficult and error-prone. Thus static analysis of parallel programs has become more and more important.

The goal of this talk is to present a generic framework for the analysis of parallel programs, especially in the presence of recursive procedures and dynamic process creation. We base our framework on dynamic pushdown networks (DPN) [1] and weighted pushdown systems (WPDS) [2].

DPN precisely model procedures and process creation, by modeling each process as a pushdown system (PDS), where rules can additionally create new PDS as a side effect. They have mainly been studied for reachability analyses [1, 3, 4]. Since the analysis of recursive procedures and synchronisation is undecidable [5], DPN do not model synchronisation between processes. However, through the addition of weights we will be able to analyse some interaction between processes.

WPDS extend PDS by labelling transitions with weights and solving the generalised pushdown predecessor (GPP) problem [2, 6, 7], which is the meet-over-all-paths solution for paths between regular sets of configurations. The weights can be used to formulate a wide range of analysis problems. The GPP problem formulation allows for a specific query depending on the shape of the entire call-stack, in contrast to standard dataflow techniques, where typically all information at the topmost program point is merged.

Analogous to WPDS we extend DPN to weighted DPN (WDPN) by annotating weights to transitions and study the corresponding GPP problem, which now allows for a query based on the state of each process in the network. Even though a WPDS is then simply a WDPN with one process, adapting the approach to solve the GPP problem from WPDS to WDPN is problematic. In general a path of a DPN is an interleaving of the transitions of arbitrary many parallel processes. Results from [1] show, that such a set of paths can not be described using a grammar as in [2] or a constraint system. Thus the solution of the GPP problem can not be computed as abstract interpretation [8] of such a system.

We avoid these problems by introducing a branching semantics for DPN, similar to the tree semantics in [4]. Transitions of newly spawned processes are

no longer mixed with the transitions of the creating process, but contained in their own branch. This results in executions which are tree shaped for single processes and form hedges, which contain a tree for each process, for configurations with multiple processes. The set of hedges connecting two regular sets of configurations can then be described using a constraint system, using an approach adapted from WPDS.

We introduce an extended weight domain to abstract these hedges, and study the analogous branching GPP (BGPP) problem, which is the meet-over-all-hedges solution, for these branching WDPN (BWDPN). We show, that if the weight domain of a WDPN and the extended weight domain of a BWDPN, both based on the same DPN, are related, the solution for the GPP problem of the WDPN can be derived from the solution of the corresponding BGPP problem of the BWDPN. The BGPP problem can be solved by abstract interpretation of the above mentioned constraint system.

Up to this point our framework of WDPN and BWDPN can solve the bitvector problems for DPN formulated in [1], the more general KILL/GEN analyses described in [9] and the shortest path analysis from [2]. In [10] a different approach to generalize WPDS to parallel programs is presented, by introducing a context bound. This leads to an underapproximation, whereas our approach handles unbounded context switches precisely.

References

1. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular symbolic analysis of dynamic networks of pushdown systems. In: CONCUR. LNCS 3653, Springer (2005)
2. Reps, T., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Sci. Comp. Prog. **58**(1-2) (2005)
3. Bouajjani, A., Esparza, J., Schwoon, S., Strejček, J.: Reachability analysis of multithreaded software with asynchronous communication. In: FSTTCS. LNCS 3821, Springer (2005)
4. Lammich, P., Müller-Olm, M., Wenner, A.: Predecessor sets of dynamic pushdown networks with tree-regular constraints. In: CAV. LNCS 5643, Springer (2009)
5. Ramalingam, G.: Context-sensitive synchronization-sensitive analysis is undecidable. ACM Trans. Program. Lang. Syst. **22**(2) (2000)
6. Lal, A., Reps, T.W.: Improving pushdown system model checking. In: CAV. LNCS 4144, Springer (2006)
7. Lal, A., Reps, T.W., Balakrishnan, G.: Extended weighted pushdown systems. In: CAV. LNCS 3576, Springer (2005)
8. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, ACM Press (1977)
9. Lammich, P., Müller-Olm, M.: Precise fixpoint-based analysis of programs with thread-creation and procedures. In: CONCUR. LNCS 4703 (2007)
10. Lal, A., Touili, T., Kidd, N., Reps, T.W.: Interprocedural analysis of concurrent programs under a context bound. In: TACAS. LNCS 4963, Springer (2008)