Model Checking Dynamic Pushdown Networks*

Fu Song[1] and Tayssir Touili[2]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
fsong@sei.ecnu.edu.cn
[2] LIAFA, CNRS and Université Paris Diderot
touili@liafa.univ-paris-diderot.fr

Abstract. A Dynamic Pushdown Network (DPN) is a set of pushdown systems (PDSs) where each process can dynamically create new instances of PDSs. DPNs are a natural model of multi-threaded programs with (possibly recursive) procedure calls and thread creation. Thus, it is important to have model-checking algorithms for DPNs. We consider in this work model-checking DPNs against single-indexed LTL and CTL properties of the form $\bigwedge f_i$ s.t. f_i is a LTL/CTL formula over the PDS i. We consider the model-checking problems w.r.t. simple valuations (i.e, whether a configuration satisfies an atomic proposition depends only on its control location) and w.r.t. regular valuations (i.e., the set of the configurations satisfying an atomic proposition is a regular set of configurations). We show that these model-checking problems are decidable. We propose automata-based approaches for computing the set of configurations of a DPN that satisfy the corresponding single-indexed LTL/CTL formula.

1 Introduction

Multithreading is a commonly used technique for modern software. However, multithreaded programs are known to be error prone and difficult to analyze. Dynamic Pushdown Networks (DPN) [4] are a natural model of multi-threaded programs with (possibly recursive) procedure calls and thread creation. A DPN consists of a finite set of pushdown systems (PDSs), each of them models a sequential program (process) that can dynamically create new instances of PDSs. Therefore, it is important to investigate automated methods for verifying DPNs. While existing works concentrate on the reachability problem of DPNs [4,18,17,9,15,24], model checking for the Linear Temporal Logic (LTL) and the Computation Tree Logic (CTL) which can describe more interesting properties of program behaviors has not been tackled yet for DPNs.

In general, the model checking problem is undecidable for double-indexed properties, i.e., properties where atomic propositions are interpreted over the control states of two or more threads [11]. This undecidability holds for pushdown networks even without thread creation. To obtain decidable results, in this paper, we consider single-indexed LTL and CTL model checking for DPNs, where a single-index LTL or CTL formula is a formula of the form $\bigwedge f_i$ such that f_i is a LTL/CTL formula over the PDS i.

* This work is partially funded by ANR grant ANR-08-SEGI-006, Shanghai Knowledge Service Platform for Trustworthy Internet of Things No. ZF1213, NSFC Project No.91118007, Civil Aerospace Project 125 and NSFC Project No.61021004.

C.-c. Shan (Ed.): APLAS 2013, LNCS 8301, pp. 33–49, 2013.

A DPN satisfies $\bigwedge f_i$ iff every PDS i that runs in the network satisfies the subformula f_i. We first consider LTL model-checking for DPNs with simple valuations where whether a configuration of a PDS i satisfies an atomic proposition depends only on the control state of the configuration. Then, we consider LTL model-checking for DPNs with regular valuations where the set of configurations of a PDS satisfying an atomic proposition is a regular set of configurations. Finally, we consider CTL model-checking for DPNs with simple and regular valuations. We show that these model-checking problems are decidable. We propose automata-based approaches for computing the set of configurations of a DPN that satisfy the corresponding single-indexed LTL/CTL formula.

It is non-trivial to do LTL/CTL model checking for DPNs, since the number of instances of PDSs can be unbounded. Checking independently whether all the different PDSs satisfy the corresponding subformula f_i is not correct. Indeed, we do not need to check whether an instance of a PDS j satisfies f_j if this instance is not created during a run. To solve this problem, we extend the automata-based approach for standard LTL/CTL model-checking for PDSs [2,8,7,20]. For every process i, we compute a finite automaton $\mathcal{A}_i$ recognizing all the configurations from which there exists a run σ of the process i that satisfies f_i. $\mathcal{A}_i$ also memorizes the set of all the initial configurations of the instances of PDSs that are dynamically created during the run σ. Then, to check whether a DPN satisfies a single-indexed LTL/CTL formula, it is sufficient to check whether the initial configurations of the processes are recognized by the corresponding finite automata and whether the set of generated instances of PDSs that are stored in the automata also satisfy the formula. This condition is recursive. To solve it, we compute the largest set $\mathcal{D}_{fp}$ of the dynamically created initial configurations that satisfy the formula f. Then, to check whether a DPN satisfies f, it is sufficient to check whether the initial configurations of the different processes are recognized by the corresponding finite automata and whether the dynamically created initial configurations that are stored in the automata are in $\mathcal{D}_{fp}$.

To compute the finite automata $\mathcal{A}_i s$, we extend the automata-based approaches for standard LTL [2,7,8] and CTL [20] model-checking for PDSs. For every i, $1 \leq i \leq n$, we construct a Büchi Dynamic PDS (resp. alternating Büchi Dynamic PDS) which is a synchronization of the PDS i and the LTL (resp. CTL) formula f_i. Büchi Dynamic PDS (resp. alternating Büchi Dynamic PDS) is an extension of Büchi PDS (resp. alternating Büchi PDS) with the ability to create new instances of PDSs during the run. The finite automata $\mathcal{A}_i s$ we are looking for correspond to the languages accepted by these Büchi Dynamic PDSs (resp. alternating Büchi Dynamic PDSs). Then, we show how to solve these language problems and compute the finite automata $\mathcal{A}_i s$.

Related Work. The DPN model was introduced in [4]. Several other works use DPN and its extensions to model multi-threaded programs [4,9,17,18,24]. All these works only consider reachability issues. Ground Tree Rewrite Systems [10] and process rewrite systems [5,19] are two models of multi-threaded programs with procedure calls and threads creation. However, [19] only considers reachability problem and [10,5] only consider subclasses of LTL. We consider LTL and CTL model checking problems.

Pushdown networks with communication between processes are studied in [3,6,1,22]. These works consider systems with a fixed number of threads. [15,16] use

parallel flow graphs to model multi-threaded programs. However, all these works only consider reachability. [25] considers safety properties of multi-threaded programs.

[11,12,13] study single-index LTL/CTL and double-indexed LTL model checking problems for networks of pushdown systems that synchronize via a finite set of nested locks. [14] considers model-checking on properties that are expressed in a kind of finite automata for such networks of pushdown systems. These works don't consider dynamic threads creation.

Outline. Section 2 gives the basic definitions. Section 3 and Section 4 show LTL and CTL model-checking for DPNs, respectively. Due to lack of space, proofs are omitted and can be found in the full version of this paper [21].

2 Preliminaries

2.1 Dynamic Pushdown Networks

Definition 1. *A Dynamic Pushdown Network (DPN) $\mathcal{M}$ is a set $\{\mathcal{P}_1, ..., \mathcal{P}_n\}$ s.t. for every i, $1 \leq i \leq n$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a dynamic pushdown system (DPDS), where P_i is a finite set of control locations s.t. $P_k \cap P_i = \emptyset$ for $k \neq i$, Γ_i is the stack alphabet, and Δ_i is a finite set of transition rules in the following forms: (a) $q\gamma \hookrightarrow p_1\omega_1$ or (b) $q\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2$ s.t. $q, p_1 \in P_i, \gamma \in \Gamma_i, \omega_1 \in \Gamma_i^*, p_2\omega_2 \in P_j \times \Gamma_j^*$ for some j, $1 \leq j \leq n$.*

A *global configuration* of $\mathcal{M}$ is a *multiset* $\mathcal{G}$ over $\bigcup_{i=1}^{n} P_i \times \Gamma_i^*$. Each element $q\omega \in P_i \times \Gamma_i^* \cap \mathcal{G}$ denotes that an instance of $\mathcal{P}_i$ running in parallel in the network is at the *local configuration* $q\omega$, i.e., $\mathcal{P}_i$ is at the control location q and its stack content is ω. If $\omega = \gamma u$ for $\gamma \in \Gamma_i$ and there is in Δ_i a transition (a) $q\gamma \hookrightarrow p_1\omega_1$ or (b) $q\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2$ s.t. $p_2\omega_2 \in P_j \times \Gamma_j$, then the instance of $\mathcal{P}_i$ can move from $q\omega$ to the control location p_1 and replace γ by ω_1 at the top of its stack, i.e., $\mathcal{P}_i$ moves to $p_1\omega_1 u$. The other instances in parallel in the network stay at the same local configurations. In addition, transition (b) will create a new instance of $\mathcal{P}_j$ starting from $p_2\omega_2$. Formally, a DPDS $\mathcal{P}_i$ induces an *immediate successor relation* $\Longrightarrow_i$ as follows: for every $\omega \in \Gamma_i^*$, if $q\gamma \hookrightarrow p_1\omega_1 \in \Delta_i$, then $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega$; if $q\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2 \in \Delta_i$, then $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega \triangleright \{p_2\omega_2\}$. To unify the presentation, if $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega$, we sometimes write $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega \triangleright \emptyset$ instead. The transitive and reflexive closure of $\Longrightarrow_i$ is denoted by $\Longrightarrow_i^*$. Formally, for every $p\omega \in P_i \times \Gamma_i^*$, $p\omega \Longrightarrow_i^* p\omega \triangleright \emptyset$; and if $p\omega \Longrightarrow_i p_1\omega_1 \triangleright D_1$ and $p_1\omega_1 \Longrightarrow_i^* p_2\omega_2 \triangleright D_2$, then $p\omega \Longrightarrow_i^* p_2\omega_2 \triangleright D_1 \cup D_2$. $\Longrightarrow_i^+$ is defined as usual.

A DPDS $\mathcal{P}_i$ can be seen as a pushdown system (PDS) with the ability of dynamically creating new instances of PDSs. The initial local configuration of a newly created instance is called DCLIC (for Dynamically Created Local Initial Configuration).

A local run of an instance of $\mathcal{P}_i$ from a local configuration c_0 is a sequence of local configurations $c_0c_1...$ over $\mathcal{P}_i \times \Gamma_i^*$ s.t. for every $j \geq 0$, $c_j \Longrightarrow_i c_{j+1} \triangleright D$ for some D. A global run ρ of $\mathcal{M}$ from a global configuration $\mathcal{G}$ is a (potentially infinite) set of local runs. Initially, ρ contains exactly the local runs starting from the local configurations in $\mathcal{G}$. Whenever a DCLIC c is created by some local run of ρ, a new local run starting from c is added into ρ. For every i, $1 \leq i \leq n$, let $\wp(\sigma) = i$ iff σ is a local run of an instance of $\mathcal{P}_i$, and $\wp(p\omega) = \wp(p) = i$ iff $p \in P_i$. Let $\mathcal{D}_i = \{p_2\omega_2 \in \bigcup_{i=1}^{n} P_i \times \Gamma_i^* \mid q\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2 \in \Delta_i\}$ be the set of potential DCLICs of the DPDS $\mathcal{P}_i$.

2.2 LTL and Büchi Automata

From now on, we fix a set of atomic propositions AP.

Definition 2. *The set of LTL formulas is given by (where $a \in$ AP):*

$$\psi ::= a \mid \neg\psi \mid \psi \wedge \psi \mid \boldsymbol{X}\psi \mid \psi \boldsymbol{U} \psi.$$

Given an ω-word $\eta = \alpha_0\alpha_1...$ over 2^{AP}, let $\eta(k)$ denote α_k, and η_k denote the *suffix* of η starting from α_k. $\eta \models \psi$ (η satisfies ψ) is inductively defined as follows: $\eta \models a$ iff $a \in \eta(0)$; $\eta \models \neg\psi$ iff $\eta \not\models \psi$; $\eta \models \psi_1 \wedge \psi_2$ iff $\eta \models \psi_1$ and $\eta \models \psi_2$; $\eta \models \mathbf{X}\psi$ iff $\eta_1 \models \psi$; $\eta \models \psi_1 \mathbf{U} \psi_2$ iff there exists $k \geq 0$ such that $\eta_k \models \psi_2$ and for every j, $1 \leq j < k$, $\eta_j \models \psi_1$.

Definition 3. *A* Büchi automaton *(BA) $\mathcal{B}$ is a tuple $(G, \Sigma, \theta, g^0, F)$ where G is a finite set of states, Σ is the input alphabet, $\theta \subseteq G \times \Sigma \times G$ is a finite set of transitions, $g^0 \in G$ is the initial state and $F \subseteq G$ is a finite set of accepting states.*

A run of $\mathcal{B}$ over an ω-word $\alpha_0\alpha_1...$ is a sequence of states $q_0q_1...$ s.t. $q_0 = g^0$ and $(q_i, \alpha_i, q_{i+1}) \in \theta$ for every $i \geq 0$. A run is accepting iff it infinitely often visits some states in F.

It is well-known that given a LTL formula f, one can construct a BA B_f s.t. $\Sigma = 2^{AP}$ recognizing all the ω-words that satisfy f [23].

2.3 Single-Indexed LTL for DPNs

Let $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$ be a DPN. A *single-indexed* LTL formula is a formula f of the form $\bigwedge_{i=1}^{n} f_i$ s.t. for every i, $1 \leq i \leq n$, f_i is a LTL formula in which the validity of the atomic propositions depends only on the DPDS $\mathcal{P}_i$. Let $\lambda : AP \longrightarrow 2^{\bigcup_{i=1}^{n} P_i \times \Gamma_i^*}$ be a valuation which assigns to each atomic proposition a set of local configurations. A local run $p_0\omega_0 p_1\omega_1...$ of $\mathcal{P}_i$ satisfies f_i iff the ω-word $\alpha_0\alpha_1...$ where for every $j \geq 0$, $\alpha_j = \{a \in AP \mid p_j\omega_j \in \lambda(a)\}$, satisfies f_i. A local configuration c of $\mathcal{P}_i$ satisfies f_i iff $\mathcal{P}_i$ has a local run σ from c that satisfies f_i. If D is the set of DCLICs created during the run σ, we write $c \models_D f_i$. $\mathcal{M}$ satisfies f iff it has a global run ρ such that for every i, $1 \leq i \leq n$, each local run of $\mathcal{P}_i$ in ρ satisfies the formula f_i.

2.4 Multi-automata and Predecessors

From now on, we fix a DPN $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$ where for every i, $1 \leq i \leq n$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$, and a single-indexed LTL formula $f = \bigwedge_{i=1}^{n} f_i$. To check whether $\mathcal{M}$ satisfies f is non-trivial. Indeed, it is not correct to check independently whether each $\mathcal{P}_i$ satisfies f_i. Instead, we need to check whether there exists a global run ρ from a global configuration $\mathcal{G}$ s.t. an instance of $\mathcal{P}_i$ satisfies the formula f_i only if it is an instance in $\mathcal{G}$ or it is dynamically created during the run ρ. Thus, it is important to memorize the set of DCLICs that are created during a run. To this aim, we introduce the function $pre_{\mathcal{P}_i} : 2^{P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}} \longrightarrow 2^{P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}}$ as follows. $pre_{\mathcal{P}_i}(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times \Gamma_i^*,\ s.t.\ c \Longrightarrow_i c' \rhd D_1$ and $(c', D_2) \in U\}$. Intuitively, if $\mathcal{P}_i$ moves from c to c' and generates the DCLIC D_1 and $(c', D_2) \in U$, then $(c, D_1 \cup D_2) \in pre_{\mathcal{P}_i}(U)$. The transitive and reflexive closure of $pre_{\mathcal{P}_i}$ is denoted by $pre^*_{\mathcal{P}_i}$. Formally, $pre^*_{\mathcal{P}_i}(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times \Gamma_i^*,\ s.t.\ c \Longrightarrow_i^* c' \rhd D_1$ and $(c', D_2) \in U\}$. Let $pre^+_{\mathcal{P}_i}(U) = pre^*_{\mathcal{P}_i}(pre_{\mathcal{P}_i}(U))$.

To finitely represent (infinite) sets of local configurations of DPDSs and DCLICs generated by DPDSs, we use Multi-automata and Alternating Multi-automata.

Definition 4. *An Alternating Multi-automaton* (AMA) *is a tuple* $\mathcal{A}_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$*, where* Q_i *is a finite set of states,* $I_i \subseteq P_i$ *is a finite set of initial states corresponding to the control locations of the DPDS* $\mathcal{P}_i$*,* $Acc_i \subseteq Q_i$ *is a finite set of final states,* $\delta_i \subseteq (Q_i \times \Gamma_i) \times 2^{\mathcal{D}_i} \times 2^{Q_i}$ *is a finite set of transition rules.*
A MA is a AMA $\mathcal{A}_i$ *s.t.* $\delta_i \subseteq (Q_i \times \Gamma_i) \times 2^{\mathcal{D}_i} \times Q_i$.

We write $p \xrightarrow{\gamma/D}_i \{q_1, ..., q_m\}$ instead of $(p, \gamma, D, \{q_1, ..., q_m\}) \in \delta_i$, where D is a set of DCLICs. We define the relation $\longrightarrow_i^* \subseteq (Q_i \times \Gamma_i^*) \times 2^{\mathcal{D}_i} \times 2^{Q_i}$ as the smallest relation s.t.: (1) $q \xrightarrow{\epsilon/\emptyset}_i^* \{q\}$ for every $q \in Q_i$, (2) if $q \xrightarrow{\gamma/D}_i \{q_1, ..., q_m\}$ and $q_k \xrightarrow{\omega/D_k}_i^* S_k$ for k, $1 \leq k \leq m$, then $q \xrightarrow{\gamma\omega/D \cup \bigcup_{k=1}^m D_k}_i^* \bigcup_{k=1}^m S_k$. Let $L(\mathcal{A}_i)$ be the set of tuples $(p\omega, D) \in P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ s.t. $p \xrightarrow{\omega/D}_i^* S$ for some $S \subseteq Acc_i$. A set $W \subseteq P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ is *regular* iff there exists an AMA $\mathcal{A}_i$ s.t. $L(\mathcal{A}_i) = W$. A set of local configurations $C \subseteq P_i \times \Gamma_i^*$ is *regular* iff $C \times \{\emptyset\}$ is a regular set.

Given a DPDS $\mathcal{P}_i$ and a regular set $W \subseteq P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ accepted by a MA $A_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$, we can construct a MA $A_i^{pre^*} = (Q_i, \Gamma_i, \delta_i', I_i, Acc_i)$ that exactly accepts $pre^*_{\mathcal{P}_i}(W)$. W.l.o.g., we assume that A_i has no transition leading to an initial state and that $P_i = I_i$. $A_i^{pre^*}$ is constructed by the following saturation procedure (an adaption of the saturation procedure of [2]).

- *For every* $p\gamma \hookrightarrow p_1\omega_1 \in \Delta_i$ *and* $p_1 \xrightarrow{\omega_1/D_1}_i^* q$*, add a new rule* $p \xrightarrow{\gamma/D}_i q$;
- *For every* $p\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2 \in \Delta_i$ *and* $p_1 \xrightarrow{\omega_1/D_1}_i^* q$*, add a new rule* $p \xrightarrow{\gamma/D \cup \{p_2\omega_2\}}_i q$.

The procedure adds only new transitions to A_i. Since the number of states is fixed, the number of possible new transitions is finite. Thus, the saturation procedure always terminates. We can show that each transition can be processed only once. Thus, the number of transition rules added into $A_i^{pre^*}$ is at most $O(|\Delta_i| \cdot |Q_i|^2 \cdot 2^{|\mathcal{D}_i|})$. The intuition behind this procedure is that, for every $\omega' \in \Gamma_i^*$: suppose $p\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2 \in \Delta_i$ and the tuple $(p_1\omega_1\omega', D)$ is accepted by the automaton, i.e., $p_1 \xrightarrow{\omega_1/D_1}_i^* q \xrightarrow{\omega'/D_2}_i^* g$ for some $g \in Acc_i$ and $D = D_1 \cup D_2$. Then, we add the new transition rule $p \xrightarrow{\gamma/D_1 \cup \{p_2\omega_2\}}_i q$ that allows the automaton to accept $(p\gamma\omega', D \cup \{p_2\omega_2\})$, i.e., $p \xrightarrow{\gamma/D_1 \cup \{p_2\omega_2\}}_i q \xrightarrow{\omega'/D_2}_i^* g$. The case $p\gamma \hookrightarrow p_1\omega_1 \in \Delta_i$ is similar. Thus, we obtain the following theorem.

Theorem 1. *Given a MA* A_i *recognizing a regular set* W *of the DPDS* $\mathcal{P}_i$*, we can construct a MA* $A_i^{pre^*}$ *recognizing* $pre^*_{\mathcal{P}_i}(W)$ *in time* $O(|\Delta_i| \cdot |Q_i|^2 \cdot 2^{|\mathcal{D}_i|})$.

3 Single-Indexed LTL Model-Checking for DPNs

In this section, we consider LTL model checking w.r.t. a labeling function $l : \bigcup_{i=1}^n P_i \longrightarrow 2^{AP}$ assigning to each control location a set of atomic propositions. In this case, the valuation λ_l (called simple valuation) is defined as follows: for every $a \in AP$, $\lambda_l(a) = \{p\omega \in \bigcup_{i=1}^n P_i \times \Gamma_i^* \mid a \in l(p)\}$. A global configuration $\mathcal{G}$ satisfies $f = \bigwedge f_i$ iff $\mathcal{M}$

has a global run ρ from $\mathcal{G}$ s.t. every local run σ of ρ satisfies $f_{\wp(\sigma)}$ where $\wp(\sigma)$ denotes the index of the DPDS which corresponds to the local run σ. Checking whether $\mathcal{G}$ satisfies f is non-trivial since the number of local runs of ρ can be unbounded. We cannot check all the different instances of the DPDSs independently. Indeed, we don't have to check whether an instance of $\mathcal{P}_i$ (for some $i, 1 \leq i \leq n$) satisfies f_i if this instance is not created during the execution. We can solve this problem in a naive way as follows: Given an initial global configuration $\mathcal{G}$, we can guess the set of DCLICs $D \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ which are created in a global run from $\mathcal{G}$ such that the global run satisfies f. Then, it is sufficient to check that every local configuration $c \in \mathcal{G} \cup D$ satisfies the LTL formula $f_{\wp(c)}$ when disallowing the transition rules which create a DCLIC outside of D and discarding the DCLICs inside of D. Checking whether c satisfies $f_{\wp(c)}$ could be solved by LTL model-checking for PDSs [2,7] if we discard the DCLICs of the DPDS. However, this naive technique is very complicated as it necessitates an exponential number of calls to the LTL model checking algorithm of PDSs. Moreover, it is very complex. We have to consider all the possible sets of DCLICs whose number is at most $O(2^{|\bigcup_{i=1}^{n} \mathcal{D}_i|})$, and for each set D of DCLICs, we have to perform at most $O(|\bigcup_{i=1}^{n} \mathcal{D}_i|)$ times of LTL model-checking algorithm for PDSs, where LTL model-checking for PDSs is in time $O(|P_{\wp(d)}|^2 \cdot |\Delta_{\wp(d)}| \cdot 2^{|f_{\wp(d)}|})$ [2,7]. Thus, the complexity of checking whether $\mathcal{G}$ satisfies f or not will be $O(2^{|\bigcup_{i=1}^{n} \mathcal{D}_i|} \cdot \sum_{d \in \bigcup_{i=1}^{n} \mathcal{D}_i \cup \mathcal{G}} (|P_{\wp(d)}|^2 \cdot |\Delta_{\wp(d)}| \cdot 2^{|f_{\wp(d)}|}))$.

To overcome these problems, we propose in this section a *direct* algorithm. We compute for every i, $1 \leq i \leq n$, a MA $\mathcal{A}_i$ such that $(c, D) \in L(\mathcal{A}_i)$, where c is a local configuration of $\mathcal{P}_i$ and $D \subseteq \mathcal{D}_i$ is a set of DCLICs, iff $\mathcal{P}_i$ has a local run σ from c that satisfies f_i such that D is the set of DCLICs created during the local run σ. Then, a global configuration $\mathcal{G}$ satisfies $f = \bigwedge f_i$ iff for every configuration $c \in \mathcal{G}$, there exists a set of DCLICs D_c s.t. $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$ and every $d \in D_c$ satisfies f. This condition is recursive. However, it can be effectively checked since there is only a finite number of DCLICs. Checking this condition naively is not efficient. To obtain a more efficient procedure, we compute the largest set $\mathcal{D}_{fp} \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ of DCLICs such that $d \in \mathcal{D}_{fp}$ iff d is a DCLIC and there exists a global run of $\mathcal{M}$ starting from d that satisfies f. Then, to check whether a global configuration $\mathcal{G}$ satisfies f, it is sufficient to check for every $c \in \mathcal{G}$ whether there exists $D_c \subseteq \mathcal{D}_{fp}$ s.t. $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$.

3.1 Computing the MAs $\mathcal{A}_i$

To compute the MAs $\mathcal{A}_i$, for i, $1 \leq i \leq n$, we extend the automata-based approach for standard LTL model-checking for PDSs [2,7]. We first compute a Büchi automaton (BA) $\mathcal{B}_i$ that corresponds to the formula f_i, for i, $1 \leq i \leq n$. Then, we synchronize the BAs with the DPDSs to obtain Büchi DPDSs. The MAs $\mathcal{A}_i$ we are looking for correspond to the languages accepted by these Büchi DPDSs.

Definition 5. *A Büchi DPDS (BDPDS) is a tuple $\mathcal{BP}_i = (P_i, \Gamma_i, \Delta_i, F_i)$, where $(P_i, \Gamma_i, \Delta_i)$ is a DPDS and $F_i \subseteq P_i$ is a finite set of accepting control locations.*

A BDPDS is a kind of DPDS with a Büchi acceptance condition F_i. Runs of a BDPDS are defined as local runs for DPDSs. A run σ of $\mathcal{BP}_i$ is accepting iff σ *infinitely often* visits some control locations in F_i. Let $L(\mathcal{BP}_i)$ be the set of all the pairs $(c, D) \in P_i \times$

$\Gamma_i^* \times 2^{\mathcal{D}_i}$ s.t. $\mathcal{BP}_i$ has an accepting run from c and the run generates the set of DCLICs D.

Let $\mathcal{B}_i = (G_i, 2^{AP}, \theta_i, g_i^0, F_i)$ be the BA recognizing all the ω-words that satisfy f_i. We compute a BDPDS $\mathcal{BP}_i$ such that $\mathcal{P}_i$ has a local run from $p\omega$ that satisfies f_i and generates a set of DCLICs D iff $([p, g_i^0]\omega, D) \in L(\mathcal{BP}_i)$. We define $\mathcal{BP}_i = (P_i \times G_i, \Gamma_i, \Delta_i', F_i')$ as follows: for every $p \in P_i$, $[p, g] \in F_i'$ iff $g \in F_i$; and for every $(g_1, l(p), g_2) \in \theta_i$, we have:

1. $[p, g_1]\gamma \hookrightarrow [p_1, g_2]\omega_1 \in \Delta_i'$ iff $p\gamma \hookrightarrow p_1\omega_1 \in \Delta_i$;
2. $[p, g_1]\gamma \hookrightarrow [p_1, g_2]\omega_1 \rhd D \in \Delta_i'$ iff $p\gamma \hookrightarrow p_1\omega_1 \rhd D \in \Delta_i$.

Intuitively, $\mathcal{BP}_i$ is a product of $\mathcal{P}_i$ and the BA $\mathcal{B}_i$. $\mathcal{B}_i$ has an accepting run $g_0 g_1 ...$ over an ω-word $l(p_0)l(p_1)...$ that corresponds to a local run $\sigma = p_0\omega_0\ p_1\omega_1 ...$ of $\mathcal{P}_i$ iff $\mathcal{BP}_i$ has an accepting run $\sigma' = [p_0, g_0]\omega_0\ [p_1, g_1]\omega_1 ...$, and D is the set of DCLICs created during the run σ iff D is the set of DCLICs created during the run σ'. Suppose the run of $\mathcal{P}_i$ is at $p_j\omega_j$, then the run of $\mathcal{B}_i$ can move from g_j to g_{j+1} iff $(g_j, l(p_j), g_{j+1}) \in \theta_i$. This is ensured by Items 1 and 2 expressing that $\mathcal{BP}_i$ can move from $[p_j, g_j]\omega_j$ to $[p_{j+1}, g_{j+1}]\omega_{j+1}$ iff $(g_j, l(p_j), g_{j+1}) \in \theta_i$. The accepting control locations $F_i' = \{[p, g] \mid p \in P_i, g \in F_i\}$ ensures that the run of $\mathcal{B}_i$ visits infinitely often some states in F_i iff the run of $\mathcal{BP}_i$ visits infinitely often some control locations F_i'. Item 2 ensures that the run of $\mathcal{P}_i$ creates a DCLIC $p_2\omega_2$ iff the run of $\mathcal{BP}_i$ creates this DCLIC. Thus, we obtain the following theorem.

Lemma 1. *$\mathcal{P}_i$ has a local run from $p\omega$ that satisfies f_i and creates a set of DCLICs D iff $([p, g_i^0]\omega, D) \in L(\mathcal{BP}_i)$, where $\mathcal{BP}_i$ can be constructed in time $O(|\Delta_i| \cdot 2^{|f_i|})$.*

The complexity follows from the fact that the number of transition rules of $\mathcal{BP}_i$ is at most $O(|\Delta_i| \cdot 2^{|f_i|})$.

Computing $L(\mathcal{BP}_i)$: Let us fix an index i, $1 \leq i \leq n$. We show that computing $L(\mathcal{BP}_i)$ boils down to $pre^*_{\mathcal{P}_i}$ computations.

Proposition 1. *Let $\mathcal{BP}_i = (P_i, \Gamma_i, \Delta_i, F_i)$ be a BDPDS, $\mathcal{BP}_i$ has an accepting run from $c \in P_i \times \Gamma_i^*$ and D is the set of DCLICs created during this run iff $\exists D_1, D_2, D_3 \subseteq \mathcal{D}_i$ s.t. $D = D_1 \cup D_2 \cup D_3$, and*

$(\alpha_1) : c \Longrightarrow_i^* p\gamma\omega \rhd D_1$ for some $\omega \in \Gamma_i^*$;

$(\alpha_2) : p\gamma \Longrightarrow_i^+ gu \rhd D_2$ and $gu \Longrightarrow_i^* p\gamma v \rhd D_3$, for some $g \in F_i, v \in \Gamma_i^*$.

Intuitively, an accepting run from c will reach a configuration $p\gamma\omega$ (Item α_1) followed by a repeatedly executed cycle (Item α_2) which is a sequence of configurations with an accepting location g. The execution of the cycle returns to the control location p with the same symbol γ at the top of the stack. The rest of the stack will never be popped during this cycle. Repeatedly executing the cycle yields an accepting run (since $g \in F_i$) and the set of DCLICs generated during this cycle is $D_2 \cup D_3$. Thus, the set of DCLICs created by the accepting run starting from c is $D_1 \cup D_2 \cup D_3$. To compute $L(\mathcal{BP}_i)$, we reformulate the above conditions as follows:

Proposition 2. *Let $\mathcal{BP}_i = (P_i, \Gamma_i, \Delta_i, F_i)$ be a BDPDS, $\mathcal{BP}_i$ has an accepting run from $c \in P_i \times \Gamma_i^*$ and D is the set of DCLICs created during this run iff $\exists D_1, D_2' \subseteq \mathcal{D}_i$ s.t. $D = D_1 \cup D_2'$, and*

$(\beta_1) : (c, D_1) \in pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\})$;

$(\beta_2) : (p\gamma, D_2') \in pre^+_{\mathcal{P}_i}((F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}) \cap pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\}))$ *(note that $D_2' = D_2 \cup D_3$).*

Intuitively, items β_1 and β_2 are reformulations of items α_1 and α_2, respectively. By Proposition 2, we can get that $L(\mathcal{BP}_i) = \{(c, D_1 \cup D_2') \in P_i \times \Gamma_i \times 2^{\mathcal{D}_i} \mid$ Items β_1 and β_2 hold$\}$. Since $F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ and $\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\}$ are regular sets, using Theorem 1, we can construct two MAs A' and A'' accepting $pre^+_{\mathcal{P}_i}((F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}) \cap pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\}))$ and $pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\})$. The intersection $(F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}) \cap pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\})$ is easy to compute. Since $F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ denotes all the configurations whose control locations are accepting, we only need to let the initial states of A'' be the states of F_i. Since the set $P_i \times \Gamma_i \times 2^{\mathcal{D}_i}$ is finite, we can determine all the tuples $(p\gamma, D_2') \in P_i \times \Gamma_i \times 2^{\mathcal{D}_i}$ s.t. Item β_2 holds. The set of pairs (c, D_1) is the union of all the sets $pre^*_{\mathcal{P}_i}(\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\})$. Thus, we can get $L(\mathcal{BP}_i)$. For every BDPDS $\mathcal{P}_i$ and MA A_i, $pre^*_{\mathcal{P}_i}(L(A_i))$ and $pre^+_{\mathcal{P}_i}(L(A_i))$ can be computed in time $O(|\Delta_i| \cdot |Q_i|^2 \cdot 2^{|\mathcal{D}_i|})$, where $|Q_i| = O(|P_i|)$. Thus, we get that:

Lemma 2. *For every BDPDS $\mathcal{BP}_i = (P_i, \Gamma_i, \Delta_i, F_i)$, we can construct a MA $\mathcal{A}_i$ in time $O(|\Delta_i| \cdot |\Gamma_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|})$ such that $L(\mathcal{A}_i) = L(\mathcal{BP}_i)$.*

From Lemma 1 and Lemma 2, we get:

Theorem 2. *Given a DPN $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a labelling function l, we can compute MAs $\mathcal{A}_1, ..., \mathcal{A}_n$ in time $O(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|}))$ s.t. for every i, $1 \leq i \leq n$, every $p\omega \in P_i \times \Gamma_i^*$ and $D \subseteq \mathcal{D}_i$, $p\omega \models_D f_i$ iff $([p, g_i^0]\omega, D) \in L(\mathcal{A}_i)$.*

3.2 Single-Indexed LTL Model-Checking for DPNs with Simple Valuations

Given a DPN $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$ and a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$, by Theorem 2, we can construct a set of MAs $\{\mathcal{A}_1, ..., \mathcal{A}_n\}$ s.t. for every i, $1 \leq i \leq n$, and every local configuration $p\omega \in P_i \times \Gamma_i^*$, $p\omega \models_D f_i$ iff $([p, g_i^0]\omega, D) \in L(\mathcal{A}_i)$. Then, to check whether a global configuration $\mathcal{G}$ satisfies f, we need to check whether for every local configuration $c \in \mathcal{G}$, there exists a set of DCLICs D_c s.t. $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$ and every DCLIC $d \in D_c$ satisfies f, i.e., there exists a set of DCLICs D_d s.t. $(d, D_d) \in L(\mathcal{A}_{\wp(d)})$, etc. This condition is recursive. It can be solved, because the number of DCLICs is finite. To obtain a more efficient procedure, we compute the maximal set of DCLICs $\mathcal{D}_{fp}$ s.t. for every $d \in \bigcup_{i=1}^n \mathcal{D}_i$, d satisfies f iff $d \in \mathcal{D}_{fp}$. Then, to check whether $\mathcal{G}$ satisfies f, it is sufficient to check whether for every $c \in \mathcal{G}$, there exists $D_c \subseteq \mathcal{D}_{fp}$ s.t. $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$.

Let $\{\mathcal{A}_1, ..., \mathcal{A}_n\}$, s.t. for every i, $1 \leq i \leq n$, $\mathcal{A}_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$, be the set of the computed MAs. Intuitively, $\mathcal{D}_{fp}$ should be equal to the set of local configurations $p\omega \in \bigcup_{i=1}^n \mathcal{D}_i$ s.t. there exists $D \subseteq \mathcal{D}_{fp}$ s.t. $p\omega \models_D f_{\wp(p)}$, i.e., $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$. Thus, $\mathcal{D}_{fp}$ can be defined as the greatest fixpoint of the

function $F(X) = \{p\omega \in \mathcal{D}_I \mid \exists D \subseteq X \text{ s.t. } ([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$. This set can then be computed iteratively as follows: $\mathcal{D}_{fp} = \bigcap_{j\geq 0} D_j$, where $D_0 = \mathcal{D}_I$ and $D_{j+1} = \{p\omega \in \mathcal{D}_I \mid \exists D \subseteq D_j, ([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$ for every $j \geq 0$. Since $\bigcup_{i=1}^n \mathcal{D}_i$ is a finite set, and for every $j \geq 0$, D_{j+1} is a subset of D_j, there always exists a fixpoint $m \geq 0$ such that $D_m = D_{m+1}$. Then, we can get that $\mathcal{D}_{fp} = D_m$.

For every $p\omega \in \bigcup_{i=1}^n \mathcal{D}_i$ and $D \subseteq \mathcal{D}_{\wp(p)}$, to avoid checking whether $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$ at each step when computing $D_0, D_1, ...$, we can compute all these tuples that satisfy this condition once and store them in a hash table. We can show that whether or not $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$ can be decided in time $\mathbf{O}(|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|})$. Thus, we can get the hash table in time $\mathbf{O}(\sum_{p\omega\in\bigcup_{i=1}^n \mathcal{D}_i}(|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$. Given D_j and the hash table, we can compute D_{j+1} in time $\mathbf{O}(\sum_{p\omega\in\bigcup_{i=1}^n \mathcal{D}_i} 2^{|\mathcal{D}_{\wp(p)}|})$. Thus we can get $\mathcal{D}_{fp}$ in time $\mathbf{O}(\sum_{p\omega\in\mathcal{D}_I}(|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_I|}) + |\mathcal{D}_I|^2 \cdot 2^{|\mathcal{D}_I|})$.

Theorem 3. *We can compute $\mathcal{D}_{fp}$ in time $\mathbf{O}(\sum_{p\omega\in\bigcup_{i=1}^n \mathcal{D}_i}(|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|} + |\bigcup_{i=1}^n \mathcal{D}_i| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$ s.t. for every $c \in \bigcup_{i=1}^n \mathcal{D}_i$, c satisfies the single-indexed LTL formula f iff $c \in \mathcal{D}_{fp}$.*

Then, from Theorem 3 and Theorem 2, we get the following theorem.

Theorem 4. *Given a DPN $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a labelling function l, we can compute MAs $\mathcal{A}_1, ..., \mathcal{A}_n$ in time $O(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|}))$ s.t. for every global configuration $\mathcal{G}$, $\mathcal{G}$ satisfies f iff for every $p\omega \in \mathcal{G}$, there exists $D \subseteq \mathcal{D}_{fp}$ s.t. $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$.*

You can see that the complexity of our technique is better than the one of the naive approach given at the beginning of Section 3.

3.3 Single-Indexed LTL Model-Checking with Regular Valuations

We generalize single-indexed LTL model checking for DPNs w.r.t. simple valuations to a more general model checking problem where the set of configurations in which an atomic proposition holds is a regular set of local configurations. Formally, a regular valuation is a function $\lambda : AP \longrightarrow 2^{\bigcup_{i=1}^n P_i \times \Gamma_i^*}$ s.t. for every $a \in AP$, $\lambda(a)$ is a regular set of local configurations of $\mathcal{P}_i$ for i, $1 \leq i \leq n$. The previous construction can be extended to deal with this case. For this, we follow the approach of [8]. We compute, for i, $1 \leq i \leq n$, a new DPDS $\mathcal{P}'_i$, which is a kind of synchronization of the DPDS $\mathcal{P}_i$ and the *deterministic* finite automata corresponding to the regular valuations. This allows to determine whether atomic propositions hold at a given step by looking only at the top of the stack of $\mathcal{P}'_i$, for every i, $1 \leq i \leq n$. By doing this, we can reduce single-indexed LTL model checking for DPNs with regular valuations to single-indexed LTL model checking for DPNs with simple valuations. Due to lack of space, we omit the details. They can be found in the full version of this paper [21].

Theorem 5. *Given a DPN $\mathcal{M} = \{\mathcal{P}_1, ..., \mathcal{P}_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a regular valuation λ, we can compute MAs $\mathcal{A}_1, ..., \mathcal{A}_n$ in time $\mathbf{O}(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |States_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|}))$ s.t. for every global configuration $\mathcal{G}$, $\mathcal{G}$ satisfies f iff for every $p\omega \in \mathcal{G}$, there exists $D \subseteq \mathcal{D}_{fp}$ s.t. $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$, where $|States_i|$ denotes the number of states of the automata corresponding to the regular valuation λ.*

4 Single-Indexed CTL Model Checking for DPNs

In this section, we consider single-indexed CTL model-checking for DPNs with regular valuations. Single-indexed CTL model-checking for DPNs with simple valuations is a special case.

4.1 Single-Indexed CTL

For technical reasons, we suppose that CTL formulas are given in positive normal form, i.e., only atomic propositions are negated. Indeed, any CTL formula can be translated into positive normal form by pushing the negations inside. Moreover, we use the *release* operator $\mathbf{R}$ as the dual of the until operator $\mathbf{U}$. Let AP be a finite set of atomic propositions. The set of CTL formulas is given by (where $a \in AP$):

$$\psi ::= a \mid \neg a \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathbf{AX}\psi \mid \mathbf{EX}\psi \mid \mathbf{A}[\psi \mathbf{U} \psi] \mid \mathbf{E}[\psi \mathbf{U} \psi] \mid \mathbf{A}[\psi \mathbf{R} \psi] \mid \mathbf{E}[\psi \mathbf{R} \psi].$$

The other standard CTL operators can be expressed by the above operators. E.g., $\mathbf{EF}\psi = \mathbf{E}[true\mathbf{U}\psi]$, $\mathbf{AF}\psi = \mathbf{A}[true\mathbf{U}\psi]$, $\mathbf{EG}\psi = \mathbf{E}[false\mathbf{R}\psi]$ and $\mathbf{AG}\psi = \mathbf{A}[false\mathbf{R}\psi]$. The closure $cl(\psi)$ of ψ is the set of all the subformulas of ψ including ψ. Let $At(\psi) = \{a \in AP \mid a \in cl(\psi)\}$ and $cl_{\mathbf{R}}(\psi) = \{\phi \in cl(\psi) \mid \phi = \mathbf{E}[\psi_1 \mathbf{R} \psi_2] \text{ or } \phi = \mathbf{A}[\psi_1 \mathbf{R} \psi_2]\}$.

Let $\lambda : AP \rightarrow 2^{\bigcup_{i=1}^{n} P_i \times \Gamma_i^*}$ a regular valuation assigning to each atomic proposition a regular set of local configurations. A local configuration c satisfies a CTL formula f_i, (denoted $c \models^{\lambda} f_i$), iff there exists $D \subseteq \mathcal{D}_i$ s.t. $c \models_D^{\lambda} f_i$ holds, where $\models_D^{\lambda}$ is inductively defined in Figure 1. Intuitively, $c \models_D^{\lambda} f_i$ means that c satisfies f_i and the executions that made c satisfy f_i create the set of DCLICs D, i.e., when a transition rule $q\gamma \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2$ is used to make f_i satisfied, $p_2\omega_2$ is in D. We write $c \models_D f_i$ instead of $c \models_D^{\lambda} f_i$ when λ is clear from the context.

A *single-indexed* CTL formula f is a formula of the form $\bigwedge f_i$ s.t. for every i, $1 \leq i \leq n$, f_i is a CTL formula in which the validity of the atomic propositions depends only on the DPDS $\mathcal{P}_i$. A global configuration $\mathcal{G}$ satisfies $f = \bigwedge f_i$ iff for every $c \in \mathcal{G}$, there exists a set of DCLICs $D \subseteq \mathcal{D}_{\wp(c)}$ s.t. $c \models_D f_{\wp(c)}$ and for every $d \in D$, d also satisfies f.

4.2 Alternating BDPDSs

Definition 6. *An Alternating BDPDS (ABDPDS) is a tuple $\mathcal{BP}'_i = (P'_i, \Gamma_i, \Delta'_i, F_i)$, where P'_i is a finite set of control locations, Γ_i is the stack alphabet, $F_i \subseteq P'_i$ is a set of accepting control locations, Δ'_i is a finite set of transition rules in the form of $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \{q_1u_1, ..., q_ku_k\}$ s.t. $p\gamma \in P'_i \times \Gamma_i, \{p_1\omega_1, ..., p_h\omega_h\} \subseteq P'_i \times \Gamma_i^*$ and $\{q_1u_1, ..., q_ku_k\} \subseteq \mathcal{D}_i$.*

An ABDPDS $\mathcal{BP}'_i$ induces a relation $\mapsto_i \subseteq (P'_i \times \Gamma_i^*) \times (2^{P'_i \times \Gamma^*} \times 2^{\mathcal{D}_i})$ defined as follows: for every $\omega \in \Gamma_i^*$, if $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \{q_1u_1, ..., q_ku_k\} \in \Delta_i$, then $p\gamma\omega \mapsto_i \{p_1\omega_1\omega, ..., p_h\omega_h\omega\} \triangleright \{q_1u_1, ..., q_ku_k\}$. Intuitively, if $\mathcal{BP}'_i$ is at the configuration $p\gamma\omega$, it can fork into h copies in the configurations $p_1\omega_1\omega, ..., p_h\omega_h\omega$ and creates k new instances of ABDPDSs starting from the DCLICs $q_1u_1, ..., q_ku_k$, respectively. We sometimes write $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\}$ if $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \emptyset \in \Delta_i$.

$c \models^{\lambda}_{\emptyset} a \iff c \in \lambda(a)$;
$c \models^{\lambda}_{\emptyset} \neg a \iff c \notin \lambda(a)$;
$c \models^{\lambda}_{D} \psi_1 \wedge \psi_2 \iff \exists D_1, D_2 \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i \ s.t.\ D = D_1 \cup D_2,\ c \models^{\lambda}_{D_1} \psi_1$ and $c \models^{\lambda}_{D_2} \psi_2$;
$c \models^{\lambda}_{D} \psi_1 \vee \psi_2 \iff c \models^{\lambda}_{D} \psi_1$ or $c \models^{\lambda}_{D} \psi_2$;
$c \models^{\lambda}_{D} \mathbf{AX}\ \psi \iff$ For every $c_1, ..., c_m \in P_i \times \Gamma_i^*$ s.t. for j , $1 \leq j \leq m, \exists D_j, D'_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i,\ c \Longrightarrow_i c_j \rhd D'_j,\ c_j \models^{\lambda}_{D_j} \psi$ and $D = \bigcup_{j=1}^{m} (D_j \cup D'_j)$;
$c \models^{\lambda}_{D} \mathbf{EX}\ \psi \iff$ There exist $c' \in P_i \times \Gamma_i^*,\ D', D'' \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i\ s.t.\ c \Longrightarrow_i c' \rhd D'',\ c' \models^{\lambda}_{D'} \psi$ and $D = D' \cup D''$;
$c \models^{\lambda}_{D} \mathbf{A}[\psi_1 \mathbf{U} \psi_2] \iff$ For every path $\sigma = c_0 c_1 ...$ with $c_0 = c$, for every $m \geq 1, \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i,\ s.t.\ c_{m-1} \Longrightarrow c_m \rhd D'_m$, and $\exists k \geq 0,\ s.t.\ \exists D_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i, c_k \models^{\lambda}_{D_k} \psi_2, \forall j,\ 0 \leq j < k, c_j \models^{\lambda}_{D_j} \psi_1$ and $D = \bigcup_{\sigma} (\bigcup_{j=1}^{k} D'_j \cup \bigcup_{j=0}^{k} D_j)$;
$c \models^{\lambda}_{D} \mathbf{E}[\psi_1 \mathbf{U} \psi_2] \iff$ There exists a path $\sigma = c_0 c_1 ...$ with $c_0 = c$, for every $m \geq 1,\ \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and $\exists k \geq 0,\ s.t.\ \exists D_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i,\ c_k \models^{\lambda}_{D_k} \psi_2, \forall j,\ 0 \leq j < k, c_j \models^{\lambda}_{D_j} \psi_1$, and $D = \bigcup_{j=1}^{k} D'_j \cup \bigcup_{j=0}^{k} D_j$;
$c \models^{\lambda}_{D} \mathbf{A}[\psi_1 \mathbf{R} \psi_2] \iff$ For every path $\sigma = c_0 c_1 ...$ with $c_0 = c$, for every $m \geq 1,\ \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and either $\forall j \geq 0, \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i,\ c_j \models^{\lambda}_{D_j} \psi_2$ and $D_\sigma = \bigcup_{j \geq 1} D'_j \cup \bigcup_{j \geq 0} D_j$, or $\exists k \geq 0,\ \exists D''_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i\ s.t.\ c_k \models^{\lambda}_{D''_k} \psi_1$ and $\forall j,\ 0 \leq j \leq k,\ \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i, c_j \models^{\lambda}_{D_j} \psi_2$, $D_\sigma = \bigcup_{j=0}^{k} D_j \cup D''_k \cup \bigcup_{j=1}^{k} D'_j . D = \bigcup_{\sigma} D_\sigma$;
$c \models^{\lambda}_{D} \mathbf{E}[\psi_1 \mathbf{R} \psi_2] \iff$ There exists a path $\sigma = c_0 c_1 ...$ with $c_0 = c$, for every $m \geq 1, \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and either $\forall j \geq 0,\ \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i, c_j \models^{\lambda}_{D_j} \psi_2$ and $D = \bigcup_{j \geq 1} D'_j \cup \bigcup_{j \geq 0} D_j$, or $\exists k \geq 0, \exists D''_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i\ s.t.\ c_k \models^{\lambda}_{D''_k} \psi_1$ and $\forall j,\ 0 \leq j \leq k, \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i, c_j \models^{\lambda}_{D_j} \psi_2$, and $D = \bigcup_{j=0}^{k} D_j \cup D''_k \cup \bigcup_{j=1}^{k} D'_j$.

Fig. 1. Semantics of CTL

A run of $\mathcal{BP}'_i$ from a configuration $p\omega \in P'_i \times \Gamma_i^*$ is a tree rooted by $p\omega$, the other nodes are labeled by elements of $P'_i \times \Gamma_i^*$. If a node is labelled by qu whose children are $p_1\omega_1, ..., p_m\omega_m$, then, necessarily, $qu \mapsto \{p_1\omega_1, ..., p_m\omega_m\} \rhd D$ for some $D \subseteq \mathcal{D}_i$. The run is *accepting* iff each branch of this run *infinitely often* visits some control locations in F_i. Let $L(\mathcal{BP}'_i)$ be the set of all the pairs $(c, D) \in P'_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ s.t. $\mathcal{BP}'_i$ has an accepting run from c and that creates the set of DCLICs D.

4.3 Computing Corresponding Alternating BDPDSs

To perform single-indexed CTL model-checking for DPNs with regular valuations, we follow the approach for LTL model-checking for DPNs. But, in this case, we need alternating MAs and Alternating BDPDSs, since CTL formulas can be translated to alternating Büchi automata. We compute a set of AMAs $\mathcal{A}'_1, ..., \mathcal{A}'_n$ s.t. for every i, $1 \leq i \leq n$ and every local configuration $p\omega$ of $\mathcal{P}_i$, $p\omega \models_D f_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{A}'_i)$. Later, we compute the largest set of DCLICs $\mathcal{D}'_{fp}$ such that a DCLIC d satisfies f iff $d \in \mathcal{D}'_{fp}$. Then, to check whether a global configuration $\mathcal{G}$ satisfies f, it is sufficient to check whether for every $p\omega \in \mathcal{G}$, there exists $D \subseteq \mathcal{D}'_{fp}$ s.t. $([p, f_{\wp(p)}]\omega, D) \in L(\mathcal{A}'_{\wp(p)})$. To compute the AMAs, we construct a set of alternating BDPDSs $\mathcal{BP}'_i$ which are synchronizations of the DPDSs $\mathcal{P}_i$ with formulas f_i s.t. the AMAs we are looking for correspond to the languages accepted by these alternating BDPDSs $\mathcal{BP}'_i s$. We first show how to compute the alternating BDPDSs $\mathcal{BP}'_i$. Then, we show how to compute the languages of these alternating BDPDSs $\mathcal{BP}_i s$, i.e. the AMAs.

We fix an index i, $1 \leq i \leq n$. We construct an ABDPDS $\mathcal{BP}'_i$ s.t. for every $p\omega \in P'_i \times \Gamma_i^*$, $p\omega \models_D f_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{BP}'_i)$. We suppose w.l.o.g. that the DPDS $\mathcal{P}_i$ has a bottom-of-stack $\sharp$ which is never popped from the stack. For every $a \in At(f_i)$, since

$\lambda(a)$ is a regular set of local configurations of $\mathcal{P}_i$, let $M_a = (Q_a, \Gamma_i, \delta_a, I_a, Acc_a)$ be a MA s.t. $L(M_a) = \lambda(a) \times \{\emptyset\}$, and $M_{\neg a} = (Q_{\neg a}, \Gamma_i, \delta_{\neg a}, I_{\neg a}, Acc_{\neg a})$ a MA s.t. $L(M_{\neg a}) = (P_i \times \Gamma_i^* \setminus \lambda(a)) \times \{\emptyset\}$, i.e., the set of configurations where a does not hold. To distinguish between all the initial states p in M_a and $M_{\neg a}$, we write p_a and $p_{\neg a}$ instead. W.l.o.g., we assume that the set of states $Q_a s$, and $Q_{\neg a} s$ are disjoint for every $a \in At(f_i)$.

Let $\mathcal{BP}'_i = (P'_i, \Gamma_i, \Delta'_i, F_i)$ be the ABDPDS such that $P'_i = P_i \times cl(f_i) \cup \bigcup_{a \in At(f_i)}(Q_a \cup Q_{\neg a})$; $F_i = P_i \times cl_{\mathbf{R}}(f_i) \cup \bigcup_{a \in At(f_i)}(Acc_a \cup Acc_{\neg a})$; and Δ'_i is the smallest set of transition rules s.t. for every control location $p \in P_i$, every subformula $\psi \in cl(f_i)$ and every $\gamma \in \Gamma_i$, we have:

1. if $\psi = a$ or $\psi = \neg a$, where $a \in At(f_i)$; $[p, \psi]\gamma \hookrightarrow \{p_\psi \gamma\} \in \Delta'_i$;
2. if $\psi = \psi_1 \wedge \psi_2$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_1]\gamma, [p, \psi_2]\gamma\} \in \Delta'_i$;
3. if $\psi = \psi_1 \vee \psi_2$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_1]\gamma\} \in \Delta'_i$ and $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma\} \in \Delta'_i$;
4. if $\psi = \mathbf{EX}\psi_1$; $[p, \psi]\gamma \hookrightarrow \{[p', \psi_1]\omega\} \rhd \{p''\omega'\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$; $[p, \psi]\gamma \hookrightarrow \{[p', \psi_1]\omega\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \in \Delta_i$;
5. if $\psi = \mathbf{AX}\psi_1$; $[p, \psi]\gamma \hookrightarrow \{[p', \psi_1]\omega \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \rhd \{p''\omega' \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \in \Delta'_i$;
6. if $\psi = \mathbf{E}[\psi_1 \mathbf{U} \psi_2]$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma\} \in \Delta'_i$, and $[p, \psi]\gamma \hookrightarrow \{[p, \psi_1]\gamma, [p', \psi]\omega\} \rhd \{p''\omega'\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$, $[p, \psi]\gamma \hookrightarrow \{[p, \psi_1]\gamma, [p', \psi]\omega\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \in \Delta_i$;
7. if $\psi = \mathbf{A}[\psi_1 \mathbf{U} \psi_2]$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma\} \in \Delta'_i$ and $[p, \psi]\gamma \hookrightarrow \{[p, \psi_1]\gamma, [p', \psi]\omega \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \rhd \{p''\omega' \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \in \Delta'_i$;
8. if $\psi = \mathbf{E}[\psi_1 \mathbf{R} \psi_2]$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma, [p, \psi_1]\gamma\} \in \Delta'_i$, and $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma, [p', \psi]\omega\} \rhd \{p''\omega'\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$, $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma, [p', \psi]\omega\} \in \Delta'_i$ if $p\gamma \hookrightarrow p'\omega \in \Delta_i$;
9. if $\psi = \mathbf{A}[\psi_1 \mathbf{R} \psi_2]$; $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma, [p, \psi_1]\gamma\} \in \Delta'_i$ and $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma, [p', \psi]\omega \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \rhd \{p''\omega' \mid p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i\} \in \Delta'_i$.
10. for every transition (q_1, γ, q_2) in $\bigcup_{a \in At(f_i)}(\delta_a \cup \delta_{\neg a})$; $q_1\gamma \hookrightarrow \{q_2\epsilon\} \in \Delta'_i$,
11. for every $q \in \bigcup_{a \in At(f_i)}(Acc_a \cup Acc_{\neg a})$; $q\sharp \hookrightarrow \{q\sharp\} \in \Delta'_i$.

For every $p\omega \in P'_i \times \Gamma_i^*$, $\mathcal{BP}'_i$ has an accepting run σ from $[p, f_i]\omega$ and D is the set of DCLICs created by σ iff $p\omega \models_D f_i$. The intuition behind each rule is explained as follows.

If $\psi = a \in At(f_i)$, for every $p\omega \in P'_i \times \Gamma_i^*$, $p\omega$ satisfies ψ iff $\mathcal{BP}'_i$ has an accepting run from $[p, a]\omega$. To check this, $\mathcal{BP}'_i$ moves to the initial state corresponding to p in M_a (i.e. p_a) by Item 1 allowing to check whether M_a accepts ω. Then the run of $\mathcal{BP}'_i$ from $p_a\omega$ mimics the run of M_a from the initial state p. Checking whether M_a accepts ω is ensured by Item 10. If $\mathcal{BP}'_i$ is at state q_1 with γ on the top of the stack and $q_1 \xrightarrow{\gamma} q_2$ is a transition of M_a, then $\mathcal{BP}'_i$ pops γ from the stack and moves the control location from q_1 to q_2. Popping γ from the stack allows to check the rest of the stack content. The configuration $p\omega$ is accepted by M_a iff the run of M_a reaches a final state $q \in Acc_a$, i.e., the run of $\mathcal{BP}'_i$ from $p\omega$ reaches the control location q with the empty stack, i.e., the stack only contains $\sharp$. Thus, $\mathcal{BP}'_i$ should have an infinite run from $q\sharp$ which infinitely often visits

some control locations in F_i. This is ensured by adding a loop on the configuration $q\sharp$ (Item 11) and adding q into F_i. The case $\psi = \neg a$ s.t. $a \in At(f_i)$ is similar.

If $\psi = \psi_1 \wedge \psi_2$, then, for every $p\omega \in P'_i \times \Gamma_i^*$, $p\omega$ satisfies ψ iff $p\omega$ satisfies ψ_1 and ψ_2. This is ensured by Item 2 stating that $\mathcal{BP}'_i$ has an accepting run from $[p, \psi_1 \wedge \psi_2]\omega$ iff $\mathcal{BP}'_i$ has an accepting run from $[p, \psi_1]\omega$ and $[p, \psi_2]\omega$. Item 3 is similar to Item 2.

Item 4 expresses that if $\psi = \mathbf{EX}\psi_1$, then, for every $p\gamma u \in P'_i \times \Gamma_i^*$ s.t. $\gamma \in \Gamma_i$, $p\gamma u$ satisfies ψ iff there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \triangleright p''\omega' \in \Delta_i$ such that $p'\omega u$ satisfies ψ_1. Thus, $\mathcal{BP}'_i$ should have an accepting run from $[p, \psi]\gamma u$ iff $\mathcal{BP}'_i$ has an accepting run from $[p', \psi_1]\omega u$. Moreover, if t_2 is the fired transition rule, the created DCLIC $p''\omega'$ should also be created by $\mathcal{BP}'_i$. Item 5 is analogous.

If $\psi = \mathbf{E}[\psi_1 \mathbf{U} \psi_2]$, then, for every $p\gamma u \in P'_i \times \Gamma_i^*$ s.t. $\gamma \in \Gamma_i$, $p\gamma u$ satisfies ψ iff either it satisfies ψ_2, or it satisfies ψ_1 and there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \triangleright p''\omega' \in \Delta_i$ such that $p'\omega u$ satisfies ψ. Thus, $\mathcal{BP}'_i$ has an accepting run from $[p, \psi]\gamma u$ iff either $\mathcal{BP}'_i$ has an accepting run from $[p, \psi_2]\gamma u$ or $\mathcal{BP}'_i$ has an accepting run from $[p, \psi_1]\gamma u$ and $[p', \psi]\omega u$. This is ensured by Item 6. Moreover, if t_2 is the fired transition rule, the created DCLIC $p''\omega'$ should also be created by $\mathcal{BP}'_i$. The case $\psi = \mathbf{A}[\psi_1 \mathbf{U} \psi_2]$ is analogous.

Item 8 expresses that if $\psi = \mathbf{E}[\psi_1 \mathbf{R} \psi_2]$, then, for every $p\gamma u \in P'_i \times \Gamma_i^*$ s.t. $\gamma \in \Gamma_i$, $p\gamma u$ satisfies ψ iff it satisfies ψ_2, and either it satisfies also ψ_1, or there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \triangleright p''\omega' \in \Delta_i$ such that $p'\omega u$ satisfies ψ. This guarantees that ψ_2 holds either always, or until both ψ_1 and ψ_2 hold. The fact that the state $[p, \psi]$ is in F_i ensures that paths where ψ_2 always hold are accepting. If t_2 is the fired transition rule, the created DCLIC $p''\omega'$ should also be created by $\mathcal{BP}'_i$. The intuition behind Item 9 is analogous to Item 8. Then, we obtain the following lemma.

Lemma 3. *For every i, $1 \leq i \leq n$, we can compute an ABDPDS $\mathcal{BP}'_i$ with $O(|P_i| \cdot |f_i| + \sum_{a \in At(f_i)}(|Q_a| + |Q_{\neg a}|))$ states and $O((|P_i| \cdot |\Gamma_i| + |\Delta_i|)|f_i| + \sum_{a \in At(f_i)}(|\delta_a| + |\delta_{\neg a}|))$ transition rules such that for every $(p\omega, D) \in P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$, $p\omega \models_D f_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{BP}'_i)$.*

4.4 Computing $L(\mathcal{BP}'_i)$

Let us fix an index i, $1 \leq i \leq n$, the AMA $\mathcal{A}'_i$ we are looking for corresponds to $L(\mathcal{BP}'_i)$. To compute this language, it is insufficient to simply compute the set of configurations from which $\mathcal{BP}'_i$ has an accepting run, since we also need to memorize the set of DCLICs created during the run of $\mathcal{BP}'_i$. To this aim, we follow the automata-based approach for CTL model-checking of PDSs presented in [20]. We first characterize the set $L(\mathcal{BP}'_i)$, then we compute the AMA $\mathcal{A}'_i$ such that $L(\mathcal{A}'_i) = L(\mathcal{BP}'_i)$.

Characterizing $L(\mathcal{BP}'_i)$: To characterize $L(\mathcal{BP}'_i)$, we introduce the function $pre_{\mathcal{BP}'_i} : 2^{P'_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}} \longrightarrow 2^{P'_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}}$ as follows: $pre_{\mathcal{BP}'_i}(U) = \{(c, D) \mid c \mapsto_i \{c_1, ..., c_m\} \triangleright D_0, \forall j : 1 \leq j \leq m, (c_j, D_j) \in U, \text{ and } D = \bigcup_{j=0}^{m} D_j\}$. The transitive and reflexive closure of $pre_{\mathcal{BP}'_i}$ is denoted by $pre^*_{\mathcal{BP}'_i}$. Formally, $pre^*_{\mathcal{BP}'_i}(U) = \{(c, D) \mid (c, D) \in U$ or there exist $c_1, ..., c_m$ *s.t.* $c \mapsto_i \{c_1, ..., c_m\} \triangleright D_0, \forall j : 1 \leq j \leq m, (c_j, D_j) \in pre^*_{\mathcal{BP}'_i}(U)$, and $D = \bigcup_{j=0}^{m} D_j\}$. Let $pre^+_{\mathcal{BP}'_i}(U) = pre^*_{\mathcal{BP}'_i}(pre_{\mathcal{BP}'_i}(U))$.

Let $Y_{\mathcal{BP}'_i} = \bigcap_{j \geq 1} Y_j$ where $Y_0 = P'_i \times \Gamma_i^* \times \{\emptyset\}$, $Y_{j+1} = pre^+_{\mathcal{BP}'_i}(Y_j \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$ for every $j \geq 0$. Intuitively, $(c, D) \in Y_1$ iff $\mathcal{BP}'_i$ has a run from c s.t. each path of this

Algorithm 1. Computation of $Y_{\mathcal{BP}'_i}$.

Input : An ABDPDS $\mathcal{BP}'_i = (P'_i, \Gamma_i, \Delta'_i, F_i)$;
Output: An AMA $\mathcal{A}'_i = (Q_i, \Gamma_i, \delta_i, I_i, \{q_f\})$ s.t. $L(\mathcal{A}'_i) = Y_{\mathcal{BP}'_i}$;

1 Let $k := 0, \delta_i := \{(q_f, \gamma, \emptyset, \{q_f\})$ for every $\gamma \in \Gamma_i\}$, and $\forall p \in P'_i$, $p^0 := q_f$;
2 **repeat** we call this loop $loop_1$
3 $\quad k := k + 1$;
4 $\quad$ Add a new transition rule $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$ in δ_i for every $p \in F_i$;
5 $\quad$ **repeat** we call this loop $loop_2$
6 $\quad\quad$ For every $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright D$ in Δ'_i,
7 $\quad\quad$ and every case $p_j^k \xrightarrow{\omega_j/D_j}_i{}^* R_j$ for all j, $1 \leq j \leq h$;
8 $\quad\quad\quad$ $p^k \xrightarrow{\gamma/D \cup \bigcup_{j=1}^h D_j}_i \bigcup_{j=1}^h R_j$ in δ_i
9 $\quad$ **until** *No new transition rule can be added*;
10 $\quad$ Remove from δ_i the transition rules $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$, $\forall p \in F_i$;
11 $\quad$ Replace in δ_i transition rule $p^k \xrightarrow{\gamma/D}_i R$ by $p^k \xrightarrow{\gamma/D}_i \pi^k(R)$, $\forall p \in P'_i, \gamma \in \Gamma_i, R \subseteq Q_i$;
12 **until** $k > 1$ *and* $\forall p \in P'_i, \gamma \in \Gamma_i, R \subseteq P'_i \times \{k\} \cup \{q_f\}, D \subseteq \mathcal{D}_i$, $p^k \xrightarrow{\gamma/D}_i R \in \delta_i$ *iff* $p^{k-1} \xrightarrow{\gamma/D}_i \pi^{-1}(R) \in \delta_i$;

run visits accepting control locations at least *once* and D is the set of DCLICs created during this run. $(c, D) \in Y_j$ iff $\mathcal{BP}'_i$ has a run from c s.t. each path of this run visits some control locations in F_i at least j times and D is the set of DCLICs created during this run. Since $Y_{\mathcal{BP}'_i} = \bigcap_{j \geq 1} Y_j$, for every $(c, D) \in Y_{\mathcal{BP}'_i}$, $\mathcal{BP}'_i$ has a run from c s.t. each path visits some control locations in F_i infinitely often and D is the set of all the DCLICs created during this run. Thus, we get:

Proposition 3. $L(\mathcal{BP}'_i) = Y_{\mathcal{BP}'_i}$.

Computing $Y_{\mathcal{BP}'_i}$: We show that $Y_{\mathcal{BP}'_i}$ can be represented by an AMA $\mathcal{A}'_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$ where $Q_i \subseteq P'_i \times \mathbb{N} \cup \{q_f\}$ and q_f is the unique final state, i.e., $Acc_i = \{q_f\}$. Let q^k denote $(q, k) \in P'_i \times \mathbb{N}$. Intuitively, to compute $Y_{\mathcal{BP}'_i}$, we will compute iteratively the different $Y_j s$. The iterative procedure computes different AMAs. To force termination, we use an acceleration based on the projection functions π^{-1} and π^k: for every $S \subseteq Q_i$,

$$\pi^{-1}(S) = \begin{cases} \{q^k \mid q^{k+1} \in S\} \cup \{q_f\} \text{ if } q_f \in S \text{ or } \exists q^1 \in S, \\ \{q^k \mid q^{k+1} \in S\} \qquad \text{else.} \end{cases}$$

$$\pi^k(S) = \{q^k \mid \exists j,\ 1 \leq j \leq k \text{ s.t. } q^j \in S\} \cup \{q_f \mid q_f \in S\}.$$

Algorithm 1 computes an AMA $\mathcal{A}'_i$ recognizing $Y_{\mathcal{BP}'_i}$. Let us explain the intuition behind the different lines of this algorithm. Let A_0 be the automaton obtained after the initialization (Line 1). It is clear that A_0 accepts Y_0. Let A_k be the AMA obtained at step k (a step starts at Line 3). For every $p \in P'_i$, state p^k denotes state p at step k, i.e., A_k recognizes a tuple $(p\omega, D)$ iff $p^k \xrightarrow{\omega/D}_i{}^* \{q_f\}$. Suppose the algorithm is at the beginning

of the k^{th} step ($loop_1$). Line 4 adds the ϵ-transition $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$ for every $p \in F_i$. Then, we obtain $L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$. $loop_2$ (Lines 5-9) is the saturation procedure that computes $pre^*_{\mathcal{BP}'_i}(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$. Line 10 removes the ϵ-transition $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$ for every $p \in F_i$. After this, we obtain $pre^+_{\mathcal{BP}'_i}(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$. Thus, in case of termination, the algorithm outputs $Y_{\mathcal{BP}'_i}$. The substitution at Line 11 is used to force termination. Thus, we can show the following theorem.

Theorem 6. *Algorithm 1 always terminates and produces $Y_{\mathcal{BP}'_i}$.*

Proof Sketch. The proof follows the proof of [20]. Algorithm 1 follows the idea of the algorithm of [20]. computing an AMA recognizing the language of an ABDPDS when transition rules are in the form of $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\}$, i.e., $\mathcal{D}_i = \emptyset$. The main differences are:

To compute $pre^*_{\mathcal{BP}'_i}(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$, instead of using the following saturation procedure given in [2] that computes reachable configurations of *Alternating* PDSs:

If $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_m\omega_m\} \in \Delta'_i$ and $p_j^k \xrightarrow{\omega_j/\emptyset}_i^* R_j$, for j, $1 \leq j \leq m$, add $p^k \xrightarrow{\gamma/\emptyset}_i \cup_{j=1}^m R_j$ in δ_i.

We use the following saturation procedure:

If $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \rhd D \in \Delta'_i$ and $p_j^k \xrightarrow{\omega_j/D_j}_i^* R_j$ for j, $1 \leq j \leq h$, add $p^k \xrightarrow{\gamma/D \cup \bigcup_{j=1}^h D_j} \cup_{j=1}^h R_j$ in δ_i.

The idea behind our saturation procedure is the following: suppose $p\gamma \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \rhd D \in \Delta'_i$ and for every j, $1 \leq j \leq h$, $(p_j\omega_j\omega', D_j)$ is in $L(A'_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ (i.e., $p_j^k \xrightarrow{\omega_j/D'_j}_i^* R_j \xrightarrow{\omega'/D''_j}_i^* \{q_f\}$ and $D_j = D'_j \cup D''_j$). Then, Lines 3-6 add the new transition rule $p^k \xrightarrow{\gamma(D \cup \bigcup_{j=1}^h D'_j)}_i \bigcup_{j=1}^h R_j$ that allows to accept $(p\gamma\omega', D \cup \bigcup_{j=1}^h D_j)$, i.e., $(p\gamma\omega', D \cup \bigcup_{j=1}^h D_j) \in pre^*_{\mathcal{BP}'_i}(\{(p_1\omega_1\omega', D_1), ..., (p_j\omega_j\omega', D_j)\})$. □

Complexity. Following [20], we can show that $loop_2$ can be done in time $O(|P'_i| \cdot |\Delta'_i| \cdot 2^{4|P'_i|+|\mathcal{D}_i|})$. The substitution (Line 11) and termination condition (Line 12) can be done in time $O(|\Gamma_i| \cdot |P'_i| \cdot 2^{2|P'_i|+|\mathcal{D}_i|})$ and $O(|\Gamma_i| \cdot |P'_i| \cdot 2^{|P'_i|+|\mathcal{D}_i|})$, respectively. Putting all these estimations together, the global complexity of Algorithm 1 is $O(|P'_i|^2 \cdot |\Delta'_i| \cdot |\Gamma_i| \cdot 2^{5|P'_i|+|\mathcal{D}_i|})$.

By Proposition 3 and Theorem 6, we get:

Lemma 4. *Given an ABDPDS $\mathcal{BP}'_i$, we can construct an AMA $\mathcal{A}'_i$ with $O(|\Gamma_i| \cdot |P'_i| \cdot 2^{|P'_i|+|\mathcal{D}_i|})$ transitions and $O(|P'_i|)$ states in time $O(|P'_i|^2 \cdot |\Delta'_i| \cdot |\Gamma_i| \cdot 2^{5|P'_i|+|\mathcal{D}_i|})$ s.t. $L(\mathcal{BP}'_i) = L(\mathcal{A}'_i)$.*

From Lemma 4 and Lemma 3, we get:

Lemma 5. *We can compute AMAs $\mathcal{A}'_1, ..., \mathcal{A}'_n$ in time $O(\sum_{i=1}^n((|P_i| \cdot |f_i| + k)^2 \cdot ((|P_i| \cdot |\Gamma_i| + |\Delta_i|)|f_i| + d) \cdot |\Gamma_i| \cdot 2^{5(|P_i| \cdot |f_i|+k)+|\mathcal{D}_i|}))$ s.t. for every i, $1 \leq i \leq n$, $p\omega \in P_i \times \Gamma_i^*$, $p\omega \models_D f_i$ iff $([p, f_i], D) \in L(\mathcal{A}'_i)$, where $k = \sum_{a \in At(f_i)}(|Q_a| + |Q_{\neg a}|)$ and $d = \sum_{a \in At(f_i)}(|\delta_a| + |\delta_{\neg a}|)$.*

4.5 CTL Model-Checking For DPNs with Regular Valuations

By Lemma 5, we obtain a set of AMAs $\{\mathcal{A}'_1, ..., \mathcal{A}'_n\}$ s.t. for every i, $1 \leq i \leq n$ and every local configuration $p\omega \in P_i \times \Gamma_i^*$, $p\omega \models_D f_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{A}'_i)$. Following

the approach for single-indexed LTL model-checking for DPNs, to obtain an efficient procedure, we compute the largest set $\mathcal{D}'_{fp}$ of DCLICs s.t. for every $d \in \bigcup_{i=1}^{n} \mathcal{D}_i$, d satisfies f iff $d \in \mathcal{D}'_{fp}$. Then, to check whether a global configuration $\mathcal{G}$ satisfies f, it is sufficient to check whether for every $p\omega \in \mathcal{G}$, there exists $D \subseteq \mathcal{D}'_{fp}$ s.t. $([p, f_{\wp(p)}]\omega, D) \in L(\mathcal{A}'_{\wp(p)})$. $\mathcal{D}'_{fp}$ can be computed as done in Section 3.2. We can show that:

Theorem 7. *We can compute AMAs* $\mathcal{A}'_1, ..., \mathcal{A}'_n$ *in time* $O(\sum_{i=1}^{n}((|P_i| \cdot |f_i| + k)^2 \cdot ((|P_i| \cdot |\Gamma_i| + |\Delta_i|)|f_i| + d) \cdot |\Gamma_i| \cdot 2^{5(|P_i| \cdot |f_i| + k) + |\mathcal{D}_i|}))$ *s.t. for every global configuration* $\mathcal{G}$, $\mathcal{G}$ *satisfies* f *iff for every* $p\omega \in \mathcal{G}$, *there exists* $D \subseteq \mathcal{D}'_{fp}$ *such that* $([p, f_{\wp(p)}]\omega, D) \in L(\mathcal{A}'_{\wp(p)})$, *where* $k = \sum_{a \in At(f_i)}(|Q_a| + |Q_{\neg a}|)$ *and* $d = \sum_{a \in At(f_i)}(|\delta_a| + |\delta_{\neg a}|)$.

References

1. Atig, M.F., Bouajjani, A., Touili, T.: On the reachability analysis of acyclic networks of pushdown systems. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 356–371. Springer, Heidelberg (2008)
2. Bouajjani, A., Esparza, J., Maler, O.: Reachability Analysis of Pushdown Automata: Application to Model Checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
3. Bouajjani, A., Esparza, J., Touili, T.: A generic approach to the static analysis of concurrent programs with procedures. In: POPL, pp. 62–73 (2003)
4. Bouajjani, A., Müller-Olm, M., Touili, T.: Regular symbolic analysis of dynamic networks of pushdown systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 473–487. Springer, Heidelberg (2005)
5. Bozzelli, L., Kretínský, M., Rehák, V., Strejcek, J.: On decidability of LTL model checking for process rewrite systems. Acta Inf. 46(1) (2009)
6. Chaki, S., Clarke, E., Kidd, N., Reps, T., Touili, T.: Verifying concurrent message-passing C programs with recursive calls. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 334–349. Springer, Heidelberg (2006)
7. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithm for model checking pushdown systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer, Heidelberg (2000)
8. Esparza, J., Kucera, A., Schwoon, S.: Model checking LTL with regular valuations for pushdown systems. Inf. Comput. 186(2), 355–376 (2003)
9. Gawlitza, T.M., Lammich, P., Müller-Olm, M., Seidl, H., Wenner, A.: Join-lock-sensitive forward reachability analysis for concurrent programs with dynamic process creation. In: VMCAI, pp. 199–213 (2011)
10. Göller, S., Lin, A.W.: The complexity of verifying ground tree rewrite systems. In: LICS, pp. 279–288 (2011)
11. Kahlon, V., Gupta, A.: An Automata-Theoretic Approach for Model Checking Threads for LTL Properties. In: LICS, pp. 101–110 (2006)
12. Kahlon, V., Gupta, A.: On the analysis of interacting pushdown systems. In: POPL, pp. 303–314 (2007)
13. Kahlon, V., Ivančić, F., Gupta, A.: Reasoning about threads communicating via locks. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 505–518. Springer, Heidelberg (2005)
14. Kidd, N., Lammich, P., Touili, T., Reps, T.: A decision procedure for detecting atomicity violations for communicating processes with locks. In: Păsăreanu, C.S. (ed.) Model Checking Software. LNCS, vol. 5578, pp. 125–142. Springer, Heidelberg (2009)

15. Lammich, P., Müller-Olm, M.: Precise fixpoint-based analysis of programs with thread-creation and procedures. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 287–302. Springer, Heidelberg (2007)
16. Lammich, P., Müller-Olm, M.: Conflict analysis of programs with procedures, dynamic thread creation, and monitors. In: Alpuente, M., Vidal, G. (eds.) SAS 2008. LNCS, vol. 5079, pp. 205–220. Springer, Heidelberg (2008)
17. Lammich, P., Müller-Olm, M., Wenner, A.: Predecessor sets of dynamic pushdown networks with tree-regular constraints. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 525–539. Springer, Heidelberg (2009)
18. Lugiez, D.: Forward analysis of dynamic network of pushdown systems is easier without order. Int. J. Found. Comput. Sci. 22(4), 843–862 (2011)
19. Mayr, R.: Process rewrite systems. Inf. Comput. 156(1-2), 264–286 (2000)
20. Song, F., Touili, T.: Efficient CTL model-checking for pushdown systems. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 434–449. Springer, Heidelberg (2011)
21. Song, F., Touili, T.: Model Checking Dynamic Pushdown Networks. Research report (2012), http://www.liafa.univ-paris-diderot.fr/~song/dpn-full.pdf
22. Touili, T., Atig, M.F.: Verifying parallel programs with dynamic communication structures. Theor. Comput. Sci. 411(38-39), 3460–3468 (2010)
23. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. J. Comput. Syst. Sci. 32(2), 183–221 (1986)
24. Wenner, A.: Weighted dynamic pushdown networks. In: Gordon, A.D. (ed.) ESOP 2010. LNCS, vol. 6012, pp. 590–609. Springer, Heidelberg (2010)
25. Yahav, E.: Verifying safety properties of concurrent java programs using 3-valued logic. In: POPL, pp. 27–40 (2001)