Model Checking Dynamic Pushdown Networks

Fu Song and Tayssir Touili

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 $i$.
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 $A_i$ recognizing all the configurations from which there exists a run $\sigma$ of the process $i$ that satisfies $f_i$. $A_i$ also memorizes the set of all the initial configurations of the instances of PDSs that are dynamically created during the run $\sigma$. 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 $D_{fp}$ 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 $D_{fp}$.

To compute the finite automaton $A_i$, 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 $A_i$ 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 $A_i$.

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 [3,6,12]. 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) \( M \) is a set \( \{ P_1, \ldots, P_n \} \) s.t. for every \( i, 1 \leq i \leq n \), \( P_i = (P_i, \Gamma_i, \Delta_i) \) 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 \( k \neq i \), \( \Gamma_i \) is the stack alphabet, and \( \Delta_i \) is a finite set of transition rules in the following forms: (a) \( q \gamma \rightarrow p_1 \omega_1 \) or (b) \( q \gamma \rightarrow p_1 \omega_1 \triangleright p_2 \omega_2 \) 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 \( M \) is a multiset \( G \) over \( \bigcup_{i=1}^n P_i \times \Gamma_i^* \). Each element \( q\omega \in P_i \times \Gamma_i^* \cap G \) denotes that an instance of \( P_i \) running in parallel in the network is at the local configuration \( q\omega \), i.e., \( P_i \) is at the control location \( q \) and its stack content is \( \omega \). If \( \omega = \gamma u \) for \( \gamma \in \Gamma_i \) and there is in \( \Delta_i \) a transition (a) \( q \gamma \rightarrow p_1 \omega_1 \) or (b) \( q \gamma \rightarrow p_1 \omega_1 \triangleright p_2 \omega_2 \) s.t. \( p_2 \omega_2 \in P_j \times \Gamma_j \), then the instance of \( P_i \) can move from \( q\omega \) to the control location \( p_1 \omega_1 \) and replace \( \gamma \) by \( \omega_1 \) at the top of its stack, i.e., \( P_i \) 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 \( P_j \) starting from \( p_2 \omega_2 \).

Formally, a DPDS \( P_i \) induces an immediate successor relation \( \rightarrow \) as follows: for every \( \omega \in \Gamma_i^* \), if \( q \gamma \rightarrow p_1 \omega_1 \) then \( q\omega \rightarrow p_1 \omega_1 \omega \); if \( q \gamma \rightarrow p_1 \omega_1 \triangleright p_2 \omega_2 \), then \( q\omega \rightarrow p_1 \omega_1 \triangleright \{ p_2 \omega_2 \} \). To unify the presentation, if \( q\gamma \rightarrow p_1 \omega_1 \omega, \gamma \in \Gamma_i \), we sometimes write \( q\gamma \rightarrow p_1 \omega_1 \omega \triangleright \emptyset \) instead. The transitive and reflexive closure of \( \rightarrow \) is denoted by \( \Rightarrow \). Formally, for every \( \omega \in P_i \times \Gamma_j^* \), \( p\omega \Rightarrow p\omega \triangleright \emptyset \); and if \( p\omega \rightarrow p_1 \omega_1 \triangleright D_1 \) and \( p_1 \omega_1 \rightarrow p_2 \omega_2 \triangleright D_2 \), then \( p\omega \Rightarrow p_2 \omega_2 \triangleright D_1 \cup D_2 \). \( \Rightarrow \) is defined as usual.

A DPDS \( P_i \) 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 \( P_i \) from a local configuration \( c_0 \) is a sequence of local configurations \( c_0 c_1 \ldots \) over \( P_i \times \Gamma_i^* \) s.t. for every \( j \geq 0 \), \( c_j \rightarrow c_{j+1} \triangleright D \) for some \( D \). A global run \( \rho \) of \( M \) from a global configuration \( G \) is a (potentially infinite) set of local runs. Initially, \( \rho \) contains exactly the local runs starting from the local configurations in \( G \). Whenever a DCLIC \( c \) is created by some local run of \( \rho \), a new local run starting from \( c \) is added into \( \rho \). For every i, \( 1 \leq i \leq n \), let \( \phi(\sigma) = i \) iff \( \sigma \) is a local run of an instance of \( P_i \), and \( \phi(p\omega) = \phi(p) = i \) iff \( p \in P_i \). Let \( D_i = \{ p_2 \omega_2 \in \bigcup_{i=1}^n P_i \times \Gamma_i^* \mid q\gamma \rightarrow p_1 \omega_1 \triangleright p_2 \omega_2 \in \Delta_i \} \) be the set of potential DCLICs of the DPDS \( P_i \).
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 $a \in AP$):

$$\psi ::= a \mid \neg \psi \mid \psi \land \psi \mid X\psi \mid \psi_U \psi.$$ 

Given an $\omega$-word $\eta = \alpha_0\alpha_1\ldots$ over $2^{AP}$, let $\eta(k)$ denote $\alpha_k$, and $\eta_k$ denote the suffix of $\eta$ starting from $\alpha_k$. $\eta \models \psi$ ($\eta$ satisfies $\psi$) is inductively defined as follows: $\eta \models a$ iff $a \in \eta(0)$; $\eta \models \neg \psi$ iff $\eta \not\models \psi$; $\eta \models \psi_1 \land \psi_2$ iff $\eta \models \psi_1$ and $\eta \models \psi_2$; $\eta \models X\psi$ iff $\eta_1 \models \psi$; $\eta \models \psi_U\psi_2$ 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) $B$ is a tuple $(G, \Sigma, \theta, g^0, F)$ where $G$ is a finite set of states, $\Sigma$ 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 $B$ over an $\omega$-word $\alpha_0\alpha_1\ldots$ is a sequence of states $q_0q_1\ldots$ s.t. $q_0 = g^0$ and $(q_i, \sigma_i, q_{i+1}) \in \theta$ 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 $\omega$-words that satisfy $f$.

2.3 Single-Indexed LTL for DPNs

Let $M = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ 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 $\lambda : AP \rightarrow 2^{U_{\infty}P \times G^\ast}$ be a valuation which assigns to each atomic proposition a set of local configurations. A local run $p_0\alpha_0 p_1\alpha_1\ldots$ of $\mathcal{P}_i$ satisfies $f_i$ iff the $\omega$-word $\alpha_0\alpha_1\ldots$ where for every $j \geq 0$, $\alpha_j = \{a \in AP \mid p_j \alpha_j \in \lambda(a)\}$, satisfies $f_i$. A local configuration $c$ of $\mathcal{P}_i$ satisfies $f_i$ iff $\mathcal{P}_i$ has a local run $\sigma$ from $c$ that satisfies $f_i$. If $D$ is the set of DCLICs created during the run $\sigma$, we write $c \models_D f_i$. $M$ satisfies $f$ iff it has a global run $\rho$ such that for every $i$, $1 \leq i \leq n$, each local run of $\mathcal{P}_i$ in $\rho$ satisfies the formula $f_i$.

2.4 Multi-automata and Predecessors

From now on, we fix a DPN $M = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ 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 $M$ 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 $\rho$ from a global configuration $G$ s.t. an instance of $\mathcal{P}_i$ satisfies the formula $f_i$ only if it is an instance in $G$ or it is dynamically created during the run $\rho$. Thus, it is important to memorize the set of DCLICs that are created during a run. To this aim, we introduce the function $pre_{\mathcal{P}_i} : 2^{P_i \times G^\ast_{\infty} \times 2^{\Delta_i}} \rightarrow 2^{P_i \times G^\ast_{\infty} \times 2^{\Delta_i}}$ as follows. $pre_{\mathcal{P}_i}(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times G^\ast_{\infty}, s.t. c \rightarrow_{i} c' \triangleright D_1 \text{ and } (c', D_2) \in U\}$. 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, $pre_{\mathcal{P}_i}^+(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times G^\ast_{\infty}, s.t. c \rightarrow^*_i c' \triangleright D_1 \text{ and } (c', D_2) \in U\}$. 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 \( \mathcal{A}_i = (Q_i, \Gamma_i, \delta_i, I_i, \text{Acc}_i) \), 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 \( \mathcal{P}_i \), \( \text{Acc}_i \subseteq Q_i \) is a finite set of final states, \( \delta_i \subseteq (Q_i \times \Gamma_i) \times 2^{D_i} \times 2^{Q_i} \) is a finite set of transition rules.

A MA is a AMA \( \mathcal{A}_i \) s.t. \( \delta_i \subseteq (Q_i \times \Gamma_i) \times 2^{D_i} \times Q_i \).

We write \( \frac{\gamma/D_i}{p} \{q_1, \ldots, q_m\} \) instead of \((p, \gamma, D_i, \{q_1, \ldots, q_m\}) \in \delta_i \), where \( D \) is a set of DCLICs. We define the relation \( \rightarrow_i^* \subseteq (Q_i \times \Gamma_i)^* \times 2^{D_i} \times 2^{Q_i} \) as the smallest relation s.t.: (1) \( q \xrightarrow[\omega/0^+]{\omega/D_i} \{q\} \) for every \( q \in Q_i \), (2) if \( q \xrightarrow[\gamma/D_i]{\gamma} \{q_1, \ldots, q_m\} \) and \( q_k \xrightarrow[\omega/D_k^+]{\omega} \) \( S_k \) for \( k, 1 \leq k \leq m \), then \( q \xrightarrow[\gamma/\omega/D_i \cup \omega/D_1 \cup \ldots \cup \omega/D_m]{\gamma/\omega} \{q_1, \ldots, q_m\} \). Let \( L(\mathcal{A}_i) \) be the set of tuples \( (p, \omega, D) \in P_i \times \Gamma_i^* \times 2^{D_i} \), s.t. \( p \xrightarrow[\omega/D_i^+]{\omega} S \) for some \( S \subseteq \text{Acc}_i \). A set \( W \subseteq P_i \times \Gamma_i^* \times 2^{D_i} \) 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 \( C \times \{0\} \) is a regular set.

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

- For every \( p \gamma \xleftarrow{} p_1 \omega_1 \in \mathcal{A}_i \) and \( p_1 \xrightarrow[\omega_1/D_i^+]{\omega_1} q \), add a new rule \( p \xrightarrow[\gamma/D_i^+]{\gamma} q \);
- For every \( p \gamma \xleftarrow{} p_1 \omega_1 \triangleright p_2 \omega_2 \in \mathcal{A}_i \) and \( p_1 \xrightarrow[\omega_1/D_i^+]{\omega_1} q \), add a new rule \( p \xrightarrow[\gamma/D_i \cup \omega_2^+]{\gamma} q \).

The procedure adds only new transitions to \( \mathcal{A}_i \). 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 \( \mathcal{A}_{i}^{pre} \) is at most \( O(|\mathcal{A}_i| \cdot |Q_i|^2 \cdot 2^{2|D_i|}) \). The intuition behind this procedure is that, for every \( \omega' \in \Gamma_i^* \), suppose \( p \gamma \xleftarrow{} p_1 \omega_1 \triangleright p_2 \omega_2 \in \mathcal{A}_i \) and the tuple \((p_1 \omega_1, D)\) is accepted by the automaton, i.e., \( p_1 \xrightarrow[\omega_1/D_i^+]{\omega_1} q \xrightarrow[\omega/D_k^+]{\omega} g \) for some \( g \in \text{Acc}_i \) and \( D = D_1 \cup D_2 \). Then, we add the new transition rule \( p \xrightarrow[\gamma/D_i \cup \omega_2^+]{\gamma} q \) that allows the automaton to accept \((p \gamma \omega', D \cup \{p_2 \omega_2\})\), i.e., \( p \xrightarrow[\gamma/D_i \cup \omega_2^+]{\gamma} g \). The case \( p \gamma \xleftarrow{} p_1 \omega_1 \in \mathcal{A}_i \) is similar. Thus, we obtain the following theorem.

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

### 3 Single-Indexed LTL Model-Checking for DPNs

In this section, we consider LTL model checking w.r.t. a labeling function \( l : \bigcup_{i=1}^{n} P_i \rightarrow 2^{AP} \) assigning to each control location a set of atomic propositions. In this case, the valuation \( \lambda_i \) (called simple valuation) is defined as follows: for every \( a \in AP \), \( \lambda_i(a) = \{p \omega \in \bigcup_{i=1}^{n} P_i \times \Gamma_i^* \mid a \in l(p)\} \). A global configuration \( G \) satisfies \( f = \bigwedge f_i \) iff \( M \)
has a global run $\rho$ from $G$ s.t. every local run $\sigma$ of $\rho$ satisfies $f_{\varphi(\sigma)}$ where $\varphi(\sigma)$ denotes the index of the DPDS which corresponds to the local run $\sigma$. Checking whether $G$ satisfies $f$ is non-trivial since the number of local runs of $\rho$ 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 $P_i$ (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 $G$, we can guess the set of DCLICs $D \subseteq \bigcup_{i=1}^{n} D_i$ which are created in a global run from $G$ such that the global run satisfies $f$. Then, it is sufficient to check that every local configuration $c \in G \cup D$ satisfies the LTL formula $f_{\varphi(c)}$ when disallowing the transition rules which create a DCLIC outside of $D$ and discarding the DCLICs inside of $D$. Checking whether $c$ satisfies $f_{\varphi(c)}$ could be solved by LTL model-checking for PDSs [27] 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 $O(2^{\lvert \bigcup_{i=1}^{n} D_i \rvert})$, and for each set $D$ of DCLICs, we have to perform at most $O(\lvert \bigcup_{i=1}^{n} D_i \rvert)$ times of LTL model-checking algorithm for PDSs, where LTL model-checking for PDSs is in time $O(|P_{\varphi(d)}|^2 \cdot |A_{\varphi(d)}| \cdot 2^{|f_{\varphi(c)}|})$. Thus, the complexity of checking whether $G$ satisfies $f$ or not will be $O(2^{\sum_{i=1}^{n} |D_i|} \cdot \sum_{d \in \bigcup_{i=1}^{n} D_i \cup G} (|P_{\varphi(d)}|^2 \cdot |A_{\varphi(d)}| \cdot 2^{|f_{\varphi(c)}|}))$.

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 $P_i$ and $D \subseteq D_i$ is a set of DCLICs, iff $P_i$ has a local run $\sigma$ from $c$ that satisfies $f_i$ such that $D$ is the set of DCLICs created during the local run $\sigma$. Then, a global configuration $G$ satisfies $f = \bigwedge f_i$ iff for every configuration $c \in G$, there exists a set of DCLICs $D_c$ s.t. $(c, D_c) \in L(\mathcal{A}_{\varphi(c)})$ 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 $D_{fp} \subseteq \bigcup_{i=1}^{n} D_i$ of DCLICs such that $d \in D_{fp}$ iff $d$ is a DCLIC and there exists a global run of $M$ starting from $d$ that satisfies $f$. Then, to check whether a global configuration $G$ satisfies $f$, it is sufficient to check for every $c \in G$ whether there exists $D_c \subseteq D_{fp}$ s.t. $(c, D_c) \in L(\mathcal{A}_{\varphi(c)})$.

### 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 [27]. 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 $\mathcal{B}P_i = (P_i, \Gamma_i, A_i, F_i)$, where $(P_i, \Gamma_i, A_i)$ 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 $\sigma$ of $\mathcal{B}P_i$ is accepting iff $\sigma$ infinitely often visits some control locations in $F_i$. Let $L(\mathcal{B}P_i)$ be the set of all the pairs $(c, D) \in P_i \times \mathcal{C}$.
\( \Gamma_i \times 2^{\mathcal{D}_i} \) s.t. \( \mathcal{B}_i \) has an accepting run from \( c \) and the run generates the set of DCLICs \( D \).

Let \( \mathcal{B}_i = (G_i, 2^{\mathcal{A}_i}, \theta_i, g_0^i, F_i) \) be the BA recognizing all the \( \omega \)-words that satisfy \( f_i \). We compute a BDPDS \( \mathcal{B}_i \) such that \( \mathcal{B}_i \) has a local run from \( p_0 \) that satisfies \( f_i \) and generates a set of DCLICs \( D \) iff \( ([p, g^i_0] \omega) D \in L(\mathcal{B}_i) \). We define \( \mathcal{B}_i = (P_i \times G_i, \Gamma_i, A'_i, F'_i) \) 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. \([p, g_1] \gamma \leftrightarrow [p_1, g_2] \omega_1 \in A'_i \) iff \( p_1 \omega_1 \in A_i \);
2. \([p, g_1] \gamma \leftrightarrow [p_1, g_2] \omega_1 \triangleright D \in A'_i \) iff \( p_1 \omega_1 \triangleright D \in A_i \).

Intuitively, \( \mathcal{B}_i \) is a product of \( \mathcal{P}_i \) and the BA \( \mathcal{B}_i \). \( \mathcal{B}_i \) has an accepting run \( g_0 g_1 \ldots \) over an \( \omega \)-word \( l(p_0) l(p_1) \ldots \) that corresponds to a local run \( \sigma = p_0 \omega_0 p_1 \omega_1 \ldots \) of \( \mathcal{P}_i \) iff \( \mathcal{B}_i \) has an accepting run \( \sigma' = [p_0, g_0] \omega_1 [p_1, g_1] \omega_1 \ldots \) and \( D \) is the set of DCLICs created during the run \( \sigma' \) iff \( D \) is the set of DCLICs created during the run \( \sigma' \). 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 \( \mathcal{B}_i \) 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 \( \mathcal{B}_i \) visits infinitely often some local 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 \( \mathcal{B}_i \) creates this DCLIC. Thus, we obtain the following theorem.

**Lemma 1.** \( \mathcal{P}_i \) has a local run from \( p_0 \) that satisfies \( f_i \) and creates a set of DCLICs \( D \) iff \( ([p, g^i_0] \omega, D) \in L(\mathcal{B}_i) \), where \( \mathcal{B}_i \) can be constructed in time \( O(|A_i| \cdot 2^{|B|}) \).

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

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

**Proposition 1.** Let \( \mathcal{B}_i = (P_i, \Gamma_i, A_i, F_i) \) be a BDPDS, \( \mathcal{B}_i \) has an accepting run from \( c \in \Gamma \) 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

\[
\begin{align*}
(a_1) & : c \rightarrow_{\mathcal{B}_i}^* p \gamma \omega \triangleright D_1 \text{ for some } \omega \in \Gamma_i^*; \\
(a_2) & : p \gamma \rightarrow_{\mathcal{B}_i}^* g \triangleright D_2 \text{ and } g \rightarrow_{\mathcal{B}_i}^* p \gamma v \triangleright D_3, \text{ for some } g \in F_i, v \in \Gamma_i^*.
\end{align*}
\]

Intuitively, an accepting run from \( c \) will reach a configuration \( p \gamma \omega \) (Item \( a_1 \)) followed by a repeatedly executed cycle (Item \( a_2 \)) 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 \( \gamma \) 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 \( L(\mathcal{B}_i) \), we reformulate the above conditions as follows:
Proposition 2. Let $\mathcal{BP}_i = (P_i, \Gamma_i, A_i, F_i)$ 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 $\exists D_1, D_2 \subseteq D_i$ s.t. $D = D_1 \cup D_2$, and
\[
\begin{align*}
(\beta_1) : (c, \hat{D}_i) & \in \text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i \times \{\emptyset\}); \\
(\beta_2) : (p\gamma, D'_2) & \in \text{pre}_{fp}^+ ((F_i \times \Gamma_i ^* \times 2^D_i) \cap \text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\})) \text{ (note that } D'_2 = D_2 \cup D_3). 
\end{align*}
\]

Intuitively, items $\beta_1$ and $\beta_2$ are reformulations of items $\alpha_1$ and $\alpha_2$, respectively. By Proposition 2 we can get that $L(\mathcal{BP}_i) = \{(c, D_1 \cup D_2) \in P_i \times \Gamma_i \times 2^D_i |$ Items $\beta_1$ and $\beta_2$ hold$\}$. Since $F_i \times \Gamma_i ^* \times 2^D_i$ and $\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\}$ are regular sets, using Theorem 1 we can construct two MAs $A'$ and $A''$ accepting $\text{pre}_{fp}^+ ((F_i \times \Gamma_i ^* \times 2^D_i) \cap \text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\}))$ and $\text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\})$. The intersection $(F_i \times \Gamma_i ^* \times 2^D_i) \cap \text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\})$ is easy to compute. Since $F_i \times \Gamma_i ^* \times 2^D_i$ 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 $P_i \times \Gamma_i \times 2^D_i$ is finite, we can determine all the tuples $(p\gamma, D'_2) \in P_i \times \Gamma_i \times 2^D_i$ s.t. Item $\beta_2$ holds. The set of pairs $(c, D_1)$ is the union of all the sets $\text{pre}_{fp}^+ (\{p\} \times \gamma \Gamma_i ^* \times \{\emptyset\})$. Thus, we can get $L(\mathcal{BP}_i)$. For every BDPDS $\mathcal{P}_i$ and MA $A_i$, $\text{pre}_{fp}^+ (L(A_i))$ and $\text{pre}_{fp}^+ (L(A_i))$ can be computed in time $O(|A_i| \cdot |Q_i|^3 \cdot 2^{2D_i})$, where $|Q_i| = O(|P_i|)$. Thus, we get that:

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

From Lemma 1 and Lemma 2, we get:

Theorem 2. Given a DPN $\mathcal{M} = \{P_1, \ldots, P_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a labelling function $l$, we can compute MAs $A_1, \ldots, A_n$ in time $O(\sum_{i=1}^n (|A_i| \cdot |P_i|^3 \cdot 2^{2D_i}))$ s.t. for every $i, 1 \leq i \leq n$, every $p\omega \in P_i \times \Gamma_i$ and $D \subseteq D_i$, $p\omega \models_D f_i$ iff $([p, g_i^0], \omega, D) \in L(A_i)$.

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

Given a DPN $\mathcal{M} = \{P_1, \ldots, P_n\}$ and a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$, by Theorem 2 we can construct a set of MAs $\{A_1, \ldots, A_n\}$ 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 $([p, g_i^0], \omega, D) \in L(A_i)$. Then, to check whether a global configuration $G$ satisfies $f$, we need to check whether for every local configuration $c \in G$, there exists a set of DCLICs $D_c$ s.t. $(c, D_c) \in L(\mathcal{A}_{\varphi(c)})$ and every DCLIC $d \in D_c$ satisfies $f$, i.e., there exists a set of DCLICs $D_d$ s.t. $(d, D_d) \in L(\mathcal{A}_{\varphi(d)})$, 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 $D_{fp}$ s.t. for every $d \subseteq D_{fp}$ s.t. $d$ satisfies $f$ iff $d \in D_{fp}$. Then, to check whether $G$ satisfies $f$, it is sufficient to check whether for every $c \in G$, there exists $D_c \subseteq D_{fp}$ s.t. $(c, D_c) \in L(\mathcal{A}_{\varphi(c)})$.

Let $\{A_1, \ldots, A_n\}$, s.t. for every $i, 1 \leq i \leq n$, $A_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$, be the set of the computed MAs. Intuitively, $D_{fp}$ should be equal to the set of local configurations $p\omega \in \bigcup_{i=1}^n D_i$ s.t. there exists $D \subseteq D_{fp}$ s.t. $p\omega \models_D f_{\varphi(p)}$, i.e., $([p, s_{\varphi(p)}^0], \omega, D) \in L(\mathcal{A}_{\varphi(p)})$. Thus, $D_{fp}$ can be defined as the greatest fixpoint of the
function \( F(X) = \{ p \omega \in D_I \mid \exists D \subseteq X \text{ s.t. } ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \} \). This set can then be computed iteratively as follows: \( D_{fp} = \bigcap_{j \geq 0} D_j \), where \( D_0 = D_I \) and \( D_{j+1} = \{ p \omega \in D_I \mid \exists D \subseteq D_j, ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \} \) for every \( j \geq 0 \). Since \( \bigcup_{i=1}^{m} D_i \) 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 \( D_{fp} = D_m \).

For every \( p \omega \in \bigcup_{i=1}^{m} D_i \) and \( D \subseteq D_{\phi(p)} \), to avoid checking whether \( ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \) at each step when computing \( D_0, D_1, \ldots \), 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 \( ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \) can be decided in time \( O(|\omega| \cdot |\delta_{\phi(p)}| \cdot |Q_{\phi(p)}| \cdot 2^{2|\omega|}) \).

Thus, we can get the hash table in time \( O(\sum_{p \omega \in \bigcup_{i=1}^{m} D_i} (|\omega| \cdot |\delta_{\phi(p)}| \cdot |Q_{\phi(p)}| \cdot 2^{2|\omega|})) \). Given \( D_j \) and the hash table, we can compute \( D_{j+1} \) in time \( O(\sum_{p \omega \in \bigcup_{i=1}^{m} D_i} (|\omega| \cdot |\delta_{\phi(p)}| \cdot |Q_{\phi(p)}| \cdot 2^{2|\omega|}) + |D_j| \cdot 2^{2|\omega|}) \).

**Theorem 3.** We can compute \( D_{fp} \) in time \( O(\sum_{p \omega \in \bigcup_{i=1}^{m} D_i} (|\omega| \cdot |\delta_{\phi(p)}| \cdot |Q_{\phi(p)}| \cdot 2^{2|\omega|}) + |\bigcup_{i=1}^{m} D_i| \cdot 2^{2|\omega|})) \) s.t. for every \( c \in \bigcup_{i=1}^{m} D_i \), \( c \) satisfies the single-indexed LTL formula \( f \) iff \( c \in D_{fp} \).

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

**Theorem 4.** Given a DPN \( M = \{ P_1, \ldots, P_n \} \), a single-indexed LTL formula \( f = \bigwedge_{i=1}^{n} f_i \) and a labelling function \( l \), we can compute MAs \( A_1, \ldots, A_n \) in time \( O(\sum_{i=1}^{n} (|A_i| \cdot 2^{|l_i|} \cdot |I_i| \cdot |S_{states_i}| \cdot |P_i| \cdot 2^{2|\omega_i|})) \) s.t. for every global configuration \( \mathcal{G} \), \( \mathcal{G} \) satisfies \( f \) iff for every \( p \omega \in \mathcal{G} \), there exists \( D \subseteq D_{fp} \) s.t. \( ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \), where \( |S_{states_i}| \) denotes the number of states of the automata corresponding to the regular valuation \( \lambda \).

### 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 \( \lambda : AP \rightarrow 2^{\sum_{i=1}^{n} P_i \times I_i} \) s.t. for every \( a \in AP \), \( \lambda(a) \) is a regular set of local configurations of \( P_i \) 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 \( P'_i \), which is a kind of synchronization of the DPDS \( P_i \) 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 \( P'_i \), 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 \( M = \{ P_1, \ldots, P_n \} \), a single-indexed LTL formula \( f = \bigwedge_{i=1}^{n} f_i \) and a regular valuation \( \lambda \), we can compute MAs \( A_1, \ldots, A_n \) in time \( O(\sum_{i=1}^{n} (|A_i| \cdot 2^{|l_i|} \cdot |I_i| \cdot |S_{states_i}| \cdot |P_i| \cdot 2^{2|\omega_i|})) \) s.t. for every global configuration \( \mathcal{G} \), \( \mathcal{G} \) satisfies \( f \) iff for every \( p \omega \in \mathcal{G} \), there exists \( D \subseteq D_{fp} \) s.t. \( ([p, g_{\phi(p)}^0]_\omega, D) \in L(A_{\phi(p)}) \), where \( |S_{states_i}| \) denotes the number of states of the automata corresponding to the regular valuation \( \lambda \).
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 $R$ as the dual of the until operator $U$. Let $AP$ be a finite set of atomic propositions. The set of CTL formulas is given by (where $a \in AP$):

$$\psi ::= a \mid \neg a \mid \psi \land \psi \mid \psi \lor \psi \mid AX\psi \mid ED\psi \mid A[\psi U \psi] \mid E[\psi U \psi] \mid A[\psi R \psi] \mid E[\psi R \psi].$$

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

Let $\lambda : AP \rightarrow 2^{\cup_{i=1}^{p} \times \Gamma_i}$ 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 $c \models^\lambda f_i$), if there exists $D \subseteq D_i$ s.t. $c \models^\lambda f_i$ holds, where $\models^\lambda$ is inductively defined in Figure[1] Intuitively, $c \models^\lambda f_i$ 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 $qy \hookrightarrow p_1\omega_1 \triangleright p_2\omega_2$ is used to make $f_i$ satisfied, $p_2\omega_2$ is in $D$. We write $c \models^\lambda f_i$ instead of $c \models^\lambda f_i$ when $\lambda$ is clear from the context.

A single-indexed CTL formula $f$ is a formula of the form $\land f_i$ 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 $P_i$. A global configuration $G$ satisfies $f = \land f_i$ iff for every $c \in G$, there exists a set of DCLICs $D \subseteq D_{\phi(c)}$ s.t. $c \models^\lambda f_{\phi(c)}$ and for every $d \in D, d$ also satisfies $f$.

4.2 Alternating BDPDSs

**Definition 6.** An Alternating BDPDS (ABDPDS) is a tuple $BP_i = (P_i, \Gamma_i, A_i, F_i)$, where $P_i$ is a finite set of control locations, $\Gamma_i$ is the stack alphabet, $F_i \subseteq P_i$ is a set of accepting control locations, $A_i$ is a finite set of transition rules in the form of $py \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \{q_1 u_1, ..., q_k u_k\}$ s.t. $py \in P_i \times \Gamma_i, \{p_1\omega_1, ..., p_h\omega_h\} \subseteq P_i \times \Gamma_i$ and $\{q_1 u_1, ..., q_k u_k\} \subseteq D_i$.

An ABDPDS $BP_i$ induces a relation $\Rightarrow_i \subseteq (P_i \times \Gamma_i) \times (2^{P_i \times \Gamma_i} \times 2^{D_i})$ defined as follows: for every $\omega \in \Gamma_i^*$, if $py \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \{q_1 u_1, ..., q_k u_k\} \in A_i$, then $py \omega \Rightarrow_i \{p_1\omega_1\omega, ..., p_h\omega_h\omega\} \triangleright \{q_1 u_1, ..., q_k u_k\}$. Intuitively, if $BP_i$ is at the configuration $py\omega$, 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 $q_1 u_1, ..., q_k u_k$, respectively. We sometimes write $py \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\}$ if $py \hookrightarrow \{p_1\omega_1, ..., p_h\omega_h\} \triangleright \emptyset \in A_i$. 
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 $p_1\omega_1, \ldots, p_m\omega_m$, then, necessarily, $qu \leftrightarrow \{p_1\omega_1, \ldots, p_m\omega_m\} \triangleright D$ for some $D \subseteq D_j$. 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 $(c, D) \in P_i' \times \Gamma_i^* \times 2^{D_j}$ s.t. $\mathcal{BP}'_i$ has an accepting run from $c$ and that creates the set of DCLICs $D$.

4.3 Computing Corresponding Alternating BDPDs

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 BDPDs, since CTL formulas can be translated to alternating Büchi automata. We compute a set of AMAs $\mathcal{A}'_1, \ldots, \mathcal{A}'_n$ s.t. for every $i$, $1 \leq i \leq n$ and every local configuration $p\omega$ of $\mathcal{P}_i$, $p\omega \models D_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{A}'_i)$. Later, we compute the largest set of DCLICs $D'_j$ such that a DCLIC $D$ satisfies $f$ iff $d \in D'_j$. Then, to check whether a global configuration $G$ satisfies $f$, it is sufficient to check whether for every $p\omega \in G$, there exists $D \subseteq D'_j$ s.t. $([p, f_i]\omega, D) \in L(\mathcal{A}'_i)$ to compute the AMAs, we construct a set of alternating BDPDs $\mathcal{BP}'_i$ which are synchronizations of the DPDs $\mathcal{P}_i$ with formulas $f_i$ s.t. the AMAs we are looking for correspond to the languages accepted by these alternating BDPDs $\mathcal{BP}'_i$. We first show how to compute the alternating BDPDs $\mathcal{BP}'_i$. Then, we show how to compute the languages of these alternating BDPDs $\mathcal{BP}'_i$ s.t. i.e. the AMAs.

We fix an index $i$, $1 \leq i \leq n$. We construct an ABDPD $\mathcal{BP}'_i$ s.t. for every $p\omega \in P_i' \times \Gamma_i^*$, $p\omega \models D_i$ iff $([p, f_i]\omega, D) \in L(\mathcal{BP}'_i)$. We suppose w.l.o.g. that the DPD $\mathcal{P}_i$ has a bottom-of-stack $\#$ which is never popped from the stack. For every $a \in \text{At}(f_i)$, since
\[\lambda(a)\] is a regular set of local configurations of \(P_i\), let \(M_a = (Q_a, \Gamma_i, \delta_a, I_a, \text{Acc}_a)\) be a MA s.t. \(L(M_a) = \lambda(a) \times \emptyset\), and \(M_{-a} = (Q_{-a}, \Gamma_i, \delta_{-a}, I_{-a}, \text{Acc}_{-a})\) a MA s.t. \(L(M_{-a}) = (P_i \times \Gamma_i \setminus \lambda(a)) \times \emptyset\), i.e., the set of configurations where \(a\) does not hold. To distinguish between all the initial states \(p\) in \(M_a\) and \(M_{-a}\), we write \(p_a\) and \(p_{-a}\) instead. W.l.o.g., we assume that the set of states \(Q_{a,s}\), and \(Q_{-a,s}\) are disjoint for every \(a \in \text{At}(f_i)\).

Let \(BP'_{i} = (P'_i, \Gamma_i, \Delta'_i, F_i)\) be the ABDPDS such that \(P'_i = P_i \times \text{cl}(f_i) \cup \bigcup_{a \in \text{At}(f_i)} (Q_a \cup Q_{-a}); F_i = P_i \times \text{cl}(f_i) \cup \bigcup_{a \in \text{At}(f_i)} (\text{Acc}_a \cup \text{Acc}_{-a});\) and \(\Delta'_i\) is the smallest set of transition rules s.t. for every control location \(p \in P_i\), every subformula \(\psi \in \text{cl}(f_i)\) and every \(\gamma \in \Gamma_i\), we have:

1. if \(\psi = a\) or \(\psi = \neg a\), where \(a \in \text{At}(f_i)\); \([p, \psi]_\gamma \leftarrow \{p \psi \gamma\} \in \Delta'_i;\)
2. if \(\psi = \psi_1 \land \psi_2\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma, [p, \psi_2]_\gamma\} \in \Delta'_i;\)
3. if \(\psi = \psi_1 \lor \psi_2\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma \lor [p, \psi_2]_\gamma\} \in \Delta'_i;\)
4. if \(\psi = \text{EX}\psi_1\); \([p, \psi]_\gamma \leftarrow \{[p', \psi_1]_\omega \lor \{p'' \omega\} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i;\)
5. if \(\psi = \text{AX}\psi_1\); \([p, \psi]_\gamma \leftarrow \{[p', \psi_1]_\omega \mid p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i\} \Rightarrow \{p'' \omega \mid p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i\} \in \Delta'_i;\)
6. if \(\psi = \text{E}[\psi_1 \text{U}\psi_2]\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma \in \Delta'_i\) and \([p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma, [p', \psi]_\omega \lor \{p'' \omega\} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i, [p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma, [p', \psi]_\omega \} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \in \Delta_i;\)
7. if \(\psi = \text{A[}\psi_1 \text{R}\psi_2]\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma \in \Delta'_i\) and \([p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma, [p', \psi]_\omega \lor \{p'' \omega\} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i, [p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma, [p', \psi]_\omega \} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \in \Delta_i;\)
8. if \(\psi = \text{E}[\psi_1 \text{R}\psi_2]\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma \in \Delta'_i\) and \([p, \psi]_\gamma \leftarrow \{[p, \psi_1]_\gamma, [p', \psi]_\omega \lor \{p'' \omega\} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i, [p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma, [p', \psi]_\omega \} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \in \Delta_i;\)
9. if \(\psi = \text{A[}\psi_1 \text{R}\psi_2]\); \([p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma \in \Delta'_i\) and \([p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma, [p', \psi]_\omega \lor \{p'' \omega\} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \lor p'' \omega \in \Delta_i, [p, \psi]_\gamma \leftarrow \{[p, \psi_2]_\gamma, [p', \psi]_\omega \} \in \Delta'_i\) if \(p \gamma \not\rightarrow p' \omega \in \Delta_i;\)
10. for every transition \((q_1, \gamma, q_2)\) in \(\bigcup_{a \in \text{At}(f_i)} (\delta_a \cup \delta_{-a}); q_1 \gamma \leftarrow \{q_2 \epsilon\} \in \Delta'_i;\)
11. for every \(q \in \bigcup_{a \in \text{At}(f_i)} (\text{Acc}_a \cup \text{Acc}_{-a}); q \# \leftarrow \{q \#\} \in \Delta'_i;\)

For every \(p\omega \in P'_i \times \Gamma_i\), \(BP'_{i}\) has an accepting run \(\sigma\) from \([p, f_i]_\omega\) and \(D\) is the set of DCLICs created by \(\sigma\) if \(p \omega \models_D f_i\). The intuition behind each rule is explained as follows.

If \(\psi = a\) is in \(\text{At}(f_i)\), for every \(p \omega \in P'_i \times \Gamma_i\), \(p \omega\) satisfies \(\psi\) if \(BP'_{i}\) has an accepting run from \([p, a]_\omega\). To check this, \(BP'_{i}\) 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 \(\omega\). Then the run of \(BP'_{i}\) from \(p_a\omega\) mimics the run of \(M_a\) from the initial state \(p\). Checking whether \(M_a\) accepts \(\omega\) is ensured by Item 10. If \(BP'_{i}\) is at state \(q_1\) with \(\gamma\) on the top of the stack and \(q_1 \gamma \rightarrow q_2\) is a transition of \(M_a\), then \(BP'_{i}\) pops \(\gamma\) from the stack and moves the control location from \(q_1\) to \(q_2\). Popping \(\gamma\) 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 \(q \in \text{Acc}_a\), i.e., the run of \(BP'_{i}\) from \(p \omega\) reaches the control location \(q\) with the empty stack, i.e., the stack only contains \(\#\). Thus, \(BP'_{i}\) should have an infinite run from \(q \#\) which infinitely often visits.
some control locations in $F_i$. This is ensured by adding a loop on the configuration $q^\#_i$ (Item 11) and adding $q$ into $F_i$. The case $\psi = \neg a \ s.t. \ a \in A(f_i)$ is similar.

If $\psi = \psi_1 \land \psi_2$, then, for every $p\omega \in P'_i \times \Gamma^*_i$, $p\omega$ satisfies $\psi$ iff $p\omega$ satisfies $\psi_1$ and $\psi_2$. This is ensured by Item 2 stating that $BP'_i$ has an accepting run from $[p, \psi_1 \land \psi_2]_\omega$ iff $BP'_i$ 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 $\psi = \text{EX}\psi_1$, then, for every $p\psi u \in P'_i \times \Gamma^*_i$ s.t. $\gamma \in \Gamma$, $p\psi u$ satisfies $\psi$ iff there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \mapsto p''\omega' \in \Delta_i$ such that $p''\omega'$ satisfies $\psi_1$. Thus, $BP'_i$ should have an accepting run from $[p, \psi_1]_\omega u$ iff $BP'_i$ 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 $BP'_i$. Item 5 is analogous.

If $\psi = \text{E} \psi_1 U \psi_2$, then, for every $p\psi u \in P'_i \times \Gamma^*_i$ s.t. $\gamma \in \Gamma$, $p\psi u$ satisfies $\psi$ iff either it satisfies $\psi_2$, or it satisfies $\psi_1$ and there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \mapsto p''\omega' \in \Delta_i$ such that $p''\omega'$ satisfies $\psi$. Thus, $BP'_i$ has an accepting run from $[p, \psi_1]_\omega u$ if either $BP'_i$ has an accepting run from $[p, \psi_2]_\omega u$ or $BP'_i$ has an accepting run from $[p, \psi_1]_\omega u$ and $[p', \psi_2]_\omega u$. 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 $BP'_i$. The case $\psi = A[\psi_1 U \psi_2]$ is analogous.

Item 8 expresses that if $\psi = \text{E} \psi_1 R \psi_2$, then, for every $p\psi u \in P'_i \times \Gamma^*_i$ s.t. $\gamma \in \Gamma$, $p\psi u$ satisfies $\psi$ iff it satisfies $\psi_2$, and either it satisfies also $\psi_1$, or there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \mapsto p''\omega' \in \Delta_i$ such that $p''\omega'$ satisfies $\psi$. This guarantees that $\psi_2$ holds either always, or until both $\psi_1$ and $\psi_2$ hold. The fact that the state $[p, \psi]$ is in $F_i$ ensures that paths where $\psi_2$ always hold are accepting. If $t_2$ is the fired transition rule, the created DCLIC $p''\omega'$ should also be created by $BP'_i$. 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 $BP'_i$ with $O(|P_i| \cdot |f_i| + \sum_{a \in A(f_i)}(|Q_a| + |Q_{\omega a}|))$ states and $O(|P_i| \cdot |f_i| + |A_i|)|f_i| + \sum_{a \in A(f_i)}(|\delta_a| + |\delta_{\omega a}|))$ transition rules such that for every $(p\omega, D) \in P_i \times \Gamma^*_i \times 2^{D_i}$, $p\omega \models_D f_i$ iff $([p, f_i]_\omega, D) \in L(BP'_i)$.

### 4.4 Computing $L(BP'_i)$

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

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

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

**Input:** An ABPDS $2BP'_1 = (P'_1, \Gamma_i, \Delta'_i, F_i)$;

**Output:** An AMA $\mathcal{A}'_i = (Q_i, \Gamma_i, \delta_i, I_i, \{q_f\})$ s.t. $L(\mathcal{A}'_i) = Y_{2BP'_1}$;

1. Let $k := 0, \delta_i := \{(q_f, \gamma, \emptyset, \{q_f\}) \text{ for every } \gamma \in \Gamma_i\}$, and $\forall p \in P'_i, p^0 := q_f$;
2. repeat we call this loop $loop_1$
3. $k := k + 1$;
4. Add a new transition rule $p^k \xrightarrow{\epsilon/0_i} \{p^{k-1}\}$ in $\delta_i$ for every $p \in F_i$;
5. repeat we call this loop $loop_2$
6. For every $p\gamma \leftarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \triangleright D$ in $\Delta'_i$,
7. and every case $p_j^k \xrightarrow{\omega_j/D_i} R_j$ for all $j$, $1 \leq j \leq h$;
8. $p^k \xrightarrow{\gamma/D_i(\bigcup_{j=1}^{h} D_j)} R_i \cup \bigcup_{j=1}^{h} R_j \in \delta_i$;
9. until No new transition rule can be added;
10. Remove from $\delta_i$ the transition rules $p^k \xrightarrow{\epsilon/0_i}, \{p^{k-1}\}, \forall p \in F_i$;
11. Replace in $\delta_i$ 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 $\forall p \in P'_i, \gamma \in \Gamma_i, R \subseteq P'_i \times \{k\} \cup \{q_f\}, D \subseteq D_i, p^k \xrightarrow{\gamma/D_i}, R \in \delta_i$ 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 $2BP'_1$ 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 $Y_{2BP'_i} = \bigcap_{j \geq 1} Y_j$, for every $(c, D) \in Y_{2BP'_i}, 2BP'_1$ 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.** $L(2BP'_i) = Y_{2BP'_i}$.

**Computing $Y_{2BP'_i}$:** We show that $Y_{2BP'_i}$ 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 $q^k_i$ denote $(q, k) \in P'_i \times \mathbb{N}$. Intuitively, to compute $Y_{2BP'_i}$, we will compute iteratively the different $Y_{j}$. The iterative procedure computes different AMAs. To force termination, we use an acceleration based on the projection functions $\pi^{-1}$ and $\pi^k$: for every $S \subseteq Q_i$,

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

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

Algorithm[1] computes an AMA $\mathcal{A}'_i$ recognizing $Y_{2BP'_i}$. 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 $p^k \xrightarrow{\epsilon/D_i} \{q_f\}$. Suppose the algorithm is at the beginning
of the $k^{th}$ step (loop 1). Line 4 adds the $\epsilon$-transition $p^{k} \xrightarrow{\epsilon(0)} i \{p^{k-1}\}$ for every $p \in F_i$. Then, we obtain $L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{D_i}$. loop 2 (Lines 5-9) is the saturation procedure that computes $pre_{\text{grp}}^+(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{D_i})$. Line 10 removes the $\epsilon$-transition $p^{k} \xrightarrow{\epsilon(0)} i \{p^{k-1}\}$ for every $p \in F_i$. After this, we obtain $pre_{\text{grp}}^+(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{D_i})$. Thus, in case of termination, the algorithm outputs $Y_{\text{grp}}$. 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 $Y_{\text{grp}}$.

**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 $p \gamma \leftarrow \{p_1 \omega_1, ..., p_m \omega_m\}$, i.e., $D_i = \emptyset$. The main differences are:

To compute $pre_{\text{grp}}^+(L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{D_i})$, instead of using the following saturation procedure given in [2] that computes reachable configurations of Alternating PDSs:

If $p \gamma \leftarrow \{p_1 \omega_1, ..., p_m \omega_m\} \in A'$ and $p^{k} \xrightarrow{\omega_j} \gamma R_j$, for $1 \leq j \leq m$, add $p^{k} \xrightarrow{\gamma(0)} \bigcup_{j=1}^{m} R_j$ in $\delta_i$.

We use the following saturation procedure:

If $p \gamma \leftarrow \{p_1 \omega_1, ..., p_m \omega_m\} \rightarrow D \in A'$ and $p^{k} \xrightarrow{\omega_j/D_j} \gamma R_j$, for $1 \leq j \leq h$, add $p^{k} \xrightarrow{\gamma(D_j \cup \bigcup_{j=1}^{h} D_j)} \bigcup_{j=1}^{h} R_j$ in $\delta_i$.

The idea behind our saturation procedure is the following: suppose $p \gamma \leftarrow \{p_1 \omega_1, ..., p_m \omega_m\} \rightarrow D \in A'$ and for every $j$, $1 \leq j \leq h$, $(p_j \omega_j \omega', D_j)$ is in $L(A_{k-1}) \cap F_i \times \Gamma_i^* \times 2^{D_i}$ (i.e., $p^{k,j} \xrightarrow{\omega_j/D_j} \gamma R_j \xrightarrow{\omega_j' / D_j'} \gamma q_j$ and $D_j = D_j' \cup D''_j$). Then, Lines 3-6 add the new transition rule $p^{k} \xrightarrow{\gamma(D_j \cup \bigcup_{j=1}^{h} D_j)} \bigcup_{j=1}^{h} R_j$ that allows to accept $(p \gamma \omega', D \cup \bigcup_{j=1}^{h} D_j)$, i.e., $(p \gamma \omega', D \cup \bigcup_{j=1}^{h} D_j) \in pre_{\text{grp}}^+(\{(p_1 \omega_1 \omega', D_1), ..., (p_m \omega_m \omega', D_m)\})$. □

**Complexity.** Following [20], we can show that loop 2 can be done in time $O(|P_i| \cdot |A'| \cdot 2|P_i|^{2} + |D_i|)$, i.e., $(p \gamma \omega', D \cup \bigcup_{j=1}^{h} D_j)$ (Line 11) and termination condition (Line 12) can be done in time $O(|\Gamma|^* \cdot |P_i|^* \cdot 2|P_i|^{2} + |D_i|)$ and $O(|\Gamma|^* \cdot |P_i|^* \cdot 2|P_i|^{2} + |D_i|)$, respectively. Putting all these estimations together, the global complexity of Algorithm 1 is $O(|\Gamma|^* \cdot |P_i|^* \cdot 2|P_i|^{2} + |D_i|)$.

By Proposition 5 and Theorem 6 we get:

**Lemma 4.** Given an ABDPDS $\mathcal{BP}'$, we can construct an AMA $\mathcal{R}'$ with $O(|\Gamma|^* \cdot |P'_i|^* \cdot 2|P'_i|^* + |D'_i|)$ transitions and $O(|P'_i|)$ states in time $O(|\Gamma_i|^* \cdot |P'_i|^* \cdot 2|P_i|^* + |D_i|)$ since $L(\mathcal{BP}'_i) = L(\mathcal{A}')$.

From Lemma 4 and Lemma 3 we get:

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

### 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 $p \omega \in P_i \times \Gamma_i^*$, $p \omega \models f_i$ 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 $D'_{fp}$ of DCLICs s.t. for every $d \in \bigcup_{i=1}^{n} D_i$, $d$ satisfies $f$ iff $d \in D'_{fp}$. Then, to check whether a global configuration $G$ satisfies $f$, it is sufficient to check whether for every $p\omega \in G$, there exists $D \subseteq D'_{fp}$ s.t. $(p, f_{\wp}(p)\omega, D) \in L(\mathcal{A}'_{\wp(p)})$. $D'_{fp}$ 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 $O(\sum_{i=1}^{n} ((|P_i| \cdot |f_i| + k)^2 \cdot (|P_i| \cdot |L_i| + |A_i| |f_i| + d) \cdot |\Gamma_i| \cdot 2^{(|P_i| |f_i| + k) + |Q_i|}))$ s.t. for every global configuration $G$, $G$ satisfies $f$ iff for every $p\omega \in G$, there exists $D \subseteq D'_{fp}$ such that $(p, f_{\wp}(p)\omega, D) \in L(\mathcal{A}'_{\wp(p)})$, where $k = \sum_{a \in \mathcal{A}(f_i)} (|Q_a| + |Q_{\neg a}|)$ and $d = \sum_{a \in \mathcal{A}(f_i)} (|\delta_a| + |\delta_{\neg a}|)$.

**References**


