Model-Checking Bounded Multi-Pushdown Systems[⋆]

Kshitij Bansal[1] and Stéphane Demri[1,2]

[1]New York University, USA [2]LSV, CNRS, France

Abstract. We provide complexity characterizations of model checking multi-pushdown systems. We consider three standard notions for boundedness: context boundedness, phase boundedness and stack ordering. The logical formalism is a linear-time temporal logic extending well-known logic `CaRet` but dedicated to multi-pushdown systems in which abstract operators are parameterized by stacks. We show that the problem is ExpTime-complete for context-bounded runs and unary encoding of the number of context switches; we also prove that the problem is 2ExpTime-complete for phase-bounded runs and unary encoding of the number of phase switches. In both cases, the value k is given as an input, which makes a substantial difference in the complexity. [1]

1 Introduction

Verification problems for pushdown systems, systems with a finite automaton and an unbounded stack, have been extensively studied and decidability can be obtained as in the case for finite-state systems. For instance, computing $\text{pre}^\star(X)$ (set of configurations *reaching a* regular set X), $\text{post}^\star(X)$ (correspondingly, configurations *accessible from* X), reachability and LTL model checking have been shown to be decidable [8,18]. These have also been implemented, for instance in the model-checker Moped [18]. It can be argued that they are natural models for modeling recursive programs. Two limitations though of the model are the inability to model programs with infinite domains (like integers) and modeling concurrency. Having an infinite automaton to handle the former limitation leads to undecidability. An approach to tackle this has been to abstract infinite-state programs to Boolean programs using, for instance, predicate abstraction. The model is repeatedly refined, as needed, like in tools SLAM, SatAbs etc. For concurrency, a natural way to extend this model would be to consider pushdown automata with multiple stacks, which has seen significant interest in the recent past [2,6,9,10]. This is the main object of study in this paper which we call *multi-pushdown systems (MPDS).*

The difficulty of model-checking MPDS is that a pushdown system with even two stacks and with a singleton stack alphabet is sufficient to model a Turing

[⋆] Work partially supported by projects *ARCUS IdF/Inde* and *EU Seventh Framework Programme* under grant agreement No. PIOF-GA-2011-301166 (DATAVERIF).

[1] Omitted proofs and additional material can be found in the technical report [5].

machine, hence making the problem of even testing reachability undecidable. This is not a unique situation and similar issues exists with other abstractions, like model-checking problems on counter systems; other models of multithreaded programs are also known to admit undecidable verification problems. That is why subclasses of runs have been introduced as well as problems related to the search for 'bounded runs' that may satisfy a desirable or undesirable property. For instance, context-bounded model-checking (bound on the number of context switches) [17] allows to regain decidability.

This paper focuses on the study of model-checking problems for MPDS based on LTL-like dialects, naturally allowing to express liveness properties, when some bounds are fixed. Though decidability of these problems has been established in some recent works we aim to provide optimal computational complexity analysis for LTL-like properties. In particular, we consider a LTL-like specification language based on `CaRet` [1], which strikes to us as fitting given the interest of the model in program verification. As in [14], `CaRet` generalized to multiple stacks and called Multi-`CaRet` is considered. Under this logic, we show model-checking problem of MPDS restricted to k-**context bounded runs** is in ExpTime, when k is encoded in unary. Since this problem is a generalization of LTL model checking pushdown systems which is known to be ExpTime-hard, this is an optimal result. Viewed as an extension of [8], we consider both a more general model and a more general logic, while still preserving the complexity bounds. At a technical level, we focus on combining several approaches in order to achieve optimal complexity bounds. In particular, we combine the approach taken in `CaRet` model-checking of recursive state machines machines, ideas from reachability analysis of multi-pushdown systems [18] and the techniques introduced in [8,18]. We also consider less restrictive notions, showing optimal 2-ExpTime for k-**phase bounded runs** [12] when k is in unary. Note that in all restrictions we consider, k is given as an input and not as a parameter of the problem, which makes a substantial difference when complexity analysis is provided. When k is encoded in binary, the bounds are 2-ExpTime and 3-ExpTime for context and phase boundedness respectively. For a third notion of **ordered multi-pushdown systems** [3], model-checking is in 2ExpTime.

Related work. In [16], decidability results are found for classes of automata with auxiliary storage based on MSO property, see also [15]. This includes MPDS with bounded context and ordered MPDS. Unlike our ExpTime bound, the complexity is non-elementary in the size of the formula. This stems from the use of celebrated Courcelle's Theorem, which has parameterized complexity non-elementary, the parameter being the size of formula plus the tree-width.

More closely related to our approach of generalizing the automata-based approach for LTL are two recent works [4,14]; indeed model-checking of linear-time properties for MPDS under several boundedness hypothesis has been the subject of several recent studies. In [4], LTL model-checking on multi-pushdown systems when runs are k-scope-bounded is shown ExpTime-complete. Scope-boundedness strictly extends context-boundedness and therefore Corollary 7(I) and [4, Theorem 7] are closely related even though Corollary 7(I) deals with the richer Multi-CaRet and it takes into account specifically context-boundedness. By contrast, [14] introduces an extension of CaRet that is expressively identical to the variant we consider in our paper (models are multiply nested words). Again, it deals with scope-boundedness and Corollary 7(I) and [14, Theorem 6] are closely related even though Corollary 7(I) takes into account context-boundedness specifically, which leads to a slightly different result. Similarly, upper bounds [14, Theorem 7] about ordered multiply nested words, is related to upper bound we provide in Corollary 8 for OBMC. Nevertheless, as technical contributions, we first deal with context-boundedness, phase-boundedness and ordered MPDS uniformly independent of the notion of boundedness by following an automata-based approach reducing to the corresponding repeated reachability problem. In second step, we provide optimal complexity bounds by building on analysis for context-boundedness on [8,18] whereas for ordered MPDS it relies on [2]. Finally, our construction allows us to add regularity constraints on stack contents, extending notions from [11], that are known to go beyond first-order language, by an adaptation of the case for Multi-CaRet.

2 Preliminaries

We write $[N]$ to denote the set $\{1, 2, \ldots, N\}$. We also use a boldface as a shorthand for elements indexed by $[N]$, for e.g., $\boldsymbol{a} = \{a_i \mid i \in [N]\}$. For a finite word $w = a_1 \ldots a_k$ over the alphabet Σ, we write $|w|$ to denote its *length* k. For $0 \leq i < |w|$, $w(i)$ represents the $(i+1)$-th letter of the word, here a_{i+1}.

Pushdown systems provide a natural execution model for programs with recursion. A generalization with multiple stacks allows us to model threads, formally defined next. A *multi-pushdown system* (MPDS) is a tuple of the form $P = (G, N, \Gamma, \Delta_1, \ldots, \Delta_N)$, for some $N \geq 1$ such that G is a non-empty finite set of *global states*, Γ is the finite *stack alphabet* containing the distinguished letter $\perp$, for every $s \in [N]$, Δ_s is the *transition relation* acting on the s-th stack where Δ_s is a relation included in $G \times \Gamma \times G \times \mathfrak{A}(\Gamma)$ with $\mathfrak{A}(\Gamma)$ defined as $\mathfrak{A}(\Gamma) \overset{\text{def}}{=} \bigcup_{a \in \Gamma} \{\mathsf{call}(a), \mathsf{return}(a), \mathsf{internal}(a)\}$. Elements of the set $\mathfrak{A}(\Gamma)$ are to be thought of as *actions* modifying the stack with alphabet Γ. A *configuration* c of P is the global state along with contents of the N stacks, i.e. c belongs to $G \times (\Gamma^*)^N$. For every $s \in [N]$, we write $\rightarrow_s$ to denote the *one-step relation* w.r.t. the s-th stack. Given two configurations $c = (g, w_1, \ldots, w_s a, \ldots w_N)$ and $c' = (g', w_1, \ldots, w'_s, \ldots, w_N)$, $c \rightarrow_s c'$ iff $(g, a, g', \mathfrak{a}(b)) \in \Delta_s$ where $\mathfrak{a}(b)$ reflects the change in the stack enforcing one of the conditions below: $w_s = w'_s$, $\mathfrak{a} = \mathsf{return}$ and $a = b$, or $w'_s = w_s b$ and $\mathfrak{a} = \mathsf{internal}$, or $w'_s = w_s ab$ and $\mathfrak{a} = \mathsf{call}$. The letter $\perp$ from the stack alphabet plays a special role; indeed the initial content of each stack is precisely $\perp$. Moreover, $\perp$ cannot be pushed, popped or replaced by any other symbol. This is a standard way to constrain the transition relations and to check for 'emptiness' of the stack. We write $\rightarrow_P$ to denote the relation $(\bigcup_{s \in [N]} \rightarrow_s)$. Given a configuration c, there may exist c_1, c_2 and $i_1 \neq i_2 \in [N]$ such that $c \rightarrow_{i_1} c_1$ and $c \rightarrow_{i_2} c_2$, which is the fundamental property to consider such models as adequate for modeling concurrency. An infinite *run* is an ω-

sequence of configurations $c_0, c_1, c_2, \ldots$ s.t. for every $i \geq 0$, we have $c_i \rightarrow_P c_{i+1}$. If $c_i \rightarrow_s c_{i+1}$, then we say that for that step, the s-th stack is *active*. Similar notions can be defined for finite runs. A standard problem on MPDS is the state reachability problem: given a MPDS P, a configuration c and a global state g, is there a run from c to some configuration c' s.t. the state of c' is g?

An *enhanced* multi-pushdown system is a multi-pushdown system of the form $P = (G \times [N], N, \Gamma, \Delta_1, \ldots, \Delta_N)$ s.t. for every $s \in [N]$, $\Delta_s \subseteq (G \times \{s\}) \times \Gamma \times (G \times [N]) \times \mathfrak{A}(\Gamma)$. In such systems, the global state contains enough information to determine the next *active* stack. Observe that the way the one-step relation is defined, we do not necessarily need to carry this information as part of the finite control (see Lemma 1). We do that in order to enable us to assert about active stack in our logic (see Section 3), and for technical convenience.

Lemma 1. *Given $P = (G, N, \Gamma, \boldsymbol{\Delta})$, one can construct in polynomial time an enhanced $P' = (G \times [N], N, \Gamma, \boldsymbol{\Delta'})$ such that (I) for every infinite run of P of the form $c_0 \rightarrow_{s_0} c_1 \rightarrow_{s_1} \cdots c_t \rightarrow_{s_t} c_{t+1} \cdots$ there is an infinite run $c'_0 \rightarrow_{s_0} c'_1 \rightarrow_{s_1} \cdots c'_t \rightarrow_{s_t} c'_{t+1} \cdots$ of P' such that $(\star)$ for $t \geq 0$, if $c_t = (g_t, \{w^t_s\}_s)$, then $c'_t = ((g_t, s_t), \{w^t_s\}_s)$ and (II) similarly, for every infinite run of P' of the form $c'_0 \rightarrow_{s_0} c'_1 \rightarrow_{s_1} \cdots c'_t \rightarrow_{s_t} c'_{t+1} \cdots$ there is an infinite run $c_0 \rightarrow_{s_0} c_1 \rightarrow_{s_1} \cdots c_t \rightarrow_{s_t} c_{t+1} \cdots$ of P such that $(\star)$.*

The proof is by an easy verification. In the sequel, w.l.o.g., we consider enhanced MPDS only since all the properties that can be expressed in our logical languages are linear-time properties. For instance, there is a logspace reduction from the state reachability problem to its restriction to enhanced MPDS.

State reachability problem is known to be undecidable by a simple reduction from the non-emptiness problem for intersection of context-free grammars. This has motivated works on restrictions on runs so that decidability can be regained (for state reachability problem and for model-checking problems). We recall below standard notions for boundedness; other notions can be found in [13,9]. Definitions are provided for infinite runs but they can be adapted to finite runs.

For the notion of k-boundedness, a phase is understood as a sub-run such that a single stack is active (see e.g. [17]). Let $\rho = c_0 \rightarrow_{s_0} c_1 \rightarrow_{s_1} \cdots c_t \rightarrow_{s_t} c_{t+1} \cdots$ be an infinite run and $k \geq 0$. We say that ρ is *k-bounded* if there exist positions $i_1 \leq i_2 \leq \ldots \leq i_{k-1}$ such that $s_t = s_{t+1}$ for all $t \in \mathbb{N} \setminus \{i_1 \ldots i_{k-1}\}$. In the notion of k-phase-boundednessdefined below, a phase is understood as a sub-run such that return actions are performed on a single stack, see e.g. [12]. Let $\rho = c_0 \rightarrow_{s_0} c_1 \rightarrow_{s_1} \cdots c_t \rightarrow_{s_t} c_{t+1} \cdots$ be an infinite run and $k \geq 0$. We say that ρ is *k-phase-bounded* if there is a partition $Y_1, \ldots, Y_\alpha$ of $\mathbb{N}$ with $\alpha \leq k$ such that for every $j \in [1, \alpha]$ there is $s \in [N]$ s.t. for every $i \in Y_j$, if a return action is performed from c_i to c_{i+1}, then it is done on the sth stack. Finally, in the notion of order-boundedness defined below, the stacks are linearly ordered and a return action on a stack can only be performed if the smallest stacks are empty, see e.g. [3]. Let P be a multi-pushdown system and $\preceq = ([N], \leq)$ be a total ordering. Let $\rho = c_0 \rightarrow_{s_0} c_1 \rightarrow_{s_1} \cdots c_t \rightarrow_{s_t} c_{t+1} \cdots$ be an infinite run. We say that ρ is *$\preceq$-bounded* if for every $t \in \mathbb{N}$ that a return is performed on the s-th stack, all the stacks strictly smaller than s w.r.t. $\preceq$ are empty.

3 Specification Language Multi-`CaRet`

Below, we introduce Multi-`CaRet`, an extension of the logic `CaRet` proposed in [1], and dedicated to runs of MPDS (instead of for runs of recursive state machines as done in [1]). The logic below can be seen as a fragment of MSO and therefore the decidability results from [6,16] apply to the forthcoming model-checking problems. However, our definition makes a compromise between a language of linear properties that extends the logic from [1] and the most expressive logic for which our model-checking problems are known to be decidable. The logic below is expressively identical as well as syntactically and semantically similar to one in [14], except for the presence of regular constraints.

Models of Multi-`CaRet` are infinite runs of multi-pushdown systems. For each (enhanced) multi-pushdown system $P = (G \times [N], N, \varGamma, \varDelta_1, \ldots, \varDelta_N)$, the fragment Multi-`CaRet`$(P)$ of `CaRet` that uses syntactic resources from P (namely G and $[N]$). Multi-`CaRet` is defined as the union of all the sub-languages Multi-`CaRet`(P). The grammar $\phi := g \mid s \mid \mathsf{call} \mid \mathsf{return} \mid \mathsf{internal} \mid \phi \vee \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi\mathsf{U}\phi \mid \mathsf{X}^{\mathsf{a}}_{s}\phi \mid \phi\mathsf{U}^{\mathsf{a}}_{s}\phi \mid \mathsf{X}^{\mathsf{c}}_{s}\phi \mid \phi\mathsf{U}^{\mathsf{c}}_{s}\phi$, defines formulas of Multi-`CaRet`(P), with $s \in [N]$, $g \in G$. Models of Multi-`CaRet`(P) formulae are ω-sequences in $\left(G \times [N] \times (\varGamma^*)^N\right)^\omega$, which can be obviously understood as infinite runs of P.

Semantics. Given an infinite run $\rho = c_0 c_1 \ldots c_t \ldots$ with $c_t = (g_t, s_t, w^t_1, \ldots, w^t_N)$ for every position $t \in \mathbb{N}$, the satisfaction relation $\rho, t \models \phi$ with ϕ in Multi-`CaRet`(P) is defined inductively as follows (successor relations are defined just below and obvious clauses are omitted):

$$
\begin{array}{ll}
\rho, t \models g & \text{iff } g_t = g \quad \text{and} \quad \rho, t \models s \text{ iff } s_t = s \\
\rho, t \models \mathfrak{a} & \text{iff } \left(\mathfrak{a}, \left|w^{t+1}_{s_t}\right| - \left|w^{t}_{s_t}\right|\right) \in \{(\mathsf{call}, 1), (\mathsf{internal}, 0), (\mathsf{return}, -1)\} \\
\rho, t \models \phi_1 \mathsf{U} \phi_2 & \text{iff there is a sequence of positions } i_0 = t,\ i_1 \ldots,\ i_k, \text{ s.t.} \\
& \quad \text{for } j < k,\ i_{j+1} = \mathrm{succ}_\rho(i_j),\ \rho, i_j \models \phi_1 \text{ and } \rho, i_k \models \phi_2
\end{array}
$$

For $b \in \{\mathsf{a}, \mathsf{c}\}$ and $s \in [N]$:

$$
\begin{array}{ll}
\rho, t \models \mathsf{X}^{b}_{s}\phi & \text{iff } \mathrm{succ}^{b,s}_{\rho}(t) \text{ is defined and } \rho, \mathrm{succ}^{b,s}_{\rho}(t) \models \phi \\
\rho, t \models \phi_1 \mathsf{U}^{a}_{s} \phi_2 & \text{iff there exists a sequence of positions } t \leq i_0 < i_1 \\
& \quad \cdots < i_k, \text{ where } i_0 \text{ smallest such with } s_{i_0} = s, \text{ for} \\
& \quad j < k,\ i_{j+1} = \mathrm{succ}^{a,s}_{\rho}(i_j),\ \rho, i_j \models \phi_1 \text{ and } \rho, i_k \models \phi_2 \\
\rho, t \models \phi_1 \mathsf{U}^{c}_{s} \phi_2 & \text{iff there exists a sequence of positions } t \geq i_0 > i_1 \\
& \quad \cdots > i_k, \text{ where } i_0 \text{ greatest such with } s_{i_0} = s, \text{ for} \\
& \quad j < k,\ i_{j+1} = \mathrm{succ}^{c,s}_{\rho}(i_j),\ \rho, i_j \models \phi_1 \text{ and } \rho, i_k \models \phi_2
\end{array}
$$

Definition for $\models$ uses three successor relations: *global* successor relation, *abstract* successor relation that jumps to the first future position after a return action at the same level, if any, and the *caller* successor relation that jumps to the latest past position before a call action at the same level, if any. Here are the definitions: $\mathrm{succ}_\rho(t) \stackrel{\text{def}}{=} t + 1$ for every $t \in \mathbb{N}$; $\mathrm{succ}^{\mathsf{c},s}_{\rho}(t)$ (caller of s-th stack):

largest $t' < t$ s.t. $s_{t'} = s$ and $\left|w_s^{t'}\right| = |w_s^t| - 1$. If such a t' does not exist, then $\mathrm{succ}_\rho^{\mathtt{c},s}(t)$ is undefined; and $\mathrm{succ}_\rho^{\mathtt{a},s}(t)$ is defined when s is active at position t:

1. If $\left|w_s^{t+1}\right| = |w_s^t| + 1$ (call), then $\mathrm{succ}_\rho^{\mathtt{a},s}(t)$ is the smallest $t' > t$ such that $s_{t'} = s$ and $\left|w_s^{t'}\right| = |w_s^t|$. If there is no such t' then $\mathrm{succ}_\rho^{\mathtt{a},s}(t)$ is undefined.
2. If $\left|w_s^{t+1}\right| = |w_s^t|$ (internal), then $\mathrm{succ}_\rho^{\mathtt{a},s}(t)$ is the smallest $t' > t$ such that $s_{t'} = s$ (first position when sth stack is active).
3. If $\left|w_s^{t+1}\right| = |w_s^t| - 1$ (return), then $\mathrm{succ}_\rho^{\mathtt{a},s}(t)$ is undefined.

In the sequel, we write $\rho \models \phi$ whenever $\rho, 0 \models \phi$.

Adding regularity constraints. We define Multi-$\mathtt{CaRet}^{reg}$ as the extension of Multi-$\mathtt{CaRet}$ in which regularity constraints on stack contents can be expressed. Logic Multi-$\mathtt{CaRet}^{reg}$ is defined from Multi-$\mathtt{CaRet}$ by adding atomic formulae of the form $\mathtt{in}(s, \mathcal{A})$ where s is a stack identifier and $\mathcal{A}$ is a finite-state automaton over the stack alphabet Γ. The satisfaction relation $\models$ is extended accordingly: $\rho, t \models \mathtt{in}(s, \mathcal{A})$ iff $w_s^t \in \mathrm{L}(\mathcal{A})$ where $\mathrm{L}(\mathcal{A})$ is the set of finite words accepted by $\mathcal{A}$. Note that regularity constraints can be expressed on each stack.

Let us introduce the model-checking problems considered herein. The model-checking problem for MPDS (MC) is defined s.t. it takes as inputs a MPDS P, a configuration $\left(g, (\bot)^N\right)$ and a formula ϕ in Multi-$\mathtt{CaRet}(P)$ and asks whether there is an infinite run ρ from $\left(g, (\bot)^N\right)$ such that $\rho \models \phi$. We know that the model-checking problem for MPDS is undecidable whereas its restriction to a single stack is ExpTime-complete [1]. Now, let us turn to bounded model-checking problems. Bounded model-checking problem for MPDS (BMC) is defined such that it takes as inputs P, a configuration $\left(g, (\bot)^N\right)$, a formula ϕ in Multi-$\mathtt{CaRet}(P)$ and a bound $k \in \mathbb{N}$ and it asks whether there is an infinite k-bounded run ρ from $\left(g, (\bot)^N\right)$ such that $\rho \models \phi$. Note that $k \in \mathbb{N}$ is an input and not a parameter of BMC. This makes a significant difference for complexity since usually complexity can increase when passing from being a constant to being an input. Phase-bounded model-checking problem (PBMC) is defined similarly by replacing in the above definition 'k-bounded run' by 'k-phase-bounded run'. Similarly, we can obtain a definition with order-boundedness. Order-bounded model-checking problem for multi-pushdown systems (OBMC) is defined such that it takes as inputs P, a configuration $\left(g, (\bot)^N\right)$, a formula ϕ in Multi-$\mathtt{CaRet}(P)$ and a total ordering $\preceq = ([N], \leq)$ and it asks whether there is an infinite $\preceq$-bounded run ρ from $\left(g, (\bot)^N\right)$ such that $\rho \models \phi$.

The problem of repeated reachability of MPDS, written REP, is defined in the expected way with a generalized Büchi acceptance condition related to states. We refer to the problem restricted to k-bounded runs by BREP. Obviously, the variants with other notions of boundedness can be defined too. Now, the simplified version of Multi-$\mathtt{CaRet}$ consists of the restriction of Multi-$\mathtt{CaRet}$ in which atomic formulae are of the form (g, s) when enhanced MPDS are involved. For every $\mathcal{P}$ in {MC,BMC,PBMC,OBMC}, there is a logspace reduction to $\mathcal{P}$ restricted to formulae from the simplified language. The proof idea consists in adding to global states information about the next active stack and about the type of action. In the sequel, w.l.o.g., we restrict ourselves to the simplified languages. By [16], we conclude that BMC, PBMC and OBMC are decidable (use of Courcelle's Theorem). However, it provides non-elementary upper bounds. As a main result of our paper, we show that BMC is ExpTime-complete when k is encoded in unary even in presence of regular constraints.

4 From Model-Checking to Repeated Reachability

Herein, we reduce the problem of model checking (MC) to the problem of repeated reachability (REP) while noting complexity features that are helpful later on (Theorem 4). This generalizes Vardi-Wolper reduction from LTL model-checking into non-emptiness for generalized Büchi automata, similarly to the approach followed in [14]; not only we have to tailor the reduction to Multi-`CaRet` and to MPDS but also we aim at getting tight complexity bounds afterwards. The instance of MC that we have is a MPDS P, a formula ϕ and initial state (g_0, i_0). For the instance of REP we will reduce to, we will denote the MPDS by $\widehat{P}$, the set of acceptance sets by $\mathcal{F}$ and the set of initial states by I_0.

Augmented Runs. Let ρ be a run of the multi-pushdown system $P = (G \times [N], N, \Gamma, \boldsymbol{\Delta})$ with $\rho \in (G \times [N] \times (\Gamma^*)^N)^\omega$. The multi-pushdown system $\widehat{P}$ is built in such a way that its runs correspond exactly to runs from P but augmented with pieces of information related to the satisfaction of subformulas (taken from the closure set $\mathrm{Cl}(\phi)$ elaborated on shortly), whether a stack is dead or not (using a tag from $\{\mathsf{alive}, \mathsf{dead}\}$) and whether the current call will ever be returned or not (using a tag from $\{\mathsf{noreturn}, \mathsf{willreturn}\}$). These additional tags will suffice to reduce the existence of a run satisfying ϕ to the existence of a run satisfying a generalized Büchi condition. First, we define from ρ an "augmented run" $\gamma(\rho)$ which is an infinite sequence from $(\widehat{G} \times [N] \times (\widehat{\Gamma}^*)^N)^\omega$ where $\widehat{G} = G \times \mathcal{P}(\mathrm{Cl}(\phi))^N \times \{\mathsf{noreturn}, \mathsf{willreturn}\}^N \times \{\mathsf{alive}, \mathsf{dead}\}^N$ and $\widehat{\Gamma} = \Gamma \times \mathcal{P}(\mathrm{Cl}(\phi)) \times \{\mathsf{noreturn}, \mathsf{willreturn}\}$. By definition, an augmented run is simply an ω-sequence but it remains to check that indeed, it will be also a run of the new system. We will see that $\widehat{G} \times [N]$ is the set of global states of $\widehat{P}$ and $\widehat{\Gamma}$ is the stack alphabet of $\widehat{P}$. Before defining $\gamma(\cdot)$ which maps runs to augmented runs, let us introduce the standard notion for *closure* but slightly tailored to our needs. Each global state is partially made of sets of formulas that can be viewed as future obligations. This is similar to what is done for LTL and is just a variant of Fischer-Ladner closure. An obligation for a stack is a set of subformulas that is locally consistent; such consistent sets are called atoms and they are defined below as well as the notion of closure. Given a formula ϕ, its *closure*, denoted $\mathrm{Cl}(\phi)$, is the smallest set that contains ϕ, the elements of $G \times [N]$, and satisfies the following properties ($b \in \{a, c\}$ and $s \in [N]$): (i) if $\neg\phi' \in \mathrm{Cl}(\phi)$ or $\mathsf{X}\phi' \in \mathrm{Cl}(\phi)$ or $\mathsf{X}^b_s\phi' \in \mathrm{Cl}(\phi)$ then $\phi' \in \mathrm{Cl}(\phi)$; (ii) if $\phi' \vee \phi'' \in \mathrm{Cl}(\phi)$, then $\phi', \phi'' \in \mathrm{Cl}(\phi)$; (iii) if $\phi' \mathsf{U} \phi'' \in \mathrm{Cl}(\phi)$, then ϕ', ϕ'', and $\mathsf{X}(\phi' \mathsf{U} \phi'')$ are in $\mathrm{Cl}(\phi)$; (iv) if $\phi' \mathsf{U}^b_s \phi'' \in \mathrm{Cl}(\phi)$, then ϕ', ϕ'', and $\mathsf{X}^b_s(\phi' \mathsf{U}^b_s \phi'')$ are in $\mathrm{Cl}(\phi)$; (v) if $\phi' \in \mathrm{Cl}(\phi)$ and ϕ' in not of the form $\neg\phi''$, then $\neg\phi' \in \mathrm{Cl}(\phi)$. The number of formulas in $\mathrm{Cl}(\phi)$ is linear in the size of ϕ and P. An *atom* of ϕ, is a set $A \subseteq \mathrm{Cl}(\phi)$ that satisfies the following properties: (a) for $\neg\phi' \in \mathrm{Cl}(\phi)$, $\phi' \in A$ iff $\neg\phi' \notin A$; (b) or $\phi' \vee \phi'' \in \mathrm{Cl}(\phi)$, $\phi' \vee \phi'' \in A$ iff ($\phi' \in A$ or $\phi'' \in A$); (c) or $\phi' \mathsf{U} \phi'' \in \mathrm{Cl}(\phi)$, $\phi' \mathsf{U} \phi'' \in A$ iff $\phi'' \in A$ or ($\phi' \in A$ and $\mathsf{X}(\phi' \mathsf{U} \phi'') \in A$); (d) A contains exactly one element from $G \times [N]$. Let $\mathrm{Atoms}(\phi)$ denote the set of atoms of ϕ, along with empty set (used as special atom, use will become clear later). Note that there are $2^{\mathcal{O}(|\phi|)}$ atoms of ϕ.

We write $((g^t, s^t), \boldsymbol{w}^t)$ to denote the t-th configuration of ρ. We define the augmented run $\gamma(\rho)$ so that its t-th configuration is of the form $((\widehat{g^t}, s^t), \widehat{\boldsymbol{w}^t})$ with $\widehat{g^t} = (g^t, \boldsymbol{A}^t, \boldsymbol{r}^t, \boldsymbol{d}^t)$ and $\widehat{w^t_j} = (w^t_j, v^t_j, u^t_j)$ for every j in $[N]$. We say that the stack j is *active* at time t if $s^t = j$. Then, we define $\mathsf{dead\text{-}alive}$ tag to be dead if and only if the stack is not active at or after the corresponding position. The idea of the closure as we discussed is to maintain the set of subformulas that hold true at each step. We will expect it to be the empty set if the stack is dead. As for $\mathsf{willreturn\text{-}noreturn}$ tag, it reflects whether a call action has a "matching" return. This is similar to the $\{\infty, ret\}$ tags in [1]. This may be done by defining tag to be $\mathsf{noreturn}$ if stack will never become smaller than what it is now. Finally, the formulas and $\mathsf{willreturn\text{-}noreturn}$ tag on the stack are defined to be what they were in the global state at the time when the corresponding letter was pushed.

$$\forall t \geq 0,\ j \in [N] : (d^t_j = \mathsf{dead}) \ \overset{\text{def}}{\Leftrightarrow}\ (\forall t' \geq t,\ s^{t'} \neq j). \tag{1}$$

$$\forall t \geq 0,\ j \in [N] \ \text{ with } d^t_j = \mathsf{alive},\ \psi \in \mathrm{Cl}(\phi):$$

$$\psi \in A^t_j \overset{\text{def}}{\Leftrightarrow} \rho, t' \models \psi \text{ where } t' \text{ is the least } t' \geq t \text{ with } s^{t'} = j. \tag{2}$$

$$\forall t \geq 0,\ j \in [N] \ \text{ with } d^t_j = \mathsf{dead}\text{: } A^t_j \overset{\text{def}}{=} \emptyset. \tag{3}$$

$$\forall t \geq 0,\ j \in [N] : (r^t_j = \mathsf{noreturn}) \overset{\text{def}}{\Leftrightarrow} (\forall t' \geq t,\ \left|w^{t'}_j\right| \geq \left|w^t_j\right|). \tag{4}$$

$$\forall t \geq 0,\ j \in [N] : v^t_j \overset{\text{def}}{=} A^{t_1}_j A^{t_2}_j \dots A^{t_l}_j \text{ and } u^t_j \overset{\text{def}}{=} d^{t_1}_j d^{t_2}_j \dots d^{t_l}_j,$$

$$\text{where for } k \text{ in } [l]\text{: } t_k \text{ is largest } t_k \leq t \text{ such that } \left|w^{t_k}_j\right| = k - 1. \tag{5}$$

Construction. We construct a system which simulates the original system with accepting runs having augmentations faithful to the semantics described above in (1)-(5). We define the multi-pushdown system $\widehat{P}$ as $(\widehat{G} \times [N], N, \widehat{\Gamma}, \widehat{\boldsymbol{\Delta}})$ with the states and alphabet as defined earlier, and each transition relation $\widehat{\Delta_s}$ is defined s.t. $(\widehat{g}, s, \widehat{a}, \widehat{g'}, s', \mathfrak{a}(\widehat{a'}))$ is in $\widehat{\Delta_s} \overset{\text{def}}{\Leftrightarrow}$ conditions from Fig. 1 are satisfied. The set $\mathcal{F}$ is defined by the following sets of accepting states:

(a) For each $\psi = \phi_1 \mathsf{U} \phi_2 \in \mathrm{Cl}(\phi)$, we define $F^1_\psi \overset{\text{def}}{=} \{(\widehat{g}, s) \mid \phi_2 \in A_s \text{ or } \psi \notin A_s\}$.

(b) For each abstract-until formula $\psi = \phi_1 \mathsf{U}^{\mathsf{a}}_s \phi_2 \in \mathrm{Cl}(\phi)$, we define $F^2_\psi \overset{\text{def}}{=} \{(\widehat{g}, s) \mid r_s = \mathsf{noreturn} \text{ and } (\phi_2 \in A_s \text{ or } \psi \notin A_s)\}$.

(c) For each $j \in [N]$, we define $F^3_j \overset{\text{def}}{=} \{(\widehat{g}, s) \mid j = s\} \cup \{(\widehat{g}, s) \mid d_j = \mathsf{dead}\}$.

(d) For each $j \in [N]$, $F^4_j \overset{\text{def}}{=} \{(\widehat{g}, s) \mid d_j = \mathsf{dead}\} \cup \{(\widehat{g}, s) \mid j = s, d_s = \mathsf{noreturn}\}$.

Lemma 2. *Let ρ be a run of P. Then, $\gamma(\rho)$ is a run of $\widehat{P}$ such that for every $F \in \mathcal{F}$, there is a global state in F that is repeated infinitely often.*

1. $((g,s), a_S, (g',s'), \mathfrak{a}(a'_S)) \in \Delta_s$
2. $d_s = \mathsf{alive}$
3. $\forall j \neq s$, $d_j = d'_j$
4. If $\mathfrak{a} = \mathsf{call}$, then $r_s = \mathsf{willreturn} \Rightarrow r'_s = \mathsf{willreturn}$ and $a'_r = r_s$
5. If $\mathfrak{a} = \mathsf{internal}$, then $r'_s = r_s$ and $a'_r = a_r$
6. If $\mathfrak{a} = \mathsf{return}$, then $r_s = \mathsf{willreturn}$ and $r'_s = a_r$
7. $\forall j \neq s$, $r_j = r'_j$
8. $(g,s) \in A_s$
9. $\forall j \neq s$, $A_j = A'_j$
10. $\mathsf{X}A_s \subseteq A'_{s'}$ $(= A_{s'})$
11. If $\mathfrak{a} = \mathsf{call}$, then $a'_A = A_s$
12. If $\mathfrak{a} = \mathsf{internal}$, then $\mathsf{X}^{\mathsf{a}}_s A_s \subseteq A'_s$ and $a'_A = a_A$.
13. If $\mathfrak{a} = \mathsf{return}$, then $\mathsf{X}^{\mathsf{a}}_s a_A \subseteq A'_s$
14. Further, $\mathsf{X}^{\mathsf{a}}_s A_s = \emptyset$ if
    (a) $\mathfrak{a} = \mathsf{call}$ and $r_s = \mathsf{noreturn}$, or
    (b) $\mathfrak{a} = \mathsf{return}$, or
    (c) $d'_s = \mathsf{dead}$
15. If $\mathfrak{a} = \mathsf{call}$, $\mathsf{X}^{\mathsf{c}}_s A'_{s'} = (\mathsf{X}^{\mathsf{c}}_s \operatorname{Atoms}(\phi)) \cap A_s$
16. If $\mathfrak{a} = \mathsf{internal}$, then $\mathsf{X}^{\mathsf{c}}_s A'_{s'} = \mathsf{X}^{\mathsf{c}}_s A_s$.
17. Let $b \in \{\mathsf{a}, \mathsf{c}\}$. Let $\psi \in \mathrm{Cl}(\phi)$, $\psi = \phi_1 \mathsf{U}^b_s \phi_2$. Then, $\psi \in A_s$ iff either $\phi_2 \in A_s$ or $(\phi_1 \in A_s$ and $\mathsf{X}^{\mathsf{a}}_s \psi \in A_s)$.
18. Let $b \in \{\mathsf{a}, \mathsf{c}\}$. Let $\psi \in \mathrm{Cl}(\phi)$, $\psi = \phi_1 \mathsf{U}^b_j \phi_2$ with $j \neq s$. Then $\psi \in A_s$ iff $\psi \in A'_s$.
19. $\forall j$: If $d_j = \mathsf{dead}$, then $A_j = \emptyset$ and $r_j = \mathsf{noreturn}$.

Fig. 1. Conditions for $\widehat{\Delta_s}$. $\mathsf{X}A = \{\psi \mid \mathsf{X}\psi \in A\}$, $\mathsf{X}^{\mathsf{a}}_1 A = \{\psi \mid \mathsf{X}^{\mathsf{a}}_1 \psi \in A\}$ and $\widehat{a} = (a_S, a_A, a_r)$ (similarly, $\widehat{a'}$).

Lemma 3. *Let $\widehat{\rho}$ be a run of $\widehat{P}$ satisfying the acceptance condition $\mathcal{F}$. Then, $\widehat{\rho}$ projected over states of P, denoted $\Pi(\widehat{\rho})$, is a run of P and $\gamma(\Pi(\widehat{\rho})) = \widehat{\rho}$.*

From Lemmas 2 and 3 the soundness and completeness of the reduction follow if we define the set of new initial states I_0 for the REP problem as states with initial state (g_0, i_0) for the MC problem and ϕ present in the part tracking formulas that hold true: $I_0 = \{((g_0, \boldsymbol{A}, \boldsymbol{d}, \boldsymbol{r}), i_0) \mid \phi \in A_{i_0}\}$. This gives an exponential-time reduction from MC to REP as well as their bounded variants. Theorem 4 below can be viewed as a counterpart of [14, Theorem 3].

Theorem 4. *Let P be a MPDS with initial configuration $(g, (\bot)^N)$ and ϕ be a Multi-*`CaRet` *formula. Let $\widehat{P}$ be the system built from P, g and ϕ, I_0 be the associated set of initial states and $\mathcal{F}$ be the acceptance condition. **(I)** If ρ_1 is a run of P from $(g, (\bot)^N)$ then $\rho_2 = \gamma(\rho_1)$ is a run of $\widehat{P}$ satisfying $\mathcal{F}$ and (A)-(C) hold true. **(II)** If ρ_2 is a run of $\widehat{P}$ from some configuration with global state in I_0 and satisfying $\mathcal{F}$, then $\Pi(\rho_2)$ is a run of P and (A)-(C) hold true too.*

*Conditions (A)–(C) are defined as follows: **(A)** ρ_1 is k-bounded iff ρ_2 is k-bounded, for all $k \geq 0$; **(B)** ρ_1 is k-phase-bounded iff ρ_2 is k-phase-bounded, for all $k \geq 0$; **(C)** ρ_1 is $\preceq$-bounded iff ρ_2 is $\preceq$-bounded, for all total orderings of the stacks $\preceq = ([N], \leq)$.*

Note that at each position, ρ_1 and ρ_2 work on the same stack and perform the same type of action (call, return, internal move), possibly with slightly different letters. This is sufficient to guarantee the satisfaction of the conditions (A)–(C).

5 Complexity Analysis with Bounded Runs

Bounded Repeated Global State Reachability Problem. We evaluate the complexity of BREP as well as its variant restricted to a single accepting global state, written $\mathrm{BREP}_{\mathrm{single}}$. There is a logspace reduction from BREP to $\mathrm{BREP}_{\mathrm{single}}$ by copying the MPDS as many times as the cardinality of $\mathcal{F}$ (as done to reduce non-emptiness problem for generalized Büchi automata to non-emptiness problem for standard Büchi automata). This allows us to conclude about the complexity upper bound for BMC itself but it is worth noting that the MPDS obtained by synchronization has an exponential number of global states and therefore a refined complexity analysis is required to get optimal upper bounds. In order to analyze the complexity for $\mathrm{BREP}_{\mathrm{single}}$, we take advantage of proof techniques that are introduced earlier and for which we provide a complexity analysis that will suit our final goal. Namely, existence of an infinite k-bounded run s.t. a final global state (g_f, i_f) is repeated infinitely often is checked: (1) by first guessing a sequence of intermediate global states witnessing context switches of length at most $k+1$, (2) by computing the (regular) set of reachable configurations following that sequence and then (3) by verifying whether there is a reachable configuration leading to an infinite run s.t. (g_f, i_f) is repeated infinitely often and no stack switch is performed. The principle behind (2) is best explained in [17] but we provide a complexity analysis using the computation of $\mathrm{post}^{\star}(X)$ along the lines of [18]. Sets $\mathrm{post}^{\star}(X)$ need to be computed at most k times, which might cause an exponential blow-up (for instance if at each step the number of states were multiplied by a constant). Actually, computing $\mathrm{post}^{\star}$ adds an additive factor at each step, which is essential for our analysis. Let us define $\mathrm{BREP}_{\mathrm{single}}$: it takes as inputs P, a configuration $\left((g,i),(\bot)^N\right)$, a global state (g_f, i_f) and $k \in \mathbb{N}$ and it asks whether there is an infinite k-bounded run ρ from $\left((g,i),(\bot)^N\right)$ s.t. (g_f, i_f) is repeated infinitely often.

Proposition 5. *$\mathrm{BREP}_{\mathrm{single}}$ can be solved in time $\mathcal{O}(|P|^{k+1} \times p(k,|P|))$ for some polynomial $p(\cdot,\cdot)$.*

The proof of Proposition 5 is at the heart of our complexity analysis and it relies on constructions from [8,18]. We take advantage of it with the input system $\widehat{P}$.

Corollary 6. *(I) BMC with k encoded with a unary representation is* ExpTime*-complete. (II) BMC with k in binary encoding is in 2*ExpTime.

Note that [6, Theorem 15] would lead to an ExpTime upper bound for BMC if k is not part of the input, see the ExpTime upper bound for the problem NESTED-TRACE-SAT($\mathcal{L}^-, k$) from [6]; in our case k *is* indeed part of the input and in that case, the developments in [6] will lead to a 2ExpTime bound by using the method used for NESTED-TRACE-SAT($\mathcal{L}^-, k$) even if k is encoded in unary. Indeed, somewhere in the proof, the path expression $succ_{\leq k}$ is exponential in the value k. Hence, Corollary 6(I) is the best we can hope for when k is part of the input of the model-checking problem. We write BMC^{reg} to denote the extension of BMC in which Multi-`CaRet` is replaced by Multi-$\texttt{CaRet}^{reg}$.

Corollary 7. *(I) BMC^{reg} with k encoded with an unary representation is* ExpTime*-complete. (II) BMC^{reg} with k in binary encoding is in 2*ExpTime*.*

Complexity Results for Other Boundedness Notions. We focus on the complexity analysis for OBMC and PBMC. Let OREP$_{\text{single}}$ be the variant of BREP$_{\text{single}}$ with ordered MPDS: it takes as inputs an ordered multi-pushdown system P, a configuration $\big((g,i),(\bot)^N\big)$, a global state (g_f,i_f) and it asks whether there is an infinite run ρ from $\big((g,i),(\bot)^N\big)$ such that (g_f,i_f) is repeated infinitely often. According to [2, Theorem 11], OREP$_{\text{single}}$ restricted to ordered multi-pushdown systems with k stacks can be checked in time $\mathcal{O}(|P|^{2^{d\ k}})$ where d is a constant. Our synchronized product $\widehat{P}$ is exponential in the size of formulas (see Section 4), whence OBMC is in 2ExpTime too (k is linear in the size of our initial P). Condition (C) from Theorem 4 needs to be used here.

Corollary 8. *OBMC is in* 2ExpTime*.*

The same complexity upper bound can be shown with regularity constraints.

Now, let us consider k-bounded-phase runs. Let us define PBREP$_{\text{single}}$ in a similar way: it takes as inputs a MPDS P, a configuration $\big((g,i),(\bot)^N\big)$, a global state (g_f,i_f) and $k \in \mathbb{N}$ and it asks whether there is an infinite k-phase-bounded run ρ from $\big((g,i),(\bot)^N\big)$ such that (g_f,i_f) is repeated infinitely often. In [3, Section 5], it is shown that non-emptiness for k-phase MPDS can be reduced to non-emptiness for ordered MPDS with $2k$ stacks. By inspecting the proof, we can conclude: a similar reduction can be performed for reducing the repeated reachability of a global state, and non-emptiness of k-phase P with N stacks is reduced to non-emptiness of one of N^k instances of P' with $2k$ stacks and each P' is in polynomial-size in $k+|P|$. Therefore, PBREP$_{\text{single}}$ is in 2ExpTime. Indeed, there is an exponential number of instances and checking non-emptiness for one of them can be done in double exponential time. By combining the different complexity measures above, checking an instance of PBREP$_{\text{single}}$ with $\widehat{P}$ requires time in $\mathcal{O}(N^k \times \left|\widehat{P}\right|^{2^{d\ 2k}})$ which is double-exponential in the size of P. Consequently, bounded model-checking with bounded-phase MPDS is in 2ExpTime too if the number of phases is encoded in unary.

Corollary 9. *(I) PBMC where k is encoded in unary is in* 2ExpTime*. (II) PBMC where k is encoded in binary is in* 3ExpTime*.*

Again, the same complexity upper bounds apply when regularity constraints are added. Note that an alternative proof of Corollary 9(I) can be found in the recent paper [7] where fragments of MSO are taken into account.

6 Conclusion

We showed that model-checking over MPDS with k-bounded runs is ExpTime-complete when k is an input bound encoded in unary, otherwise the problem is in 2ExpTime with a binary encoding. The logical language is a version of

CaRet in which abstract temporal operators are related to calls and returns and parameterized by the stacks, and regularity constraints on stack contents are present too. A 2ExpTime upper bound is also established with ordered MPDS or with k-phase bounded runs.

Acknowledgments. We thank the reviewers for their time and helpful comments. K. Bansal also thanks LSV & ENS Cachan (France), and Clark Barrett in New York, for making the internship in Summer 2011 and hence, this work, possible.

References

1. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *TACAS'04*, vol. 2988 of *LNCS*, pp. 467–481. Springer, 2004.
2. M. Atig. Global model checking of ordered multi-pushdown systems. In *FST&TCS'10*, pp. 216–227. LIPICS, 2010.
3. M. Atig, B. Bollig, and P. Habermehl. Emptiness of multi-pushdown automata is 2ETIME-complete. In *DLT'08*, vol. 5257 of *LNCS*, pp. 121–133. Springer, 2008.
4. M. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan. Linear-time model-cheking for multithreaded programs under scope-bounding. In *ATVA'12*, vol. 7561 of *LNCS*, pp. 152–166. Springer, 2012.
5. K. Bansal and S. Demri. A note on the complexity of model-checking bounded multi-pushdown systems. Technical Report TR2012-949, NYU, Dec 2012.
6. B. Bollig, A. Cyriac, P. Gastin, and M. Zeitoun. Temporal logics for concurrent recursive programs: Satisfiability and model checking. In *MFCS'11*, vol. 6907 of *LNCS*, pp. 132–144, 2011.
7. B. Bollig, D. Kuske, and R. Mennicke. The complexity of model-checking multi-stack systems. 2012. Submitted.
8. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model-checking. In *CONCUR'97*, vol. 1243 of *LNCS*, pp. 135–150. Springer, 1997.
9. A. Cyriac, P. Gastin, and K. N. Kumar. MSO decidability of multi-pushdown systems via split-width. In *CONCUR'12*, vol. 7454 of *LNCS*, pp. 547–561, 2012.
10. J. Esparza and P. Ganty. Complexity of pattern-based verification for multithreaded programs. In *POPL'11*, pp. 499–510. ACM, 2011.
11. J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. In *TACS'01*, vol. 2215 of *LNCS*, pp. 316–339, 2001.
12. S. La Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS'07*, pp. 161–170. IEEE, 2077.
13. S. La Torre and M. Napoli. Reachability of multistack pushdown systems with scope-bounded matching relations. In *CONCUR'11*, vol. 6901 of *LNCS*, pp. 203–218. Springer, 2011.
14. S. La Torre and M. Napoli. A temporal logic for multi-threaded programs. In *TCS 2012*, vol. 7604 of *LNCS*, pp. 225–239. Springer, 2012.
15. S. La Torre and G. Parlato. Scope-bounded multistack pushdown systems: fixed-point, sequentialization and tree-width. In *FSTTCS'12*, pp. 173–184. LIPICS, 2012.
16. P. Madhusudan and G. Parlato. The tree width of auxiliary storage. In *POPL'11*, pp. 283–294. ACM, 2011.
17. S. Qaader and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS'05*, vol. 3440 of *LNCS*, pp. 93–107. Springer, 2005.
18. S. Schwoon. *Model-checking pushdown systems.* PhD thesis, TUM, 2002.