Acceleration in Multi-Pushdown Systems

Mohamed Faouzi Atig[1] K. Narayan Kumar[2] and Prakash Saivasan[2]

[1] Uppsala University, Sweden / mohamed_faouzi.atig@it.uu.se
[2] Chennai Mathematical Institute, 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. [38, 16, 39, 20, 9, 23]). 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(s) 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 [37]. 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 other verification problems. Subsequently, other classes generalizing the bounded context assumption have been proposed (see [30, 31, 10, 11, 41, 40, 25]).

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 ([9, 23]).

The configuration of a MPDS can be represented as a tuple of words giving the current state and the contents of the each of the stacks. We can then represent sets of configurations by *recognizable* or *regular* languages [5]. Given a 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 [37]. 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 $post^*(I)$ be the same. For eg. 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 to be a very useful (e.g., [12, 2, 8, 1, 4, 27, 7, 6, 22, 3, 34, 35, 26, 21]) in the analysis of a variety of infinite state systems. We propose to use accelerations as an under-approximation technique 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 sets as well as rational sets of configurations are stable w.r.t. bounded context executions. Next 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 [12] and the class of bounded semilinear languages [14]. This forms a very expressive class, for eg. the 1-dimensional version is equivalent to the class of semilinear bounded languages (see [14]). We show that n-CSREs are closed under union, intersection and concatenation. Furthermore, we have the decidability of the emptiness, membership problem as well as the inclusion problem for n-CSREs. Then, 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 executions.

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 (the most general class considered here and a n-dimensional version of Parikh automata) 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. We provide details of the constructions and sketch the ideas behind the proofs leaving the details to the appendix.

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 decid-

able 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 [19, 18, 24]. 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). Furthermore, our approach has the advantage to be incremental (in contrast with the approach used [19, 18, 24]): Assume that we have already computed the set C of reachable configurations induced by the bounded language $w_1^* w_2^* \cdots w_n^*$, then in order to compute the set of reachable configurations induced by the bounded language $w_1^* w_2^* \cdots w_n^* w_{n+1}^*$, we only need to compute the effect of iterating the loop w_{n+1} from C. While the approach used in [19, 18, 24] will require to do the analysis from scratch for the bounded language $w_1^* w_2^* \cdots w_n^* w_{n+1}^*$.

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 $g(a) = \perp$ 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 $\mathbf{uv} = (u_1 v_1, \ldots, u_n v_n)$. 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 $\delta \subseteq (Q \times (\Sigma_1 \cup \{\varepsilon\}) \times \cdots \times (\Sigma_n \cup \{\varepsilon\}) \times Q)$, 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 $(q_0, \mathbf{u}_1, q_1), (q_1, \mathbf{u}_2, q_2), \ldots, (q_{n-1}, \mathbf{u}_n, q_n) \in \delta$ such that $\mathbf{w} = \mathbf{u}_1 \mathbf{u}_2 \cdots \mathbf{u}_n$. The run π is accepting if $q_n \in F$. 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 [5]. 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., [5]). 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 $O(|A|.|A'|)$, 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 $\mathcal{C}f(M)$. 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 $u_j = v_j = az$ for some z, for all $j \in [1..n]$.

For a sequence of transitions $\sigma = t_1 t_2 \ldots t_m \in \Delta^*$ and two configurations $c, c' \in \mathcal{C}f(M)$, we write $c \xrightarrow{\sigma}_M c'$ to denote that one of the following two cases holds: (1) $\sigma = \varepsilon$ and $c = c'$, or (2) there are configurations $c_0, \cdots, c_m \in \mathcal{C}f(M)$ 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 $C \subseteq \mathcal{C}f(M)$ 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 of configurations c' such that $c \xrightarrow{\sigma}_M c'$ where $c \in C$ and $\sigma \in \Theta^*$. We use $Post_{\Theta^*}(C)$ to denote the set $\{c' \mid c \xrightarrow{\sigma}_M c', c \in C, \sigma \in \Theta^*\}$. Observe that the global reachability problem is the acceleration problem with the set of initial configurations as C and $\Theta = \Delta$.

4 Context-Bounding as an Acceleration Problem

In the following, we show that context-bounding analysis [37, 36, 32, 29] for an MPDS $M = (n, Q, \Gamma, \Delta)$ can be formulated as an acceleration problem wrt. the class of rational/regular configurations. Given two configuration $c, c' \in \mathcal{C}f(M)$ 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 [37]. 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 [15, 33].

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, synchronzing 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 we 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 $j > i$.

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 Section 2. As an immediate consequence of this result and the fact that the membership problem for rational/regular languages can be checked in polynomial time, we can deduce that the k-context reachability problem can be decided in polynomial in the size of M and exponential in k (as in [37]).

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 $\mathtt{Eff}_i : (\Gamma^* \times \Gamma^* \times \Delta^*) \rightharpoonup (\Gamma^* \times \Gamma^*)$. We first define $\mathtt{Eff}_i$ 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, $\mathtt{Eff}_i(u, v, t)$ computes the effect of $\sigma.t$ on stack i. Given $u, v \in \Gamma^*$ and $t \in \Delta$, we define $\mathtt{Eff}_i((u, v), t)$ as follows:

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

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

Our aim is to compute the complete effect of some sequence σ on stack i and this is given by $\mathtt{Eff}_i((\varepsilon, \varepsilon), \sigma)$. We shall refer to this as $\mathtt{Summ}(i, \sigma)$. The next lemma formalizes our intuition about $\mathtt{Summ}$ 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 $c = (p, w_1, \ldots, w_n)$ and $c' = (p', w'_1, \ldots, w'_n)$ 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 $\mathtt{Summ}(i, \sigma) = (u_i, v_i)$.*

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 $\texttt{Summ}(i,\sigma) = \bot$, then $\texttt{Summ}(i,\sigma^j) = \bot$ 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 $\texttt{Summ}$ 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 $y_i = \varepsilon$. In case of the latter we let y_i be the unique word such that $v_i = u_i y_i$ and $x_i = \varepsilon$. 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 show 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 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 $i \in [1..m-1]$ 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 $i \in [1..m-1]$. 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 $i \in [1..m]$. 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 $\texttt{Summ}(i,\theta) = \perp$ 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 $c = (p, w_1, \ldots, w_n)$ and $c' = (p', w'_1, \ldots, w'_n)$ 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 are 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 of 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) and prove some closure properties. CSRE definable languages form an expressive class equivalent to the bounded semi-linear languages defined in [14] and the class of languages accepted by 1-CQDD introduced in [12]. 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: $t ::= 0 \mid 1 \mid x \mid t - t \mid t + t$. The set of formulae of the Presburger arithmetic is defined to be $\varphi ::= t \leq t \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\, \varphi$.

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 $\{(i_1, i_2, \ldots, i_k) \mid \mu \models \varphi(x_1, x_2, \ldots, x_k)$ 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 [12] (see [14]), 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 [14] 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 $e_i = (\varepsilon, u_1, u_2, \ldots, u_k, (y = i \wedge \varphi(y, y_1, y_2, \ldots, y_k)))$.*

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 $(\varepsilon, u_1, u_2, \ldots, u_k, \varphi(y, y_1, y_2, \ldots, y_k))$ 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 $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u_n}, \varphi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n))$ 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 n-languages accepted by n-dim CSREs enjoys the same closure properties as the class of CSREs. Furthermore we have the same decidability results.

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 the case of 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 iterations may be set parametrically. The construction is also in polynomial time.

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 $e[\mathbf{w}, j] = (\mathbf{u}_1[1 \leftarrow \varepsilon], \ldots, \mathbf{u}_n[1 \leftarrow \varepsilon], (\bigwedge_{1 \leq i \leq n} \mathbf{x}_i[1] = j \wedge \varphi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)))$.*

We now have all the technical 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 e_q such that $(q, w_1, \ldots, w_n) \in C$ 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 $L(f(p)) = \{(w_1, \ldots, w_n) \mid (q, w_1, \ldots, w_n) \in C\}$ 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 $i \in [1..m]$ 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 $i \in [1..m]$. 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 $Post_{\theta}(C)$. First, let us assume that $\texttt{Summ}(i, \theta) = \perp$ for some $i \in [1..n]$. Then it is easy to see that $Post_{\theta}(C) = \emptyset$ 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 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 $((\varepsilon, w_2, \ldots, w_{\ell_1}), \ldots, (\varepsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi''(x_1, \ldots, x_{\ell_n}))$. Next, we simultaneously add the content v_i to stack i, $1 \leq i \leq n$ as follows: Let the n-CSRE e' be $((v_1, \varepsilon, w_2, \ldots, w_{\ell_1}), \ldots, (v_n, \varepsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi'(y_1, x_1, \ldots, x_{k_1}, \ldots, y_n, x_{\ell_{n-1}+1}, \ldots, x_{\ell_n}))$ where $\varphi' = \varphi'' \wedge \bigwedge_{1 \leq h \leq n} y_i = 1$.

It is easy to see that $Post_{\theta^*}(C)$ is CSRE representable by the function f' defined as follows $f'(q) = f(q)$, $f'(q_m') = e'$, 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 $i \in [1..m]$. 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 $\texttt{Summ}(i,\theta) = \perp$ 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 $\texttt{Iter}(i,\sigma) = (x_i, y_i)$

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 $((\varepsilon, w_2, \ldots, w_{\ell_1}), \ldots, (\varepsilon, w_{\ell_{n-1}+2}, \ldots, w_{\ell_n}), \varphi_1(x_1, \ldots, x_{\ell_n}))$. 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 $((\varepsilon, w'_2, \ldots, w'_{j_1}), \ldots, (\varepsilon, w'_{j_{n-1}+2}, \ldots, w'_{j_n}), \varphi_2(z_1, \ldots, z_{j_n}))$. We now exploit the parametrized nature of $e_1[(x_1, \ldots, x_n)]$ stated in Lemma 8. We let $e' = ((v_1, y_1, \varepsilon, w'_2, \ldots, w'_{j_1}), \ldots, (v_n, y_n, \varepsilon, w'_{j_{n-1}+2}, \ldots, w'_{j_n}), \varphi'(t_1, t'_1, z_1, \ldots, z_{k_1}, \ldots, t_n, t'_n, z_{\ell_{n-1}+1}, \ldots, z_{j_n}))$ where $\varphi' = \varphi_2 \wedge \bigwedge_{1 \leq h \leq n} t_i = 1 \wedge (z_1 = z_{j_1+1} = \cdots = z_{j_{n-1}+1} = t'_1 = t'_2 = \cdots = t'_n)$.

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 runs.

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 introduce the class of *constrained* rational languages (as an extension of constrained (or Parikh) automata languages to the settings of multi-dimensional words [28, 13]). We show that the class of constrained rational languages is stable with respect to bounded-context runs and simple loops. Even better, we show that the class of constrained rational languages is stable with respect to bounded context-switches sets, a generalization of loops and contexts. The following section is structured as follows: First, we give the formal definition of the class of constrained rational language and state some of its properties. Then, we present the class of bounded context-switches. 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 the class of constrained rational languages. These properties can be inferred from the properties of rational languages [5] and Parikh/constrained automata [28, 13, 43].

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 a tuple $\Lambda = (\tau_0, \tau_1, \ldots, \tau_{2m})$ with $m \in \mathbb{N}$ where (1) for every $i \in [0..m]$, we have $\tau_{2i} \subseteq \Delta_{i_j}$ for some $i_j \in [1..n]$ with $i_0 = i_{2m}$, 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 realised 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.

From an MPDS M, 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. We formalize these arguments now.

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 $M_i = (1,Q,\Gamma,\Delta_i \cup \{\tau_0,\tau_1,\ldots,\tau_{2m}\}|_i)$ be a PDS simulating the i-th stack while taking into account the effect 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 $i \in [1..n]$, $(q,u_i,q',v_i,\sharp^m) \in L(T_i)$.*

Remark: The ideas used in the proof of Lemma 11 can be used to compute more details such as the number of occurrences of every transition (instead of number of iterations of $L(\Lambda)$). However, without the special structure of $L(\Lambda)$, this does not seem to lead to a corresponding generalization of Lemma 12.

Now, we are ready to prove Theorem 6. Let us assume now that we are given a $(n+1)$-tape constrained finite-state 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 finite state automaton $\mathcal{C}'$ accepting the set $Post_{(L(\Lambda))^*}(C)$. To do that, we proceed as follows: We first compose C with the synchronized 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 starting control 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 starting state stored at the first tape from $\mathcal{C}''_n$ since it is not any more needed. So, let $\mathcal{G}$ be $(3n)$-tape constrained automaton such that $\mathcal{G} = \Pi_{[2..3n]}(\mathcal{C}''_n)$.

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 $\iota'' = ([1..3n] \setminus \{3i+1 \mid i \in [1..n-1]\})$. Let us assume that $\mathcal{G}''$ is of the form (A', ϕ') where $A' = (P', Q, \Gamma, \{\sharp\}, \Gamma, \sharp, \ldots, \Gamma, \{\sharp\}, \delta', p'_0, F')$. Furthermore, let us assume that for every $i \in [1..n]$, δ'_i is the subset of δ' containing only the transitions of the form $(p, \mathbf{v}, p') \in \delta'$ such that $\mathbf{v}[2i+1] = \sharp$. Observe that by definition, we have $\mathbf{v}[j] = \varepsilon$ 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 $\{2i+1 | i \in [1..n]\}$, we need to augment 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. LNCS, vol. 1855, pp. 419–434. Springer (2000)
2. Bardin, S., Finkel, A., Leroux, J., Petrucci, L.: FAST: acceleration from theory to practice. STTT 10(5), 401–424 (2008)
3. Bardin, S., Finkel, A., Leroux, J., Schnoebelen, P.: Flat acceleration in symbolic model checking. In: ATVA. LNCS, vol. 3707, pp. 474–488. Springer (2005)
4. Bérard, B., Fribourg, L.: Reachability analysis of (timed) petri nets using real arithmetic. In: CONCUR. LNCS, vol. 1664, pp. 178–193. Springer (1999)
5. Berstel, J.: Transductions and context-free langages. TeubnerStudienbucher Informatik (1979)
6. Boigelot, B.: On iterating linear transformations over recognizable sets of integers. Theor. Comput. Sci. 309(1-3), 413–468 (2003)
7. Boigelot, B.: Domain-specific regular acceleration. STTT 14(2), 193–206 (2012)
8. Boigelot, B., Wolper, P.: Symbolic verification with periodic sets. In: CAV. LNCS, vol. 818, pp. 55–67. Springer (1994)
9. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: CONCUR. LNCS, vol. 1243, pp. 135–150. Springer (1997)
10. Bouajjani, A., Esparza, J., Schwoon, S., Strejcek, J.: Reachability analysis of multithreaded software with asynchronous communication. In: FSTTCS. LNCS, vol. 3821, pp. 348–359. Springer (2005)
11. Bouajjani, A., Fratani, S., Qadeer, S.: Context-bounded analysis of multithreaded programs with dynamic linked structures. In: CAV. LNCS, vol. 4590, pp. 207–220 (2007)
12. 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)
13. 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)
14. Cadilhac, M., Finkel, A., McKenzie, P.: Bounded parikh automata. Int. J. Found. Comput. Sci. 23(8), 1691–1710 (2012)
15. Caucal, D.: On the regular structure of prefix rewriting. Theor. Comput. Sci. 106(1), 61–86 (1992)
16. Esparza, J., Knoop, J.: An automata-theoretic approach to interprocedural data-flow analysis. In: FoSSaCS. LNCS, vol. 1578, pp. 14–30. Springer (1999)
17. Esparza, J., Ganty, P., Kiefer, S., Luttenberger, M.: Parikh's theorem: A simple and direct automaton construction. Inf. Process. Lett. 111(12), 614–619 (2011)
18. Esparza, J., Ganty, P., Majumdar, R.: A perfect model for bounded verification. In: LICS. pp. 285–294. IEEE Computer Society (2012)
19. Esparza, J., Ganty, P., Poch, T.: Pattern-based verification for multithreaded programs. ACM Trans. Program. Lang. Syst. 36(3), 9:1–9:29 (2014)
20. Esparza, J., Kiefer, S., Schwoon, S.: Abstraction refinement with Craig interpolation and symbolic pushdown systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS. LNCS, vol. 3920, pp. 489–503. Vienna, Austria (2006)
21. Finkel, A.: A generalization of the procedure of karp and miller to well structured transition systems. In: ICALP. LNCS, vol. 267, pp. 499–508. Springer (1987)
22. Finkel, A., Leroux, J.: How to compose presburger-accelerations: Applications to broadcast protocols. In: FSTTCS. LNCS, vol. 2556, pp. 145–156. Springer (2002)

23. Finkel, A., Willems, B., Wolper, P.: A direct symbolic approach to model checking pushdown systems. Electr. Notes Theor. Comput. Sci. 9, 27–37 (1997)
24. Ganty, P., Majumdar, R., Monmege, B.: Bounded underapproximations. FMSD 40(2), 206–231 (2012)
25. Heußner, A., Leroux, J., Muscholl, A., Sutre, G.: Reachability analysis of communicating pushdown systems. In: FOSSACS. LNCS, vol. 6014, pp. 267–281. Springer (2010)
26. Karp, R.M., Miller, R.E.: Parallel program schemata. J. Comput. Syst. Sci. 3(2), 147–195 (1969)
27. Kelly, W., Pugh, W., Rosser, E., Shpeisman, T.: Transitive closure of infinite graphs and its applications. Journal of Parallel Programming 24(6), 579–598 (1996)
28. Klaedtke, F., Rueß, H.: Monadic second-order logics with cardinalities. In: ICALP, LNCS, vol. 2719, pp. 681–696. Springer (2003)
29. La Torre, S., Madhusudan, P., Parlato, G.: Reducing context-bounded concurrent reachability to sequential reachability. In: Bouajjani, A., Maler, O. (eds.) CAV. LNCS, vol. 5643, pp. 477–492. Springer (2009)
30. Lal, A., Reps, T.W.: Reducing concurrent analysis under a context bound to sequential analysis. FMSD 35(1), 73–97 (2009)
31. Lal, A., Touili, T., Kidd, N., Reps, T.W.: Interprocedural analysis of concurrent programs under a context bound. In: TACAS. LNCS, vol. 4963, pp. 282–298. Springer (2008)
32. Lal, A., Reps, T.W.: Reducing concurrent analysis under a context bound to sequential analysis. FMSD 35(1), 73–97 (2009)
33. Lal, A., Touili, T., Kidd, N., Reps, T.W.: Interprocedural analysis of concurrent programs under a context bound. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS. LNCS, vol. 4963, pp. 282–298. Springer (2008)
34. Leroux, J.: Acceleration for petri nets. In: ATVA. LNCS, vol. 8172, pp. 1–4. Springer (2013)
35. Leroux, J., Sutre, G.: Flat counter automata almost everywhere! In: ATVA. LNCS, vol. 3707, pp. 489–503. Springer (2005)
36. Musuvathi, M., Qadeer, S.: Iterative context bounding for systematic testing of multithreaded programs. In: PLDI. pp. 446–455. ACM (2007)
37. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: TACAS. LNCS, vol. 3440, pp. 93–107. Springer (2005)
38. Reps, T., Schwoon, S., Jha, S.: Weighted pushdown systems and their application to interprocedural dataflow analysis. In: SAS. LNCS, vol. 2694, pp. 189–213. Springer (2003)
39. Song, F., Touili, T.: Pushdown model checking for malware detection. STTT 16(2), 147–173 (2014)
40. Torre, S.L., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: TACAS. LNCS, vol. 4963, pp. 299–314. Springer (2008)
41. Torre, S.L., Madhusudan, P., Parlato, G.: Reducing context-bounded concurrent reachability to sequential reachability. In: CAV. LNCS, vol. 5643, pp. 477–492. Springer (2009)
42. Verma, K.N., Seidl, H., Schwentick, T.: On the complexity of equational horn clauses. In: CADE. LNCS, vol. 3632, pp. 337–352. Springer (2005)
43. Wong, K.: Parikh automata with pushdown stack. Diploma thesis, RWTH Aachen (2004)

A Context-Bounding as an Acceleration Problem

A.1 Proof of Lemma 1

This lemma was established in [15, 33]. In the following, we provide a sketch of its proof.

Lemma 1. It is possible to construct, in polynomial time in the size of M_i, an 4-tape finite state automaton $T = (Q_T, Q, \Gamma, Q, \Gamma, \delta_T, q_0', F_T)$, 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 $\pi \in \Delta_i^*$.

Proof. Before going into the proof, we will introduce some notations. Let $\gamma \in \Gamma^*$ be a stack content. We will say $(q, \alpha) \rightarrow_\gamma (q', \beta)$ iff one of the following two cases hold: (1) $(q, \alpha) = (q', \beta)$ and γ is a suffix of α and β, or (2) there are configurations $(q_1, w_1), \cdots, (q_{m+1}, w_{m+1}) \in Cf(M)$ such that $(q, \alpha) = (q_1, w_1)$, $(q', \beta) = (q_{m+1}, w_{m+1})$, γ is a suffix of w_1, w_2, …, w_{m+1}, and $(q_j, w_j) \xrightarrow{t_j}_M (q_{j+1}, w_{j+1})$ for some transition $t_j \in \Delta_i$ for all $j \in [1..m]$.

Let $\mathcal{U} = \{(q, a, q') \mid (q, a) \xrightarrow{\pi}_{M_i} (q', \varepsilon), a \in \Gamma, \pi \in \Delta_i^*\}$ be set such that if $(q, a, q') \in \mathcal{U}$ then there is a run that starts at q, consumes a from stack and reaches q', similarly let $\mathcal{V} = \{(q, a, q') \mid (q, \varepsilon) \xrightarrow{\pi}_{M_i} (q', a), a \in \Gamma, \pi \in \Delta_i^*\}$. Observe the set $\mathcal{U}$ and $\mathcal{V}$ are effectively constructible in polynomial time by reduction to the reachability problem for pushdown systems.

Clearly if $(q, u) \xrightarrow{\pi}_{M_i} (q', v)$ then there is a configuration (q'', w) such that $(q, u) \rightarrow_w (q'', w) \rightarrow_w (q', v)$ i.e. any run can be split into decreasing part and an increasing part, where the decreasing part takes you to a configuration with lowest possible stack height and increasing part starts from this configuration. This can also be seen as $(q, u) \xrightarrow{\pi}_{M_i} (q', v)$ iff $(q, u'w) \rightarrow_w (q'', w) \rightarrow_w (q', v'w)$ for some w such that $u = u'w$ and $v = v'w$. Hence it is easy to see that $L(T) = \{(q, uw, q', vw) \mid (q, u) \rightarrow (q'', \varepsilon) \wedge (q'', \varepsilon) \rightarrow (q', v) \wedge w \in \Gamma^*\}$. We will now give the construction of T. The states of T are $Q_T = \{s\} \cup (Q \times (Q \cup \{\perp\})) \cup \{f\}$. The initial state is s and the final state is f. From the initial state on reading state from the first tape, moves to that state i.e. we add for all $q \in Q$, $(s, (q, \varepsilon, \varepsilon, \varepsilon), (q, \perp)) \in \delta_T$. Subsequently it starts simulating moves of decreasing phase, i.e. we add $((q, \perp), (\varepsilon, a, \varepsilon, \varepsilon), (q', \perp)) \in \delta_T$ if $(q, a, q') \in \mathcal{U}$. It is easy to see that $(q, \perp) \xrightarrow{(\varepsilon, w, \varepsilon, \varepsilon)}_T (q', \perp)$ is a run in T iff $(q, w) \rightarrow_{M_i} (q', \varepsilon)$. We add transition that guesses the intermediate point from where the automata starts simulating the increasing phase from there onwards. To this effect, we add for all $q' \in Q$, $((q, \perp), (\varepsilon, \varepsilon, q', \varepsilon), (q, q')) \in \delta_T$. To simulate the increasing phase, we add for all $(q', a, q'') \in \mathcal{V}$, transition $((q, q''), (\varepsilon, \varepsilon, \varepsilon, a), (q.q')) \in \delta_T$. As in previous case, it is easy to see that $(q, q') \xrightarrow{(\varepsilon, \varepsilon, \varepsilon, v)}_T (q, q'')$ iff $(q'', \varepsilon) \rightarrow_{M_i} (q', v)$. Finally we add for all $q \in Q$, $((q, q), (\varepsilon, \varepsilon, \varepsilon, \varepsilon), f) \in \delta_T$ and for all $a \in \Gamma$, $(f, (\varepsilon, a, \varepsilon, a), f) \in \delta_T$, these transitions guesses the intermediate point and checks if the word below is same in even stacks. The correctness of the construction follows from the fact that, there is an accepting run of the form $s \xrightarrow{(q, \varepsilon, \varepsilon, \varepsilon)}_T (q, \perp) \xrightarrow{(\varepsilon, u, \varepsilon, \varepsilon)}_T (q'', \perp) \xrightarrow{(\varepsilon, \varepsilon, q', \varepsilon)}_T (q'', q') \xrightarrow{(\varepsilon, \varepsilon, \varepsilon, v)}_T (q'', q'') \rightarrow_T f \xrightarrow{(\varepsilon, w, \varepsilon, w)}_T f$ iff $(q, uw) \rightarrow_w (q'', w) \rightarrow_w (q', vw)$

B Accelerating Transition Sequences: Case of regular/rational sets

B.1 Proof of Theorem 2

Theorem 2 There is 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).

Proof. Let us first show that the set of regular languages is not closed wrt. acceleration of simple sequence of transitions. To do that, consider an MPDS $M = (2, \{q, q'\}, \{a, b\}, \Delta)$ where Δ only contains the following two transitions $(q, push_1(a), q')$ and $(q', push_2(b), q)$. Let $\theta = (q, push_1(a), q')(q', push_2(b), q)$. It is easy to see that $Post_{\theta^*}(\{(q, \varepsilon, \varepsilon)\}) = \{(q, a^i, b^i) \mid i \in \mathbb{N}\}$ which is not regular.

Now let us show the non-closure of rational languages. Consider an MPDS $M = (2, \{q, q'\}, \{a, b\}, \Delta)$ where Δ only contains the following two rules $(q, push_1(b), q')$ and $(q', pop_2(a), q)$. Let $\theta = (q, push_1(b), q')(q', pop_2(a), q)$ and $C = \{(q, a^i, a^i) \mid i \in \mathbb{N}\}$. Observe that C is a rational set of configurations. It is easy to see that $Post_{\theta^*}(C) = \{(q, b^j a^i, a^{i-j}) \mid 0 \leq j \leq i\}$ which is not rational.

B.2 Proof of Lemma 2

Lemma 2. Let $c = (p, w_1, \ldots, w_n)$ and $c' = (p', w'_1, \ldots, w'_n)$ 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 $\texttt{Summ}(i, \sigma) = (u_i, v_i)$.

Proof. We will prove this lemma by induction on the length of the run and for a fixed i. We will assume that the sequence of transition considered is state-wise compatible.

Base case $|\sigma| = 0$: For the base case we consider the zero length run. $\texttt{Summ}(i, \varepsilon)$ is defined as $\texttt{Eff}((\varepsilon, \varepsilon), \varepsilon) = (\varepsilon, \varepsilon)$. Hence holds trivially.

Length greater than 0, case $\sigma.\tau$:

($\Rightarrow$)

Let $c \xrightarrow{\sigma} c'' \xrightarrow{\tau} c'$ with $c = (p, w_1, \ldots, w_n)$, $c'' = (p'', w''_1, \cdots, w''_n)$ and $c' = (p', w'_1, \ldots, w'_n)$. By induction, we have $\texttt{Summ}(i, \sigma) = (u_i, v_i)$, with $w_i = u_i u'_i$ and $w''_i = v_i u'_i$.

- case where $\tau \notin \Delta_i$ or $Op(\tau = nop_i)$ then we have nothing to prove.
- case where $Op(\tau) = push_i(a)$, clearly $w'_i = a.w''_i$ hence we have that $w'_i = a.v_i.u'_i$. Further, by definition we have that $\texttt{Summ}(i, \sigma.\tau) = (u_i, a.v_i)$.
- case where $Op(\tau) = pop_i(a)$, clearly $a.w'_i = w''_i$. We have two cases to consider
  1) where $v_i = \varepsilon$ in this case, $w''_i = u'_i = a u''_i$. But notice that $w_i = u_i.u'_i = u_i.a.u''_i$. Also by definition we get $\texttt{Summ}(i, \sigma.\tau) = (u_i a, \varepsilon)$.
  2) where $v_i \neq \varepsilon$, in this case we have $a w'_i = w''_i = v_i u'_i$, hence v_i is of the form $v_i = a v''_i$ and $w'_i = v''_i u_i$. Further by definition, we have that $\texttt{Summ}(i, \sigma\tau) = (u_i, v''_i)$.

($\Leftarrow$)

Let $\texttt{Summ}(i, \sigma) = (u_i, v_i)$, we will assume by induction existence of a run $c \xrightarrow{\sigma} c''$ such that $c = (p, w_1, \ldots, w_n)$, $c'' = (p'', w''_1, \cdots, w''_n)$ with $w_i = u_i u'_i$ and $w''_i = v_i u'_i$.

– case where $\tau \notin \Delta_i$ or $Op(\tau = nop_i)$ is straight forward.
– case where $Op(\tau) = push_i(a)$ then $\texttt{Summ}(i, \sigma\tau) = (u_i, av_i)$. Clearly $c_i'' \xrightarrow{\tau} c_i'$ where $c_i' = (p', w_1', \ldots, w_n')$. Further by definition of $\xrightarrow{\tau}$, we have that $w_i' = aw_i'' = av_iu_i'$
– case where $Op(\tau) = pop_i(a)$ and $v_i = \varepsilon$ then $\texttt{Summ}(i, \sigma\tau) = (u_i.a, \varepsilon)$. We have by induction hypothesis, $w_i = u_iau_i'$ and $w_i'' = au_i'$. Hence $c_i'' \xrightarrow{\tau} c_i'$ where $aw_i' = w_i'' = au_i'$
– case where $Op(\tau) = pop_i(a)$ and $v_i \neq \varepsilon$ then $\texttt{Summ}(i, \sigma\tau) = (u_i, v_i')$, clearly $v_i' = av_i$ and hence $w_i'' = av_iu_i'$. Now since we have an a on top of stack, we have $c'' \rightarrow c'$, with $w_i = v_i'u_i'$.

B.3 Proof of Lemma 3

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, $\texttt{Summ}(i, \sigma^j) = (u_ix_i^{j-1}, v_iy_i^{j-1})$ with $\texttt{Summ}(i, \sigma) = (u_i, v_i)$ and $\texttt{Iter}(i, \sigma) = (x_i, y_i)$.*

Proof. Firstly we will assume that $\texttt{Summ}(i, \theta)$ is well-defined. Given any summary sequence of transitions, $\sigma = \tau_1 \cdots \tau_m$, we define a set $\texttt{Nest}(\sigma) \subseteq [1..m]$ as follows. For any $j \in [1..m-1]$ and for any $a \in \Gamma$, if $\tau_j = (q_1, pop_i(a), q_2)$ and $\tau_{j+1} = (q_3, push_i(a), q_4)$ for some $q_1, q_2, q_3, q_4 \in Q$, then $j, j+1 \in \texttt{Nest}(\sigma)$. For any $k, j \in [1..m]$, if $k+1 \in \texttt{Nest}(\sigma)$ and $j-1 \in \texttt{Nest}(\sigma)$ and $\tau_k = (q_1, pop_i(a), q_2)$ and $\tau_j = (q_3, push_i(a), q_4)$ for some $q_1, q_2, q_3, q_4 \in Q$, then $k, j \in \texttt{Nest}(\sigma)$. Clearly $\texttt{Nest}(\sigma)$ captures all the intermediate positions of σ that are well matched. We define for all $i \in [1..m-1]$, if $i \in Nest(\sigma)$ or $Op(\tau_i) = nop_i$ then $\texttt{Unmatched}(i, \sigma) = \texttt{Unmatched}(i+1, \sigma)$, $\texttt{Unmatched}(i, \sigma) = \tau_i.\texttt{Unmatched}(i+1, \sigma)$ otherwise and for $i = m$, if $i \in Nest(\sigma)$ then $\texttt{Unmatched}(m, \sigma) = \varepsilon$, $\texttt{Unmatched}(i, \sigma) = \tau_i$ otherwise. Clearly $\texttt{Unmatched}(1, \sigma)$ gives us set of transitions in σ that are not well matched. Given a transition τ,we will use $\Gamma(\tau)$ to return the stack alphabet of the operation.

Given a sequence of transitions say $\tau_1\tau_2 \cdots \tau_n$, we say they are state wise compatible iff target state of τ_i is same as the source state of τ_{i+1}. The following lemma states that in any sequence of transition, $\texttt{Summ}(i, \theta)$ is well defined iff barring the transitions that are well-matched, all transitions that pop elements from stack occur before ones that push into the stack. This is directly follows from the definition of $\texttt{Summ}$.

Lemma 13. *Given a sequence of transitions θ that are state wise compatible, $\texttt{Summ}(i, \theta)$ is well defined iff $\texttt{Unmatched}(1, \theta|_{\Delta_i}) \in (Q \times \cup_{a \in \Gamma} pop_i(a) \times Q)^*.(Q \times \cup_{a \in \Gamma} push_i(a) \times Q)^*$. Let $\texttt{Unmatched}(1, \theta|_{\Delta_i}) = \alpha_i.\beta_i$ where $\alpha_i \in (Q \times \cup_{a \in \Gamma} pop_i(a) \times Q)^*$ and $\beta_i \in (Q \times \cup_{a \in \Gamma} push_i(a) \times Q)^*$. Infact we can easily show that $\texttt{Summ}(i, \theta) = (\Gamma(\alpha_i), \Gamma(\beta_i)^R)$*

Proof. ($\Rightarrow$) We will induct on the length of the transition sequence θ. If $|\theta| = 0$ then there is nothing to prove. For the induction case, we will assume that $\theta = \sigma.\tau$, let $\texttt{Summ}(i, \sigma) = (u_i, v_1)$ and let $\texttt{Unmatched}(1, \theta|_{\Delta_i}) = \alpha\beta \in (Q \times \cup_{a \in \Gamma} pop_i(a) \times Q)^*.(Q \times \cup_{a \in \Gamma} push_i(a) \times Q)^*$. We will further assume that $\alpha = \tau_{j_1}\tau_{j_2} \cdots \tau_{j_n}$ and $\beta = \tau'_{j_1}\tau'_{j_2} \cdots \tau'_{j_m}$.

– case there $Op(\tau) = push_i$ or $Op(\tau) = nop_i$ is simple and straight forward.

– case where $Op(\tau) = pop_i$, if $\beta = \varepsilon$, it is easy to see that $\texttt{Unmatched}(1, \sigma\tau|_{\Delta_i}) = \alpha\tau$ and clearly $\alpha\tau \in (Q \times \cup_{a\in\Gamma} pop_i(a) \times Q)^*$ and $\texttt{Summ}(i, \sigma\tau) = \Gamma(\alpha\tau)$. If β is not ε then clearly τ matches with τ'_{j_m}, hence $\texttt{Unmatched}(1, \sigma\tau|_{\Delta_i}) = \alpha\beta'$ where $\beta' = \tau^i_{j_1}\tau^i_{j_2}\cdots\tau^i_{j_{m-1}}$. This is clearly in the required form.

($\Rightarrow$)

Again we will show by induction on length of transition sequence θ. If $|\theta| = 0$ then there is nothing to prove. For the induction case, we will assume that $\theta = \sigma.\tau$, let $\texttt{Unmatched}(1, \sigma|_{\Delta_i}) = \alpha\beta$. By induction, we have $\texttt{Summ}(i, \sigma) = (u_i, v_1)$ with $u_i = \Gamma(\alpha)$ and $v_i = \Gamma(\beta)^R$

– case there $Op(\tau) = push_i$ or $Op(\tau) = nop_i$ is simple and straight forward.
– case there $Op(\tau) = pop_i$ and $\beta = \varepsilon$ is straight forward.
– case there $Op(\tau) = pop_i$ and $\beta \neq \varepsilon$ is straight forward. Let $\Gamma(\tau) = a$, then $\Gamma(\beta) = v_i'^R.a$, which means $v_i = av'_i$. From this and definition of $\texttt{Summ}$, we get $\texttt{Summ}(i, \sigma\tau) = (u_i, v'_i)$

It is also easy to see that $\texttt{Summ}(i, \theta.\theta)$ is well defined iff $\texttt{Unmatched}(1, \theta.\theta|_{\Delta_i}) \in (Q \times \cup_{a\in\Gamma} pop_i(a) \times Q)^*.(Q \times \cup_{a\in\Gamma} push_i(a) \times Q)^*$. But $\texttt{Unmatched}(1, \theta.\theta|_{\Delta_i}) = \texttt{Unmatched}(1, \alpha_i.\beta_i.\alpha_i.\beta_i)$. Hence we have that either $\Gamma(\alpha_i)$ is prefix of $\Gamma(\beta_i)^R$ or $\Gamma(\beta_i)^R$ is a prefix of $\Gamma(\alpha_i)$.

If $\Gamma(\alpha_i)$ is prefix of $\Gamma(\beta_i)$ then $\alpha_i.\beta_i.\alpha_i.\beta_i = \alpha_i.\beta^1_i.\beta^2_i.\alpha_i.\beta_i$ where $\Gamma(\beta^2_i)^R = \Gamma(\alpha_i)$. clearly $\beta^2_i.\alpha_i$ is well matched hence we have $\texttt{Unmatched}(1, \alpha_i.\beta_i.\alpha_i.\beta_i) = \alpha_i.\beta^1_i.\beta_i$ and $\texttt{Summ}(i, \theta.\theta) = (u_i, v_i.x_i)$, where $x_i = \Gamma(\beta^1_i)^R$. In general, it is easy to see that $(\alpha_i.\beta_i)^j = (\alpha_i.\beta^1_i.\beta^2_i)^{j-1}.\alpha_i.\beta_i$ with $\texttt{Unmatched}(1, (\alpha_i.\beta_i)^j) = (\alpha_i.(\beta^1_i)^{j-1}.\beta_i)$ and $\texttt{Summ}(i, \theta^j) = (u_i, v_i.x_i^{j-1})$

Similarly, If $\Gamma(\beta_i)$ is prefix of $\Gamma(\alpha_i)$ then $\alpha_i.\beta_i.\alpha_i.\beta_i = \alpha_i.\beta_i.\alpha^1_i.\alpha^2_i.\beta_i$ where $\Gamma(\beta_i)^R = \Gamma(\alpha^1_i)$ and $\beta_i.\alpha^1_i$ is well matched. Hence $\texttt{Unmatched}(1, \alpha_i.\beta_i.\alpha_i.\beta_i) = \alpha_i.\alpha^2_i.\beta_i$, in general we have $(\alpha_i.\beta_i)^j = \alpha_i.\beta_i.(\alpha^1_i.\alpha^2_i.\beta_i)^{j-1}$. Hence the result follows.

B.4 Proof of Lemma 4

Lemma 4. Let $j \geq 1$ and $c = (p, w_1, \ldots, w_n)$ and $c' = (p', w'_1, \ldots, w'_n)$ 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 are defined as above.

Proof. The proof follows directly from lemma-3 and lemma-2. We will prove this inducting on j

base case $j = 1$: Base case directly follows from lemma-2 which states, $c \xrightarrow{\theta} c'$ with $w_i = u_i w''_i$ and $w'_i = v_i w''_i$ iff $\texttt{Summ}(i, \theta) = (u_i, v_i)$.

case $j \geq 1$: By lemma-3 if $\texttt{Summ}(i, \theta\theta)$ is defined then $\texttt{Summ}(i, \theta^j) = (u_i x_i^{j-1}, v_i y_i^{j-1})$ is defined for all $j > 1$, where $(x_i, y_i) = \texttt{Iter}(i, \theta)$. Now applying lemma-2, we get that $c \xrightarrow{\theta^j} c'$ iff for every $i \in [1..n]$, we have $w_i = u_i x_i^{j-1} w''_i$ and $w'_i = v_i y_i^{j-1} w''_i$ such that $\texttt{Summ}(i, \theta^j) = (u_i x_i^{j-1}, v_i y_i^{j-1})$.

B.5 Presburger Arithmetic

In the following, we present a technical lemma that states that certain tuples of natural numbers defined in terms of iterations of words is definable in Presburger arithmetic. Before that let us recall some definitions about pushdown automata.

A pushdown automaton. A pushdown automaton $\mathcal{P}$ is a tuple $(Q,\Sigma,\Gamma,\Delta,q_0,F)$ where Q is a finite set of states, Σ is a finite alphabet, Γ is the stack alphabet, $\Delta \subseteq Q \times \Gamma^* \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$, $q_0 \in Q$ is the initial state, and F is the set of accepting states. A configuration of a pushdown automaton is defined by a pair (q,w) where $q \in Q$ is a state and $w \in \Gamma^*$ is the stack content.

Given two configuration (q,w) and (q',w') of $\mathcal{P}$ and a transition $t = (q,u,a,v,q') \in \Delta$, we use $(q,w) \xrightarrow{t}_{\mathcal{P}} (q',w')$ to denote that there is a word $w'' \in \Gamma^*$ such that $w = uw''$ and $w' = vw''$.

Let $\sigma \in \Delta^*$. Let c and c' be two configurations of $\mathcal{P}$. Then, we use $c \xrightarrow{\sigma}_{\mathcal{P}} c'$ to denote that one of the following two cases holds: (1) $\sigma = \varepsilon$ and $c = c'$, or (2) there is a sequence of letters $t_1,t_2,\ldots,t_n \in \Delta$ and a sequence of configurations $c_0,c_1,\ldots,c_n$ such that the following conditions are satisfied: (1) $c_0 = c$, (2) $c_n = c'$, and (3) $c_i \xrightarrow{t_{i+1}}_{\mathcal{P}} c_{i+1}$ for all $i \in [0..n-1]$.

We define the function $\mathit{Label} : \Delta \to \Sigma \cup \{\varepsilon\}$ that associate for each transition the letter that is labelling it: Formally, given a transition $t = (q,u,a,v,q') \in \Delta$, we set $\mathit{Label}(t) = a$.

A word $w \in \Sigma^*$ is accepted by the pushdown automaton if and only if $(q_0,\varepsilon) \xrightarrow{\sigma}_{\mathcal{P}} (q,\varepsilon)$ for some $q \in F$ and $\sigma \in \Delta^*$ such that $\mathit{Label}(\sigma) = w$.

The language accepted by the pushdown automaton $\mathcal{P}$, denoted $L(\mathcal{P})$ is defied as the set if words accepted by $\mathcal{P}$.

Given a language L, we use $\mathit{Parikh}(L)$ to denote the Parikh image of L, i.e. $\mathit{Parikh}(L) = \{\mathit{Parikh}(\sigma) \mid \sigma \in L\}$. In the following we recall a result from [42].

Proposition 1. *Given a pushdown automaton $\mathcal{P} = (Q,\Sigma,\Gamma,\Delta,q_0,F)$, it is possible to construct, in polynomial time in the size of $\mathcal{P}$, an existential Presburger formula ϕ such that $\mathit{var}(\phi) = \Sigma$ and $\mathit{Parikh}(L(\mathcal{P})) = \{\mu \mid \mu \models \phi\}$.*

Now we are ready to state and prove a technical lemma that states that certain tuples of natural numbers defined in terms of iterations of words is definable in Presburger arithmetic.

Lemma 14. *Given two sequences of words $v_1,\ldots,v_\ell$ and $u_1,\ldots,u_k$ over Γ, it is possible to construct, in polynomial time in $|v_1v_2\ldots v_\ell u_1u_2\ldots u_k|$, an existential Presburger formula $\varphi(x_1,x_2,\ldots,x_\ell,y_1,y_2,\ldots,y_k)$ such that $\mu \models \varphi$ iff $v_1^{\mu(x_1)}\cdots v_\ell^{\mu(x_\ell)} = u_1^{\mu(y_1)}\cdots u_k^{\mu(y_k)}$.*

Proof. To prove this lemma, we we will reduce the problem to computing an existential Presburger formula recognizing the Parikh image of a pushdown automaton $\mathcal{P} = (Q,\Sigma,\Gamma,\Delta,q_0,F)$. Then, we will use Proposition 1 to show that such an existential Presburger formula can be constructed in polynomial time.

The pushdown automaton $\mathcal{P}$ is defined as follows. The set of states Q is defined as the set $\{q_0,q_1,\ldots,q_\ell,q_{\ell+1},q_{\ell+2}\ldots,q_{\ell+k}\}$ with $F = \{q_{\ell+k}\}$. The input alphabet Σ is defined by the set of variables $x_1,x_2,\ldots,x_\ell$ and $y_1,y_2,\ldots,y_k$. The pushdown

works in phases: In the first phase, the pushdown automaton is at the state q_0 and can push the reverse of the word v_1 (denoted v_1^R) into its stack while outputting x_1 (i.e., $(q_0, \varepsilon, x_1, v_1^R, q_0) \in \Delta$). In nondeterministic manner the pushdown can decide to move to the simulation of the second phase by moving its state from q_0 to q_1 (i.e., $(q_0, \varepsilon, \varepsilon, \varepsilon, q_1) \in \Delta$).

In a phase i, with $1 < i \leq \ell$, the pushdown automaton $\mathcal{P}$ is at the state q_{i-1} and and can push the reverse of the word v_i (denoted v_i^R) into its stack while outputting x_i (i.e., $(q_{i-1}, \varepsilon, x_i, v_i^R, q_{i-1}) \in \Delta$). In nondeterministic manner the pushdown can decide to move to the simulation of the second phase by moving its state from q_{i-1} to q_i (i.e., $(q_{i-1}, \varepsilon, \varepsilon, \varepsilon, q_i) \in \Delta$).

In the phase $\ell + 1$, the pushdown automaton is at the state q_ℓ and can start popping the reverse of the word u_k (denoted u_k^R) from its stack while outputting y_k (i.e., $(q_\ell, u_k^R, x_i, \varepsilon, q_\ell) \in \Delta$). In nondeterministic manner the pushdown can decide to move to the simulation of the next phase $\ell + 2$ by moving its state from q_ℓ to $q_{\ell+1}$ (i.e., $(q_\ell, \varepsilon, \varepsilon, \varepsilon, q_{\ell+1}) \in \Delta$).

In a phase i, with $\ell + 1 < i \leq \ell + k$, the pushdown automaton $\mathcal{P}$ is at the state q_{i-1} and and can pop the reverse of the word $u_{k+\ell+1-i}$ (denoted $u_{k+\ell-i}^R$) from its stack while outputting $y_{k+\ell+1-i}$ (i.e., $(q_{i-1}, u_{k+\ell+1-i}^R, y_{k+\ell+1-i}, \varepsilon, q_{i-1}) \in \Delta$). In nondeterministic manner the pushdown can decide to move to the simulation of the next phase by moving its state from q_{i-1} to q_i (i.e., $(q_{i-1}, \varepsilon, \varepsilon, \varepsilon, q_i) \in \Delta$).

Finally, the set Δ is defined as the smallest set satisfying the above condition.

Let φ be the formula recognizing the Parikh image of the pushdown automaton $\mathcal{P}$ constructed as stated in Proposition 1. Then it is easy to see that $\mu \models \varphi$ iff $v_1^{\mu(x_1)} \cdots v_\ell^{\mu(x_\ell)} = u_1^{\mu(y_1)} \cdots u_k^{\mu(y_k)}$.

C Acceleration of Constrained Simple Regular Expressions

C.1 Proof of Lemma 5

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.

Proof. Let us assume two CSRE $e_1 = (v_1, v_2, \ldots, v_\ell, \varphi_1(x_1, x_2, \ldots, x_\ell))$ and $e_2 = (u_1, u_2, \ldots, u_k, \varphi_2(y_1, y_2, \ldots, y_k))$. We assume w.l.o.g. that $x_i \neq y_j$ for all i and j.

- **[Concatenation]** Let $e_1 \cdot e_2$ be the CSRE expression defined by the tuple $(v_1, v_2, \ldots, v_\ell, u_1, \ldots, u_k, \varphi(x_1, \ldots, x_\ell, y_1, \ldots, y_k))$ where $\varphi = \varphi_1(x_1, \ldots, x_\ell) \wedge \varphi_2(y_1, \ldots, y_k)$. It is easy to see that $L(e_1 \cdot e_2) = L(e_1) \cdot L(e_2)$ and that the size of $e_1 \cdot e_2$ is linear in $|e_1| + |e_2|$.
- **[Union]** Let $e_1 + e_2$ be the CSRE expression defined by the tuple $(v_1, v_2, \ldots, v_\ell, u_1, \ldots, u_k, \varphi(x_1, \ldots, x_\ell, y_1, \ldots, y_k))$ such that $\varphi = (\varphi_1(x_1, \ldots, x_\ell) \wedge \wedge_{1 \leq j \leq k} y_j = 0) \vee (\varphi_2(y_1, \ldots, y_k) \wedge \wedge_{1 \leq i \leq \ell} x_i = 0)$ It is easy to see that $L(e_1 + e_2) = L(e_1) \cup L(e_2)$ and that the size of $e_1 + e_2$ is linear in $|e_1| + |e_2|$.

– **[Intersection]** We show there is a CSRE accepting $L(e_1) \cap L(e_2)$. Let $e_1 = (v_1, \ldots, v_\ell, \varphi_1(x_1, \ldots, x_k))$ and $e_2 = (u_1, \ldots, u_k, \varphi_2(y_1, \ldots, y_k))$ with $x_i \neq y_j$ for i, j. Let $\varphi_3(x_1, \ldots, x_\ell, y_1, \ldots, y_k)$ be the existential Presburger formula obtained by the application of Lemma 14 to the two sequences of words $v_1, \ldots, v_\ell$ and $u_1, \ldots, u_k$. Then the CSRE expression $e_1 \cap e_2$ defined to be $(v_1, \ldots, v_\ell, \varphi(x_1, \ldots, x_\ell))$ with $\varphi = \exists y_1.\exists y_2. \ldots .\exists y_n.\varphi_3(x_1, \ldots, x_\ell, y_1, \ldots, y_k) \wedge \varphi_1(x_1, \ldots, x_\ell) \wedge \varphi_2(y_1, \ldots, y_k)$ defines the intersection of the languages defined by e_1 and e_2 and that the size of $e_1 \cap e_2$ is linear in $|e_1| + |e_2|$.

  Let us know prove the second of the lemma concerning the decision problem.
  The emptiness and membership can be easily shown to be decidable and NP-complete. A lower bound for these two problem can be obtained by a straightforward reduction from the satisfiability problem of the class of existential Presburger formula.

– **[Inclusion]** Let us now show that the inclusion problem is also decidable. The lemma is an immediate corollary of the following three results: (1) for any CSRE expression e, it is possible to effectively construct a bounded Parikh automaton accepting $L(e)$ (see [14] for the bounded Parikh automaton definition), (2) for any bounded Parikh automaton, it is possible to construct a bounded deterministic automaton recognizing the same language (see Theorem 11, Corollary 12 and Section 5.3 in [14] for the bounded Parikh automaton definition), and (3) the class of deterministic Parikh automata is closed under intersection and complement and its emptiness problem is decidable (see [13]).

C.2 Proof of Lemma 6

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 $e_i = (\varepsilon, u_1, u_2, \ldots, u_k, (y = i \wedge \varphi(y, y_1, y_2, \ldots, y_k)))$.

Proof. Let us assume that the CSRE e is of the form $(w_1, w_2, \ldots, w_n, \phi(x_1, x_2, \ldots, x_n))$. Prefixes of the form w^j may end in the middle of an occurance of some w_i and so, in order to compute left-quotients, it is useful to expand this CSRE into union of a set of more elaborate ones, taking into account the position of such splits. For every $i \in [1..n]$ and $u, v \in \Sigma^*$ such that $w_i = uv$, we define the CSRE $sp(i, u, v)$ as follows:

$$(w_1, \ldots, w_i, u, v, w_i, w_{i+1}, \ldots, w_n, \phi \wedge \psi_{(i,u,v)}(x_1, \ldots, x'_i, x_u, x_v, x''_i, x_{i+1}, \ldots, x_n))$$

where

$$\psi_{i,u,v} = (x_i = x'_i + x''_i + x_u + x_v) \wedge ((x_u = x_v = 0) \vee (x_u = x_v = 1))$$

We note that the size of $\psi_{i,u,v}$ is constant. It is easy to see that $L(e) = \bigcup_{i \in [1..n]} \bigcup_{w_i = uv} L(sp(i, u, v))$.

Next, we use Lemma 14 to obtain a Presburger formula $\varphi'_{(i,u,v)}(y,x_1,\ldots,x_i,x_u)$ such that $\mu \models \varphi_{(i,u,v)}(y,x_1,\ldots,x_i,x_u)$ iff $w_1^{\mu(x_1)}w_2^{\mu(x_2)}\ldots w_{i-1}^{\mu(x_{i-1})}w_i^{\mu(x_i)}u^{\mu(x_u)} = w^{\mu(y)}$. We modify the formula to also insist that $x_u = 1$, and refer to the resulting formula as $\varphi_{(i,u,v)}$. The size of $\varphi_{(i,u,v)}$ is polynomial in $|w_1\ldots w_iu| + |w|$. Now, we combine the CSRE $sp(i,u,v)$ with the formula $\varphi_{(i,u,v)}$ as follows: Let $sp'(i,u,v)$ be the following CSRE defined

$$(w,v,w_i,w_{i+1},\ldots,w_n,\exists x_1\exists x_2\ldots\exists x_i\exists x_u.\ \phi\wedge\psi_{(i,u,v)}\wedge\varphi_{(i,u,v)})$$

The free variables of the formula in $sp'(i,u,v)$, in order, are $y,x_v,x''_i,x_{i+1},\ldots,x_n$ ($n-i+3$ in all).

Notice that this CSRE $sp'(i,u,v)$ defines the language consisting of words of the form $w^{\mu(y)}vw_i^{\mu(x''_i)}w_{i+1}^{\mu(x_{i+1})}\ldots w_n^{\mu(x_n)}$ such that there are $\mu(x_1),\mu(x_2)\ldots\mu(x_i),\mu(x_u)\in\mathbb{N}$ such that 1) $w^{\mu(y)} = w_1^{\mu(x_1)}w_2^{\mu(x_2)}\ldots w_i^{\mu(x_i)}u$ and 2) $w_1^{\mu(x_1)}w_2^{\mu(x_2)}\ldots w_i^{\mu(x_i)}u.v.w_i^{\mu(x''_i)}w_{i+1}^{\mu(x_{i+1})}\ldots w_n^{\mu(x_n)} \in L(sp(i,u,v))$. Further, the sum of the length of the words in $sp'(i,u,v)$ is linear in the sum of the lengths of the words in e plus the $|w|$, the number of variables increases at most by a constant, and the size of the formula is $|\phi|$ plus the size of $|\varphi_{i,u,v}| + |\psi_{i,u,v}|$ (it increases additively by a polynomial in $|w_1\ldots w_iu| + |w|$.)

Next we show how to combine the various CSRE's $sp'(i,u,v)$, $1\le i\le n$, $uv = w_i$ into a single CSRE with the desired property. We describe this for the case of two such CSREs and the generalization to a collection is similar.

Given $sp'_{(i,u,v)}$ and $sp'_{(i',u',v;)}$ we first relabel all the variables in the Presburger formulas of the two CSREs to be disjoint leaving only the variable y as it is. So, let their respective free variables be $y,p_1,p_2,\ldots p_{n-i+2}$ and $y,q_1,q_2,\ldots,q_{n-i'+2}$. The resulting CSRE is then given by

$$(w,v,w_i,w_{i+1},\ldots,w_n,v',w_{i'},\ldots,w_n,\chi(y,p_1,p_2,\ldots,p_{n-i+2},q_1,q_2,\ldots,q_{n-i'+2}))$$

where χ is

$$(\phi\wedge\psi_{i,u,v}\wedge\varphi_{(i,u,v)}\wedge(\bigwedge_{j=1}^{n-i'+2} q_j = 0))\vee(\phi\wedge\psi_{i',u',v'}\wedge\varphi_{(i',u',v')}\wedge(\bigwedge_{j=1}^{n-i+2} p_j = 0))$$

$$= \phi\wedge((\psi_{i,u,v}\wedge\varphi_{(i,u,v)}\wedge(\bigwedge_{j=1}^{n-i'+2} q_j = 0))\vee(\psi_{i',u',v'}\wedge\varphi_{(i',u',v')}\wedge(\bigwedge_{j=1}^{n-i+2} p_j = 0))$$

This describes the language $L(sp'_{(i,u,v)})\cup L(sp'_{(i',u',v')})$.

We can easily generalize this to combine all the $sp'_{(i,u,v)}$s using a disjoint set of variables (except y) for each of them in the similar manner. Let the sum of the length of the words $w_1,\ldots,w_n$ in e be K. The resulting expression:

1. has a variable y whose value may be set to the number of iterations of w by which we desire to left quotient.
2. whose sum of length of the words is polynomial in $K+|w|$

3. whose number of variables increases by a polynomial in $K + |w|$
4. whose formula has increased in length w.r.t. to ϕ by an addition whose size is polynomial in $K + |w|$.

This completes the proof of the lemma.

C.3 Proof of Lemma 7

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.

Proof. The proofs are similar to the ones of Lemma 5 and we only show the closure under concatenation here. Consider two n-CSREs $\mathbf{e_1} = ((u_1,\ldots,u_{\ell_1}),\ldots,(u_{\ell_{n-1}+1},\ldots,u_{\ell_n}),\varphi_1(x_1,\ldots,x_{\ell_n}))$ and $\mathbf{e_2} = ((v_1,\ldots,v_{k_1}),\ldots,(v_{k_{n-1}+1},\ldots,v_{k_n}),\varphi_2(y_1,\ldots,y_{k_n}))$ with $x_i \neq y_j$ for i,j. We will show that there is an CSRE accepting the language $L(\mathbf{e_1}) \cdot L(\mathbf{e_2})$. Let $\mathbf{e_1}\cdot\mathbf{e_2}$ be the CSRE expression defined by the tuple $((u_1,\ldots,u_{\ell_1},v_1,\ldots,v_{k_1}),\ldots,(u_{\ell_{n-1}+1},\ldots,u_{\ell_n},u_{k_{n-1}+1},\ldots,u_{k_n}),\varphi(x_1,\ldots,x_{k_1},$ $y_1,\ldots y_{\ell_1},x_{k_1+1},\ldots,x_{k_2},y_{\ell_1+1},\ldots,y_{\ell_2},\ldots,x_{k_n},y_{\ell_{n-1}+1},\ldots,y_{\ell_n}))$ where $\varphi = \varphi_1(x_1,\ldots,x_{k_n}) \wedge \varphi_2(y_1,\ldots,y_{\ell_n})$. It is easy to see that $L(\mathbf{e_1}\cdot\mathbf{e_2}) = L(\mathbf{e_1})\cdot L(\mathbf{e_2})$.

We will show that it is possible to reduce emptiness, membership and inclusion problems to their corresponding ones for CSRE. The idea is to encode the n-dim CSRE as a CSRE using a special symbol $\sharp$ (not appearing in the alphabet) to separate all the different tapes. To that aim, we define the function $Encode$ that maps any n-dim CSRE to an CSRE. Let $\mathbf{e} = ((u_1,\ldots,u_{k_1}),\ldots,(u_{k_{n-1}+1},\ldots,u_{k_n}),\varphi(x_1,\ldots,x_{k_n}))$ be an n-dim CSRE over the alphabet Σ. We assume that $\sharp \notin \Sigma$. Then we define $Encode(\mathbf{e})$ to be the CSRE $(u_1,\ldots,u_{k_1},\sharp,\ldots,\sharp,u_{k_{n-1}+1},\ldots,u_{k_n},\varphi'(x_1,\ldots,x_{k_1},y_1,x_{k_1+1},\ldots,x_{k_2},y_2,\ldots,$ $y_{n-1},x_{k_{n-1}+1},\ldots,x_{k_n}))$ where $\varphi' = \varphi(x_1,\ldots,x_{k_n}) \wedge \wedge_{1\leq i\leq n-1} y_i = 1$. It is easy to see that, for any n-dim word $(w_1,\ldots,w_n)$ and any n-dim CSREs $\mathbf{e}$ we have $(w_1,\ldots,w_n) \in L(\mathbf{e_1})$ iff $w_1\sharp\cdots\sharp w_n \in L(Encode(\mathbf{e_1}))$. Hence, $L(\mathbf{e_1}) = \emptyset$ iff $L(Encode(\mathbf{e_1})) = \emptyset$ and $L(\mathbf{e_1}) \subseteq L(\mathbf{e_2})$ iff $L(Encode(\mathbf{e_1})) \subseteq L(Encode(\mathbf{e_2}))$. Thus, the emptiness, membership and containment problems are decidable.

C.4 Proof of Lemma 8

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 $e[\mathbf{w},j] = (\mathbf{u}_1[1 \leftarrow \varepsilon],\ldots,\mathbf{u}_n[1 \leftarrow \varepsilon],(\bigwedge_{1\leq i\leq n}\mathbf{x}_i[1] = j \wedge \varphi(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n)))$.

Proof. (sketch) First suppose, we only quotient the 1st component by iterations of a w_1. We can do this by simply following the proof of Lemma 6 almost identically. In the resulting n-CSRE, the increase in size is according to the items 2, 3 and 4 listed at the

end of that proof. The increase is additive, increasing by a polynomial in the sum of the lengths of the words in $\mathbf{u_1}$ and $|w_1$, and provides a variable y_1 (as indicated in item 1) to control the number of iterations of w_1 by which the first component is to be quotiented.

Repeating this now for the second component, again following the same line as in the proof of Lemma 6 will result in an additive increase in size by a polynomial in the sum of the lengths of the words in $\mathbf{u_2}$ and $|w_2|$ and provide a variable y_2 to control the iterations of w_2 and so on. After n steps on is left with a CSRE which has only grown polynomially in $|\mathbf{u}_1|+\ldots|\mathbf{u}_n|+|w_1|+\ldots+|w_n|$.

We can then existentially quantify the $y_2 \ldots y_n$, add a conjunct of the form $y_1 = y_2 \wedge y_1 = y_3 \ldots y_1 = y_n$ and drop the empty words corresponding to the positions referred to by $y_2, \ldots, y_n$ to arrive at the desired formula.

C.5 Proof of Theorem 5

Theorem 5

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

Proof. Consider an PDS $M = (1, \{q\}, \{a,b\}, \Delta)$ where Δ only contains the following two rules $(q, push_1(b), q)$ and $(q, push_1(a), q)$. It is easy to see that M pushes any word in $(a+b)^*$ into its stack. Let $C = \{(q,a)\}$. Observe that C is a CSRE representable sets of configurations. It is easy to see that $Post_{\theta^*}(C) = \{(q, wa) \mid w \in (a+b)^*\}$ which is not CSRE representable set. □

D Acceleration of Bounded-Context Sets

D.1 Proof of Lemma 9

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.*

Proof. We will now prove the closures w.r.t. constrained rational languages.

- **Union:** Let $\mathcal{A} = (A, \varphi_1)$ and $\mathcal{B} = (B, \varphi_2)$ be two constrained rational automata such that $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$ and $B = (Q^B, \Sigma_1, \cdots, \Sigma_n, \delta^B, q_0^B, F^B)$. Without loss of generality, we will assume that $Q^A \cap Q^B = \emptyset$ and hence $\delta^A \cap \delta^B = \emptyset$. We construct the constrained rational automaton $\mathcal{C} = (C, \varphi)$ such that $L(\mathcal{C}) = L(\mathcal{A}) \cup L(\mathcal{B})$ as follows: $C = (Q^A \cup Q^B \cup \{s_0\}, \Sigma_1, \cdots, \Sigma_n, \delta^C, s_0, F^A \cup F^B)$ where $\delta^C = \delta^A \cup \delta^B \cup \{(s_0, \varepsilon^n, q_0^A), (s_0, \varepsilon^n, q_0^B)\}$ and $\varphi = \varphi_1 \vee \varphi_2$. The, it is easy to see that $L(\mathcal{C}) = L(\mathcal{B}) \cup L(\mathcal{A})$. Note that the overall size of $\mathcal{C}$ is linear in size of $\mathcal{A}$ and $\mathcal{B}$.
- **Concatenation:** Let $\mathcal{A} = (A, \varphi_1)$ and $\mathcal{B} = (B, \varphi_2)$ be two constrained rational automata such that $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$ and $B = (Q^B, \Sigma_1, \cdots, \Sigma_n, \delta^B, q_0^B, F^B)$. Without loss of generality, we will assume that $Q^A \cap Q^B = \emptyset$ and hence $\delta^A \cap \delta^B = \emptyset$. We construct the constrained rational automaton $\mathcal{C} = (C, \varphi)$ such that $L(\mathcal{C}) = L(\mathcal{A}) \cdot L(\mathcal{B})$ as follows:

$C = (Q^A \cup Q^B \cup, \Sigma_1, \cdots, \Sigma_n, \delta^C, q_0^A, F^B)$, $\delta^C = \delta^A \cup \delta^B \cup \{(q, \varepsilon^n, q_0^B) \mid q \in F^A\}$ and $\varphi = \varphi_1 \wedge \varphi_2$. The correctness of the construction follows immediately from the construction. Note that the over all size of $\mathcal{C}$ is linear in size of $\mathcal{A}$ and $\mathcal{B}$.

- **Emptiness:** Emptiness follows directly from the fact that n-tape constrained automaton $\mathcal{A}$ over $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ can be seen as 1-tape constrained automaton $\mathcal{B}$ over the alphabet $(\Sigma_1 \cup \{\varepsilon\}) \times \cdots \times (\Sigma_n \cup \{\varepsilon\})$ for which the emptiness problem is well-known to be decidable [28, 13]).
- **Membership:** This follows immediately from the decidability of the emptiness problem and the closure of the class of constrained languages wrt. intersection with regular languages (see Lemma 10). In fact any multi-dimensional language consisting of finite words is regular.
- **Undecidability of emptiness of intersection:** This directly follows from the fact that emptiness of intersection of rational language is undecidable.

D.2 Proof of lemma 10

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

Proof. We will now prove the claim of the above lemma:

- **Permutation:** Given a constrained rational automaton $\mathcal{A} = (A, \varphi)$ with $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$, and a permutation $h : [1..n] \mapsto [1..n]$, we can construct another constrained rational automaton $\mathcal{B} = (B, \varphi')$ with $B = (Q^A, \Sigma_{h(1)}, \cdots, \Sigma_{h(n)}, \delta^B, q_0^A, F^A)$ such that $L(\mathcal{B}) = \{(w_{h(1)}, w_{h(2)}, \cdots, w_{h(n)}) \mid (w_1, w_2, \cdots, w_n) \in L(\mathcal{A})\}$. The states, initial state and final states of out B automaton are the same as that of A. Along with the transition relation δ^B, we will also construct below a bijective function $g : \delta^A \mapsto \delta^B$ mapping the transitions of A to that of B. This will be useful for constructing the Presburger formula. For any $\tau = (q, (a_1, \cdots, a_n), q') \in \delta^A$, we add $\tau' = (q, (a_{h(1)}, \cdots, a_{h(n)}), q') \in \delta^B$ and let $g(\tau) = \tau'$. Let us assume that $\varphi(t_1, t_2, \ldots, t_m)$ where $t_1, t_2, \ldots, t_m \in \delta^A$ are the free variables of φ. Then the formula φ' as $\varphi(t_1/g(t_1), t_2/g(t_2), \ldots, t_m/g(t_m))$. It is then easy to see that $L(\mathcal{B}) = \{(w_{h(1)}, w_{h(2)}, \cdots, w_{h(n)}) \mid (w_1, w_2, \cdots, w_n) \in L(\mathcal{A})\}$. Note that the over all size of $\mathcal{B}$ is linear in size of $\mathcal{A}$
- **Projection:** Given a constrained rational automaton $\mathcal{A} = (A, \varphi)$ with $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$ and a set of indices $\iota = \{i_1 < i_2 < \ldots i_m\} \subset \{1, \ldots, n\}$, we can construct another constrained rational automaton $\mathcal{B} = (B, \varphi')$ with $B = (Q^B, \Sigma_2, \cdots, \Sigma_n, \delta^B, q_0^B, F^B)$ such that $L(\mathcal{B}) = \{(w_{i_1}, w_{i_2}, \cdots, w_{i_m}) \mid (w_1, w_1, \cdots, w_n) \in L(\mathcal{A})\}$. The states of B are given by $Q^B = Q^A \cup \{p_\tau \mid \tau \in \delta^A\}$, i.e. we add $|\delta^A|$ new states to the automaton along with the states of A. The initial state of B is the same as the initial state of A i.e. $q_0^B = q_0^A$. Similarly the final state of B is the same as the final states of A i.e. $F^A = F^B$. Along with the transition relation, we will also describe a function $g : \delta^A \mapsto \delta^B$ that will help us to construct the Presburger formula. For any $\tau = (q, (a_1, \cdots, a_n), q') \in \delta^A$, we add $\tau_1 = (q, (a_{i_1}, \cdots, a_{i_m}), p_\tau) \in$

δ^B and $\tau_2 = (p_\tau, (\varepsilon, \cdots, \varepsilon), q') \in \delta^B$. We let $g(\tau) = \tau_1$. Now the Presburger formula φ' is defined as $\varphi' = \varphi(\tau_1/g(\tau_1), \tau_2/g(\tau_2), \ldots, \tau_\ell/g(\tau_\ell))$, where $\tau_1 \cdots \tau_\ell$ are the transitions of A. Note that the complexity of such an construction is at most polynomial. The number states of B is the sum $|\delta^A|$ and $|A|$. Furthermore, we have that δ^A can be in the order of $Q^2 \times \Sigma$ where $\Sigma = \bigcup_{i \in [1..n]} \Sigma_i$.

– **Composition:** Let $\mathcal{A} = (A, \varphi_1)$ and $\mathcal{B} = (B, \varphi_2)$ be two constrained rational automata such that $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$ and $B = (Q^B, \Sigma_1, \Sigma'_2, \cdots, \Sigma'_m, \delta^B, q_0^B, F^B)$. Then, we can construct a constrained rational automaton $\mathcal{C} = (C, \varphi)$ such that $C = (Q^C, \Sigma_1, \cdots, \Sigma_n, \Sigma'_2, \cdots, \Sigma'_m, \delta^C, q_0^C, F^C)$ and $L(\mathcal{C}) = \{(w_1, \cdots, w_n, u_2, \cdots, u_m) \mid (w_1, \cdots, w_n) \in L(\mathcal{A}) \wedge (w_1, u_2, \cdots, u_m) \in L(\mathcal{B})\}$. Observe that we assume that the composition between $\mathcal{A}$ and $\mathcal{B}$ is done over the first tape since the class of constrained rational languages is closed under permutation. The states of the automata C are given by $Q^C = Q^A \times Q^B$, the initial state is given by $q_0^C = (q_0^A, q_0^B)$ and the set of final states is given by $F^C = F^A \times F^B$. We will define the transition relation δ^C along with two partial functions $g : \delta_C \mapsto \delta_A$ and $h : \delta_C \mapsto \delta_B$. For every $q, q' \in Q^A$, $p, p' \in Q^B$ and $a_1 \in \Sigma_1$, if $\tau_1 = (q, (a_1, \cdots, a_n), q') \in \delta^A$ and $\tau_2 = (p, (a_1, b_2, \cdots, b_m), p') \in \delta^B$ then we add $\tau' = ((q, p), (a_1, \cdots, a_n, b_2, \cdots, b_m), (q', p')) \in \delta^C$. We also let $g(\tau') = \tau_1$ and $h(\tau') = \tau_2$.
For any $\tau = (q, (\varepsilon, a_2, \cdots, a_n), q') \in \delta^A$, we add $\tau' = ((q, p), (\varepsilon, a_2, \cdots, a_n, \varepsilon^{m-1}), (q', p)) \in \delta^C$ and we let $g(\tau') = \tau$. Similarly for any $\tau = (q, (\varepsilon, b_2, \cdots, b_m), q') \in \delta^B$, we add $\tau' = ((q, p), (\varepsilon^n, b_2, \cdots, b_n), (q, p')) \in \delta^C$ and we also let $h(\tau') = \tau$. For any $\tau \in \delta_A$, we will refer to $g^{-1}(\tau)$ to mean $g^{-1}(\tau) = \{\tau' \mid \tau' \in \delta_C \wedge g(\tau') = \tau\}$. We use similar notation for $h^{-1}(\tau)$ with $\tau \in \delta_B$. Now the Presburger formula φ is defined as $\varphi = \varphi_1((\tau/\sum_{\tau' \in g^{-1}(\tau)} \tau')_{\tau \in \delta^A}) \wedge \varphi_1((\tau/\sum_{\tau' \in h^{-1}(\tau)} \tau')_{\tau \in \delta^B})$.

– **Intersection with regular:** Given a constrained rational automata $\mathcal{A} = (A, \varphi')$ with $A = (Q^A, \Sigma_1, \cdots, \Sigma_n, \delta^A, q_0^A, F^A)$ and a regular n-dim language L. Since the intersection operator is distributive over the union operator and the closure of the class of rational languages under union, we can assume w.l.o.g. that L is of the form $L(A_1) \times L(A_2) \times \cdots \times L(A_n)$ where for every $i \in [1..n]$, $A_i = (Q_i, \Sigma_i, \delta_i, q_i^{init}, F_i)$ is a finite state automaton. Let $A'_i = (Q_i, \Sigma_1, \ldots, \Sigma_n, \delta'_i, q_i^{init}, F_i)$ be the n-tape automaton such that (1) $\delta'_i \subseteq \{\varepsilon\}^{i-1} \times \Sigma_i \cup \{\varepsilon\}^{n-i}$ and (2) $(q, (\varepsilon^{i-1}, a, \varepsilon^{n-i}), q') \in \delta'_i$ iff $(q, a, q') \in \delta_i$. Then, it is easy to set that $L \cap L(\mathcal{A}) = L((((((\mathcal{A} \circ_{(1,1)} (A'_1, \texttt{true})) \circ_{(2,2)} (A'_2, \texttt{true})) \circ_{(3,3)} (A'_3, \texttt{true})) \cdots \circ_{(n,n)} A'_n))$. Since the class of constrained the class of constrained automata is closed under composition, we obtain that it is possible to construct a constrained rational automaton $\mathcal{A}'$ such that $L(\mathcal{A}') = L \cap L(\mathcal{A})$. Furthermore, it is easy to see that the size of A' is exponential in $|A_1| + |A_2| + \cdots + |A_n|$ and polynomial in the size of $\mathcal{A}$. Observe that in fact the complexity is polynomial in $(|A_1| + |A_2| + \cdots + |A_n|)^n$ and the size of $\mathcal{A}$.

D.3 Proof of Lemma 11

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 in-*

cluded 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.

Proof. We need to build a rational automata T such that $L(T) = \{(q,u,q',v,\sharp^m) \mid (q,u) \xrightarrow{\pi}_{\mathcal{P}} (q'v) \wedge \pi \in (L(\Lambda))^m\}$. We will use the same notations as in Lemma 1. It is easy to see that if $(q,u) \xrightarrow{\pi} (q'v)$ then clearly such a run can be split as $(q,u'w) \xrightarrow{\sigma_1}_w (q'',w) \xrightarrow{\sigma_2}_w (q',v'w)$ such that $u = u'w, v = v'w$. Note that the intermediate position need not fall in the Λ boundary. Let $u' = a_1 \cdots a_\ell$ and $v' = b_1 \cdots b_m$ then clearly we can further split the run into increasing and decreasing phase as $(q,a_1 \cdots a_n w) \xrightarrow{\beta_1}_{a_2 \cdots a_n w} (q_1, a_2 \cdots a_n w) \xrightarrow{\beta_2}_{a_3 \cdots a_n w} \cdots \xrightarrow{\beta_\ell}_w (q'',w)$ and $(q'',w) \xrightarrow{\alpha_1}_w (q'_1, b_n w) \xrightarrow{\alpha_2}_{b_n w} \cdots \xrightarrow{\alpha_m}_{b_2 \cdots b_n w} (q, v'w)$. Again note that these intermediate sequences need not fall at the Λ boundary. Given a run (say $(q,a_1 \cdots a_n w) \xrightarrow{\beta_1}_w (q_1, a_2 \cdots a_n w)$), the number of times Λ sequences are executed and completed in this part of run is regular. However only this information is not sufficient to concatenate sequence of such runs. We also need to keep track of the position inside Λ where the run ended.

For this purpose, we will construct a new pushdown automaton (we will use the definition in section B.5) $\mathbf{P} = (S, \{\sharp\}, \Gamma, \Delta, p_0, S)$ that has embedded in it the information of which position in Λ it is executing. The states of such a pushdown automaton are $S = P \times [0..m]$. The initial and final state of this pushdown is not important at this point of time. the transition relation Δ is defined as follows. We will also along side define a mapping $g : \Delta \mapsto R \cup \{\varepsilon\}$

- if $\tau = (p, pop_i(a), p') \in \tau_{2i}$ for some $i \in [0..m]$ then, we add $\tau' = ((p,i), a, \varepsilon, \varepsilon, (p',i)) \in \Delta$, we let $g(\tau') = \tau$
- if $\tau = (p, pop_i(a), p') \in \tau_{2i-1}$ for some $i \in [1..m]$ then, we add $\tau' = ((p,i), a, \varepsilon, \varepsilon, (p',i+1)) \in \Delta$, we let $g(\tau') = \tau$
- if $\tau = (p, push_i(a), p') \in \tau_{2i}$ for some $i \in [0..m]$ then, we add $\tau' = ((p,i), \varepsilon, \varepsilon, a, (p',i)) \in \Delta$, we let $g(\tau') = \tau$
- if $\tau = (p, push_i(a), p') \in \tau_{2i-1}$ for some $i \in [1..m]$ then, we add $\tau' = ((p,i), \varepsilon, \varepsilon, a, (p',i+1)) \in \Delta$, we let $g(\tau') = \tau$
- if $\tau = (p, nop_i(a), p') \in \tau_{2i}$ for some $i \in [0..m]$ then, we add $\tau' = ((p,i), \varepsilon, \varepsilon, \varepsilon, (p',i)) \in \Delta$, we let $g(\tau') = \tau$
- if $\tau = (p, nop_i(a), p') \in \tau_{2i-1}$ for some $i \in [1..m]$ then, we add $\tau' = ((p,i), \varepsilon, \varepsilon, \varepsilon, (p',i+1)) \in \Delta$, we let $g(\tau') = \tau$
- We also add $\tau = ((p,m), \varepsilon, \sharp, \varepsilon, (p,0)) \in \Delta$ and we let $g(\tau) = \varepsilon$

We extend the function g in the straightforward manner to sequence of transitions. Clearly, $\pi = ((p,i),u) \xrightarrow{\sigma}_{\mathbf{P}} ((p',j),v)$ iff $(p,u) \xrightarrow{g(\sigma)}_{\mathcal{P}} (p,v)$ such that $g(\sigma) \in \tau_{2i}^* \tau_{2i+1} \tau_{2i+2}^* \tau_{2i+3} \cdots \tau_{2m}^* \Lambda^k \tau_0^* \tau_1 \cdots \tau_{2j}^*$, for some $k \in \mathbb{N}$.

For any $i, j \in [0..m]$, let $L^-((p,i),a,(p',j)) = \{Label(\sigma) \mid ((p,i),a) \xrightarrow{\sigma}_{\mathbf{P}} ((p',j),\varepsilon)\}$. The above language recognises $Label(\sigma) = \sharp^k$ if there is an execution sequence that starts at state p and position τ_{2i} in Λ, executes rest of

sequence of Λ, iterates Λ^{k-1}, executes the Λ sequence up to τ_{2j}, reaches state p' and in the process removes the letter a from the stack. Clearly such a language is effectively regular and at most exponential in size of $\mathbf{P}$, since Parikh image of pushdown automata are effectively regular [17]. We will assume the finite state automaton recognizing such a language to be $B^-((p,i),a,(p',j))$.

Similarly let $L^+((p,i),a,(p',j)) = \{Label(\sigma) \mid ((p,i),\varepsilon) \xrightarrow{\sigma}_{\mathbf{P}} ((p',j),a)\}$, which is also regular for the same reason as above. We will assume the finite state automata recognizing the Parikh image of $L^+((p,i),a,(p',j))$ to be $B^+((p,i),a,(p',j))$.

We now show how to construct the rational automata T. Towards this purpose, we will introduce some notations. For any $p,p' \in Q$ and $i,j \in [0..m]$, we will refer to states of $B^{?\in\{+,-\}}((p,i),a,(p',j))$ as $State(B^{?\in\{+,-\}}((p,i),a,(p',j)))$. Similarly, we will refer to initial, final states and transitions as $Init(B^{?\in\{+,-\}}((p,i),a,(p',j)))$, $Final(B^{?\in\{+,-\}}((p,i),a,(p',j)))$ and $\Delta(B^{?\in\{+,-\}}((p,i),a,(p',j)))$ respectively. We will further assume that the states of such automata are distinct. We will simply use St^- and St^+ to refer to set of all states of all possible B^- and B^+ automata

The idea behind construction of rational automata T is as follows. We need to ensure that $L(T) = \{(q,u,q',v,\sharp^m) \mid (q,u) \xrightarrow{\pi}_{\mathcal{P}} (q',v) \wedge \pi \in (L(\Lambda))^m\}$. But note that $(q,u) \xrightarrow{\pi}_{\mathcal{P}} (q',v)$ iff $((q,0),u) \xrightarrow{\sigma}_{\mathbf{P}} (q',m),v)$ for some σ such that $\pi = g(\sigma)$. Now we can split this run as $((q,0),u'w) \xrightarrow{\alpha_1}_w ((q'',j),w) \xrightarrow{\alpha_2}_w (q',m),v'w)$ for some $j \in [0..m]$ where w is never popped during the two runs. Now let $u' = a_1 \cdots a_\ell$ and $v' = b_1 \cdots b_m$. Then the above run can further be split as $((q,0),a_1 \cdots a_\ell w) \xrightarrow{\sigma_1}_{a_2 \cdots a_\ell w} ((q_1,i_1),a_2 \cdots a_\ell w) \xrightarrow{\sigma_2}_{a_3 \cdots a_\ell w} \cdots \xrightarrow{\sigma_\ell}_{w_\ell} ((q'',j),w)$
$\xrightarrow{\sigma'_1}_w ((q'_1,i'_1),b_m w) \xrightarrow{\sigma'_2}_{b_m w} (q'_2,b_{m-1}b_m w) \cdots \xrightarrow{\sigma'_m}_{b_2 \cdots b_m w} ((q',m),v'w)$,
for some $i_1,i_2,\cdots,i_n,i'_1,i'_2 \cdots i'_m \in [1..m]$. Observe that the languages $Label(\sigma_1),Label(\sigma_2),\cdots,Label(\sigma'_m)$ are all regular. Now the construction of T amounts to guessing these intermediate points of the decreasing and increasing phase, while keeping track of the Parikh image of the languages generated during these runs on the 5^{th} tape. Finally we need to check that the intermidiary points between the decreasing and increasing phase are the same (which is done in a similar manner as in the proof of Lemma 1). The expoential blow up is due to the size of finite state automata $B^{?\in\{+,-\}}((p,i),a,(p',j))$.

Now the states of T is given by $Q^T = ((Q \times [0..m]) \cup St^-) \times ((Q \times [0..m]) \cup St^+ \cup \{\bot\}) \cup \{s,f\}$. It contains the states to recognise the decreasing slope, increasing slope and a final state. The initial state is s and set of final states is $F_T = \{f\}$. We add $(s,(q,\varepsilon,\varepsilon,\varepsilon,\varepsilon),((q,0),\bot)) \in \delta_T$ to start the simulation of the decreasing phase. We add for all state $q \in Q$ and $i,j \in [0..m]$, for all $a \in \Gamma$ and $q,q' \in Q$, we add $(((q,i),\bot),(\varepsilon,a,\varepsilon,\varepsilon,\varepsilon),(Init(B^-((q,i),a,(q',j)),\bot))) \in \delta_T$, this guesses the intermediate points of the decreasing phase and simulates the finite state automata on tape 5. Similarly we add $((Final(B^-((q,i),a,(q',j))),\bot),(\varepsilon,\varepsilon,\varepsilon,\varepsilon,\varepsilon),((q',j),\bot)) \in \delta_T$, this ensures that on completing the simulation of automaton, the control returns back. The above transitions have the following effect. Fom the initial state s on reading a state q on tape-1, we start the decreasing phase by entering the state $((q,0),\bot)$. From any state of the form $((q,i),\bot)$, on reading a letter a from tape-2, it guesses the resulting state (q',j) of a run that ends up consuming a from stack and simulates the automata $B^-((q,i),a,(q',j))$. We return to (q',j) after com-

pleting to simulate the automata. Towards simulating the automata, we add for all $(p,a,p') \in \Delta(B^-((q,i),a,(q',j)))$, the transition $((p,\bot),(\varepsilon,\varepsilon,\varepsilon,\varepsilon,a),(p',\bot)) \in \Delta'$. Note that $s \xrightarrow{(q,\varepsilon^4))} ((q,0),\bot) \xrightarrow{(\varepsilon,u,\varepsilon,\varepsilon,\sharp^m)} ((q',j),\bot)$ iff $((q,0),u) \xrightarrow{\sigma}_{\mathbf{P}} ((q',j),\varepsilon)$ with $Label(\sigma) = \sharp^m$.

Similarly we add set of transitions to deal with the increasing slope. We add $(((q,i),\bot),(\varepsilon,\varepsilon,q',\varepsilon,\varepsilon),((q,i),(q',m))) \in \delta_T$. We add for all state $p \in Q$, $k \in [0..m]$, $i,j \in [0..m]$, $a \in \Gamma$ and $q,q' \in Q$, we add $(((p,k),(q,i)),(\varepsilon,\varepsilon,\varepsilon,a,\varepsilon),((p,k),Init(B^+((q',j),a,(q,i))))) \in \delta_T$. Similarly we add $(((p,k),Final(B^+((q,i),a,(q',j)))),(\varepsilon,\varepsilon,\varepsilon,\varepsilon,\varepsilon),((p,k),(q,i))) \in \delta_T$. Towards simulating the automata, we add for all $(p,a,p') \in \Delta(B^+((q,i),a,(q',j)))$, the transition $(((q,k),p),(\varepsilon,\varepsilon,\varepsilon,\varepsilon),a,((q,k),p')) \in \Delta'$. We also add $((q,k),(q,k),\varepsilon^5,f) \in \Delta'$ and $(f,\varepsilon,a,\varepsilon,a,\varepsilon,f) \in \Delta'$.

D.4 Proof of Lemma 12

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 $i \in [1..n]$, $(q,u_i,q',v_i,\sharp^m) \in L(T_i)$.*

Proof. ($\Rightarrow$) Let $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q',v_1,\ldots,v_n)$ be a run in M such that $\pi \in (L(\Lambda))^m$. By definition, $\pi|_i$ is clearly a valid sequence of moves in M_i ($|_i$ is extended to a sequence in straight forward manner) such that $\pi|_i \in (L(\Lambda|_i))^m$. We can easily prove by induction on length of the run that if $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q',v_1,\ldots,v_n)$ then $(q,u_i) \xrightarrow{\pi|_i} (q',v_i)$. By definition of $L(T_i)$, this means that $(q,u_i,q',v_i,\sharp^m) \in L(T_i)$.

($\Leftarrow$)

The idea for the other direction is as follows. If $(q,u_i,q',v_i,\sharp^m) \in L(T_i)$, then by definition there are individual runs in pushdown systems such that this run is in $Ł(\Lambda|_i)^m$. Clearly $\Lambda|_i$ contains some real transitions and some projected transitions. Note that the real transitions are also part of our multi-pushdown system M. If we can find a shuffle of all the real transition sequences of each of these runs such that the ordering among the sequence obey the original ordering in the individual run, they are state-wise compatible and they form a word in $L(\Lambda)^m$, then it is easy to see that such a transition sequence is also a valid transition sequence in the multi-pushdown system M. We will attempt below to find one such valid sequence.

Let $j \in [1..n]$ be the index of the stack such that $\tau_0,\tau_{2m} \subseteq \Delta_j$. In the following, we will prove a stronger claim than the one stated by Lemma 12. In fact we will prove that if for every $i \in [1..n]$, $(q,u_i,q_i,v_i,\sharp^m) \in L(T_i)$ then $(q,u_i,q_j,v_i,\sharp^m) \in L(T_i)$ for all $i \in [1..n]$, and $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$ for some sequence $\pi \in (L(\Lambda))^m$.

The proof is done by induction on m.

Base Cases: The base case is when $m = 0$. This means that $(q,u_i) = (q_i,v_i)$ for all $i \in [1..n]$ and hence $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$ for $\pi = \varepsilon$. Furthermore it is easy to see that $(q,u_i,q_j,v_i,\sharp^m) \in L(T_i)$ holds for all $i \in [1..n]$.

Let us assume that $m = 1$. Let us assume that $(q,u_i,q_i,v_i,\sharp) \in L(T_i)$ for all $i \in [1..n]$. From Lemma 11, this implies that for every $i \in [1..n]$, $(q,u_i) \xrightarrow{\pi_i}_{M_i} (q_i,v_i)$ with $\sigma \in (L(\Lambda))$. Then we can rewrite the run $(q,u_i) \xrightarrow{\pi_i}_{M_i} (q_i,v_i)$

as follows $(q,u_i) \xrightarrow{\alpha_{(i,0)}}_{M_i} (p_{(i,1)},u_{(i,1)}) \xrightarrow{\alpha_{(i,1)}}_{M_i} (p_{(i,2)},u_{(i,2)}) \xrightarrow{\alpha_{(i,2)}}_{M_i} \cdots \xrightarrow{\alpha_{(i,2m-1)}}_{M_i} (p_{(i,2m)},u_{(i,2m)}) \xrightarrow{\alpha_{(i,2m)}}_{M_i} (q_i,v_i)$ where $\alpha_{(i,\ell)} \in \tau_\ell^*$ for all $\ell \in [0..2m]$ such that $\alpha_{(i,2r-1)} = \tau_{2r-1}$ for all $r \in [1..m]$. Then, it is easy to see that for every $i \neq j$, $(p_{(i,2m)},u_{(i,2m)}) \xrightarrow{\alpha_{(j,2m)|i}}_{M_i} (q_j,v_i)$ with $v_i = u_{(i,2m)}$. This implies that $(q,u_i) \xrightarrow{\alpha_{(i,0)}\cdot\alpha(i,1)\cdots\alpha_{(i,2m-1)}\cdot\alpha_{(j,2m)|i}}_{M_i} (q_j,v_i)$. Hence, $(q,u_i,q_j,v_i,\sharp) \in L(T_i)$ for all $i \in [1..n]$ and so we have proved the first part of the claim.

Let $s \in [0..2m]$, i_s be the index of the active stack (i.e., $\tau_s \subseteq \Delta_{i_s}$). Then we can show that for every $s \in [1..2m-1]$, $(p_{(1,s)},u_{(1,s)},u_{(2,s)},\cdots,u_{(n,s)}) \xrightarrow{\alpha_{(i_s,s)}}_M (p_{(1,s+1)},u_{(1,s+1)},u_{(2,s+1)},\cdots,u_{(n,s+1)})$. Furthermore, we can show $(q,u_1,u_2,\ldots,u_n) \xrightarrow{\alpha_{(j,0)}}_M (p_{(1,1)},u_{(1,1)},u_{(2,1)},\cdots,u_{(n,1)})$ and $(p_{(1,2m)},u_{(1,2m)},u_{(2,2m)},\cdots,u_{(n,2m)}) \xrightarrow{\alpha_{(j,2m)}}_M (q_j,v_1,v_2,\ldots,v_n)$. Now combining all these runs we get that $(q,u_1,u_2,\ldots,u_n) \xrightarrow{\alpha_{(j,0)}\alpha_{(i_1,1)}\cdots\alpha_{i_{2m-1},2m-1}\alpha_{(j,2m)}} (q_j,v_1,v_2,\ldots,v_n)$. This complete the proof for the base case.

Induction Step: Let us assume now that if for every $i \in [1..n]$, $(q,u_i,q_i',v_i,\sharp^m) \in L(T_i)$ then $(q,u_i,q_j',v_i,\sharp^m) \in L(T_i)$ for all $i \in [1..n]$, and $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$ for some sequence $\pi \in (L(\Lambda))^m$.

Let us show that if for every $i \in [1..n]$, $(q,u_i,q_i,v_i,\sharp^{m+1}) \in L(T_i)$ then $(q,u_i,q_j,v_i,\sharp^{m+1}) \in L(T_i)$ for all $i \in [1..n]$, and $(q,u_1,\ldots,u_n) \xrightarrow{\pi}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$ for some sequence $\pi \in (L(\Lambda))^{m+1}$.

Since $(q,u_i,q_i,v_i,\sharp^{m+1}) \in L(T_i)$ for all $i \in [1..n]$, this implies that $(q,u_i) \xrightarrow{\pi_i}_{M_i} (q_i,v_i)$ for some $\pi_i \in L(\Lambda|_i)^{m+1}$. We can then split the run $(q,u_i) \xrightarrow{\pi_i}_{M_i} (q_i,v_i)$ as follows: $(q,u_i) \xrightarrow{\pi_i'}_{M_i} (q_i',w_i) \xrightarrow{\pi''}_{M_i} (q_i,v_i)$ such that $\pi_i' \in L(\Lambda|_i)^m$ and $\pi_i'' \in L(\Lambda|_i)$.

Since for every $i \in [1..n]$, $(q,u_i) \xrightarrow{\pi_i'}_{M_i} (q_i',w_i)$ such that $\pi_i' \in L(\Lambda|_i)^m$, we have $(q,u_i,q_i',w_i,\sharp^m) \in L(T_i)$ from Lemma 11. Now we can apply the induction hypothesis to these runs and we get that $(q,u_i,q_j',w_i,\sharp^m) \in L(T_i)$ for all $i \in [1..n]$, and $(q,u_1,\ldots,u_n) \xrightarrow{\sigma}_{\mathcal{M}} (q_j',w_1,\ldots,w_n)$ for some sequence $\pi \in (L(\Lambda))^m$.

On the other hand, since $\tau_0 \subseteq \Delta_j$, we can show that $(q_j',w_i) \xrightarrow{\pi'''}_{M_i} (q_i,v_i)$ such that $\pi_i'' \in L(\Lambda|_i)^m$. This implies that $(q_j',w_i,q_i,v_i,\sharp) \in L(T_i)$ from Lemma 11. Now we can apply the induction hypothesis to these runs and we get that $(q_j',w_i,q_j,v_i,\sharp) \in L(T_i)$ for all $i \in [1..n]$, and $(q_j',w_1,\ldots,w_n) \xrightarrow{\sigma'}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$ for some sequence $\pi \in (L(\Lambda))^m$.

Since $(q,u_i,q_j',w_i,\sharp^m) \in L(T_i)$ and $(q_j',w_i,q_j,v_i,\sharp) \in L(T_i)$ for all $i \in [1..n]$, we get that $(q,u_i) \xrightarrow{\pi_i'}_{M_i} (q_j',w_i)$ and $(q_j',w_i) \xrightarrow{\pi'''}_{M_i} (q_j,v_i)$. Combining these two runs we get $(q,u_i) \xrightarrow{\pi'\cdot\pi'''}_{M_i} (q_j,v_i)$ for all $j \in [1..n]$.

Now, we can also combine the following two runs $(q,u_1,\ldots,u_n) \xrightarrow{\sigma}_{\mathcal{M}} (q_j',w_1,\ldots,w_n)$ and $(q_j',w_1,\ldots,w_n) \xrightarrow{\sigma'}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$, we get $(q,u_1,\ldots,u_n) \xrightarrow{\sigma\cdot\sigma'}_{\mathcal{M}} (q_j,v_1,\ldots,v_n)$. □