Acceleration in Multi-PushDown Systems

Mohamed Faouzi Atig[1(✉)], K. Narayan Kumar[2], and Prakash Saivasan[2]

[1] Uppsala University, Uppsala, Sweden
mohamed_faouzi.atig@it.uu.se
[2] Chennai Mathematical Institute, Chennai, India
{kumar,saivasan}@cmi.ac.in

Abstract. Multipushdown systems (MPDS) are formal models of multi-threaded recursive programs. They are turing powerful and hence one considers under-approximation techniques in their analysis. We study the use of loop accelerations in conjunction with bounded context analysis.

1 Introduction

Sequential recursive programs are usually modeled as pushdown systems (PDSs) and algorithmic techniques developed for PDSs have been used to solve a number of problems related to the verification of such programs (e.g. [14,20,23,26,40,41]). Extending this idea to multi-threaded recursive programs requires multi-pushdown systems (MPDSs), i.e. automata with multiple pushdown stores. Unfortunately, MPDSs are turing powerful. The main technique used to circumvent this problem is that of under-approximation. The idea is to identify a subset of behaviours and restrict the verification only to this subset. An underapproximation is interesting only if the verification problem when restricted to this subset is decidable and in addition the subset covers interesting behaviours. This idea came to the fore with the *bounded context analysis* proposed in [39]. A context switch occurs when the automaton switches from accessing one stack to another (or equivalently, the execution of a multi-threaded program switches from scheduling one thread to another). Placing an a priori bound on the number of context switches results in the decidability of reachability and other verification problems. Subsequently, other classes generalizing the bounded context assumption have been proposed (see [2–6,15,28,32–35]).

Recall that the configuration of a PDS can be seen as a word (giving the current state and the contents of the stack). In the *global model checking problem* the aim is to compute from (a representation of) the set of initial configurations (I) (a representation of) the set of configurations reachable from I (denoted $post^*(I)$). For PDSs if the initial set of configurations is a regular language then the set of reachable configurations is a computable regular language ([14,26]).

The configuration of a MPDS can be represented as a tuple of words giving the current state and the contents of each of the stacks. We can then represent sets of configurations by *recognizable* or *regular* languages [10]. Given a

The authors acknowledge partial support by DST-VR Project P-02/2014, TCS PhD Fellowship, Infosys Foundation and the Linnaeus centre of excellence UPMARC.

M. Chechik and J.-F. Raskin (Eds.): TACAS 2016, LNCS 9636, pp. 698–714, 2016.
DOI: 10.1007/978-3-662-49674-9_45

recognizable language representing the set of initial configurations, the set of configurations that may be reached via runs with at most k-context switches is also a (computable) recognizable language [39]. Thus, the global model checking problem is decidable and this has many applications, including the obvious one — reachability can be decided.

Note that our description of global model-checking does not require that the representations of the initial set I and the reachable set $post^*(I)$ to be the same. For instance, for PDSs, whether we use finite sets or regular sets for the initial set of configurations, the final set can be described effectively as a regular set. However, if both sets use the same description, then we say that the representation is stable. Stability is an useful property as it permits us to compose (and hence iterate finitely) the algorithm.

Another well known technique used in the verification of infinite state systems is that of loop accelerations. It is similar in spirit to global model checking but with different applications. The idea is to consider a loop of transitions (a finite sequence of transitions that lead from a control state back to the same control state). The aim is to determine the effect of iterating the loop. That is, to effectively construct a representation of the set of configurations that may be reached by valid iterations of the loop.

Loop accelerations turn out to be very useful in the analysis of a variety of infinite state systems (e.g., [1,7–9,11–13,16,24,25,29,30,36,37]). In this paper, we propose to use accelerations in the verification of MPDSs. We take this further by proposing a technique that composes the iterations of such loops with context bounded runs to obtain a new decidable under-approximation for MPDSs. Observe that there is no bound on the number of context switches under loop iterations while a context bounded run permits unrestricted recursive behaviours, not permitted by loop iterations, thus complementing each other.

We begin by showing that both regular and rational sets of configurations are stable w.r.t. bounded context runs. Then, we show that this does not extend to iterations of loops. We show that under iterations of a loop, the $post^*$ of a regular set of transitions is always rational while that of a rational set need not be rational. We then address the question of a representation that is stable w.r.t. loop accelerations. Towards this we propose a new representation for configurations called n-CSRE inspired by the CQDDs [16] and the class of bounded semilinear languages [18]. We show that n-CSREs are indeed stable w.r.t iteration of loops. This result also has the pleasant feature that the construction is in polynomial time. However, n-CSREs are not stable w.r.t bounded context runs.

As a final step we introduce a joint generalization of both loop iterations and bounded context executions called bounded context-switch sets. We show that the class of languages defined by n-dimensional constrained automata (a n-dimensional version of Parikh automata [17,31]) is stable w.r.t accelerations via bounded context-switch sets. Since membership is decidable for this class, we obtain a decidability of reachability under this generous class of behaviours. Observe that the class of n-dimensional constrained automata is not closed under intersection and that the inclusion problem is undecidable.

Related Work. To the best of our knowledge this is the first study of accelerations in the setting of MPDSs. By using ideas from acceleration we have obtained a decidable under-approximation that significantly extends the notion of context-bounding, and which seems incomparable to many other classes considered in literature. The closest work is the pattern-based (or bounded) verification for MPDSs [21,22,27]. The pattern-based verification checks the correctness of the program for the set of the executions described as a bounded language (i.e., $w_1^* w_2^* \cdots w_n^*$). Our loop acceleration result allow to compute the set of reachable configurations induced by a bounded language and hence solving the global reachability problem for pattern-based verification for MPDSs (and providing a new proof for its decidability).

2 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers. For $i, j \in \mathbb{N}$ with $i \leq j$, we use $[i..j]$ to denote the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. Let A and B two sets. For a partial function $g : A \rightharpoonup B$ and $a \in A$, we write $g(a) = \bot$ if g is undefined on a. Let Σ be a finite alphabet. As usual Σ^* denotes the set of all finite words over Σ and ϵ denotes the empty word. Let $u \in \Sigma^*$, we use $Parikh(u)$ to denote the mapping that associates to each letter a in Σ, the number of occurrences of a in u.

Next we extend these notions to higher dimensions. Let $\Sigma_1, \ldots, \Sigma_n$ be n finite alphabets. A n-dim word $\mathbf{u}$ over $\Sigma_1, \ldots, \Sigma_n$ is a tuple $(u_1, u_2, \ldots, u_n)$ with $u_i \in \Sigma_i^*$. For every $j \in [1..n]$, we use $\mathbf{u}[j]$ to denote the word u_j. Let $i \in [1..n]$ and $w \in \Sigma_i^*$, we use $\mathbf{u}[i \leftarrow w]$ to denote the n-dim word $(u_1, u_2, \ldots, u_{i-1}, w, u_{i+1}, \ldots, u_n)$. A n-dim language is a set of n-dim words. Given two n-dim words $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, their concatenation is defined by $\mathbf{uv} = (u_1v_1, \ldots, u_nv_n)$. The concatenation of two n-dim languages L_1, L_2 is defined as expected to be $L_1.L_2 = \{\mathbf{uv} \mid \mathbf{u} \in L_1 \wedge \mathbf{v} \in L_2\}$.

A n-tape finite state automaton over $\Sigma_1, \ldots, \Sigma_n$ is defined as $A = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, q_0, F)$ where Q is a finite set of states, q_0 is the initial state, F is the set of final states, and $\delta \subseteq (Q \times (\Sigma_1 \cup \{\epsilon\}) \times \cdots \times (\Sigma_n \cup \{\epsilon\}) \times Q)$, is the transition relation. A run π of A over a n-dim word $\mathbf{w}$ over $\Sigma_1, \ldots, \Sigma_n$ is a sequence of transitions $(q_0, \mathbf{u}_1, q_1), (q_1, \mathbf{u}_2, q_2), \ldots, (q_{m-1}, \mathbf{u}_m, q_m) \in \delta$ such that $\mathbf{w} = \mathbf{u}_1\mathbf{u}_2 \cdots \mathbf{u}_m$. The run π is accepting if $q_m \in F$. The language of A, denoted by $L(A)$, is the set of n-dim words $\mathbf{w}$ for which there is an accepting run of A over w. A n-dim language is *rational* if it is the language of some n-tape automaton [10]. Observe that 1-tape automata are the standard finite-state automata.

An interesting subclass of rational languages are what are called *recognizable or regular* languages. A n-dim language L is *regular* if it is a finite union of products of n rational 1-dim languages (i.e. $L = \bigcup_{j=1}^m L_{(j,1)} \times \cdots \times L_{(j,n)}$ for some $m \in \mathbb{N}$ where $L_{(j,i)}$ is an 1-dim rational language over Σ_i). Observe that if $n = 1$ rational and regular languages are the same. The language $\{(a^i, b^i) \mid i \geq 0\}$ is an example of a rational language that is not regular.

Let us recall some properties of rational and regular languages (see, e.g., [10]). First, the class of regular languages, for any dimension $n \geq 1$, is closed under boolean operations. On the other hand, for every $n \geq 2$, the class of n-dim rational languages is closed under union and concatenation but not under complementation, nor under intersection. However, the emptiness and membership problems for rational languages are decidable in all dimensions and further the inclusion problem is also decidable for regular languages. The inclusion problem is undecidable for rational languages (for $n \geq 2$).

We describe some additional closure properties of rational languages that will prove useful. Rational languages are effectively closed under the permutation of indices: Let A be a n-tape automaton over $\Sigma_1, \ldots, \Sigma_n$. Given a mapping $h : [1..n] \to [1..n]$, it is possible to construct a n-tape automaton $h(A)$, linear in the size of A, such that $(w_1, ..., w_n) \in L(A)$ iff $(w_{h(1)}, \ldots, w_{h(n)}) \in L(h(A))$. Rational languages are also effectively closed under projection: Given a set of indices $\iota = \{i_1 < i_2 < \ldots i_m\} \subset \{1, \ldots, n\}$, we can construct an automaton $\Pi_\iota(A)$, linear in size of A, such that $L(\Pi_\iota(A)) = \{(w_{i_1}, w_{i_2}, \ldots, w_{i_m}) \mid (w_1, w_2, \ldots, w_n) \in L(A)\}$. Rational languages are also closed under an operation we call *composition*: Let A be as before and let A' be a rational language over $\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_m$. Let $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ be two indices s.t. $\Sigma'_j = \Sigma_i$. Then, it is possible to construct a $(n+m-1)$-tape automaton $A \circ_{(i,j)} A'$, whose size is $\mathcal{O}(|A|.|A'|)$, accepting $(w_1, \ldots, w_n, w'_1, \ldots, w'_{j-1}, w'_{j+1}, \ldots, w'_m)$ iff $(w_1, \ldots, w_n) \in L(A)$ and $(w'_1, \ldots, w'_{j-1}, w_i, w'_{j+1}, \ldots, w'_m) \in L(A')$, i.e. the composition corresponding to the synchronization of the i^{th} tape of A with the j^{th} tape of A'.

3 Multi-PushDown Systems

A *Multi-PushDown System* (MPDS) is a tuple $M = (n, Q, \Gamma, \Delta)$ where: (1) $n \geq 1$ is the number of stacks, (2) Q is the non-empty finite set of states, (3) Γ is the finite set of stack symbols, and (4) $\Delta \subseteq (Q \times (\cup_{i \in [1..n]} \Omega(i)) \times Q)$ is the transition relation. For every $i \in [1..n]$, $\Omega(i)$ is the set of operations on the stack i containing: (i) the *push operation* $push_i(a)$ ($a \in \Gamma$), (ii) the *pop operation* $pop_i(a)$ ($a \in \Gamma$), and (iii) the *internal operation* nop_i. A PushDown System (PDS) can be seen as a MPDS with $n = 1$. Let $\Delta_i = \Delta \cap (Q \times \Omega_i \times Q)$.

A configuration of the MPDS M is a $(n+1)$-tuple $(q, u_1, u_2, \cdots, u_n)$ where $q \in Q$ is the current state of M, and for every $i \in [1..n]$, $u_i \in \Gamma^*$ is the current content of the i-th stack of M. A configuration can be seen as $(n+1)$-word. The set of configurations of the MPDS M is denoted by $Cf(M)$. Given two configurations $(q, u_1, \cdots, u_n)$ and $(q', v_1, \cdots, v_n)$ of M and a transition $t \in \Delta$, we define the transition relation $\xrightarrow{t}_M$ as follows: $(q, u_1, \cdots, u_n) \xrightarrow{t}_M (q', v_1, \cdots, v_n)$ iff one of the following holds: (1) $t = (q, push_i(a), q')$, $v_i = a.u_i$ and $u_j = v_j$ for all $j \in ([1..n] \setminus \{i\})$, (2) $t = (q, pop_i(a), q')$, $u_i = a.v_i$ and $u_j = v_j$ for all $j \in ([1..n] \setminus \{i\})$, or (3) $t = (q, nop_i, q')$ and $u_j = v_j$, for all $j \in [1..n]$.

For a sequence of transitions $\sigma = t_1 t_2 \ldots t_m \in \Delta^*$ and two configurations $c, c' \in Cf(M)$, we write $c \xrightarrow{\sigma}_M c'$ to denote that one of the following two cases holds: (1) $\sigma = \epsilon$ and $c = c'$, or (2) there are configurations $c_0, \cdots, c_m \in Cf(M)$ such that $c_0 = c$, $c' = c_m$, and $c_i \xrightarrow{t_{i+1}}_M c_{i+1}$ for all $i \in [0..m-1]$. Given a set of configurations $C \subseteq Cf(M)$ and a set of sequences of transitions $\Theta \subseteq \Delta^*$, the *acceleration problem* for M, with respect to C and Θ, consists in computing the set $Post_{\Theta^*}(C) = \{c' \mid c \xrightarrow{\sigma}_M c' , c \in C , \sigma \in \Theta^*\}$.

4 Context-Bounding as an Acceleration Problem

In the following, we show that context-bounded analysis [33,34,38,39] for an MPDS $M = (n, Q, \Gamma, \Delta)$ can be formulated as an acceleration problem wrt. the class of rational/regular configurations. Given two configurations $c, c' \in Cf(M)$ and $k \in \mathbb{N}$, the k-context reachability problem consists in checking whether there is a sequence of transitions $\sigma \in \Delta^*_{i_1}\Delta^*_{i_2}\cdots\Delta^*_{i_k}$, with $i_1, i_2, \ldots, i_k \in [1..n]$, such that $c \xrightarrow{\sigma}_M c'$. The decidability of the k-context reachability problem can be seen as an immediate corollary of the decidability of the membership problem for rational languages and the following result:

Theorem 1. *Let $i \in [1..n]$. For every regular (rational) set of configurations C, the set $Post_{\Delta^*_i}(C)$ is regular (rational) and effectively constructible.*

The set $Post_{\Delta^*_i}(C)$ has been shown to be regular and effectively constructible when C is regular in [39]. In the following, we prove Theorem 1 for the case when C is rational. We write M_i for the PDS $(1, Q, \Gamma, \Delta_i)$ simulating the behavior of M only on the stack i. First we recall a result established in [19,35].

Lemma 1. *It is possible to construct, in polynomial time in the size of M_i, a 4-tape finite state automaton T, over Q, Γ, Q, Γ, such that $(q, u, q', v) \in L(T)$ iff $(q, u) \xrightarrow{\pi}_{M_i} (q', v)$ for some sequence $\pi \in \Delta^*_i$.*

Observe that Lemma 1 relates any possible starting configuration (q, u) with any configuration (q', v) reachable from (q, u) in M_i. Let us assume now that we are given a $(n+1)$-tape automaton $A = (P, Q, \Gamma, \ldots, \Gamma, \delta, p_0, F)$ accepting the set C. In the following, we show how to compute a $(n+1)$-tape finite state automaton A' accepting the set $Post_{\Delta^*_i}(C)$. To do that, we proceed as follows: We first compose A with T, synchronizing the second tape of T (containing the stack contents at the starting configuration) with the $(i+1)$-th tape of A, to construct a $(n+4)$-tape automaton $A_1 = A \circ_{(i+1,2)} T$. We also need to synchronize the starting states (i.e. the first tape of A with the first tape of T). This can be done by intersecting A_1 with the (regular) language $\bigcup_{q \in Q}\{q\} \times (\Gamma^*)^n \times \{q\} \times Q \times \Gamma^*$. Let A_2 be the automaton resulting from the intersection operation. Then, we project away the starting control state (occurring on tapes 1 and $n+2$) and the content of the $i+1$-th tape to obtain the $(n+1)$-tape automaton $A_3 = \Pi_\iota(A_2)$ where $\iota = ([1..n] \setminus \{1, i+1, n+2\})$. This is almost what is needed except that the new content of the stack i occurs at the last position instead of position $i+1$ and the control state occurs at penultimate position instead of the first position. We rearrange this using the permutation operation. We let $A' = h(A_3)$ where h is defined as follows: (1) $h(1) = n$, (2) $h(j) = j - 1$ for all $j \leq i$, (3) $h(i+1) = n+1$, and (4) $h(j) = j - 2$ for all $j > i+1$.

Observe that the size of A' is polynomial in $|A|\cdot$ — this follows from Lemma 1 and the bounds on the closure operation on rational languages mentioned in Sect. 2.

5 Accelerating Loops: Case of Regular/Rational Sets

In this section, we address the acceleration problem for the iterative execution of a sequence of transitions in the control graph of a MPDS $M = (n, Q, \Gamma, \Delta)$. More precisely, given a sequence of transitions $\theta \in \Delta^*$ and a set of configurations $C \subseteq Cf(M)$, we are interested in characterizing the set $Post_{\theta^*}(C)$.

5.1 Computing the Effect of a Sequence of Transitions

Let $M = (n, Q, \Gamma, \Delta)$ be an MPDS and $\sigma \in \Delta^*$ a sequence of transitions of the form $(q_0, op_0, q_1)(q_1, op_1, q_2) \cdots (q_{m-1}, op_{m-1}, q_m)$. Intuitively, we associate to each stack i a pair (u_i, v_i) such that the effect of executing the sequence σ on stack i is popping the word u_i and then pushing the word v_i on to it (i.e. the stack content is transformed from $u_i w$ to $v_i w$ for some w). To this end, for every $i \in [1..n]$, we introduce a partial function $\texttt{Eff}_i : ((\Gamma^* \times \Gamma^*) \times \Delta^*) \rightharpoonup (\Gamma^* \times \Gamma^*)$. We first define $\texttt{Eff}_i$ when the third argument is a transition. Roughly speaking, assuming that we have already computed the effect of a transition sequence σ on stack i to be (u, v), i.e. to pop u and push v, $\texttt{Eff}_i((u, v), t)$ computes the effect of $\sigma.t$ on stack i. Given $u, v \in \Gamma^*$ and $t \in \Delta$, we define $\texttt{Eff}_i((u, v), t)$ as follows:

- if $Op(t) = pop_i(a)$ for some $a \in \Gamma$ then
  - $\texttt{Eff}_i((u, \epsilon), t) = (u \cdot a, \epsilon)$,
  - If $v = a \cdot v'$ for some $v' \in \Gamma^*$ then $\texttt{Eff}_i((u, v), t) = (u, v')$,
  - Otherwise $\texttt{Eff}_i((u, v), t) = \perp$.
- if $Op(t) = push_i(a)$ for some $a \in \Gamma$, then $\texttt{Eff}_i((u, v), t) = (u, a \cdot v)$
- If $Op(t) = nop_i$ or $t \in \Delta \setminus \Delta_i$, then $\texttt{Eff}_i((u, v), t) = (u, v)$.

We extend the definition of $\texttt{Eff}_i$ to sequence of transitions as expected: For every two words $u, v \in \Gamma^*$, we have (1) $\texttt{Eff}_i((u, v), \epsilon) = (u, v)$, and (2) for every $\sigma' \in \Delta^*$ and $t \in \delta$, we have $\texttt{Eff}_i((u, v), \sigma' \cdot t) = \texttt{Eff}_i(\texttt{Eff}_i((u, v), \sigma'), t)$ if $\texttt{Eff}_i((u, v), \sigma') \neq \perp$ is defined, and $\texttt{Eff}_i((u, v), \sigma' \cdot t) = \perp$ otherwise.

Our aim is to compute the complete effect of some sequence σ on stack i and this is given by $\texttt{Eff}_i((\epsilon, \epsilon), \sigma)$. We shall refer to this as $\texttt{Summ}(i, \sigma)$. The next lemma formalizes our intuition about Summ and characterizes precisely when a sequence of transitions σ may be executed and computes its effect on all the stacks (if it is executable).

Lemma 2. *Let $c = (q_0, w_1, \ldots, w_n)$ and $c' = (q_m, w'_1, \ldots, w'_n)$ be two configurations of M. $c \xrightarrow{\sigma} c'$ iff for every $i \in [1..n]$, we have $w_i = u_i u'_i$ and $w'_i = v_i u'_i$ for some $u_i, v_i, u'_i \in \Gamma^*$ such that* $\texttt{Summ}(i, \sigma) = (u_i, v_i)$.

Now, we will characterize $\texttt{Summ}(i, \sigma^j)$ with $j \geq 1$, i.e., the effect of iterating the sequence σ j-times, in terms of $\texttt{Summ}(i, \sigma)$ for all $i \in [1..n]$. Observe that if $\texttt{Summ}(i, \sigma) = \perp$, then $\texttt{Summ}(i, \sigma^j) = \perp$ for all $j \geq 1$. Hence, let us assume that $\texttt{Summ}(i, \sigma) = (u_i, v_i)$ for some words $u_i, v_i \in \Gamma^*$. First, let us consider the case when the sequence σ can be iterated twice and compute its effect on all the stacks. Now, using the definition of Summ it is not difficult to conclude that $\texttt{Summ}(i, \sigma\sigma)$

is defined iff either v_i is a prefix of u_i or u_i is a prefix of v_i. We can in fact say more. If the former holds we let x_i be the unique word such that $u_i = v_i x_i$ and $y_i = \epsilon$. In case of the latter we let y_i be the unique word such that $v_i = u_i y_i$ and $x_i = \epsilon$. Then, we have $\texttt{Summ}(i, \sigma\sigma) = (u_i', v_i')$ for all $i \in [1..n]$ where $u_i' = u_i x_i$ and $v_i' = v_i y_i$. We define a partial function $\texttt{Iter} : ([1..n] \times \Delta^*) \rightharpoonup (\Gamma^* \times \Gamma^*)$ such that $\texttt{Iter}(i, \sigma)$ is the pair (x_i, y_i) as defined above when $\texttt{Summ}(i, \sigma\sigma)$ is defined, and $\texttt{Iter}(i, \sigma) = \bot$ otherwise. We can now generalize this computation of $\texttt{Summ}$ to any number of iterations of σ as shown below.

Lemma 3. *Let $i \in [1..n]$. If $\texttt{Summ}(i, \sigma\sigma)$ is well-defined then $\texttt{Summ}(i, \sigma^j)$ is well-defined for all $j \geq 1$. Furthermore, $\texttt{Summ}(i, \sigma^j) = (u_i x_i^{j-1}, v_i y_i^{j-1})$ with $\texttt{Summ}(i, \sigma) = (u_i, v_i)$ and $\texttt{Iter}(i, \sigma) = (x_i, y_i)$.*

5.2 Acceleration of Regular/Rational Sets of Configurations by Loops

In the following, we first state that the class of regular (resp. rational) sets of configurations is not closed under $Post_{\theta^*}$. Then, we show that the image by $Post_{\theta^*}$ of any regular set of configurations is a rational one.

Theorem 2. *There is an MPDS $M = (n, Q, \Gamma, \Delta)$, a regular (resp. rational) set of its configurations C and a transition sequence $\theta \in \Delta^*$ such that the set of configurations $Post_{\theta^*}(C)$ is not regular (resp. rational).*

However, whenever C is a regular set of configurations the set $Post_{\theta^*}(C)$ has a simple description. In what follows we fix a MPDS $M = (n, Q, \Gamma, \Delta)$.

Theorem 3. *For every regular set of configurations C and transition sequence $\theta \in \Delta^*$, the set $Post_{\theta^*}(C)$ is rational and effectively constructible.*

Let θ be a sequence of transitions of the form $(q_0, op_0, q_0')(q_1, op_1, q_1') \cdots (q_m, op_m, q_m')$. Since $Post_{\theta^*}(C_1 \cup C_2) = Post_{\theta^*}(C_1) \cup Post_{\theta^*}(C_2)$, we can assume w.l.o.g that C is of the form $\{q\} \times L_1 \times \cdots \times L_n$ where each L_j is an 1-dim rational language over Γ accepted by a finite state automaton A_j for all $j \in [1..n]$. The proof proceeds by cases.

Case 1: Let us assume $q_i' \neq q_{i+1}$ for some $i \in [0..m-1]$ or $q_0 \neq q$. In this case the sequence of transitions cannot be executed and hence $Post_{\theta^*}(C) = C$.

Case 2: Let us assume $q_0 \neq q_m'$, $q_0 = q$ and $q_i' = q_{i+1}$ for all $i \in [0..m-1]$. In this case, the sequence of transitions can not be iterated more than once and so we have $Post_{\theta^*}(C) = Post_\theta(C) \cup C$. We now examine the set $Post_\theta(C)$. First, let us assume that $\texttt{Summ}(i, \theta) = \bot$ for some $i \in [1..n]$. Then $Post_\theta(C) = \emptyset$ and hence $Post_{\theta^*}(C) = C$.

Let us assume now that $\texttt{Summ}(i, \theta) = (u_i, v_i)$ is well-defined for all $i \in [1..n]$. We can apply Lemma 2, to show that $Post_\theta(C) = \{q_m'\} \times L_1' \times \cdots \times L_n'$ where for every $i \in [1..n]$, $L_i' = \{w_i' \mid \exists w_i \in \Gamma^*.\ w_i' = v_i.w_i \wedge u_i w_i \in L_i\}$. It is easy to see that L_i' is an 1-dim rational language and can be accepted by an automaton A_i' whose size is polynomial in the size of A_i and the length of θ.

Case 3: Let us assume $q_0 = q'_m$, $q_0 = q$ and $q'_i = q_i$ for all $i \in [0..m-1]$. In this case, the sequence of transitions forms a loop in the control flow graph of M and hence the sequence may possibly be iterated. Observe that if the function $\texttt{Summ}(i, \theta) = \bot$ for some $i \in [1..n]$, then $Post_{\theta^*}(C) = C$. Hence, let us assume that $\texttt{Summ}(i, \theta) = (u_i, v_i)$ for all $i \in [1..n]$ so that it is well-defined for each i. Lemma 3 suggests that we should examine when $\texttt{Summ}(i, \theta\theta)$ is defined for all i. Indeed, if $\texttt{Summ}(i, \theta\theta)$ is undefined for some $i \in [1..n]$, then $Post_{\theta^*}(C) = Post_\theta(C) \cup C$ (which can be computed as shown in the previous case). So, let us further assume that $\texttt{Summ}(i, \theta\theta)$ is well-defined for all $i \in [1..n]$. Hence, the function $\texttt{Iter}(i, \sigma)$ is also well-defined. Let us assume that $\texttt{Iter}(i, \sigma) = (x_i, y_i)$

Now, we can combine Lemma 3 with Lemma 2 to give a characterization of when a sequence θ is iterable and its effect.

Lemma 4. *Let $j \geq 1$ and $c = (q, w_1, \ldots, w_n)$ and $c' = (q, w'_1, \ldots, w'_n)$ be two configurations of M. $c \xrightarrow{\theta^j} c'$ iff for every $i \in [1..n]$, we have $w_i = u_i x_i^{j-1} w''_i$ and $w'_i = v_i y_i^{j-1} w''_i$ for some $w''_i \in \Gamma^*$ with u_i, v_i, x_i and y_is defined as above.*

With this lemma in place, let L be the $(2n+1)$-dim language defined as the set containing the exactly the words of the form

$$(q, u_1 x_1^{j-1} w_1, v_1 y_1^{j-1} w_1, u_2 x_2^{j-1} w_2, v_2 y_2^{j-1} w_2, \ldots, u_n x_n^{j-1} w_n, v_n y_n^{j-1} w_n)$$

with $j \geq 1$ and $w_i \in \Gamma^*$ and where u_i, v_i, x_i and y_is are defined as above. Observe that each element of L relates a pair of configurations such that from the first we can execute the sequence θ a finite number of times to reach the second. The starting configuration is given by the first and all the even numbered positions, while the ending configuration is given by all the odd numbered positions (including the first). As a matter of fact elements of L relates exactly all such pairs in this manner. This language L is rational and we can easily compute an $(2n+1)$-tape automaton A whose size is polynomial in the size of θ and polynomial in the size of M. To compute an $(n+1)$-tape automaton A' accepting $Post_{\theta^+}(C)$, we proceed as follows: First, we define the regular language $L' = \{q\} \times L_1 \times \Gamma^* \times \cdots \times L_n \times \Gamma^*$. Then, we compute an $(2n+1)$-tape automaton A'' accepting precisely the language resulting from the intersection of the regular language L' and L. This allows us to restrict the starting configurations to be precisely those from C. The size A'' is exponential in the number of stacks and polynomial in the size of θ, and the finite state automata $A_1, \ldots, A_n$. Finally, we need to project away the tapes concerning the starting stack configurations. We let then $A' = \Pi_\iota(A'')$ with $\iota = \{2i+1 \mid i \in [0..n]\}$. We note that this step does not result in any blow up and thus the size of A' is exponential in the number of stacks and polynomial in the size of θ and $A_1, \ldots, A_n$.

Since $Post_{\theta^*}(C) = C \cup Post_{\theta^+}(C)$ and the class of rational / regular languages is closed under union, this completes the proof of Theorem 3.

6 Constrained Simple Regular Expressions

We now introduce the class of (1 dimensional) Constrained Simple Regular Expressions (CSRE). CSRE definable languages form an expressive class equivalent to the bounded semi-linear languages defined in [18] and the class of languages accepted by 1-CQDD introduced in [16]. To deal with configuration sets of MPDS we need n-dimensional CSREs and so we lift these results to that setting. We then show that the CSRE definable sets of configurations form a stable collection under acceleration by loops. However, this class is not stable w.r.t. bounded context runs. We begin by recalling some basics about *Presburger arithmetic*.

6.1 Presburger Arithmetic

Presburger arithmetic is the first-order theory of natural numbers with addition, subtraction and order. We recall briefly its definition. Let $\mathcal{V}$ be a set of variables. We use $x, y, \ldots$ to denote variables in $\mathcal{V}$. The set of terms in Presburger arithmetic is defined as follows: $t ::= 0 \,|\, 1 \,|\, x \,|\, t - t \,|\, t + t$. The set of formulae of the Presburger arithmetic is defined to be $\varphi ::= t \leq t \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \exists x.\, \varphi$.

We use the standard abbreviations: $\varphi_1 \wedge \varphi_2 = \neg(\varphi_1 \vee \varphi_2)$, $\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 \wedge \varphi_2$, and $\forall x.\, \varphi = \neg\exists x.\, \neg\varphi$. The notions of free and bound variables, and quantifier-free formula are as usual. An *existential* Presburger formula is one of the form $\exists x_1 \exists x_2 \ldots \exists x_n . \varphi$ where φ is a quantifier-free formula. We shall often write positive boolean combinations of existential Presburger formulas in place of an existential Presburger formula. Clearly, by an appropriate renaming of the quantified variables, any such formula can be converted into an equivalent existential Presburger formula. We write $var(\varphi) \subseteq \mathcal{V}$ to denote the set of free variables of φ. Given a function μ from $var(\varphi)$ to $\mathbb{N}$, the meaning of μ *satisfies* φ is as usual and we write $\mu \models \varphi$ to denote this. We write $\varphi(x_1, x_2, \ldots, x_k)$ to denote a Presburger formula φ whose free variables are (contained in) $x_1, \ldots, x_k$. Such a formula naturally *defines* a subset of $\mathbb{N}^k$ given by $\{(i_1, i_2, \ldots, i_k) \mid \mu \models \varphi(x_1, x_2, \ldots, x_k) \text{ where } \mu(x_j) = i_j, 1 \leq j \leq k\}$. We say that a subset S of $\mathbb{N}^k$ is definable in Presburger arithmetic if there is a formula φ that defines it.

6.2 Constrained Simple Regular Expression (CSRE)

A Constrained Simple Regular Expression (CSRE) e over an alphabet Σ is defined as a tuple of the form $e = (w_1, \ldots, w_m, \varphi(x_1, x_2, \ldots, x_m))$ where $w_1, \ldots, w_m$ is a non-empty sequence of words over Σ, and φ is an existential Presburger formula. The language defined by the CSRE e, denoted by $L(e)$, is the set of words of the form $w_1^{i_1} w_2^{i_2} \cdots w_m^{i_m}$ such that φ holds for the function μ defined by $\mu(x_j) = i_j$ for all $j \in [1..m]$. The size of e is defined by $|e| = |w_1 \cdots w_m| + |\varphi|$. CSREs define the same class of languages as CQDDs [16] (see [18]), however they have a much simpler presentation avoiding automata altogether and as we shall see quite amenable to a number of operations.

Next, we present some closure and decidability results for the class of CSRE definable languages. These results can be also deduced from [18] since CSREs define bounded semilinear languages.

Lemma 5. *The class of languages defined by CSREs is closed under intersection, union and concatenation. The emptiness, membership and inclusion problems for CSREs are decidable.*

From Lemma 4 it is clear that in order to compute the effect of the iteration of a sequence θ on the content of stack i one has to *left-quotient* the content of stack i by the sequence $u_i x_i^{j-1}$ and then add the sequence $v_i y_i^{j-1}$ (on the left). With this in mind we now examine left-quotients of languages defined by CSREs w.r.t. iterations of a given word. First we state a technical lemma.

Lemma 6. *Let e be a CSRE over an alphabet Σ and $w \in \Sigma^*$ be a word. Then, we can construct, in polynomial time in $|w| + |e|$, a CSRE $e' = (w, u_1, u_2, \ldots, u_k, \varphi(y, y_1, y_2, \ldots, y_k))$ such that for every $i \in \mathbb{N}$, $L(e_i) = \{w' \mid w^i w' \in L(e)\}$ where $e_i = (\epsilon, u_1, u_2, \ldots, u_k, (y = i \wedge \varphi(y, y_1, y_2, \ldots, y_k)))$.*

The key point about the above lemma is that the left-quotient of $L(e)$ w.r.t w^i, for some $i \in \mathbb{N}$, can be precisely identified as $L(e_i)$. Thus, the CSRE $(\epsilon, u_1, u_2, \ldots, u_k, \varphi(y, y_1, y_2, \ldots, y_k))$ defines the left-quotient of $L(e)$ w.r.t $\{w^i \mid i \in \mathbb{N}\}$, giving us the following corollary.

Corollary 1. *Let e be a CSRE over an alphabet Σ and $w \in \Sigma^*$ be a word. Then, we can construct, in polynomial time in $|w| + |e|$, a CSRE e' such that $L(e') = \{w' \mid \exists i \in \mathbb{N}.\, w^i w' \in L(e)\}$.*

6.3 Multi-dimensional Constrained Simple Regular Expression

Let $n \geq 1$. An n-dim CSRE e over an alphabet Σ is a of tuple of the form $((u_1, \ldots, u_{k_1}), (u_{k_1+1}, \ldots, u_{k_2}), \ldots, (u_{k_{n-1}+1}, \ldots, u_{k_n}), \varphi(x_1, \ldots, x_{k_n}))$ where: (1) $1 \leq k_1 < k_2 < \cdots < k_n$ and (2) for every $i \in [1..k_n]$, u_i is a word over Σ. An n-dim CSRE $\mathbf{e}$ accepts the n-dim language, denoted by $L(e)$, consisting of the n-dim words of the form $(u_1^{i_1} \cdots u_{k_1}^{i_{k_1}}, \cdots, u_{k_{n-1}+1}^{i_{k_{n-1}+1}} \cdots u_{k_n}^{i_{k_n}})$ such that φ holds for the function μ defined by $\mu(x_j) = i_j$ for all $j \in [1..k_n]$. In order to simply the notations, we sometimes write e as follows $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\mathbf{n}, \varphi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n))$ where $\mathbf{u}_i = (u_{k_{i-1}+1}, \ldots, u_{k_i})$ and $\mathbf{x}_i = (x_{k_{i-1}+1}, \ldots, x_{k_i})$ for all $i \in [1..n]$. In the following, we show that the class of languages accepted by n-dim CSREs enjoys the same properties as the class of CSREs.

Lemma 7. *Let $n \geq 1$. The class of n-languages defined by n-CSREs is closed under intersection, union and concatenation. The emptiness problem , membership problem as well as inclusion problem are decidable for n-dim CSREs.*

Next, we extend Lemma 6 to n-dim CSREs — n-dim CSREs are closed under left quotienting by simultaneous iterations of a tuple of words $w_i, 1 \leq i \leq n$, one for each component. Even more, this can be achieved by constructing an n-CSRE in which the number of iterations may be set parametrically.

Lemma 8. *Let $n \geq 1$. Let e be a n-dim CSRE over an alphabet Σ and $\mathbf{w} = (w_1, \ldots, w_n)$,$w_i \in \Sigma^*$. Then, we can construct, in polynomial time in $|e| + \sum_i |w_i|$, an n-dim CSRE $e[\mathbf{w}] = (\mathbf{u}_1, \ldots, \mathbf{u}_n, \varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n))$ such that $\mathbf{u}_i[1] = w_i$, for $1 \leq i \leq n$ and for every $j \in \mathbb{N}$, $L(e[\mathbf{w}, j]) = \{\mathbf{v} \mid \mathbf{v}[i \leftarrow w_i^j \mathbf{v}[i]] \in L(e)\}$, where $e[\mathbf{w}, j] = (\mathbf{u}_1[1 \leftarrow \epsilon], \ldots, \mathbf{u}_n[1 \leftarrow \epsilon], (\bigwedge_{1 \leq i \leq n} \mathbf{x}_i[1] = j \wedge \varphi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)))$.*

We now have all the ingredients necessary to study the stability of sets of configurations defined by n-dim CSREs. We say that a set C of configurations of the MPDS M is CSRE representable if there is a function f that maps any state $q \in Q$ of M to an n-dim CSRE s.t. $(q, w_1, \ldots, w_n) \in C$ iff $(w_1, \ldots, w_n) \in L(f(q))$.

6.4 Acceleration of CSRE Representable Set of Configurations

Let $M = (n, Q, \Gamma, \Delta)$ be an MPDS. We now examine the sets $Post_{\Delta_i^*}(C)$ and $Post_{\theta^*}(C)$ where Δ_i is a set of transitions on the i-th stack of M and $\theta \in \Delta^*$ where C is a CSRE representable set of configurations.

Theorem 4. *For every transition sequence $\theta \in \Delta^*$, the class of CSRE representable sets of configurations is effectively closed under $Post_{\theta^*}$. Further post set can be computed in time polynomial in the size of θ and $|M|$.*

Proof. Let θ be a sequence of transitions of the form $(q_0, op_0, q_0')(q_1, op_1, q_1') \cdots (q_m, op_m, q_m')$ and C be a CSRE representable set of configurations. Since $Post_{\theta^*}(C_1 \cup C_2) = Post_{\theta^*}(C_1) \cup Post_{\theta^*}(C_2)$, we can assume w.l.o.g that C consists of configurations of the form $(q, w_1, \ldots, w_n)$ for some fixed $q \in Q$. Let f be a function from Q to n-dim CSREs such that $L(f(p)) = \{(w_1, \ldots, w_n) \,|\, (q, w_1, \ldots, w_n) \in C\}$ if $p = q$ and $L(f(p)) = \emptyset$ otherwise. Next, we assume that $f(q) = (\mathbf{u}_1, \ldots, \mathbf{u_n}, \varphi(\mathbf{x}_1, \ldots, \mathbf{x}_n))$. The proof proceeds by cases.

Case 1: Let us assume $q_i' \neq q_i$ for some $i \in [0..m-1]$ or $q_0 \neq q$. In this case the sequence of transitions cannot be executed and hence $Post_{\theta^*}(C) = C$.

Case 2: Let us assume $q_0 \neq q_m'$, $q_0 = q$ and $q_i' = q_i$ for all $i \in [0..m-1]$. In this case, the sequence of transitions cannot be iterated more than once and so we have $Post_{\theta^*}(C) = Post_\theta(C) \cup C$. We now examine the set $Post_\theta(C)$. If $\mathtt{Summ}(i, \theta) = \bot$ for some $i \in [1..n]$, then $Post_\theta(C) = \emptyset$ and hence $Post_{\theta^*}(C) = C$.

Let us assume now that $\mathtt{Summ}(i, \theta) = (u_i, v_i)$ is well-defined for all $i \in [1..n]$. We can construct a n-CSRE e' such that $(q_m', w_1, \ldots, w_n) \in Post_{\theta^*}(C)$ iff $(w_1, \ldots, w_n) \in L(e')$ in two steps: Let $e_1 = f(q)[(u_1, u_2, \cdots, u_n), 1]$. This left quotients component i by u_i as required and the size of e_1 is polynomial in the size of θ, M and $f(q)$. Let us assume that e_1 is of the form $((\epsilon, w_2, \ldots, w_{\ell_1}), \ldots, (\epsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi''(x_1, \ldots, x_{\ell_n}))$. Next, we simultaneously add the content v_i to stack i, $1 \leq i \leq n$ as follows: Let the n-CSRE e' be $((v_1, \epsilon, w_2, \ldots, w_{\ell_1}), \ldots, (v_n, \epsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi'(y_1, x_1, \ldots, x_{k_1}, \ldots, y_n, x_{\ell_{n-1}+1}, \ldots, x_{\ell_n}))$ where $\varphi' = \varphi'' \wedge \bigwedge_{1 \leq h \leq n} y_i = 1$.

Note that $Post_{\theta^*}(C)$ is CSRE representable by the function f' s.t. $f'(q) = f(q)$, $f'(q_m') = e'$, and $\mathrm{L}(f'(p)) = \emptyset$ for all $p \notin \{q, q_m'\}$. Observe that the construction of $Post_{\theta^*}(C)$ is done in polynomial time in the sizes of θ, M and $f(q)$.

Case 3: Let us assume $q_0 = q_m'$, $q_0 = q$ and $q_i' = q_i$ for all $i \in [0..m-1]$. In this case, the sequence of transitions forms a loop in the control flow graph of M and hence the sequence may possibly be iterated. Observe that if the function $\mathtt{Summ}(i, \theta) = \bot$ for some $i \in [1..n]$, then $Post_{\theta^*}(C) = C$. Hence, let us assume that $\mathtt{Summ}(i, \theta) = (u_i, v_i)$ for all $i \in [1..n]$ so that it is well-defined for each i. Lemma 3 suggests that we should examine when $\mathtt{Summ}(i, \theta\theta)$ is defined for all i.

Indeed, if $\texttt{Summ}(i, \theta\theta)$ is undefined for some $i \in [1..n]$, then $Post_{\theta^*}(C) = Post_\theta(C) \cup C$ (which can be computed as shown in the previous case). So, let us further assume that $\texttt{Summ}(i, \theta\theta)$ is well-defined for all $i \in [1..n]$. Hence, the function $\texttt{Iter}(i, \sigma)$ is also well-defined. Let us assume that $\texttt{Iter}(i, \sigma) = (x_i, y_i)$

We then construct a n-CSRE e' such that $(q'_m, w_1, \ldots, w_n) \in Post_{\theta^+}(C)$ iff $(w_1, \ldots, w_n) \in L(e')$ in a sequence of steps: First we construct the n-CSRE expression $e_1 = f(p)[(u_1, u_2, \cdots, u_n), 1]$. Let us assume that e_1 is of the form $((\epsilon, w_2, \ldots, w_{\ell_1}), \ldots, (\epsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi_1(x_1, \ldots, x_{\ell_n}))$. Observe that the size of e_1 is polynomial in the sizes of θ, M and $f(q)$ and it simultaneously left quotients component i by u_i. Now, we must simultaneously left-quotient the ith component by x_i^j, for a fixed j and then follow this by adding simultaneously y_i^j to component i (for the same j) and then add simultaneously v_i to component i ($1 \leq i \leq n$). To achieve this we begin by applying Lemma 8 to e_1 to construct the n-CSRE expression $e_2 = e_1[(x_1, \ldots, x_n)]$. Observe that the size of e_2 is also polynomial in the sizes of θ, M and $f(q)$. Let us assume that e_2 is of the form $((\epsilon, w'_2, \ldots, w'_{j_1}), \ldots, (\epsilon, w'_{j_{n-1}+2}, \ldots, w'_{j_n}), \varphi_2(z_1, \ldots, z_{j_n}))$. We now exploit the parametrized nature of $e_1[(x_1, \ldots, x_n)]$ stated in Lemma 8. We let $e' = ((v_1, y_1, \epsilon, w'_2, \ldots, w'_{j_1}), \ldots, (v_n, y_n, \epsilon, w'_{j_{n-1}+2}, \ldots, w'_{j_n}), \varphi'(t_1, t'_1, z_1, \ldots, z_{k_1}, \ldots, t_n, t'_n, z_{\ell_{n-1}+1}, \ldots, z_{j_n}))$ where $\varphi' = \varphi_2 \wedge \bigwedge_{1 \leq h \leq n} t_i = 1 \wedge (z_1 = z_{j_1+1} = \cdots = z_{z_{j_{n-1}+1}} = t'_1 = t'_2 = \cdots = t'_n)$.

Finally, it is easy to see that $Post_{\theta^*}(C)$ is CSRE representable by the function f' such that $L(f'(q)) = L(f(q)) \cup L(e')$, and $\mathrm{L}(f'(p)) = \emptyset$ for all $p \notin \{q\}$. Observe that the size of $f'(q)$ is still polynomial in the sizes of θ, M and $f(q)$. □

Unfortunately, CSRE representable sets are not stable w.r.t. bounded context.

Theorem 5. *For every $i \in [1..n]$, the class of CSRE representable sets of configurations is not closed under $Post_{\Delta_i^*}(C)$.*

7 Acceleration of Bounded-Context Sets

In the following, we first introduce the class of *constrained* rational languages (as an extension of constrained (or Parikh) automata languages [17,31] to the settings of multi-dimensional words). Then, we present the class of bounded context-switches sets as a generalization of loops and contexts. Finally, we show that the class of constrained rational languages is stable with respect to acceleration by bounded context-switches sets.

7.1 Constrained Rational Languages

A constrained automaton is a finite-state automaton augmented with a semi-linear set to filter (or restrict) the accepting runs. We assume that this semi-linear set is described by an existential Presburger formula. In the following, we extend this model to multi-dimensional words. Let $n \geq 1$ and $\Sigma_1, \ldots, \Sigma_n$ be n finite alphabets. Formally, a n-tape constrained finite-state automaton over $\Sigma_1, \ldots, \Sigma_n$

is defined as $\mathcal{C} = (A, \varphi)$ where $A = (Q, \Sigma_1, \ldots, \Sigma_n, \delta, q_0, F)$ is a n-tape finite-state automaton and φ is an existential Presburger formula such that $var(\varphi) = \delta$. Furthermore, we assume w.l.o.g. that if $(q, \mathbf{u}, q')$ is in δ then $|\,\mathbf{u}[1]\cdot\mathbf{u}[2]\cdots\mathbf{u}[n]\,| \leq 1$. The language of $\mathcal{C}$, denoted by $L(\mathcal{C})$, is the set of n-dim words $\mathbf{w}$ for which there is an accepting run π of A over w such that $Parikh(\pi) \models \varphi$. A n-dim language is *constrained* rational if it is the language of some n-tape constrained automaton. Let us state some properties about constrained rational languages. These properties can be inferred from the properties of rational languages [10] and Parikh/constrained automata [17,31,42].

Lemma 9. *The class of constrained rational languages is closed under union and concatenation but not under intersection. The emptiness and membership problems are decidable while the emptiness of intersection problem is undecidable.*

We can extend the permutation, projection and composition operations to the context of constrained rational languages in the straightforward manner. We also show the same closure properties as in the case of rational languages.

Lemma 10. *The class of constrained rational languages is closed under permutation, projection, composition and intersection with regular languages.*

The complexity of permutation, projection, composition is at most polynomial in size of input automata whereas the intersection with regular languages is at most exponential in the size of the description of the regular language and polynomial in the size of constrained rational automaton.

7.2 Acceleration of Bounded Context-Switches Sets

Let $M = (n, Q, \Gamma, \Delta)$ be an MPDS. A bounded context-switches set over M is defined by $\Lambda = (\tau_0, \tau_1, \ldots, \tau_{2m})$ with $m \in \mathbb{N}$ where (1) for every $i \in [0..m]$, we have $\tau_{2i} \subseteq \Delta_{j_i}$ for some $j_i \in [1..n]$ with $j_0 = j_{2m}$, and (2) for every $i \in [0..(m-1)]$, $|\tau_{2i+1}| = 1$. The size of Λ is defined as the sum of the sizes of the finite sets τ_j for all $j \in [0..2m]$. The set of sequences of transitions recognized by Λ, denoted by $L(\Lambda)$, is $\tau_0^* \tau_1 \tau_2^* \cdots \tau_{2m}^*$. Observe that when $m = 0$ and $\tau_0 = \Delta_i$ for some $i \in [1..n]$, $L(\Lambda)$ corresponds to a context associated to the stack i. And whenever $\tau_{2i} = \emptyset$ for all $i \in [0..m]$, $L(\Lambda)$ is a sequence of transitions. Thus, bounded context-switches sets generalize both loops and contexts. Observe that dropping one of τ_{2i+1} from the definition of Λ will allow the simulation of unbounded unrestricted context-switch sequences and hence leads to the undecidability of the simple reachability problem. Next, we state our main theorem:

Theorem 6. *Let M be an MPDS and $\Lambda = (\tau_0, \tau_1, \ldots, \tau_{2m})$ be a bounded context-switches set over M. For every constrained rational set of configurations C, $Post_{L(\Lambda)^*}(C)$ is a constrained rational set and effectively constructible.*

The rest of this section is dedicated to the proof of Theorem 6. First we prove an extension of Lemma 1 that shows that in addition to computing pairs of the form (q, u, q', u) such that there is a run π from (q, u) to (q', u') one may in addition keep track of the number iterations of $L(\Lambda)$ seen along π.

Lemma 11. *Let $\mathcal{P} = (1, P, \Gamma, R)$ be an PDS and $\Lambda = (\tau_0, \tau_1, \ldots, \tau_{2m})$ be a bounded context-switches set over $\mathcal{P}$ such that $\tau_j \subseteq R$. Let $\sharp$ be a special symbol not included in Γ. Then it is possible to construct, in exponential time in the sizes of $\mathcal{P}$ and Λ, an 5-tape finite-state automaton $T = (Q_T, Q, \Gamma, Q, \Gamma, \{\sharp\}, \delta_\pi, q_0, F_T)$ such that $(q, u, q', v, \sharp^m) \in L(T)$ iff $(q, u) \xrightarrow{\pi}_{\mathcal{P}} (q', v)$ for some sequence $\pi \in (L(\Lambda))^m$. Furthermore, the size of T is exponential in the sizes of $\mathcal{P}$ and Λ.*

The proof of this lemma is based on the combination of the proof of Lemma 1 with the fact the Parikh images of context-free languages can be effectively realized as regular languages. This ability to compute the number of iterations of $L(\Lambda)$ along the run is important. It can be combined with the special structure of $L(\Lambda)$, which forces context-switches to occur at identified transitions and in a fixed sequence. This allows us to prove Lemma 12, leading to the proof of Theorem 6.

Now, one can construct a PDS M_i for each stack i, which simulates the moves of M on the ith stack while guessing, non-deterministically, the effect of the moves corresponding to the other stacks. Clearly, any run of M can be decomposed in to a tuple of runs, one per M_i. However, because of the special structure of $L(\Lambda)$, a converse of this statement is true for runs of the form $L(\Lambda)^*$. Any tuple of runs, one from each M_i, which agree on the number of iterations of $L(\Lambda)$ seen along the run, can be composed together to give a run M.

Let $i \in [1..n]$. For each transition $t = (q, op, q') \in \Delta$, we represent the effect of the transition t on the stack i by the transition $t|_i$ defined as follows: $t|_i = t$ if $t \in \Delta_i$, and $t|_i = (q, nop_i, q')$ otherwise. We extend this operation to sets of transitions as follows: For a set $T \subseteq \Delta$, $T|_i = \{t|_i \mid t \in T\}$.

Let $M_i = (1, Q, \Gamma, \bigcup_{j \in [0..2m]} \tau_j|_i)$ be a PDS simulating the i-th stack while taking into account the effect transitions of the other stack operations. We define also $\Lambda|_i$ to be the bounded context-switches set defined by the tuple $(\tau_0|_i, \tau_1|_i, \ldots, \tau_{2m}|_i)$. Let T_i be the 5-tape finite state automaton resulting from the application of Lemma 11 to the PDA M_i and the bounded context-switches set $\Lambda|_i$. Then synchronizing the multi-tape automata T_i on the number of occurrences of the special symbol $\sharp$ provides a relation between any possible starting configuration $(q, u_1, \ldots, u_n)$ with any configuration $(q', v_1, \ldots, v_n)$ reachable from $(q, u_1, \ldots, u_n)$ of M by firing a sequence of transitions in $L(\Lambda)^*$.

Lemma 12. *Let $m \in \mathbb{N}$. Then, $(q, u_1, \ldots, u_n) \xrightarrow{\pi}_{\mathcal{M}} (q', v_1, \ldots, v_n)$ for some sequence $\pi \in (L(\Lambda))^m$ if and only if for every $i \in [1..n]$, $(q, u_i, q', v_i, \sharp^m) \in L(T_i)$.*

Now, we are ready to prove Theorem 6. Let us assume that we are given a $(n+1)$-tape constrained automaton $\mathcal{C} = (A, \phi)$ where $A = (P, Q, \Gamma, \ldots, \Gamma, \delta, p_0, F)$ and $L(\mathcal{C}) = C$. In the following, we show how to compute a $(n+1)$-tape constrained automaton $\mathcal{C}'$ accepting the set $Post_{(L(\Lambda))^*}(C)$. To do that, we proceed as follows: We first compose C with the constrained automaton (T_1, true), synchronizing the second tape of T_1 (containing the stack contents at the starting configuration of the M_1) with the second tape of A, to construct a $(n+5)$-tape constrained automaton $\mathcal{C}_1 = \mathcal{C} \circ_{(2,2)} (T_1, \mathsf{true})$. We then need to synchronize the starting states (i.e., the first tape of A with the first tape of T_1). This can be done by intersecting $\mathcal{C}_1$ with the (regular) language $\bigcup_{q\in Q}\{q\}\times(\Gamma^*)^n\times\{q\}\times Q\times\Gamma^*\times(\{\sharp\})^*$. Let $\mathcal{C}_1'$ be the $(n+5)$-tapes resulting of this intersection. Then, we project away the starting control state occurring on the $n+2$-tape and the content of the second tape to obtain the $(n+3)$-tape constrained automaton $\mathcal{C}_1''=\Pi_\iota(\mathcal{C}_1')$ where $\iota=([1..n+5]\setminus\{2,n+2\})$.

Then, we need to compose $\mathcal{C}_1''$ with the constrained automaton (T_2,true), synchronizing the second tape of T_2 (containing the stack contents at the starting configuration of the M_2) with the second tape of $\mathcal{C}_1''$, to construct a $(n+7)$-tape constrained automaton $\mathcal{C}_2=\mathcal{C}_1''\circ_{(2,2)}(T_2,\mathsf{true})$. We then need to synchronize the starting states (i.e., the first tape of $\mathcal{C}_1''$ with the first tape of T_2). This can be done by intersecting $\mathcal{C}_2$ with the (regular) language $\bigcup_{q\in Q}\{q\}\times(\Gamma^*)^{n-1}\times Q\times\Gamma^*\times(\{\sharp\})^*\times\{q\}\times Q\times\Gamma^*\times(\{\sharp\})^*$. Let $\mathcal{C}_2'$ be the $(n+7)$-tapes resulting of this intersection. Then, we project away the state occurring on the $n+4$-tape and the content of the second tape to obtain the $(n+5)$-tape constrained automaton $\mathcal{C}_2''=\Pi_{\iota'}(\mathcal{C}_2')$ where $\iota'=([1..n+6]\setminus\{2,n+4\})$.

This procedure is then repeated for all the constrained automata (T_i,true), with $i\in[3..n]$, to obtain at the end the $(3n+1)$-tape constrained automaton $\mathcal{C}_n''$. We can also project away the state stored at the first tape from $\mathcal{C}_n''$ since it is no longer needed. So, let $\mathcal{G}=\Pi_{[2..3n]}(\mathcal{C}_n'')$ be the resulting $(3n)$-tape constrained automaton.

Now, we need to synchronize the automata (T_i,true) on their final states stored respectively at the tapes $3(i-1)+1$, with $i\in[1..n]$, of $\mathcal{G}$. To do that we intersect $\mathcal{G}$ with the (regular) language $\bigcup_{q\in Q}\{q\}\times\Gamma^*\times(\{\sharp\})^*\times\{q\}\times\Gamma^*\times(\{\sharp\})^*\times\cdots\times\{q\}\times\Gamma^*\times(\{\sharp\})^*$. Let $\mathcal{G}'$ be the $(3n)$-tapes resulting of this intersection. We can then project away the copies of the final control states and only keep its first occurrence to obtain $(2n+1)$-tape constrained automaton $\mathcal{G}''$ defined as follows: $\mathcal{G}''=\Pi_{\iota''}(\mathcal{G}')$ where $\iota''=([1..3n]\setminus\{3i+1\,|\,i\in[1..n-1]\})$. Let us assume that $\mathcal{G}''$ is of the form (A',ϕ') where $A'=(P',Q,\Gamma,\{\sharp\},\Gamma,\{\sharp\},\ldots,\Gamma,\{\sharp\},\delta',p_0',F')$. For every $i\in[1..n]$, let δ_i' be the subset of δ' containing only transitions of the form $(p,\mathbf{v},p')\in\delta'$ s.t. $\mathbf{v}[2i+1]=\sharp$ (note that $\mathbf{v}[j]=\epsilon$ for all $j\neq 2i+1$).

From Lemma 12, we need to ensure the same number of the special letters $\sharp$ in all the tapes $\{2i+1|i\in[1..n]\}$ by augmenting the formula ϕ' with additional constraints. Let $\mathcal{G}'''=(A',\phi'')$ where $\phi''=\phi'\wedge(\sum_{t_1\in\delta_1'}t_1=\sum_{t_2\in\delta_2'}t_2=\cdots=\sum_{t_n\in\delta_n'}t_n)$. Finally, the $n+1$-tape constrained finite state automaton $\mathcal{C}'$ can be constructed from $\mathcal{G}'''$ by projecting away the tapes with symbol $\sharp$ i.e. the tapes $\{2i+1|i\in[1..n]\}$. Hence, $\mathcal{C}'=\Pi_{\iota'''}(\mathcal{G}''')$ where $\iota'''=([1..n]\setminus\{2i+1|i\in[1..n]\})$.

Using the complexity results for permutation, projection, composition and the intersection with regular languages for constrained rational languages, we can show that the size of $\mathcal{C}'$ is at most double-exponential in the sizes of M and Λ.

References

1. Annichini, A., Asarin, E., Bouajjani, A.: Symbolic techniques for parametric reasoning about counter and clock systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 419–434. Springer, Heidelberg (2000)
2. Atig, M.F., Bollig, B., Habermehl, P.: Emptiness of multi-pushdown automata is 2ETIME-complete. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 121–133. Springer, Heidelberg (2008)

3. Atig, M.F.: Model-checking of ordered multi-pushdown automata. Logical Methods Comput. Sci. **8**(3), 1–31 (2012)
4. Atig, M.F., Bouajjani, A., Narayan Kumar, K., Saivasan, P.: Linear-time model-checking for multithreaded programs under scope-bounding. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 152–166. Springer, Heidelberg (2012)
5. Atig, M.F., Bouajjani, A., Qadeer, S.: Context-bounded analysis for concurrent programs with dynamic creation of threads. Logical Methods Comput. Sci. **7**(4), 107–123 (2011)
6. Atig, M.F., Narayan Kumar, K., Saivasan, P.: Adjacent ordered multi-pushdown systems. In: Béal, M.-P., Carton, O. (eds.) DLT 2013. LNCS, vol. 7907, pp. 58–69. Springer, Heidelberg (2013)
7. Bardin, S., Finkel, A., Leroux, J., Petrucci, L.: FAST: acceleration from theory to practice. STTT **10**(5), 401–424 (2008)
8. Bardin, S., Finkel, A., Leroux, J., Schnoebelen, P.: Flat acceleration in symbolic model checking. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 474–488. Springer, Heidelberg (2005)
9. Bérard, B., Fribourg, L.: Reachability analysis of (timed) Petri nets using real arithmetic. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 178–193. Springer, Heidelberg (1999)
10. Berstel, J.: Transductions and context-free langages. TeubnerStudienbucher Informatik. Springer, Heidelberg (1979)
11. Boigelot, B.: On iterating linear transformations over recognizable sets of integers. Theor. Comput. Sci. **309**(1–3), 413–468 (2003)
12. Boigelot, B.: Domain-specific regular acceleration. STTT **14**(2), 193–206 (2012)
13. Boigelot, B., Wolper, P.: Symbolic verification with periodic sets. In: Dill, D.L. (ed.) Computer Aided Verification. LNCS, vol. 818, pp. 55–67. Springer, Heidelberg (1994)
14. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
15. Bouajjani, A., Esparza, J., Schwoon, S., Strejček, J.: Reachability analysis of multithreaded software with asynchronous communication. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 348–359. Springer, Heidelberg (2005)
16. Bouajjani, A., Habermehl, P.: Symbolic reachability analysis of FiFo-channel systems with nonregular sets of configurations. Theor. Comput. Sci. **221**(1–2), 211–250 (1999)
17. Cadilhac, M., Finkel, A., McKenzie, P.: On the expressiveness of Parikh automata and related models. In: NCMA. books@ocg.at, vol. 282, pp. 103–119. Austrian Computer Society (2011)
18. Cadilhac, M., Finkel, A., McKenzie, P.: Bounded parikh automata. Int. J. Found. Comput. Sci. **23**(8), 1691–1710 (2012)
19. Caucal, D.: On the regular structure of prefix rewriting. Theor. Comput. Sci. **106**(1), 61–86 (1992)
20. Esparza, J., Knoop, J., Majumdar, R.: An automata-theoretic approach to interprocedural data-flow analysis. In: Thomas, W. (ed.) FOSSACS 1999. LNCS, vol. 1578, pp. 14–30. Springer, Heidelberg (1999)
21. Esparza, J., Ganty, P., Majumdar, R.: A perfect model for bounded verification. In: LICS, pp. 285–294. IEEE Computer Society (2012)
22. Esparza, J., Ganty, P., Poch, T.: Pattern-based verification for multithreaded programs. ACM Trans. Program. Lang. Syst. **36**(3), 9:1–9:29 (2014)

23. Esparza, J., Kiefer, S., Schwoon, S.: Abstraction refinement with craig interpolation and symbolic pushdown systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 489–503. Springer, Heidelberg (2006)
24. Finkel, A.: A generalization of the procedure of Karp and Miller to well structured. In: Ottmann, T. (ed.) ICALP 1987. LNCS, vol. 267, pp. 499–508. Springer, Heidelberg (1987)
25. Finkel, A., Leroux, J.: How to compose Presburger-accelerations: Applications to broadcast protocols. In: Agrawal, M., Seth, A.K. (eds.) FSTTCS 2002. LNCS, vol. 2556, pp. 145–156. Springer, Heidelberg (2002)
26. Finkel, A., Willems, B., Wolper, P.: A direct symbolic approach to model checking pushdown systems. Electr. Notes Theor. Comput. Sci. **9**, 27–37 (1997)
27. Ganty, P., Majumdar, R., Monmege, B.: Bounded underapproximations. FMSD **40**(2), 206–231 (2012)
28. Heußner, A., Leroux, J., Muscholl, A., Sutre, G.: Reachability analysis of communicating pushdown systems. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 267–281. Springer, Heidelberg (2010)
29. Karp, R.M., Miller, R.E.: Parallel program schemata. J. Comput. Syst. Sci. **3**(2), 147–195 (1969)
30. Kelly, W., Pugh, W., Rosser, E., Shpeisman, T.: Transitive closure of infinite graphs and its applications. J. Parallel Program. **24**(6), 579–598 (1996)
31. Klaedtke, F., Rueß, H.: Monadic second-order logics with cardinalities. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) Automata, Languages and Programming. LNCS, vol. 2719, pp. 681–696. Springer, Heidelberg (2003)
32. La Torre, S., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 299–314. Springer, Heidelberg (2008)
33. La Torre, S., Madhusudan, P., Parlato, G.: Reducing context-bounded concurrent reachability to sequential reachability. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 477–492. Springer, Heidelberg (2009)
34. Lal, A., Reps, T.W.: Reducing concurrent analysis under a context bound to sequential analysis. FMSD **35**(1), 73–97 (2009)
35. Lal, A., Touili, T., Kidd, N., Reps, T.: Interprocedural analysis of concurrent programs under a context bound. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 282–298. Springer, Heidelberg (2008)
36. Leroux, J.: Acceleration for Petri nets. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 1–4. Springer, Heidelberg (2013)
37. Leroux, J., Sutre, G.: Flat counter automata almost everywhere!. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 489–503. Springer, Heidelberg (2005)
38. Musuvathi, M., Qadeer, S.: Iterative context bounding for systematic testing of multithreaded programs. In: PLDI, pp. 446–455. ACM (2007)
39. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 93–107. Springer, Heidelberg (2005)
40. Reps, T., Schwoon, S., Jha, S.: Weighted pushdown systems and their application to interprocedural dataflow analysis. In: Cousot, R. (ed.) Static Analysis. LNCS, vol. 2694, pp. 189–213. Springer, Heidelberg (2003)
41. Song, F., Touili, T.: Pushdown model checking for malware detection. STTT **16**(2), 147–173 (2014)
42. Wong, K.: Parikh automata with pushdown stack. Diploma thesis, RWTH Aachen (2004)