PDAAAL: A Library for Reachability Analysis of Weighted Pushdown Systems

Peter G. Jensen[1], Stefan Schmid[2], Morten K. Schou[1], and Jiří Srba[1]

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark
[2] TU Berlin, Berlin, Germany and University of Vienna, Vienna, Austria

Abstract. We present PDAAAL, an open source C++ library and tool for weighted reachability analysis of pushdown systems, including generation of both shortest and longest witness traces. We consider totally ordered weight domains which have important applications, e.g. in network verification, and achieve a speedup of two orders of magnitude compared to the state-of-the-art tool WALi. Our tool further extends the state of the art by supporting the generation of the longest trace in case it exists, or reporting that no finite longest trace can be generated. PDAAAL is provided both as a stand-alone tool accepting JSON files and as a C++ library. This allows for integration in software pipelines as well as in verification tools like AalWiNes.

1 Introduction

Pushdown automata are a fundamental model in computer science and they are often used as an underlying formalism for data-flow analysis of recursive programs [1, 3, 14, 17], parsing of XML streams [11], modelling of network protocols [5, 13] and others [9, 12]. The verification questions on different types of models can be reduced to reachability analysis for pushdown systems.

In order to support quantitative extensions of such systems, we need to study weighted extensions of pushdown automata. In general, these weights are defined over an idempotent semiring and we need to consider meet-over-all-paths values for reaching certaing pushdown configurations. There is a rich literature of theory showing the application of weighted pushdown automata [9, 10, 12, 14] to various application domains and several tools for the reachability analysis of pushdown systems exist, including the tools Moped [16] used for the analysis of Java programs (in the jMoped framework [17]), WPDS++ [7] for program analysis as well as its more recent successor WALi [8] employed in tools like ICRA [9] performing interprocedureal compositional recurrence analysis and the static analysis tool Phasar [14] for C/C++ programs.

We present an open source C++ library and stand-alone tool PDAAAL for efficient reachability analysis of pushdown automata over the weight domain of totally ordered idempotent semirings. The study of such totally ordered semirings is fundamental and has important applications, e.g., in the context of the

verification of communication networks [5, 13]. PDAAAL implements the classical *pre** and *post** saturation algorithms for unweighted pushdown systems, including the new *dual** algorithms [6] and extends these algorithms for weighted reachability analysis, computing the weights of not only the shortest traces but also the longest traces, while returning such trace witnesses in case they exist. The study of longest traces is practically relevant, as it allows, for example, to perform a worst-case analysis of the routing paths in a communication network, e.g., in terms of delay or size of packet headers [5]. It is, however, also challenging to analyze, as the longest trace may be unbounded and hence impossible to compute directly. To the best of our knowledge, PDAAAL is the first tool providing the exact computation of the longest traces as the debugging information.

We introduce the formalism of weighted pushdown systems (Section 2), present the implemented algorithms and tool usage (Section 3) and compare the performance of PDAAAL with the state-of-the-art tool WALi for weighted reachability analysis (Section 4) where we observe up to two orders of magnitude faster performance. Finally, we elaborate on a specific use case in network verification (Section 5) related to MPLS networks [5].

2 Weighted Pushdown Systems and Reachability

PDAAAL can accept weights from the domain of totally ordered idempotent semirings $S = (D, \sqcap, \oplus, \top, \bot)$. An example of a weight domain for computing the shortest paths is $S_1 = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ where weights are natural numbers including infinity, the weights are additive along a single path and minimum is the meet-over-all-path operation. A domain for the computation of the longest path is $S_2 = (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0)$.

Definition 1. *A Weighted Pushdown System (WPDS) over a weight semiring* $S = (D, \sqcap, \oplus, \top, \bot)$ *is a tuple* (P, Γ, Δ) *where* P *is a finite set of* control locations*,* Γ *is a finite* stack alphabet*, and the set of* rules Δ *is a finite subset of* $(P \times \Gamma) \times D \times (P \times \Gamma^*)$*, written* $\langle p, \gamma \rangle \overset{d}{\hookrightarrow} \langle p', w \rangle$*, if* $((p, \gamma), d, (p', w)) \in \Delta$*.*

A *configuration* in a pushdown system is a pair $\langle p, w \rangle$ where $p \in P$ is the current control location and $w \in \Gamma^*$ is the stack content (head of the stack on the left). A WPDS induces a labelled transition system $T = (P \times \Gamma^*, D, \Rightarrow)$, where for all $w' \in \Gamma^*$ $\langle p, \gamma w' \rangle \overset{d}{\Rightarrow} \langle p', w w' \rangle$, provided that there is a pushdown rule $\langle p, \gamma \rangle \overset{d}{\hookrightarrow} \langle p', w \rangle$. We write $c_0 \overset{d}{\Rightarrow}^{\oplus} c_n$ if there is a path in the labelled transition system $c_0 \overset{d_1}{\Longrightarrow} \ldots \overset{d_n}{\Longrightarrow} c_n$ such that $d = d_1 \oplus \ldots \oplus d_n$. The *distance* between two configurations c and c' is given by $\delta(c, c') = \bigsqcap \{d \mid c \overset{d}{\Rightarrow}^{\oplus} c'\}$. If S is a bounded idempotent semiring (i.e. has no infinite descending chains), the distance is well defined. If S is unbounded, the supremum may not be in the domain D; for example in the semiring S_2, the distance is ∞ if there is a positive-weight loop.

The problem solved by PDAAAL is: given a WPDS (P, Γ, Δ) and two regular sets of configurations $C, C' \subseteq P \times \Gamma^*$, compute the distance $\bigsqcap\{\delta(c, c') \mid c \in C, c' \in C'\}$ and return a witness trace (if any) with this distance.

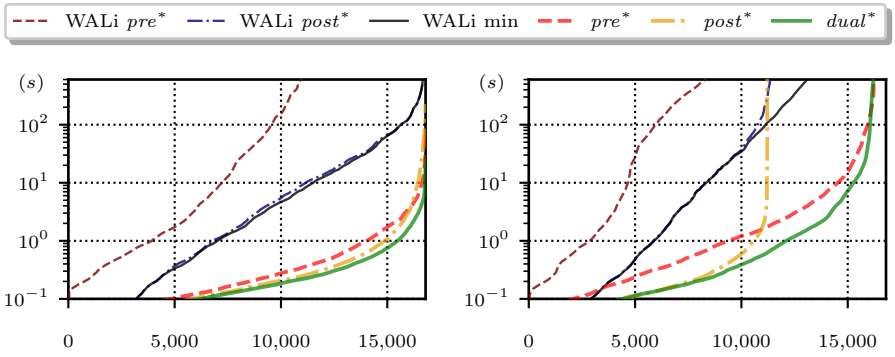
3 Implemented Algorithms and PDAAAL Architecture

It is well known that the sets of all predecessors $pre^*(C)$ and successors $post^*(C)$ of a regular set of pushdown configurations C are also regular [2]. The classical pre^* and $post^*$ saturation algorithms [1, 4, 15] solve reachability for pushdown systems without weights. Schwoon [15] describes how to find a shortest witness trace for totally ordered weight domains by using a priority queue to select the next step of the saturation. This is later generalized to bounded idempotent semirings [12], and implemented in the tool WALi [8]. Here the saturation algorithms use a workset where transitions may be added multiple times, hence possibly loosing some efficiency compared to the priority queue that exploits the total ordering. Extensions to unbounded semirings are considered in [10] by detecting that exceeding a given number of iterations of the saturation algorithm causes nontermination of the procedure. PDAAAL implements the ideas from [10] to pre^*, $post^*$ as well as $dual^*$ (combination of the first two approches) for unbounded but totally ordered weight domains.

To achieve a high performance, we employ numerous algorithmic optimizations. We extend the early termination and bidirectional-search ($dual^*$) technique from unweighted pushdowns [6] to shortest trace queries for weighted systems. The main challenge here is that the on-the-fly construction of the product automata must keep track of the weight of the best path to any state, and the saturation only terminates if the best weight of an accepting path is no higher than the current weight in the priority queue in the saturation. For longest trace queries the $dual^*$ approach simply interleaves the saturation of pre^* and $post^*$ and returns when either of them terminates. We also efficiently handle rules that apply to any top-of-stack label, using of a wildcard flag in the precondition, and adapting the pre^* and $post^*$ algorithms to efficiently handle wildcards.

PDAAAL is designed to be included as a library in other C++ projects, but it also functions as a stand-alone tool with JSON parsers for pushdown systems and $\mathcal{P}$-automata (nondeterministic automata used to represent regular sets of pushdown configurations). The tool has predefined weight domains for integers and natural numbers as well as vectors of these. In all cases, the weight semiring can either minimize or maximize the weight, depending on whether a shortest or longest trace is required. Other weight domains can be defined by the user, when PDAAAL is used as a C++ library.

As an example, a $\mathcal{P}$-automaton for the following set of pushdown configurations $\{\langle p_0, BA\rangle, \langle p_0, A\rangle, \langle p_1, A\rangle\}$ can be defined either in JSON format, or by using regular expressions for the stack, symbols '<' and '>' to denote configurations, and the symbol '|' to union multiple configuration sets: `< [p0], [B]? [A] > | < [p1], [A] >`.



(a) Shortest latency

(b) Longest latency

Fig. 1: Performance plots of WALi (pre^*,$post^*$ and minimum of both) in thin lines, compared to PDAAAL (pre^*, $post^*$ and $dual^*$) in thick lines; all instances on x-axis are independently sorted by the increasing verification time that is plotted on y-axis (log-scale) in seconds.

To run PDAAAL from the command line, an input file must be provided along with the algorithm to use: `-e 1` ($post^*$), `-e 2` (pre^*), or `-e 3` ($dual^*$) and the trace type `-t 0` (no trace), `-t 1` (any trace), `-t 2` (shortest trace), or `-t 3` (longest trace). For instance to run the $post^*$ shortest trace algorithm: `pdaaal --input example.json -e 1 -t 2`.

4 Comparison with State-of-the-Art

The first library for weighted pushdown systems, called WPDS [15], was provided by Schwoon and used in Moped version 2. Later, WPDS++ [7] was developed by Reps et al. and included further performance optimizations. The state-of-the-art tool WALi [8] was developed as a successor of WPDS++ and it is used as a backend in recent static analyzers ICRA [9] and Phasar [14] .

We compare PDAAAL to WALi by running the shortest and longest trace queries on weighted pushdown systems produced by AalWiNes [5] on a large benchmark of real communication networks from ISP providers. All together, we run 16,800 reachability queries on pushdown systems of varying sizes. WALi does not support a generation of the longest traces, unless a bound on the weight of the longest trace is known a priori. In order to enable this, we set the bound to the highest possible value of 32bit integer. On contrary, the implementation in PDAAAL is able to effectively compute a bound on the number of iterations, and hence it guarantees the termination even for unbounded longest traces.

Figure 1 shows the results comparing WALi and PDAAAL. We consider both the computation of shortest traces and longest traces where the weight domain represents the latency (which is additive along a pushdown trace). PDAAAL

supports both pre^*, $post^*$ and $dual^*$ (interleaving of pre^* and $post^*$), while WALi does not support $dual^*$. We instead present the minimum of the verification time of pre^* and $post^*$, which shows an improvement on the largest instances for the longest latency. For the shortest trace experiment, all variants of PDAAAL saturation algorithms outperform WALi by several orders of magnitude. For the longest traces, this is also the case for our $dual^*$ algorithm, even though the $post^*$ algorithm times out about at the same instance as WALi. We can also observe that our pre^* implementation is in general performing as good (or even better) than our $post^*$, while this is not the case for WALi.

PDAAAL is available on https://github.com/DEIS-Tools/PDAAAL together with specifications of input/output formats and how to run the tool. A reproducibility package is available at https://doi.org/10.5281/zenodo.6833493.

5 Applications

Pushdown automata find broad and practical applications in many domains where verification tasks are often reduced to a pushdown reachability analysis. As an example, PDAAAL can be used to model MPLS networks, a popular and widely-used type of communication network used by most Internet Service Providers for efficient traffic engineering [5]. MPLS networks interconnect a set of routers which forward packets, where packets contain stacks of labels which can be pushed and popped, and the forwarding is based on the top-of-stack label. Such networks can hence be modelled as pushdown systems.

PDAAAL can be used in combination with AalWiNes [5] as part of a what-if analysis tool (behaviour under link failures) to ensure a dependable service and policy-compliant routing. In particular, PDAAAL's support for longest traces is attractive to perform a worst-case analysis of the network's routing behavior. For example, PDAAAL can be used to compute the longest possible routes that may occur under one or multiple link failures, both in terms of the number of hops (which is directly related to the amount of bandwidth resources consumed in the network) as well as in terms of the overall delay (an important metric for latency-critical applications). Furthermore, PDAAAL can also be used to verify further quantitative metrics of interest. An online demo is available at http://demo.aalwines.cs.aau.dk.

Similar applications for the longest trace analysis also arise in other domains, allowing to perform worst-case time analyses of possible control flows in recursive programs or the execution of parsers of XML streams, shedding light on the possible overheads of such operations.

6 Conclusion

We presented PDAAAL, a tool for reachability analysis of weighted pushdown automata over possibly unbounded weight domains. Our tool can be used also as a library, and it is integrated into a recent network analysis tool AalWiNes that relies on pushdown systems produced from widely used MPLS networks.

Apart from being two orders of magnitude faster than the state-of-the-art competitor, it supports the detection of the existence of longest traces which finds practical applications in e.g., the analysis of network protocols. Our tool uses unbounded but totally ordered weight domains but despite of this limitation, it finds numerous applications and can in the case of totally ordered domains replace the backend weighted engines like Moped, WPDS++ and WALi with a generic, modern and efficient library.

References

1. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: CONCUR'97. LNCS, vol. 1243, pp. 135–150. Springer (1997)
2. Büchi, J.R.: Regular canonical systems. Archiv für mathematische Logik und Grundlagenforschung **6**(3-4), 91–111 (1964)
3. Esparza, J., Knoop, J.: An automata-theoretic approach to interprocedural dataflow analysis. In: FOSSACS'99. LNCS, vol. 1578, pp. 14–30. Springer (1999)
4. Finkel, A., Willems, B., Wolper, P.: A direct symbolic approach to model checking pushdown systems. In: INFINITY'97. ENTCS, vol. 9, pp. 27–37. Elsevier (1997)
5. Jensen, P.G., Kristiansen, D., Schmid, S., Schou, M.K., Schrenk, B.C., Srba, J.: AalWiNes: A fast and quantitative what-if analysis tool for MPLS networks. In: CoNEXT'20. p. 474–481. ACM (2020)
6. Jensen, P., Schmid, S., Schou, M., Srba, J., Vanerio, J., van Duijn, I.: Faster pushdown reachability analysis with applications in network verification. In: ATVA'21. LNCS, vol. 12971, pp. 170–186. Springer-Verlag (2021)
7. Kidd, N., Reps, T., Melski, D., Lal, A.: WPDS++: A C++ library for weighted pushdown systems. Univ. of Wisconsin (2004)
8. Kidd, N., Lal, A., Reps, T.: Wali: The weighted automaton library (2007), https://research.cs.wisc.edu/wpis/wpds/wali/
9. Kincaid, Z., Breck, J., Boroujeni, A.F., Reps, T.: Compositional recurrence analysis revisited. In: Conference on Programming Language Design and Implementation. pp. 248–262. PLDI (2017)
10. Kühnrich, M., Schwoon, S., Srba, J., Kiefer, S.: Interprocedural dataflow analysis over weight domains with infinite descending chains. In: FOSSACS'09. LNCS, vol. 5504, pp. 440–455. Springer-Verlag (2009)
11. Kumar, V., Madhusudan, P., Viswanathan, M.: Visibly pushdown automata for streaming XML. In: WWW'07. pp. 1053–1062. ACM (2007)
12. Reps, T., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Science of Computer Programming **58**(1-2), 206–263 (2005)
13. Schmid, S., Srba, J.: Polynomial-time what-if analysis for prefix-manipulating MPLS networks. In: IEEE INFOCOM'18. pp. 1799–1807. IEEE (2018)
14. Schubert, P.D., Hermann, B., Bodden, E.: PhASAR: An inter-procedural static analysis framework for C/C++. In: TACAS'19. pp. 393–410. Springer (2019)
15. Schwoon, S.: Model-checking pushdown systems. Ph.D. thesis, Technische Universität München (2002)
16. Schwoon, S.: Moped. In: http://www2.informatik.uni-stuttgart.de/fmi/szs/tools/moped/ (2002)
17. Suwimonteerabuth, D., Schwoon, S., Esparza, J.: jMoped: A java bytecode checker based on Moped. In: TACAS'05. LNCS, vol. 3440, pp. 541–545. Springer (2005)

Appendix

Recall that a monoid is an algebraic structure $(M, \bullet)$, where M is a set and $\bullet : M \times M \to M$ is a binary operator on M, such that the following two properties hold:

- Associativity: $\forall a, b, c \in M,\ (a \bullet b) \bullet c = a \bullet (b \bullet c)$,
- Identity: $\exists e \in M,\ \forall a \in M,\ e \bullet a = a = a \bullet e$. Here e is called the identity element.

In a commutative monoid, the following additional property holds:

- Commutativity: $\forall a, b \in M,\ a \bullet b = b \bullet a$

A totally ordered idempotent semiring is a tuple $\mathcal{S} = (D, \sqcap, \oplus, \top, \bot)$ where D is a set, $\top$ and $\bot$ are elements in D, and $\sqcap$ and $\oplus$ are binary operators on D, such that:

- $(D, \sqcap)$ is a commutative monoid with the identity element $\top$.
- $(D, \oplus)$ is a monoid with the identity element $\bot$.
- $\oplus$ distributes over $\sqcap$: $\forall a, b, c \in D$ we have $a \oplus (b \sqcap c) = (a \oplus b) \sqcap (a \oplus c)$ and $(a \sqcap b) \oplus c = (a \oplus c) \sqcap (b \oplus c)$.
- $\top$ is an annihilator for $\oplus$: $\forall a \in D, a \oplus \top = \top = \top \oplus a$.
- $\bot$ is an annihilator for $\sqcap$: $\forall a \in D, \bot \sqcap a = \bot$.
- The order $\sqsubseteq$ defined by: $\forall a, b \in D, a \sqsubseteq b$ iff $a \sqcap b = a$ is a total order.

Further, the totally ordered idempotent semiring is bounded, if the order $(D, \sqsubseteq)$ has no infinite descending chains.