DOI 10.1007/s00165-014-0330-y
BCS © 2015
Formal Aspects of Computing (2015) 27: 397–421

Formal Aspects of Computing

Model checking dynamic pushdown networks

Fu Song[1] and Tayssir Touili[2]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, No. 3663 Zhongshan Road(N), Shanghai, People's Republic of China
[2] LIAFA, CNRS and Université Paris Diderot, Paris, France

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 $\bigwedge f_i$ such that 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.

Keywords: Model checking, Dynamic pushdown networks, LTL, CTL

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) [BMOT05] 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 [BMOT05, Lug11, LMOW09, GLMO+11, LMO07, Wen10], 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.

Correspondence and offprint requests to: F. Song, E-mail: fsong@sei.ecnu.edu.cn

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 [KG06]. 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 [BEM97, EKS03, EHRS00, ST11]. 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 [BEM97, EHRS00, EKS03] and CTL [ST11] 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 [BMOT05]. Several other works use DPN and its extensions to model multi-threaded programs [BMOT05, GLMO+11, LMOW09, Lug11, Wen10]. All these works only consider reachability issues. Ground Tree Rewrite Systems [GL11] and process rewrite systems [BKRS09, May00] are two models of multi-threaded programs with procedure calls and threads creation. However, [May00] only considers reachability problem and [GL11, BKRS09] only consider subclasses of LTL. We consider LTL and CTL model checking problems.

Pushdown networks with communication between processes are studied in [BET03, CCK+06, ABT08, TA10]. These works consider systems with a fixed number of threads. [LMO07, LMO08] use parallel flow graphs to model multi-threaded programs. However, all these works only consider reachability. [Yah01] considers safety properties of multi-threaded programs.

[KG06, KG07, KIG05] 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. [KLTR09] 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. Sections 3 and 4 show LTL model checking for DPNs with simple valuations and regular valuations, respectively. Section 5 shows CTL model checking for DPNs. We conclude in Sect. 6.

This paper is the full version of the paper [ST13]. In this full version, we add the proofs of theorems and the technical details of the results on single-indexed LTL model checking with regular valuations.

```
     void main(){
l0 :   Socket sSocket= new Socket(445);
l1 :   Socket cSocket=null;
l2 :   while(true){
l3 :     cSocket=sSocket.accept()
l4 :     new Thread(worker,cSocket);
l5 :      if(cond) break;   }
l6 :   return;   }
```

```
     void worker(Socket s){
l'0 : String str=null;
l'1 : Resource res==s.readLine();
l'2 :  if(str=="req"){
l'3 :    // prepare response for req
l'4 :    s.send("ack"); }
l'5 :  if(str=="req'"){
l'6 :  // prepare response for req'
l'7 :    s.send("ack'"); }
l'8 : s.close();
l'9 : return; }
```

Fig. 1. A simplified concurrent server program, where $l_0, \dots, l_6, l'_0, \dots, l'_9$ are program control points

2. Preliminaries

2.1. Dynamic pushdown networks

Definition 2.1 A Dynamic Pushdown Network (DPN) $\mathcal{M}$ is a set $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ such that for every i, $1 \le i \le n$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$ is a dynamic pushdown system (DPDS), where P_i is a finite set of control locations such that $P_k \cap P_i = \emptyset$ 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) $q\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2$ such that $q, p_1 \in P_i, \gamma \in \Gamma_i, \omega_1 \in \Gamma_i^*, p_2\omega_2 \in P_j \times \Gamma_j^*$ for some j, $1 \le j \le n$.

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) $q\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2$ such that $p_2\omega_2 \in P_j \times \Gamma_j$, 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, if the transition (b) is used for moving to $p_1\omega_1 u$, then a new instance of $\mathcal{P}_j$ starting from $p_2\omega_2$ is created. 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 $q\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2 \in \Delta_i$, then $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega \rhd \{p_2\omega_2\}$. To unify the presentation, if $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega$, we sometimes write $q\gamma\omega \Longrightarrow_i p_1\omega_1\omega \rhd \emptyset$ instead. The transitive and reflexive closure of $\Longrightarrow_i$ is denoted by $\Longrightarrow_i^*$. Formally, for every $p\omega \in P_i \times \Gamma_i^*$, $p\omega \Longrightarrow_i^* p\omega \rhd \emptyset$; and if $p\omega \Longrightarrow_i p_1\omega_1 \rhd D_1$ and $p_1\omega_1 \Longrightarrow_i^* p_2\omega_2 \rhd D_2$, then $p\omega \Longrightarrow_i^* p_2\omega_2 \rhd D_1 \cup D_2$. $\Longrightarrow_i^+$ 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 $c_0c_1\dots$ over $\mathcal{P}_i \times \Gamma_i^*$ such that for every $j \ge 0$, $c_j \Longrightarrow_i c_{j+1} \rhd D$ 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, $1 \le i \le n$, 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 $\mathcal{D}_i = \{p_2\omega_2 \in \bigcup_{i=1}^n P_i \times \Gamma_i^* \mid q\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2 \in \Delta_i\}$ be the set of potential DCLICs of the DPDS $\mathcal{P}_i$.

Example 2.1 Let us consider the simplified concurrent server program shown in Fig. 1. The main process creates a socket object *sSocket* to listen on the port 445 and waits for requests from clients. When a request (connection) arrives from a client, the main process creates a new process that runs *worker* with the parameter *cSocket*, and then continues listening on the port 445 if *cond* is true (otherwise terminates). The process *worker* accepts a connection s from the main process and processes it and reads the request. If it receives *req* (resp. *req'*) from the client, then it prepares the response and sends *ack* (resp. *ack'*). Later, it closes the connection by invoking *close()* and this process terminates. Otherwise, it closes the connection directly. This program has an unbound number of processes of *Worker*. We can construct a DPN to model this program. Let $\mathcal{M} = (\mathcal{P}_1, \mathcal{P}_2)$ be the DPN such that for $i \in \{1, 2\}$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$, where $P_i = \{p_i\}$, $\Gamma_1 = \{l_0, \ldots, l_6\}$, $\Gamma_2 = \{l'_0, \ldots, l'_9\}$, Δ_1 and Δ_2 are as follows.

$$\Delta_1 = \left\{\begin{array}{llll} p_1 l_0 \hookrightarrow p_1 l_1, & p_1 l_1 \hookrightarrow p_1 l_2, & p_1 l_2 \hookrightarrow p_1 l_3, & p_1 l_3 \hookrightarrow p_1 accept_0 l_4, \\ p_1 l_4 \hookrightarrow p_1 l_5 \rhd p_2 l_0, & p_1 l_5 \hookrightarrow p_1 l_6, & p_1 l_5 \hookrightarrow p_1 l_2, & p_1 l_6 \hookrightarrow p_1 \epsilon. \end{array}\right\}$$

$$\Delta_2 = \left\{\begin{array}{llll} p_2 l'_0 \hookrightarrow p_2 l'_1, & p_2 l'_1 \hookrightarrow p_2 readLine_0 l'_2, & p_2 l'_2 \hookrightarrow p_2 l'_3, & p_2 l'_2 \hookrightarrow p_2 l'_5, \\ p_2 l'_3 \hookrightarrow p_2 l'_4, & p_2 l'_4 \hookrightarrow p_2 send_0 l'_5, & p_2 l'_5 \hookrightarrow p_2 l'_6, & p_2 l'_5 \hookrightarrow p_2 l'_8, \\ p_2 l'_6 \hookrightarrow p_2 l'_7, & p_2 l'_7 \hookrightarrow p_2 send_0 l'_8, & p_2 l'_8 \hookrightarrow p_2 close_0 l'_9, & p_2 l'_9 \hookrightarrow p_2 \epsilon. \end{array}\right\}$$

where $accept_0$, $send_0$, $close_0$ and $readLine_0$ are the entry points of the functions *accept*, *send*, *close* and *readLine* respectively. For the sake of simplicity, we do not take the constructor of *worker* into account. Intuitively, $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) models the *main* (resp. *worker*) process. The runs of $\mathcal{M}$ are an over-approximation of the program's executions.

2.2. LTL and Büchi automata

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

Definition 2.2 The set of LTL formulas is given by (where $a \in AP$):

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

Given an ω-word $\eta = \alpha_0\alpha_1 \ldots$ 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 2.3 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 $\alpha_0\alpha_1 \ldots$ is a sequence of states $q_0 q_1 \ldots$ such that $q_0 = g^0$ 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 such that $\Sigma = 2^{AP}$ recognizing all the ω-words that satisfy f [VW86].

Example 2.2 Let us consider the program in Example 2.1. One can use single-indexed LTL to express properties of this program. For instance, let us consider a property of the *worker* process: "*worker* replies ack (resp. ack') rather than ack' (resp. ack) for each request req (resp. req') from each client if this process is created by the main process". We can express this property in the LTL formula ψ_2:

$$\psi_2 = \mathbf{G}((req \Longrightarrow \mathbf{F}(ack \wedge \mathbf{G}\neg ack')) \wedge (req' \Longrightarrow \mathbf{F}(ack' \wedge \mathbf{G}\neg ack))),$$

where req, req', ack and ack' are atomic propositions associated to l'_3, l'_6, l'_4, and l'_7, respectively. It is not correct that *worker* replies ack' (resp. ack) or nothing if it receives req (resp. req'). Also, the main process should have an execution that always wait for requests from clients and processes every request. Otherwise, this server can serve only a bound number of requests. We can express this property by the LTL formula ψ_1:

$$\psi_1 = \mathbf{G}\ \mathbf{F}\ accept,$$

where $accept$ is an atomic proposition associated to l_3. Therefore, we can check the property $\psi_1 \wedge \psi_2$. Note that one cannot check ψ_1 and ψ_2, independently, since the *worker* process should satisfy ψ_2 only if it is created by the main process rather than others.

2.3. Single-indexed LTL for DPNs

Let $\mathcal{M} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ be a DPN. A *single-indexed* LTL formula is a formula f of the form $\bigwedge_{i=1}^{n} f_i$ [1] such that for every i, $1 \le i \le n$, 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 $p_0\omega_0 p_1\omega_1 \dots$ of $\mathcal{P}_i$ satisfies f_i iff the ω-word $\alpha_0\alpha_1 \dots$ where for every $j \ge 0$, $\alpha_j = \{a \in AP \mid p_j\omega_j \in \lambda(a)\}$, 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, $1 \le i \le n$, 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 $\mathcal{M} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ where for every i, $1 \le i \le n$, $\mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$, 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 $\mathcal{G}$ such that 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. Given a set $U \subseteq (P_i \times \Gamma_i^*) \times 2^{\mathcal{D}_i}$, $pre_{\mathcal{P}_i}(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times \Gamma_i^*$, such that $c \Longrightarrow_i c' \rhd D_1$ 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, given a set $U \subseteq (P_i \times \Gamma_i^*) \times 2^{\mathcal{D}_i}$, $pre^*_{\mathcal{P}_i}(U) = \{(c, D_1 \cup D_2) \mid \exists c' \in P_i \times \Gamma_i^*$, such that $c \Longrightarrow_i^* c' \rhd D_1$ 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 2.4 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, $I_i \subseteq Q_i$ 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 an AMA $\mathcal{A}_i$ such that $\delta_i \subseteq (Q_i \times \Gamma_i) \times 2^{\mathcal{D}_i} \times Q_i$.

We write $p \xrightarrow{\gamma/D}_i \{q_1, \dots, q_m\}$ instead of $(p, \gamma, D, \{q_1, \dots, q_m\}) \in \delta_i$, 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 such that: (1) $q \xrightarrow{\epsilon/\emptyset}_i^* \{q\}$ for every $q \in Q_i$, (2) if $q \xrightarrow{\gamma/D}_i \{q_1, \dots, q_m\}$ and $q_k \xrightarrow{\omega/D_k}_i^* S_k$ for every k, $1 \le k \le m$, 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 $(p\omega, D) \in P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ such that $p \xrightarrow{\omega/D}_i^* S$ 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 $\mathcal{A}_i$ such that $L(\mathcal{A}_i) = W$. A set of local configurations $C \subseteq P_i \times \Gamma_i^*$ is *regular* iff $C \times \{\emptyset\}$ is a regular set. Here "regular" means that the set of words of the stack contents in a regular set of configurations C is a regular language, i.e., the set of words $\{\omega \in \Gamma_i^* \mid p\omega \in C\}$ can be represented by a standard finite state automaton. In AMAs, the initial states are the control locations of DPDSs, and sets of DCLICs labeled to transition rules could also be regarded as input characters. Similar as the transformation of alternating finite state automata to finite state automata by performing a similar kind of powerset construction, given an AMA $\mathcal{A}_i$, we can construct a MA $\mathcal{A}'_i$ such that $L(\mathcal{A}_i) = L(\mathcal{A}'_i)$.

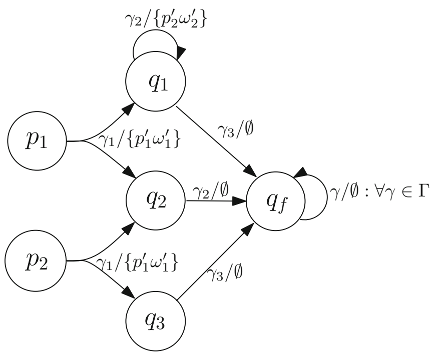
Example 2.3 Let us consider the AMA $\mathcal{A}_i = (\{p_1, p_2, q_1, q_2, q_3, q_f\}, \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}, \delta, \{p_1, p_2\}, \{q_f\})$, where δ is shown in Fig. 2a, $\mathcal{D}_i = \{p'_1\omega'_1, p'_2\omega'_2\}$. The graph representation of $\mathcal{A}_i$ is shown in Fig. 2b. Consider the configuration $p_1\gamma_1\gamma_2\gamma_3$, we have $p_1 \xrightarrow{\gamma_1\gamma_2\gamma_3/\{p'_1\omega'_1, p'_2\omega'_2\}}_i^* \{q_f\}$. Therefore, $(p_1\gamma_1\gamma_2\gamma_3, \{p'_1\omega'_1, p'_2\omega'_2\}) \in L(\mathcal{A}_i)$. Consider the configuration $p_2\gamma_1\gamma_2\gamma_3$, it is not accepted by $\mathcal{A}_i$ for any set of DCLICs. In this example,

$$L(\mathcal{A}_i) = \{(p_1\gamma_1\gamma_2^i\gamma_3\omega, \{p'_1\omega'_1, p'_2\omega'_2\}) \mid i \ge 1, \omega \in \Gamma^*\}.$$

[1] A single-indexed LTL formula $\bigwedge_{i=1}^{n} f_i$ denotes the formula $\bigwedge_{i=1}^{n} \mathbf{E}f_i$. Formulas of the form $\bigvee_{i=1}^{n} \mathbf{E}f_i$ can be verified by checking $\mathbf{E}f_i$ for every i, $1 \le i \le n$. Formulas of the form $\bigvee_{i=1}^{n} \mathbf{A}f_i$ can be verified by checking $\mathbf{E}\neg f_i$ for every i, $1 \le i \le n$.

$$\delta = \left\{ \begin{array}{ll} p_1 \xrightarrow{\gamma_1/\{p'_1\omega'_1\}}_i \{q_1, q_2\}, & p_2 \xrightarrow{\gamma_1/\{p'_1\omega'_1\}}_i \{q_2, q_3\}, \\ q_1 \xrightarrow{\gamma_2/\{p'_2\omega'_2\}}_i \{q_1\}, & q_3 \xrightarrow{\gamma_3/\emptyset}_i \{q_f\}, \\ q_1 \xrightarrow{\gamma_3/\emptyset}_i \{q_f\}, & q_2 \xrightarrow{\gamma_2/\emptyset}_i \{q_f\}, \\ q_f \xrightarrow{\gamma/\emptyset}_i \{q_f\}, \forall\gamma \in \Gamma. & \end{array} \right\}$$

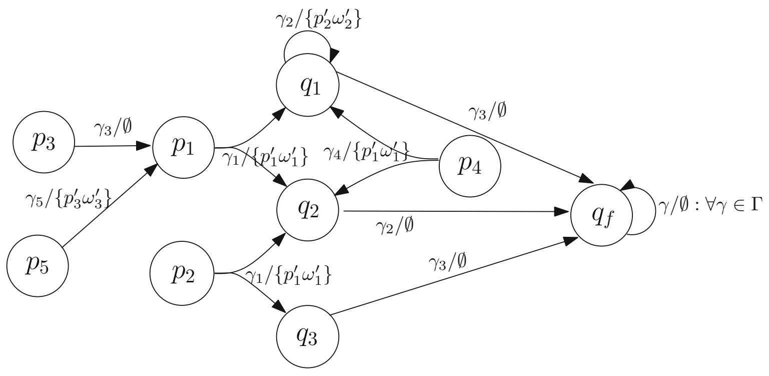
(a)



$\gamma_2/\{p'_2\omega'_2\}$
q_1
p_1
$\gamma_1/\{p'_1\omega'_1\}$
$\gamma_3/\emptyset$
q_2
$\gamma_2/\emptyset$
q_f
$\gamma/\emptyset : \forall\gamma \in \Gamma$
p_2
$\gamma_1/\{p'_1\omega'_1\}$
$\gamma_3/\emptyset$
q_3

(b)

Fig. 2. Graph representation of the AMA of Example 2.3



$\gamma_2/\{p'_2\omega'_2\}$
q_1
p_3
$\gamma_3/\emptyset$
p_1
$\gamma_3/\emptyset$
$\gamma_1/\{p'_1\omega'_1\}$
$\gamma_4/\{p'_1\omega'_1\}$
p_4
$\gamma_5/\{p'_3\omega'_3\}$
q_2
q_f
$\gamma/\emptyset : \forall\gamma \in \Gamma$
$\gamma_2/\emptyset$
p_5
p_2
$\gamma_1/\{p'_1\omega'_1\}$
$\gamma_3/\emptyset$
q_3

Fig. 3. The resulting AMA $\mathcal{A}_i^{pre^*}$

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 [BEM97]).

- *For every* $p\gamma \hookrightarrow p_1\omega_1 \in \Delta_i$ *and* $p_1 \xrightarrow{\omega_1/D}{}^*_i q$, *add a new rule* $p \xrightarrow{\gamma/D}_i q$;
- *For every* $p\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2 \in \Delta_i$ *and* $p_1 \xrightarrow{\omega_1/D}{}^*_i q$, *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 $|\Delta_i|$ of transition rules of $\mathcal{P}_i$ is finite, DCLICs are fixed by transition rules of the form $p\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2$, we get that the number of DCLICs is fixed and finite. Moveover, 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 $p\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2 \in \Delta_i$ 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.

Example 2.4 Let us consider the DPDS $\mathcal{P}_i$ such that $\Delta_i = \{p_3\gamma_3 \hookrightarrow_i p_1\epsilon,\ p_4\gamma_4 \hookrightarrow_i p_3\gamma_3\gamma_1,\ p_5\gamma_5 \hookrightarrow_i p_3\gamma_3 \rhd p'_3\gamma'_3\}$ and the AMA $\mathcal{A}_i$ as in Example 2.3. The resulting AMA $\mathcal{A}_i^{pre^*}$ is shown in Fig. 3. The saturation procedure processes as follows.

- Since $p_1 \xrightarrow{\epsilon/\emptyset}{}^*_i \{p_1\}$, the transition rule $p_3\gamma_3 \hookrightarrow_i p_1\epsilon$ allows to add $p_3 \xrightarrow{\gamma_3/\emptyset}{}_i \{p_1\}$.
- The newly added rule $p_3 \xrightarrow{\gamma_3/\emptyset}{}_i \{p_1\}$ together with $p_4\gamma_4 \hookrightarrow_i p_3\gamma_3\gamma_1$ and $p_1 \xrightarrow{\gamma_1/\{p'_1\omega'_1\}}{}_i \{q_1, q_2\}$ will create the rule $p_4 \xrightarrow{\gamma_4/\{p'_1\omega'_1\}}{}_i \{q_1, q_2\}$.
- Also, the newly added rule $p_3 \xrightarrow{\gamma_3/\emptyset}{}_i \{p_1\}$ together with $p_5\gamma_5 \hookrightarrow_i p_3\gamma_3 \rhd p'_3\gamma'_3$ will create the rule $p_5 \xrightarrow{\gamma_5/\{p'_3\omega'_3\}}{}_i \{p_1\}$.
- No new rule can be added, the saturation procedure terminates.

We have the following two lemmas that bridge the successor relation of DPDSs and MAs. For the sake of simplicity, let $c \Longrightarrow^k_i c' \rhd D$ denote that the local run of $\mathcal{P}_i$ moves from c to c' after k steps and creates the set of DCLICs D. Formally, $c \Longrightarrow^0_i c \rhd \emptyset$ for every $c \in P_i \times \Gamma^*_i$, if $c \Longrightarrow^{k-1}_i c'' \rhd D_1$ and $c'' \Longrightarrow_i c' \rhd D_2$, then $c \Longrightarrow^k_i c' \rhd D_1 \cup D_2$. Let $q \xrightarrow{u/D_1}{}^k_i g$ denote that the run of an AMA A_i can move from the state q to the state g after adding k number of transition rules into $A_i^{pre^*}$.

Lemma 2.1 *For every tuple* $(qu, D_1) \in L(A_i)$*, if* $p\omega \Longrightarrow^*_i qu \rhd D_2$*, then* $p \xrightarrow{\omega/D_1\cup D_2}{}^*_i g$ *for some final state* g *of* $A_i^{pre^*}$.

Proof. Assume $p\omega \Longrightarrow^k_i qu \rhd D_2$. The proof proceeds by induction on k.

- **Basis**. $k = 0$. Then, $p = q, \omega = u$. Since $(qu, D_1) \in L(A_i)$, we obtain that $q \xrightarrow{u/D_1}{}^0_i g$ for some final state g of A_i. This implies that $p \xrightarrow{\omega/D_1}{}^*_i g$. Note that $D_2 = \emptyset$.
- **Step**. $k \geq 1$. Then,

$$\boxed{\begin{array}{l}\text{(a) there exist local configurations } p'\omega' \in P_i \times \Gamma^*_i \text{ and } q'u' \in D_2 \text{ such that}\\ \qquad p\omega \Longrightarrow^1_i p'\omega' \rhd \{q'u'\},\ p'\omega' \Longrightarrow^{k-1}_i qu \rhd D_3 \text{ and } D_2 = D_3 \cup \{q'u'\},\end{array}}$$

or

$$\boxed{\begin{array}{l}\text{(b) there is a local configuration } p'\omega' \in P_i \times \Gamma^*_i \text{ such that}\\ \qquad p\omega \Longrightarrow^1_i p'\omega' \rhd \emptyset \text{ and } p'\omega' \Longrightarrow^{k-1}_i qu \rhd D_2.\end{array}}$$

The proof depends on whether (a) or (b) holds.

  - **Case** (a): By applying the induction hypothesis to $p'\omega' \Longrightarrow^{k-1}_i qu \rhd D_3$, we obtain that

    $p' \xrightarrow{\omega'/D_3\cup D_1}{}^*_i g$ for some final state g.

    Since $p\omega \Longrightarrow^1_i p'\omega' \rhd \{q'u'\}$, there exist $\gamma \in \Gamma_i$, $\omega_1 \in \Gamma^*_i$ and $u_1 \in \Gamma^*_i$ such that

    $\omega = \gamma\omega_1, \omega' = u_1\omega_1$ and $p\gamma \hookrightarrow p'u_1 \rhd q'u' \in \Delta_i$.

    Let q_1 be a state of $A_i^{pre^*}$, $D_4 \subseteq \mathcal{D}_i$ and $D_5 \subseteq \mathcal{D}_i$ such that

    $p' \xrightarrow{u_1/D_4}{}^*_i q_1 \xrightarrow{\omega_1/D_5}{}^*_i g$ and $D_3 \cup D_1 = D_4 \cup D_5$.

    By applying the saturation procedure to $p\gamma \hookrightarrow p'u_1 \rhd q'u'$, we obtain that

    $p \xrightarrow{\gamma/\{q'u'\}\cup D_4}{}_i q_1$.

    Hence, we get that $p \xrightarrow{\gamma\omega_1/\{q'u'\}\cup D_4\cup D_5}{}^*_i g$.

    Since $\omega = \gamma\omega_1$, $D_1 \cup D_2 = D_1 \cup D_3 \cup \{q'u'\} = D_4 \cup D_5 \cup \{q'u'\}$, we obtain that $p \xrightarrow{\omega/D_1\cup D_2}{}^*_i g$.
  - **Case** (b): By applying the induction hypothesis to $p'\omega' \Longrightarrow^{k-1}_i qu \rhd D_2$, we obtain that

    $p' \xrightarrow{\omega'/D_2\cup D_1}{}^*_i g$ for some final state g.

Since $p\omega \Longrightarrow_i^1 p'\omega' \rhd \emptyset$, there exist $\gamma \in \Gamma_i$, $\omega_1 \in \Gamma_i^*$ and $u_1 \in \Gamma_i^*$ such that

$$\omega = \gamma\omega_1, \omega' = u_1\omega_1 \quad \text{and} \quad p\gamma \hookrightarrow p'u_1 \in \Delta_i.$$

Let q_1 be a state of $A_i^{pre^*}$, $D_3, D_4 \subseteq \mathcal{D}_i$ such that

$$p' \xrightarrow{u_1/D_3}{}_i^* q_1 \xrightarrow{\omega_1/D_4}{}_i^* g \quad \text{and} \quad D_2 \cup D_1 = D_3 \cup D_4.$$

By applying the saturation procedure to $p\gamma \hookrightarrow p'u_1$, we get that

$$p \xrightarrow{\gamma/D_3}{}_i q_1.$$

Hence, we get that $p \xrightarrow{\gamma\omega_1/D_3 \cup D_4}{}^*_i g$.

Since $\omega = \gamma\omega_1$, $D_1 \cup D_2 = D_3 \cup D_4$, we obtain that $p \xrightarrow{\omega/D_1 \cup D_2}{}_i^* g$. □

Lemma 2.2 *If $p \xrightarrow{\omega/D}{}_i^* q$, then the following two properties hold:*

1. *$p\omega \Longrightarrow_i^* p'\omega' \rhd D_1$ for a local configuration $p'\omega' \in P_i \times \Gamma_i^*$, $D_1 \subseteq \mathcal{D}_i$ and $D_2 \subseteq \mathcal{D}_i$ such that $p' \xrightarrow{\omega'/D_2}{}_i^0 q$ and $D = D_1 \cup D_2$;*
2. *if q is an initial state, then $\omega' = \epsilon$ and $D_2 = \emptyset$.*

Proof. Assume $p \xrightarrow{\omega/D}{}_i^k q$. The proof proceeds by induction on k.

- **Basis**. $k = 0$. Since $p\omega \Longrightarrow_i^* p'\omega' \rhd D_1$ always hold when $p = p'$, $\omega = \omega'$, $D = D_1 = \emptyset$, $D = D_2$, we obtain that the property 1 holds. If q is an initial state, since there is no transition leading to an initial state, then $\omega' = \epsilon$.
- **Step**. $k \geq 1$. Let $t = p_1 \xrightarrow{\gamma/D_3}{}_i q'$ be the k^{th} transition rule added into A_i. Let j be the number of times that t is used in $p \xrightarrow{\omega/D}{}_i^* q$. The proof proceeds by induction on j.
  - **Basis**. $j = 0$. Then, $p \xrightarrow{\omega/D}{}_i^{k-1} q$. By applying the induction hypothesis on k, we obtain that the property (1) and the property (2) both hold.
  - **Step**. $j \geq 1$. Then, there exist $u \in \Gamma_i^*$, $v \in \Gamma_i^*$, $D_4 \subseteq \mathcal{D}_i$ and $D_5 \subseteq \mathcal{D}_i$ such that $\omega = u\gamma v$, $D = D_3 \cup D_4 \cup D_5$ and

    $$p \xrightarrow{u/D_4}{}_i^{k-1} p_1 \xrightarrow{\gamma/D_3}{}_i q' \xrightarrow{v/D_5}{}_i^* q.$$

    Since p_1 is an initial state, by applying the induction hypothesis to $p \xrightarrow{u/D_4}{}_i^{k-1} p_1$, we obtain that $pu \Longrightarrow_i^* p_1\epsilon \rhd D_4$. Since the transition rule t is added by the saturation procedure, then, either

    (a) there exist $p_2 \in P_i$ and $\omega_2 \in \Gamma_i^*$, such that $p_1\gamma \hookrightarrow p_2\omega_2 \in \Delta_i$ and $p_2 \xrightarrow{\omega_2/D_3}{}_i^{k-1} q'$,

    or

    (b) there exist $p_2 \in P_i$, $\omega_2 \in \Gamma_i^*$, $p_3\omega_3 \in \mathcal{D}_i$ and $D_6 \subseteq \mathcal{D}_i$ such that $D_3 = D_6 \cup \{p_3\omega_3\}$, $p_1\gamma \hookrightarrow p_2\omega_2 \rhd p_3\omega_3 \in \Delta_i$ and $p_2 \xrightarrow{\omega_2/D_6}{}_i^{k-1} q'$.

Thus, we obtain that either

$$\boxed{p_2 \xrightarrow{\omega_2/D_3}{}_i^{k-1} q' \xrightarrow{v/D_5}{}_i^{*} q \text{ [for case (a)],}}$$

or

$$\boxed{p_2 \xrightarrow{\omega_2/D_6}{}_i^{k-1} q' \xrightarrow{v/D_5}{}_i^{*} q \text{ [for case (b)].}}$$

Since the derivation of $p_2 \xrightarrow{\omega_2/D_3}{}_i^{k-1} q' \xrightarrow{v/D_5}{}_i^{*} q$ or $p_2 \xrightarrow{\omega_2/D_6}{}_i^{k-1} q' \xrightarrow{v/D_5}{}_i^{*} q$ uses the transition t less often than the derivation of $p \xrightarrow{\omega/D}{}_i^{k} q$, by applying the induction hypothesis, we obtain that $p_2\omega_2 v \Longrightarrow_i^* p'\omega' \rhd D_7$ and $p' \xrightarrow{\omega'/D_8}{}_i^{0} q$, where either

$$\boxed{D_7 \cup D_8 = D_3 \cup D_5 \text{ [for case (a)],}}$$

or

$$\boxed{D_7 \cup D_8 = D_6 \cup D_5 \text{ [for case (b)].}}$$

As there is no transition rule leading to an initial state, we obtain that if $\omega' = \epsilon$, then $D_8 = \emptyset$. Hence, we obtain that either

$$\boxed{\begin{array}{l} p\omega = pu\gamma v \Longrightarrow_i^* p_1\gamma v \rhd D_4,\ p_1\gamma v \Longrightarrow_i p_2\omega_2 v \rhd \emptyset \text{ and } p_2\omega_2 v \Longrightarrow_i^* p'\omega' \rhd D_7 \\ \text{for the case (a),} \end{array}}$$

or

$$\boxed{\begin{array}{l} p\omega = pu\gamma v \Longrightarrow_i^* p_1\gamma v \rhd D_4,\ p_1\gamma v \Longrightarrow_i p_2\omega_2 v \rhd \{p_3\omega_3\} \text{ and } p_2\omega_2 v \Longrightarrow_i^* p'\omega' \rhd D_7 \\ \text{for the case (b).} \end{array}}$$

Thus, we obtain that the properties (1) and (2) hold. Note that $D_2 = D_8$, and $D_1 = D_4 \cup D_7$ [for the case (a)], $D_1 = D_4 \cup D_7 \cup \{p_3\omega_3\}$ [for the case (b)]. □

Thus, from Lemmas 2.1 and 2.2, we obtain the following theorem.

Theorem 2.1 *Given a MA A_i recognizing a regular set W of the DPDS $\mathcal{P}_i$ for some i, we can construct a MA $A_i^{pre^*}$ recognizing $pre^*_{\mathcal{P}_i}(W)$ in time $O(|\Delta_i| \cdot |Q_i|^2 \cdot 2^{|\mathcal{D}_i|})$.*

Proof. Correctness:
($\Longrightarrow$) Let $(p\omega, D) \in pre_i^*(L(A_i))$. Then, $p\omega \Longrightarrow_i^* p'\omega' \rhd D_1$ for some tuple $(p'\omega', D_2) \in L(A_i)$ such that $D = D_1 \cup D_2$. By Lemma 2.1, $p \xrightarrow{\omega/D}{}_i^{*} g$ for some final state g of $A_i^{pre^*}$. So $(p\omega, D) \in L(A_i^{pre^*})$ holds.
($\Longleftarrow$) Let $(p\omega, D) \in L(A_i^{pre^*})$. Then, $p \xrightarrow{\omega/D}{}_i^{*} g$ for some final state g of $A_i^{pre^*}$. By Lemma 2.2, $p\omega \Longrightarrow_i^* p'\omega' \rhd D_1$ for some local configuration $p'\omega' \in P_i \times \Gamma_i^*$, $D_1 \subseteq \mathcal{D}_i$ such that $p' \xrightarrow{\omega'/D_2}{}_i^{0} g$ and $D = D_1 \cup D_2$.

Since $p' \xrightarrow{\omega'/D_2}{}_i^{0} g$ and g is a final state, we get that $(p'\omega', D_2) \in L(A_i)$. Thus, we obtain that $(p\omega, D) \in pre_i^*(L(A_i))$.
Complexity: It is shown in [EHRS00] that $A_i^{pre^*}$ can be computed in time $O(|\Delta_i| \cdot |Q_i|^2)$ if $\mathcal{D}_i = \emptyset$. We can extend the efficient algorithm of [EHRS00] so that $A_i^{pre^*}$ can be computed 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 $f = \bigwedge_{i=1}^n f_i$ iff $\mathcal{M}$ has a global run ρ from $\mathcal{G}$ such that 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 global runs of $\mathcal{G}$ is unbounded, and the number of local runs in each global run ρ can also be unbounded. We cannot check all the different instances of the DPDSs independently. Indeed, we do not 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 [BEM97, EHRS00] 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 $O(|P_{\wp(d)}|^2 \cdot |\Delta_{\wp(d)}| \cdot 2^{|f_{\wp(d)}|})$ [BEM97, EHRS00]. 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 $f = \bigwedge_{i=1}^{n} f_i$ iff for every configuration $c \in \mathcal{G}$, there exists a set of DCLICs D_c such that $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$ 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 $D_c \subseteq \mathcal{D}_{fp}$ such that $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$.

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 [BEM97, EHRS00]. 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 3.1 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 $(c, D) \in P_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ such that $\mathcal{BP}_i$ 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 $g_0 g_1 \ldots$ over an ω-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 $\mathcal{BP}_i$ has an accepting run $\sigma' = [p_0, g_0]\omega_0\, [p_1, g_1]\omega_1 \ldots$, 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 $F_i' = \{[p, g] \mid p \in P_i, g \in F_i\}$ ensure 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 3.1 *$\mathcal{P}_i$ 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 $L(\mathcal{BP}_i)$ can boil down to $pre^*_{\mathcal{BP}_i}$ computations.

Proposition 3.1 *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 $\exists D_1, D_2, D_3 \subseteq \mathcal{D}_i$ such that $D = D_1 \cup D_2 \cup D_3$, 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 $g \in F_i$, $v \in \Gamma_i^*$.

Proof. ($\Longrightarrow$) Let $\sigma = c_0c_1c_2\ldots$ such that for every $j \geq 0$ $c_j \Longrightarrow_i c_{j+1} \rhd I_j$ be the accepting run of $\mathcal{BP}_i$ such that $D \subseteq \mathcal{D}_i$ is the set of all the DCLICs created during this run, i.e., $D = \bigcup_{j\geq 0} I_j$. Note that D is a finite set, because $\mathcal{D}_i$ is a finite set. Let ω_j be the stack content of the local configuration c_j for every $j \geq 0$. Let σ^k denote the local configuration c_k. We construct a subsequence $c_{k_1}c_{k_2}\ldots$ such that

$$|\omega_{k_1}| = min\{|\omega_j| \mid j \geq 0\}$$
$$|\omega_{k_l}| = min\{|\omega_j| \mid j > k_{l-1}\}, \forall\, l \geq 1.$$

By this construction, we obtain that once the configuration σ^{k_l} is reached, the stack content except the topmost symbol will never change in the rest run of σ. Since $P_i \times \Gamma_i$ is finite, we can construct a subsequence $c_{j_1}c_{j_2}\ldots$ of $c_{k_1}c_{k_2}\ldots$ such that for every $h \geq 1$, the control location of c_{j_h} is p and the topmost symbol of the stack of c_{j_h} is γ.

Since σ is an accepting run, we can find an m such that

$$c_0 \Longrightarrow_i^* c_{j_1} \rhd D_4,\ c_{j_1} \Longrightarrow_i^+ c_g \rhd D_5,\ c_g \Longrightarrow_i^* c_{j_m} \rhd D_6,$$

and the control location p_g of c_g is in F_i, $D_4 = \bigcup_{h=0}^{j_1-1} I_h$, $D_5 = \bigcup_{h=j_1}^{g-1} I_h$, $D_6 = \bigcup_{h=g}^{j_m-1} I_h$, and for every $h \geq j_m$: $I_h \subseteq D_6$.

Let $c_{j_1} = p\gamma\omega$, then Item α_1 holds.

Since the stack content except the topmost symbol of the stack of c_{j_h} will never change in the rest of the run for every $h \geq 1$, there exist $u \in \Gamma_i^*$ and $v \in \Gamma_i^*$ such that $u\omega$ is the stack content of c_g, $v\omega$ is the stack content of c_{j_m}, $p\gamma \Longrightarrow_i^+ p_g u \rhd D_5$ and $p_g u \Longrightarrow_i^* p\gamma v \rhd D_6$. So Item α_2 holds.

Let $D_1 = D_4$, $D_2 = D_5$, $D_3 = D_6$. Since $D = D_4 \cup D_5 \cup D_6$, then $D = D_1 \cup D_2 \cup D_3$.

($\Longleftarrow$) To prove that $\mathcal{BP}_i$ has an accepting run from c and D is the set of DCLICs created during this run, it is sufficient to construct such an accepting run.

From Item α_2, we obtain that $p\gamma v^k\omega \Longrightarrow_i^+ p_g uv^k\omega \rhd D_2$ and $p_g uv^k \Longrightarrow_i^* p\gamma v^{k+1}\omega \rhd D_3$, for every $k \geq 0$.

From the fact that $p_g \in F_i$ and Item α_1, we deduce an accepting run and $D = D_1 \cup D_2 \cup D_3$. □

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 3.2 *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 $\exists D_1, D_2' \subseteq \mathcal{D}_i$ such that $D = D_1 \cup D_2'$, 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 3.2, we can get that $L(\mathcal{BP}_i) = \{(c, D_1 \cup D_2') \in P_i \times \Gamma_i \times 2^{\mathcal{D}_i} \mid \text{Items } \beta_1 \text{ and } \beta_2 \text{ hold}\}$. Since $F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ and $\{p\} \times \gamma\Gamma_i^* \times \{\emptyset\}$ are regular sets, using Theorem 2.1, 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 $(p\gamma, D_2') \in P_i \times \Gamma_i \times 2^{\mathcal{D}_i}$ such that 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 3.2 *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 Lemmas 3.1 and 3.2, we get:

Theorem 3.1 *Given a DPN $\mathcal{M} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a labelling function l, we can compute MAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$ in time $O(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|}))$ such that 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 $\mathcal{M} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$, by Theorem 3.1, we can construct a set of MAs $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ such that 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 such that $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$ and every DCLIC $d \in D_c$ satisfies f, i.e., there exists a set of DCLICs D_d such that $(d, D_d) \in L(\mathcal{A}_{\wp(d)})$, 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 $\mathcal{D}_{fp}$ such that 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 $D_c \subseteq \mathcal{D}_{fp}$ such that $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$.

Let $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, such that 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 $p\omega \in \bigcup_{i=1}^n \mathcal{D}_i$ such that there exists $D \subseteq \mathcal{D}_{fp}$ such that $p\omega \models_D f_{\wp(p)}$, 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 $F(X) = \{p\omega \in \mathcal{D}_I \mid \exists D \subseteq X \text{ such that } ([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$. 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 $D_0, D_1, \ldots$, 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 $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$ can be decided in time $O(|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|})$. Thus, we can get the hash table in time $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} (|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$. Given D_j and the hash table, we can compute D_{j+1} in time $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} 2^{|\mathcal{D}_{\wp(p)}|})$. Thus we can get $\mathcal{D}_{fp}$ in time $O(\sum_{p\omega \in \mathcal{D}_I} (|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_I|}) + |\mathcal{D}_I|^2 \cdot 2^{|\mathcal{D}_I|})$.

Theorem 3.2 *We can compute $\mathcal{D}_{fp}$ in time $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} (|\omega| \cdot |\delta_{\wp(p)}| \cdot |Q_{\wp(p)}| \cdot 2^{|\mathcal{D}_{\wp(p)}|} + |\bigcup_{i=1}^n \mathcal{D}_i| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$ such that for every $c \in \bigcup_{i=1}^n \mathcal{D}_i$, c satisfies the single-indexed LTL formula f iff $c \in \mathcal{D}_{fp}$.*

Proof. ($\Longrightarrow$) Suppose c satisfies f, we show that $c \in \mathcal{D}_{fp}$. Since $\mathcal{D}_{fp} = \bigcap_{j \geq 0} D_j$, where $D_0 = \bigcup_{i=1}^n \mathcal{D}_i$ and $D_{j+1} = \{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i \mid \exists D \subseteq D_j \text{ such that } ([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$ for every $j \geq 0$, it is sufficient to show that $c \in D_j$ for every $j \geq 0$. The proof proceeds by induction on j.

Algorithm 1: computing the *map* function.

```
Input  : A set of MAs $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ such that for every $i$, $1 \leq i \leq n$, $\mathcal{A}_i = (Q_i, \Gamma_i, \delta_i, I_i, Acc_i)$;
Output: A function $map : \bigcup_{i=1}^n \mathcal{D}_i \longrightarrow 2^{\bigcup_{i=1}^n \mathcal{D}_i}$ such that $D \in map(p\omega)$ iff $(p\omega, D) \in L(\mathcal{A}_{\wp(p)})$;
1 for $p\gamma_0 \dots \gamma_m \in \bigcup_{i=1}^n \mathcal{D}_i$ do
2     Let $S = Acc_{\wp(p)} \times \{\emptyset\}$;
3     for $j \leftarrow m$ to $0$ do
4         $S := \{(q, D \cup D') \mid \exists\, q \xrightarrow{\gamma_j/D}_{\wp(p)} q' \in \delta_{\wp(p)} \wedge (q', D') \in S\}$;
5     $map(p\gamma_0 \dots \gamma_m) = \{D \mid (p, D) \in S\}$;
```

- **Basis** $j = 0$: since $D_0 = \bigcup_{i=1}^n \mathcal{D}_i$ and $c \in \mathcal{D}_{\wp(c)}$, we obtain that $c \in D_0$.
- **Step** $j > 0$: Let ρ be the global run of $\mathcal{M}$ starting from c such that every local run σ of ρ satisfies $f_{\wp(\sigma)}$. Let σ_c be the local run starting from c in ρ, then σ_c satisfies $f_{\wp(c)}$. Let D_c be the set of DCLICs created during the local run σ_c, then $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$ and for every $d \in D_c$, d satisfies f. By applying the induction hypothesis, we obtain that $d \in D_{j-1}$ for every $d \in D_c$, i.e., $D_c \subseteq D_{j-1}$. Since $D_j = \{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i \mid \exists\, D \subseteq D_{j-1}$ such that $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$, we obtain that $c \in D_j$.

($\Longleftarrow$) Suppose $c \in \mathcal{D}_{fp}$, we show that c satisfies f. For this, we construct a global run ρ starting from c such that every local run σ of ρ satisfies $f_{\wp(\sigma)}$. By the definition of $\mathcal{D}_{fp}$, we can get that $\mathcal{D}_{fp} = \{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i \mid \exists\, D \subseteq \mathcal{D}_{fp}$ such that $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$.

Since $c \in \mathcal{D}_{fp}$, there exists $D_c \subseteq \mathcal{D}_{fp}$ such that $(c, D_c) \in L(\mathcal{A}_{\wp(c)})$, by Lemmas 3.1 and 3.2, $\mathcal{P}_{\wp(c)}$ has a local run $\sigma_{\wp(c)}$ starting from c such that $\sigma_{\wp(c)}$ satisfies $f_{\wp(c)}$ and D_c is the set of DCLICs created during the local run $\sigma_{\wp(c)}$. Let $\sigma_{\wp(c)}$ be the local run of ρ whenever a DCLIC c is created.

Since for every $c' \in D_c$, $c' \in \mathcal{D}_{fp}$, we can apply the same reasoning as c to c', we obtain the local run $\sigma_{c'}$ starting from c' such that the local run $\sigma_{c'}$ satisfies $f_{\wp(c')}$ and $D_{c'}$ is the set of DCLICs created during the local run $\sigma_{\wp(c')}$. Then, let $\sigma_{\wp(c')}$ be the local run of ρ whenever a DCLIC c' is created during the global run ρ. Since every DCLIC created during a local run $\sigma_{c''}$ starting from c'' such that $c'' \in \mathcal{D}_{fp}$ is also in $\mathcal{D}_{fp}$, we obtain ρ such that every local run σ of ρ satisfies $f_{\wp(\sigma)}$.

Complexity. To compute $\mathcal{D}_{fp}$, we first compute the function *map* (represented by a hash table) using Algorithm 1 such that for every $p\omega \in \bigcup_{i=1}^n \mathcal{D}_i$, $D \subseteq \mathcal{D}_{\wp(p)}$, $D \in map(p\omega)$ iff $(p\omega, D) \in L(\mathcal{A}_{\wp(p)})$. In Algorithm 1, for every $p\gamma_0 \dots \gamma_m \in \bigcup_{i=1}^n \mathcal{D}_i$, we first compute the set of DCLICs D such that $(p\gamma_0 \dots \gamma_m, D) \in L(\mathcal{A}_{\wp(p)})$ (Lines 3–4). Then, we set $map(p\gamma_0 \dots \gamma_m) = \{D \mid (p, D) \in S\}$ where S is exactly the set of pairs (q, D) such that $q \xrightarrow{\gamma_0 \cdots \gamma_m/D}{}^{*}_{\wp(p)} q'$ for some $q' \in Acc_{\wp(p)}$. When $p\omega$ is fixed, lines 3-4 need $O(|\,\omega\,| \cdot |\,\delta_{\wp(p)}\,| \cdot |\,Q_{\wp(p)}\,| \cdot 2^{|\mathcal{D}_{\wp(p)}|})$ time. Thus, we get that Algorithm 1 runs in at most $O(\sum_{p\omega \in \mathcal{D}_I} (|\,\omega\,| \cdot |\,\delta_{\wp(p)}\,| \cdot |\,Q_{\wp(p)}\,| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$ time.

Now, let us show how to compute $\mathcal{D}_{fp}$. Since $\mathcal{D}_{fp} = \bigcap_{j \geq 0} D_j$, where $D_0 = \bigcup_{i=1}^n \mathcal{D}_i$ and $D_{j+1} = \{p\omega \in D_j \mid \exists\, D \subseteq D_j,\ ([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})\}$ for every $j \geq 0$, and 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 $\mathcal{D}_{fp} = D_m$ and m is at most $|\bigcup_{i=1}^n \mathcal{D}_i\,|$. For every D_j, we can compute D_{j+1} in time $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} 2^{|\mathcal{D}_{\wp(p)}|})$, since the number of possible sets of $(p\omega, D)$ is at most $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} 2^{|\mathcal{D}_{\wp(p)}|})$. Thus, we can get $\mathcal{D}_{fp}$ in time $O(\sum_{p\omega \in \bigcup_{i=1}^n \mathcal{D}_i} (|\,\omega\,| \cdot |\,\delta_{\wp(p)}\,| \cdot |\,Q_{\wp(p)}\,| \cdot 2^{|\mathcal{D}_{\wp(p)}|} + |\bigcup_{i=1}^n \mathcal{D}_i\,| \cdot 2^{|\mathcal{D}_{\wp(p)}|}))$. □

Then, from Theorems 3.1 and 3.2, we get the following theorem.

Theorem 3.3 *Given a DPN $\mathcal{M} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$, a single-indexed LTL formula $f = \bigwedge_{i=1}^n f_i$ and a labelling function l, we can compute MAs $\mathcal{A}_1, \dots, \mathcal{A}_n$ in time $O(\sum_{i=1}^n (|\,\Delta_i\,| \cdot 2^{|f_i|} \cdot |\,\Gamma_i\,| \cdot |\,P_i\,|^3 \cdot 2^{|\mathcal{D}_i|}))$ such that 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 $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$.*

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

4. 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 $\lambda : AP \longrightarrow 2^{\bigcup_{i=1}^{n} P_i \times \Gamma_i^*}$ such that for every $a \in AP$, $\lambda(a)$ is a regular set of local configurations of $\mathcal{P}_i$ for $i,\ 1 \le i \le n$. The previous construction can be extended to deal with this case. For this, we follow the approach of [EKS03]. We compute, for $i,\ 1 \le i \le n$, 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, $1 \le i \le n$. 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.

4.1. Storing the states into the stack

We fix a DPN $\mathcal{M} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ such that for every $i,\ 1 \le i \le n, \mathcal{P}_i = (P_i, \Gamma_i, \Delta_i)$. For every $i,\ 1 \le i \le n$, we suppose w.l.o.g. that the DPDS $\mathcal{P}_i$ has a bottom-of-stack $\sharp$ that is never popped from the stack, and that for every transition rule $p\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2$ or $p\gamma \hookrightarrow p_1\omega_1$ in Δ_i, $|\ \omega_1\ | \le 2$. Indeed, every DPDS that does not satisfy this condition can be simulated by a new DPDS for which this condition holds [Sch02]. Let us fix i, $1 \le i \le n$, and let $AP_i = \{a_1, \dots, a_\iota\}$ be the set of atomic propositions used in f_i and $P_i = \{p_1, \dots, p_\kappa\}$. For every $a \in AP_i$, $p \in P_i$, let $M_a^p = (Q_a^p, \Gamma_i, \delta_a^p, s_a^p, Acc_a^p)$ be a finite automaton such that for every $\omega \in \Gamma_i^*$, $p\omega \in \lambda(a)$ (i.e., $p\omega$ satisfies a w.r.t. λ) iff the *reverse* of the word ω is accepted by M_a^p, where Q_a^p is a finite set of states, Γ_i is the input alphabet, s_a^p is the initial state, $\delta_a^p \subseteq Q_a^p \times \Gamma_i \longrightarrow Q_a^p$ is a transition function and $Acc_a^p \subseteq Q_a^p$ is a finite set of finial states. W.l.o.g., we assume that for every $a, b \in AP_i$, $p, q \in P_i$ whenever $a \neq b$ or $p \neq q$, M_a^p and M_b^q have disjoint sets of states, and we suppose that for every $(a, p) \in AP_i \times P_i$, M_a^p is deterministic and has a total transition function δ_a^p.

Since we have predicates over the stack content, to check whether or not the formula f_i holds, we need to know at each step which atomic propositions are satisfied by the stack content. To this aim, we compute a new DPDS $\mathcal{P}'_i$ which is a synchronization of the DPDS $\mathcal{P}_i$ and the finite automata M_a^p for $(a, p) \in AP_i \times P_i$ such that the stack alphabet of $\mathcal{P}'_i$ is of the form $[\gamma, \overrightarrow{S}]$, where $\overrightarrow{S} = [s_1^1, \dots, s_\iota^1, \dots, s_1^\kappa, \dots, s_\iota^\kappa]$, $s_j^k \in Q_{a_j}^{p_k}$, $1 \le j \le \iota$ and $1 \le k \le \kappa$, is a vector of states of the finite automata $M_{a_1}^{p_1}, \dots, M_{a_\iota}^{p_\kappa}$. For every $(a, p) \in AP_i \times P_i$, let $\overrightarrow{S}(a, p)$ denote the state of the finite automaton M_a^p in $\overrightarrow{S}$. A local configuration $p[\gamma_k, \overrightarrow{S_k}] \cdots [\gamma_0, \overrightarrow{S_0}]$ of $\mathcal{P}'$ is *consistent* iff for every $(a, p) \in AP_i \times P_i$, every $j,\ 1 \le j \le k-1$, $\delta_a^p(\overrightarrow{S_j}(a, p), \gamma_j) = \overrightarrow{S_{j+1}}(a, p)$ and $\overrightarrow{S_0}(a, p) = s_a^p$. Intuitively, a consistent local configuration $p[\gamma_k, \overrightarrow{S_k}] \cdots [\gamma_0, \overrightarrow{S_0}]$ denotes that the stack content is $\gamma_k \dots \gamma_0$ and for every $(a, p) \in AP_i \times P_i$, the run of the automaton M_a^p over $\gamma_0 \dots \gamma_{k-1}$ reaches the state $\overrightarrow{S_k}(a, p)$. Note that $\gamma_0 \dots \gamma_{k-1}$ is the *reverse* of the stack content $\gamma_{k-1} \dots \gamma_0$, this is why the automaton M_a^p accepts the *reverse* of the stack content. For every atomic proposition $a \in AP_i$, a consistent local configuration $p[\gamma_k, \overrightarrow{S_k}] \cdots [\gamma_0, \overrightarrow{S_0}]$ satisfies a iff there exists a state $s \in Acc_a^p$ such that $\delta_a^p(\overrightarrow{S_k}(a, p), \gamma_k) = s$. This means that whether or not a consistent local configuration satisfies an atomic proposition a depends only on the top of the stack $[\gamma_k, \overrightarrow{S_k}]$ and the control location p.

Formally, let $States_i = Q_{a_1}^{p_1} \times \cdots \times Q_{a_\iota}^{p_1} \times \cdots \times Q_{a_1}^{p_\kappa} \times \cdots \times Q_{a_\iota}^{p_\kappa}$ and $\overrightarrow{S_0} = [s_{a_1}^{p_1}, \dots, s_{a_\iota}^{p_1}, \dots, s_{a_1}^{p_\kappa}, \dots, s_{a_\iota}^{p_\kappa}]$. We compute a new DPDS $\mathcal{P}'_i = (P_i, \Gamma'_i, \Delta'_i, [\sharp, \overrightarrow{S_0}])$ as follows: $\Gamma'_i = \Gamma_i \times States_i$, $[\sharp, \overrightarrow{S_0}]$ is the bottom-of-stack of $\mathcal{P}'_i$, Δ'_i is the smallest set of transition rules satisfying the following: for every $\overrightarrow{S} \in States_i$,

ν_1: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2\epsilon \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\epsilon \in \Delta_i$;

ν_2: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2\epsilon \rhd c \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\epsilon \rhd c \in \Delta_i$;

ν_3: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2[\gamma_1, \overrightarrow{S}] \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\gamma_1 \in \Delta_i$;

ν_4: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2[\gamma_1, \overrightarrow{S}] \rhd c \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\gamma_1 \rhd c \in \Delta_i$;

ν_5: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2[\gamma_2, \overrightarrow{S'}][\gamma_1, \overrightarrow{S}] \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\gamma_2\gamma_1 \in \Delta_i$;

ν_6: $p_1[\gamma, \overrightarrow{S}] \hookrightarrow p_2[\gamma_2, \overrightarrow{S'}][\gamma_1, \overrightarrow{S}] \rhd c \in \Delta'_i$ iff $p_1\gamma \hookrightarrow p_2\gamma_2\gamma_1 \rhd c \in \Delta_i$;

where for every $(a, p) \in AP_i \times P_i$, $\delta_a^p(\overrightarrow{S}(a, p), \gamma_1) = \overrightarrow{S'}(a, p)$.

Intuitively, the run of $\mathcal{P}'_i$ reaches a consistent local configuration $p_1[\gamma_m, \overrightarrow{S_m}] \cdots [\gamma_0, \overrightarrow{S_0}]$ iff the run of $\mathcal{P}_i$ reaches the local configuration $p_1\gamma_m \ldots \gamma_0$ and for every $(a, p) \in AP_i \times P_i$, the run of the finite automaton M_a^p reaches the state $\overrightarrow{S_m}(a, p)$ over the word $\gamma_0 \ldots \gamma_{m-1}$, i.e., the reverse of the stack content $\gamma_{m-1} \ldots \gamma_0$. Moreover, the run of $\mathcal{P}'_i$ creates a DCLIC c iff the run of $\mathcal{P}'_i$ creates the DCLIC c. The intuition behind the Items $\nu_1 - \nu_6$ is explained as follows. Suppose $\mathcal{P}_i$ moves from $p_1\gamma_m\gamma_{m-1} \ldots \gamma_0$ to $p_2\gamma_{m-1} \ldots \gamma_0$ and creates a DCLIC c by the transition rule $p_1\gamma_m \hookrightarrow p_2\epsilon \rhd c$, and that for every $(a, p) \in AP_i \times P_i$, the automaton M_a^p is at state $\overrightarrow{S_{m-1}}(a, p)$ after reading the stack word $\gamma_0 \ldots \gamma_{m-2}$. Then, $\mathcal{P}'_i$ has to move from $p_1[\gamma_m, \overrightarrow{S_m}][\gamma_{m-1}, \overrightarrow{S_{m-1}}] \cdots [\gamma_0, \overrightarrow{S_0}]$ to $p_2[\gamma_{m-1}, \overrightarrow{S_{m-1}}] \cdots [\gamma_0, \overrightarrow{S_0}]$ and create the DCLIC c. This is ensured by Item ν_2. Items ν_1, ν_3 and ν_4 are similar.

If $\mathcal{P}'_i$ moves from $p_1\gamma'_m\gamma_{m-1} \ldots \gamma_0$ to $p_2\gamma_{m+1}\gamma_m\gamma_{m-1} \ldots \gamma_0$ and creates a DCLIC c by the transition rule $p_1\gamma'_m \hookrightarrow p_2\gamma_{m+1}\gamma_m \rhd c$, and that for every $(a, p) \in AP_i \times P_i$, the automaton M_a^p is at state $\overrightarrow{S_{m+1}}(a, p)$ after reading the stack word $\gamma_0 \ldots \gamma_m$ where $\delta_a^p(\overrightarrow{S_m}(a, p), \gamma_m) = \overrightarrow{S_{m+1}}(a, p)$. Then, $\mathcal{P}'_i$ moves from $p_1[\gamma'_m, \overrightarrow{S_m}][\gamma_{m-1}, \overrightarrow{S_{m-1}}] \cdots [\gamma_0, \overrightarrow{S_0}]$ to $p_2[\gamma_{m+1}, \overrightarrow{S_{m+1}}][\gamma_m, \overrightarrow{S_m}] \cdots [\gamma_0, \overrightarrow{S_0}]$ and creates the DCLIC c. This is ensured by Item ν_6. Item ν_5 is analogous.

For every $(a, p) \in AP_i \times P_i$, the fact that the finite automaton M_a^p is deterministic guarantees that the top of the stack and the control location can infer the truth of the atomic propositions. The fact that the transition function of M_a^p is total makes sure that M_a^p has always a successor state on an arbitrary input.

4.2. Reducing regular valuations to simple valuations

We can define a new valuation $\lambda' : AP \longrightarrow 2^{\bigcup_{i=1}^{n} P_i \times \Gamma_i'^*}$ as follows: for every $p\gamma \in P_i \times \Gamma_i$, $\overrightarrow{S} \in States_i$, $\lambda'(a) = \{p[\gamma, \overrightarrow{S}]\omega \mid \omega \in \Gamma_i'^*,\ \exists s \in Acc_a^p \text{ such that } \delta_a^p(\overrightarrow{S}(a, p), \gamma) = s\}$. We can show that a local run of $\mathcal{P}_i$ starting from $p\gamma_m \ldots \gamma_0$ satisfies f_i w.r.t. λ and creates a set of DCLICs D iff a local run of $\mathcal{P}'_i$ starting from a consistent local configuration $p[\gamma_m, \overrightarrow{S_m}] \cdots [\gamma_0, \overrightarrow{S_0}]$ satisfies f_i w.r.t. λ' and creates the set of DCLICs D.

Lemma 4.1 *For every i, $1 \leq i \leq n$, $p\gamma_m \ldots \gamma_0 \in P_i \times \Gamma_i^*$, $\mathcal{P}_i$ has a local run σ starting from $p\gamma_m \ldots \gamma_0$ such that D is the set of DCLICs created during this local run and σ satisfies f_i w.r.t. the regular valuation λ iff there exists a consistent local configuration $p[\gamma_m, \overrightarrow{S_m}] \cdots [\gamma_0, \overrightarrow{S_0}]$ from which $\mathcal{P}'_i$ has a local run σ' such that D is the set of DCLICs created during the local run σ' and σ' satisfies f_i w.r.t. the new valuation λ'.*

Proof. According to the construction of $\mathcal{P}'_i$, $\mathcal{P}_i$ has a local run $\sigma = c_0c_1 \ldots$ such that for every $j \geq 0$, $c_j = q_1\gamma_{m_j}^j \ldots \gamma_0^j$ and $c_j \Longrightarrow_i c_{j+1} \rhd I_j$ iff $\mathcal{P}'_i$ has a local run $\sigma' = c'_0c'_1 \ldots$ such that for every $j \geq 0$, $c'_j = q_j[\gamma_{m_j}^j, \overrightarrow{S_{m_j}^j}] \cdots [\gamma_0^j, \overrightarrow{S_0}]$ and $c'_j \Longrightarrow_i c'_{j+1} \rhd I_j$. Then, we get that $D = \bigcup_{j \geq 0} I_j$.

Now, let us show that σ satisfies f_i w.r.t. λ iff σ' satisfies f_i w.r.t. λ'. The proof proceeds by induction on the structure of f_i (note that the operators $\{\wedge, \neg, \mathbf{X}, \mathbf{U}\}$ are sufficient to express any other LTL operator).

- $\phi = a$ where a is an atomic proposition: since σ satisfies a w.r.t. λ iff $\sigma(0) = q_0\gamma_{m_0}^0 \ldots \gamma_0^0 \in \lambda(a)$, i.e., $\gamma_0^0 \ldots \gamma_{m_0}^0$ is accepted by M_a^p, and since M_a^p is deterministic, we get that M_a^p reaches the state $\overrightarrow{S_{m_0}^0}(a, p)$ after reading the word $\gamma_0^0 \ldots \gamma_{m_0-1}^0$ and the immediate successor state of $\overrightarrow{S_{m_0}^0}(a, p)$ over $\gamma_{m_0}^0$ is a final state of M_a^p. This implies that σ satisfies a w.r.t. λ iff $q_0[\gamma_{m_0}^0, \overrightarrow{S_{m_0}^0}]\omega \in \lambda'(a)$ for every $\omega \in \Gamma_i'^*$. Since σ' satisfies a w.r.t. λ' iff there exists a state $s \in Acc_a^p$ such that $\delta_a^p(\overrightarrow{S_{m_0}^0}(a, p), \gamma_{m_0}^0) = s$, we obtain that σ satisfies f_i w.r.t. λ iff σ' satisfies f_i w.r.t. λ'.
- $\phi = \phi_1 \wedge \phi_2$: σ satisfies ϕ w.r.t. λ iff σ satisfies ϕ_1 and ϕ_2 w.r.t. λ. By applying the induction hypothesis, we obtain that σ satisfies ϕ w.r.t. λ iff σ' satisfies ϕ_1 and ϕ_2 w.r.t. λ'. Thus, σ satisfies ϕ w.r.t. λ iff σ' satisfies ϕ w.r.t. λ'.
- $\phi = \mathbf{X}\phi_1$: By applying the induction hypothesis, σ_1 satisfies ϕ_1 w.r.t. λ iff σ'_1 satisfies ϕ_1 w.r.t. λ'. Since σ satisfies ϕ w.r.t. λ iff σ_1 satisfies ϕ_1 w.r.t. λ, and since σ' satisfies ϕ w.r.t. λ' iff σ'_1 satisfies ϕ_1 w.r.t. λ', we obtain that σ satisfies ϕ w.r.t. λ iff σ' satisfies ϕ w.r.t. λ'.

- $\phi = \neg\phi_1$: By applying the induction hypothesis, σ does not satisfy ϕ_1 w.r.t. λ iff σ' does not satisfy ϕ_1 w.r.t. λ'. Since σ satisfies ϕ w.r.t. λ iff σ does not satisfy ϕ_1 w.r.t. λ, and since σ' satisfies ϕ w.r.t. λ' iff σ' does not satisfy ϕ_1 w.r.t. λ', we obtain that σ satisfies ϕ w.r.t. λ iff σ' satisfies ϕ.
- $\phi = \phi_1 \mathbf{U} \phi_2$: σ satisfies ϕ w.r.t. λ iff there exists $k \geq 0$ such that for every j, $0 \leq j < k$, σ_j satisfies ϕ_1 w.r.t. λ and σ_k satisfies ϕ_2 w.r.t. λ.
  By applying the induction hypothesis, for every j, $0 \leq j < k$, σ_j satisfies ϕ_1 w.r.t. λ iff σ'_j satisfies ϕ_1 w.r.t. λ', and σ_k satisfies ϕ_2 w.r.t. λ iff σ'_k satisfies ϕ_2 w.r.t. λ'. Since σ' satisfies ϕ w.r.t. λ' iff there exists $k \geq 0$ such that for every j, $0 \leq j < k$, σ'_j satisfies ϕ_1 w.r.t. λ' and σ'_k satisfies ϕ_2 w.r.t. λ'. Thus, we obtain that σ satisfies ϕ w.r.t. λ iff σ' satisfies ϕ w.r.t. λ'.

□

4.3. Computing consistent configurations

To compute a MA $\mathcal{A}_i = (Q_i, \Gamma'_i, \delta_i, I_i, Acc_i)$ such that for every consistent local configuration $p\omega \in P_i \times \Gamma'^*_i$, $\mathcal{P}'_i$ has a local run from a consistent local configuration $p\omega$ that satisfies f_i and creates a set of DCLICs D iff $([p, g_i^0]\omega, D) \in L(\mathcal{A}_i)$, we readapt the construction of $\mathcal{BP}_i$ underlying Lemma 3.1 as follows. 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 a consistent local configuration $p\omega$ 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 $(g_1, \lambda'(p\gamma), g_2) \in \theta_i$, 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 $p\gamma \hookrightarrow p_1\omega_1 \rhd D \in \Delta_i$.

The intuition is similar to the construction underlying Lemma 3.1. The main difference is that the satisfiability of the atomic propositions depends on the control location and the top of the stack. We can get that:

Lemma 4.2 *$\mathcal{P}'_i$ has a local run from a consistent local configuration $p\omega$ 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 $O(|\Delta_i| \cdot 2^{|f_i|})$.*

However, $\mathcal{A}_i$ also accepts pairs of the form $([p, g_i^0]\omega, D)$ such that $p\omega$ is not a consistent local configuration of $\mathcal{P}'_i$. To solve this problem, we follow the approach of [EKS03] that performs a kind of synchronization of $\mathcal{A}_i$ with the automata M_a^p for $(a, p) \in AP_i \times P_i$ which will discard such pairs $([p, g_i^0]\omega, D)$. Let $\mathcal{A}'_i = ((Q_i \times States_i) \cup Q_i, \Gamma_i, \delta'_i, I_i, Acc_i \times \{\overrightarrow{S_0}\})$ be a MA such that δ'_i is defined as follows:

1. $(q_1, \overrightarrow{S_1}) \xrightarrow{\gamma/D}_i (q_2, \overrightarrow{S_2}) \in \delta'_i$ iff $q_1 \xrightarrow{[\gamma, \overrightarrow{S_2}]/D}_i q_2 \in \delta_i$ and for every $(a, p) \in AP_i \times P_i, \delta_a^p(\overrightarrow{S_2}(a, p), \gamma) = \overrightarrow{S_1}(a, p)$;
2. $q \xrightarrow{\gamma/D}_i (q', \overrightarrow{S}) \in \delta'_i$ iff $(q, \overrightarrow{S'}) \xrightarrow{\gamma/D}_i (q', \overrightarrow{S}) \in \delta'_i$ for every $q \in Q_i$.

$\mathcal{A}'_i$ has the same size than $\mathcal{A}_i$, since the immediate successor states of $\overrightarrow{S_2}(a, p)$ for $(a, p) \in AP_i \times P_i$ are uniquely determined by $\overrightarrow{S_2}$ and γ. Intuitively, $\mathcal{A}'_i$ accepts $([p_0, g_i^0]\gamma_m \ldots \gamma_0, D_1 \cup D_2)$ by a path $[p_0, g_i^0] \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m}) \xrightarrow{\gamma_{m-1}\ldots\gamma_0/D_2}{}^*_i (q_0, \overrightarrow{S_0})$ iff the configuration $p_0[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]$ is a consistent local configuration of $\mathcal{P}'$ and $([p_0, g_i^0][\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}], D_1 \cup D_2)$ is accepted by $\mathcal{A}_i$. Indeed, according to Item 1, $(q_m, \overrightarrow{S_m}) \xrightarrow{\gamma_{m-1}\ldots\gamma_0/D_2}{}^*_i (q_0, \overrightarrow{S_0})$ iff $\mathcal{A}_i$ has a path $q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_2}{}^*_i q_0$ and for every $0 \leq j \leq m-1$, every $(a, p) \in AP_i \times P_i$, $\delta_a^p(\overrightarrow{S_j}(a, p), \gamma_j) = \overrightarrow{S_{j+1}}(a, p)$, i.e., $p_0[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]$ is a consistent configuration of $\mathcal{P}'$. $[p_0, g_i^0] \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m})$ iff there exists a state $\overrightarrow{S}$ such that $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m})$ is a transition rule of $\mathcal{A}'_i$ (by Item 2). Thus, from Item 1, we get that $\mathcal{A}_i$ has the transition rule $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]/D_1}_i q_m$. Since $(q_m, \overrightarrow{S_m}) \xrightarrow{\gamma_{m-1}\ldots\gamma_0/D_2}{}^*_i (q_0, \overrightarrow{S_0})$ iff $\mathcal{A}_i$ has a path $q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_2}{}^*_i q_0$ and $p_0[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]$ is a consistent configuration of $\mathcal{P}'$,we get that $\mathcal{A}'_i$ accepts $([p_0, g_i^0]\gamma_m \ldots \gamma_0, D_1 \cup D_2)$

(i.e., $[p_0, g_i^0] \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m}) \xrightarrow{\gamma_{m-1}\cdots\gamma_0/D_2}{}^*_i (q_0, \overrightarrow{S_0})$) iff the configuration $p_0[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]$ is a consistent local configuration of $\mathcal{P}'$ and $([p_0, g_i^0][\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}], D_1 \cup D_2)$ is accepted by $\mathcal{A}_i$ (i.e., $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]/D_1}_i q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_2}{}^*_i q_0$). We can show that:

Lemma 4.3 *The configuration* $p_0[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]$ *is a consistent local configuration of* $\mathcal{P}'$ *and* $([p_0, g_i^0][\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}], D)$ *is accepted by* $\mathcal{A}_i$ *iff* $\mathcal{A}'_i$ *accepts* $([p_0, g_i^0]\gamma_m\ldots\gamma_0, D)$.

Proof. Since $[p_0, g_i^0] \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m})$ is a transition rule of $\mathcal{A}'_i$ iff there exists $\overrightarrow{S}$ such that $\mathcal{A}'_i$ has the transition rule $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_m/D_1}_i (q_m, \overrightarrow{S_m})$, it is sufficient to show that $\mathcal{A}_i$ has a path $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]/D}{}^*_i q_0$ where $\overrightarrow{S_{j+1}}(a, p) = \delta_a^p(\overrightarrow{S_j}(a, p), \gamma_j)$ for every j, $0 \leq j < m-1$, every $(a, p) \in AP_i \times P_i$ iff there exists a state $\overrightarrow{S}$ such that $\mathcal{A}'_i$ has a path $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_m\cdots\gamma_0/D}{}^*_i (q_0, \overrightarrow{S_0})$. The proof proceeds by induction on m.

- **Basis**. $m = 0$. $\mathcal{A}_i$ has a path $[p_0, g_i^0] \xrightarrow{[\gamma_0, \overrightarrow{S_0}]/D}{}^*_i q_0$ iff there exists a state $\overrightarrow{S}$ such that $\mathcal{A}'_i$ has a path $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_0/D}{}^*_i (q_0, \overrightarrow{S_0})$. This holds due to Item 1.
- **Step**. $m > 0$. $\mathcal{A}_i$ has a path $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]/D}{}^*_i q_0$ iff $\mathcal{A}_i$ has a transition rule $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]/I}_i q_m$ such that $\mathcal{A}_i$ has a path $q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_1}{}^*_i q_0$ and $D = D_1 \cup I$. This implies that there exist $[q_m, \overrightarrow{S_m}] \in Q_i \times States_i$ and $\overrightarrow{S}$ such that for every $(a, p) \in AP_i \times P_i$, $\overrightarrow{S}(a, p) = \delta_a^p(\overrightarrow{S_m}(a, p), \gamma_m)$, $\mathcal{A}'_i$ has a transition rule $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_m/I}_i [q_m, \overrightarrow{S_m}]$ and $\mathcal{A}_i$ has a path $q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_1}{}^*_i q_0$.

  By applying the induction hypothesis, $\mathcal{A}_i$ has the path $q_m \xrightarrow{[\gamma_{m-1}, \overrightarrow{S_{m-1}}]\cdots[\gamma_0, \overrightarrow{S_0}]/D_1}{}^*_i q_0$ iff $\mathcal{A}'_i$ has the path $(q_m, \overrightarrow{S_m}) \xrightarrow{\gamma_{m-1}\cdots\gamma_0/D_1}{}^*_i (q_0, \overrightarrow{S_0})$.

  Thus, $\mathcal{A}_i$ has a path $[p_0, g_i^0] \xrightarrow{[\gamma_m, \overrightarrow{S_m}]\cdots[\gamma_0, \overrightarrow{S_0}]/D}{}^*_i q_0$ iff there exists a state $\overrightarrow{S}$ such that $\mathcal{A}'_i$ has a path $([p_0, g_i^0], \overrightarrow{S}) \xrightarrow{\gamma_m\cdots\gamma_0/D}{}^*_i (q_0, \overrightarrow{S_0})$. □

From Lemmas 4.1, 4.2 and 4.3, we can get that:

Theorem 4.1 *Given a DPN* $\mathcal{M} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, *a single-indexed LTL formula* $f = \bigwedge_{i=1}^n f_i$ *and a regular valuation* λ, *we can compute MAs* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *in time* $O(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |States_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_I|}))$ *such that for every* i, $1 \leq i \leq n$, *every* $p\omega \in P_i \times \Gamma_i^*$ *and* $D \subseteq \mathcal{D}_{fp}$, $p\omega \models_D f_i$ *iff* $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$.

From Theorems 4.1 and 3.2, we can deduce the following theorem.

Theorem 4.2 *Given a DPN* $\mathcal{M} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, *a single-indexed LTL formula* $f = \bigwedge_{i=1}^n f_i$ *and a regular valuation* λ, *we can compute MAs* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *in time* $O(\sum_{i=1}^n (|\Delta_i| \cdot 2^{|f_i|} \cdot |\Gamma_i| \cdot |States_i| \cdot |P_i|^3 \cdot 2^{|\mathcal{D}_i|}))$ *such that 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* $([p, g^0_{\wp(p)}]\omega, D) \in L(\mathcal{A}_{\wp(p)})$.

5. 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.

5.1. 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**.

Definition 5.1 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 $\lambda : AP \to 2^{\bigcup_{i=1}^{n} P_i \times \Gamma_i^*}$ 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 by $c \models^{\lambda} f_i$), iff there exists $D \subseteq \mathcal{D}_i$ such that $c \models_D^{\lambda} f_i$ holds, where $\models_D^{\lambda}$ is inductively defined as follows:

- $c \models_{\emptyset}^{\lambda} a \iff c \in \lambda(a)$;
- $c \models_{\emptyset}^{\lambda} \neg a \iff c \notin \lambda(a)$;
- $c \models_D^{\lambda} \psi_1 \wedge \psi_2 \iff \exists D_1, D_2 \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ such that $D = D_1 \cup D_2$, $c \models_{D_1}^{\lambda} \psi_1$ and $c \models_{D_2}^{\lambda} \psi_2$;
- $c \models_D^{\lambda} \psi_1 \vee \psi_2 \iff c \models_D^{\lambda} \psi_1$ or $c \models_D^{\lambda} \psi_2$;
- $c \models_D^{\lambda} \mathbf{AX}\ \psi \iff$ For every $c_1, \ldots, c_m \in P_i \times \Gamma_i^*$ such that for j , $1 \le j \le m, \exists D_j, D'_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c \Longrightarrow_i c_j \rhd D'_j$, $c_j \models_{D_j}^{\lambda} \psi$ and $D = \bigcup_{j=1}^{m} (D_j \cup D'_j)$;
- $c \models_D^{\lambda} \mathbf{EX}\ \psi \iff$ There exist $c' \in P_i \times \Gamma_i^*$, and $D', D'' \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ such that $c \Longrightarrow_i c' \rhd D''$, $c' \models_{D'}^{\lambda} \psi$ and $D = D' \cup D''$;
- $c \models_D^{\lambda} \mathbf{A}[\psi_1\mathbf{U}\psi_2] \iff$ For every path $\sigma = c_0c_1\ldots$ with $c_0 = c$, for every $m \ge 1, \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and $\exists k \ge 0$, such that $\exists D_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_k \models_{D_k}^{\lambda} \psi_2, \forall j,\ 0 \le j < k$, $c_j \models_{D_j}^{\lambda} \psi_1$ and $D = \bigcup_{\sigma} (\bigcup_{j=1}^{k} D'_j \cup \bigcup_{j=0}^{k} D_j)$;
- $c \models_D^{\lambda} \mathbf{E}[\psi_1\mathbf{U}\psi_2] \iff$ There exists a path $\sigma = c_0c_1\ldots$ with $c_0 = c$, for every $m \ge 1$, $\exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and $\exists k \ge 0$, such that $\exists D_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_k \models_{D_k}^{\lambda} \psi_2, \forall j,\ 0 \le j < k$, $c_j \models_{D_j}^{\lambda} \psi_1$, and $D = \bigcup_{j=1}^{k} D'_j \cup \bigcup_{j=0}^{k} D_j$;
- $c \models_D^{\lambda} \mathbf{A}[\psi_1\mathbf{R}\psi_2] \iff$ For every path $\sigma = c_0c_1\ldots$ with $c_0 = c$, for every $m \ge 1$, $\exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and either $\forall j \ge 0, \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_j \models_{D_j}^{\lambda} \psi_2$ and $D_\sigma = \bigcup_{j \ge 1} D'_j \cup \bigcup_{j \ge 0} D_j$, or $\exists k \ge 0$, $\exists D''_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ such that $c_k \models_{D''_k}^{\lambda} \psi_1$ and $\forall j,\ 0 \le j \le k$, $\exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_j \models_{D_j}^{\lambda} \psi_2$, $D_\sigma = \bigcup_{j=0}^{k} D_j \cup D''_k \cup \bigcup_{j=1}^{k} D'_j . D = \bigcup_{\sigma} D_\sigma$;
- $c \models_D^{\lambda} \mathbf{E}[\psi_1\mathbf{R}\psi_2] \iff$ There exists a path $\sigma = c_0c_1\ldots$ with $c_0 = c$, for every $m \ge 1, \exists D'_m \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, such that $c_{m-1} \Longrightarrow c_m \rhd D'_m$, and either $\forall j \ge 0$, $\exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_j \models_{D_j}^{\lambda} \psi_2$ and $D = \bigcup_{j \ge 1} D'_j \cup \bigcup_{j \ge 0} D_j$, or $\exists k \ge 0, \exists D''_k \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$ such that $c_k \models_{D''_k}^{\lambda} \psi_1$ and $\forall j,\ 0 \le j \le k, \exists D_j \subseteq \bigcup_{i=1}^{n} \mathcal{D}_i$, $c_j \models_{D_j}^{\lambda} \psi_2$, and $D = \bigcup_{j=0}^{k} D_j \cup D''_k \cup \bigcup_{j=1}^{k} D'_j$.

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 $q\gamma \hookrightarrow p_1\omega_1 \rhd p_2\omega_2$ 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 $\bigwedge_{i=1}^{n} f_i$ [2] such that 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 $D \subseteq \mathcal{D}_{\wp(c)}$ such that $c \models_D f_{\wp(c)}$ and for every $d \in D$, d also satisfies f.

5.2. Alternating BDPDSs

Definition 5.2 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 $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd \{q_1u_1, \ldots, q_ku_k\}$ such that $p\gamma \in P'_i \times \Gamma_i$, $\{p_1\omega_1, \ldots, p_h\omega_h\} \subseteq P'_i \times \Gamma_i^*$ and $\{q_1u_1, \ldots, q_ku_k\} \subseteq \mathcal{D}_i$.

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 $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd \{q_1u_1, \ldots, q_ku_k\} \in \Delta_i$, then $p\gamma\omega \mapsto_i \{p_1\omega_1\omega, \ldots, p_h\omega_h\omega\} \rhd \{q_1u_1, \ldots, q_ku_k\}$. Intuitively, if $\mathcal{BP}'_i$ is at the configuration $p\gamma\omega$, it can fork into h copies in the configurations $p_1\omega_1\omega, \ldots, p_h\omega_h\omega$ and creates k new instances of ABDPDSs starting from the DCLICs $q_1u_1, \ldots, q_ku_k$, respectively. We sometimes write $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\}$ if $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd \emptyset \in \Delta_i$.

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, $qu \mapsto \{p_1\omega_1, \ldots, p_m\omega_m\} \rhd D$ 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 $(c, D) \in P'_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ such that $\mathcal{BP}'_i$ has an accepting run from c and that creates the set of DCLICs D.

5.3. 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 $\mathcal{A}'_1, \ldots, \mathcal{A}'_n$ such that 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 $D \subseteq \mathcal{D}'_{fp}$ such that $([p, f_{\wp(p)}]\omega, D) \in L(\mathcal{A}'_{\wp(p)})$. 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 such that 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 $\mathcal{BP}'_i$ such that 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 such that $L(M_a) = \lambda(a) \times \{\emptyset\}$, and $M_{\neg a} = (Q_{\neg a}, \Gamma_i, \delta_{\neg a}, I_{\neg a}, Acc_{\neg a})$ a MA such that $L(M_{\neg a}) = (P_i \times \Gamma_i^* \setminus \lambda(a)) \times \{\emptyset\}$, 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 such that 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 $[p, \psi]\gamma \hookrightarrow \{[p, \psi_2]\gamma\} \in \Delta'_i$;

[2] Formulas of the form $\bigvee_{i=1}^{n} f_i$ can be verified by checking f_i for $1 \leq i \leq n$.

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 $\psi = \neg a$ such that $a \in At(f_i)$ 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 $p\gamma u \in P'_i \times \Gamma_i^*$ such that $\gamma \in \Gamma_i$, $p\gamma u$ satisfies ψ iff there exists a transition $t_1 = p\gamma \hookrightarrow p'\omega \in \Delta_i$ or $t_2 = p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$ 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 $p\gamma u \in P'_i \times \Gamma_i^*$ such that $\gamma \in \Gamma_i$, $p\gamma u$ 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 $t_2 = p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$ 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 $p\gamma u \in P'_i \times \Gamma_i^*$ such that $\gamma \in \Gamma_i$, $p\gamma u$ 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 $t_2 = p\gamma \hookrightarrow p'\omega \rhd p''\omega' \in \Delta_i$ 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 5.1 *For every i, $1 \leq i \leq n$, we can compute an ABDPDS $\mathcal{BP}'_i$ with $O(\mid P_i \mid \cdot \mid f_i \mid + \sum_{a \in At(f_i)}(\mid Q_a \mid + \mid Q_{\neg a} \mid))$ states and $O\big((\mid P_i \mid \cdot \mid \Gamma_i \mid + \mid \Delta_i \mid) \mid f_i \mid + \sum_{a \in At(f_i)}(\mid \delta_a \mid + \mid \delta_{\neg a} \mid)\big)$ 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 $([p, f_i]\omega, D) \in L(\mathcal{BP}'_i)$.*

5.4. 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 [ST11]. 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: $pre_{\mathcal{BP}'_i}(U) = \{(c, D) \mid c \mapsto_i \{c_1, \ldots, c_m\} \rhd D_0,\ \forall j : 1 \leq j \leq m, (c_j, D_j) \in U, \text{ and } D = \bigcup_{j=0}^{m} D_j\}$. 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 $c_1, \ldots, c_m$ such that $c \mapsto_i \{c_1, \ldots, c_m\} \rhd D_0,\ \forall j : 1 \leq j \leq m, (c_j, D_j) \in pre^*_{\mathcal{BP}'_i}(U)$, 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 such that each path of this 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 such that 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 such that 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 5.1 $L(\mathcal{BP}'_i) = Y_{\mathcal{BP}'_i}$.

Proof. ($\Longrightarrow$) First we show that if $(c, D) \in L(\mathcal{BP}'_i)$, then $(c, D) \in Y_{\mathcal{BP}'_i}$. Since $Y_{\mathcal{BP}'_i} = \bigcap_{k \geq 1} Y_k$, it is sufficient to prove that $(c, D) \in Y_k$ for every $k \geq 1$. The proof proceeds by induction on k. Let $\mapsto_i^+$ be the transitive closure of $\mapsto_i$.

- **Basis**. $k = 1$. We show that $(c, D) \in Y_1$. Since $(c, D) \in L(\mathcal{BP}'_i)$, $\mathcal{BP}'_i$ has a run from c such that each path of this run infinitely often visits some control locations in F_i and D is the set of DCLICs created during in this run. Since the number of such $D \subseteq \mathcal{D}_i$ is finite, we can find a finite set of local configurations $p_1\omega_1, \ldots, p_m\omega_m$ such that the control locations $p_1, \ldots, p_m$ are in F_i and $c \mapsto_i^+ \{p_1\omega_1, \ldots, p_m\omega_m\} \rhd D$.
  Since $p_j\omega_j \in F_i \times \Gamma_i^*$ for every j, $1 \leq j \leq m$ and $Y_0 = P_i \times \Gamma_i^* \times \{\emptyset\}$, we can obtain that $(p_j\omega_j, \emptyset) \in Y_0 \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$, for every j, $1 \leq j \leq m$.
  Since $Y_1 = pre^+_{\mathcal{BP}'_i}(Y_0 \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$, we obtain that $(c, D) \in Y_1$.
- **Step**. $k > 1$. We show that $(c, D) \in Y_k$. Since $(c, D) \in L(\mathcal{BP}'_i)$, $\mathcal{BP}'_i$ has a run from c such that each path of this run infinitely often visits some control locations in F_i and D is the set of DCLICs created during in this run, we can find a finite set of local configurations $p_1\omega_1, \ldots, p_m\omega_m$ and sets of DCLICs $D', D_1, \ldots, D_m$ such that the control locations $p_1, \ldots, p_m$ are in F_i, $D = D' \cup \bigcup_{j=1}^{m} D_j$, $p\omega \Longrightarrow_i^+ \{p_1\omega_1, \ldots, p_m\omega_m\} \rhd D'$, and $(p_j\omega_j, D_j) \in L(\mathcal{BP}'_i)$ for every j, $1 \leq j \leq m$.
  By applying the induction hypothesis (induction on k) to $(p_j\omega_j, D_j) \in L(\mathcal{BP}'_i)$ for every j, $1 \leq j \leq m$, we obtain that $(p_j\omega_j, D_j) \in Y_{k-1}$ for every j, $1 \leq j \leq m$.
  Since $Y_k = pre^+_{\mathcal{BP}'_i}(Y_{k-1} \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$, we obtain that $(c, D) \in Y_k$.

($\Longleftarrow$) Let us show that if $(c, D) \in Y_{\mathcal{BP}'_i}$, then $(c, D) \in L(\mathcal{BP}'_i)$. It is sufficient to construct an accepting run from c such that D is the set of DCLICs created during this run.

Since $Y_{\mathcal{BP}'_i} = \bigcap_{j \geq 1} Y_j$ 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 $j \geq 0$, we obtain that $Y_{\mathcal{BP}'_i} = pre^+_{\mathcal{BP}'_i}(Y_{\mathcal{BP}'_i} \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$.

Since $(c, D) \in Y_{\mathcal{BP}'_i} = pre^+_{\mathcal{BP}'_i}(Y_{\mathcal{BP}'_i} \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i})$, there exists a set of tuples $\{(p_1\omega_1, D_1), \ldots, (p_m\omega_m, D_m)\} \subseteq Y_{\mathcal{BP}'_i} \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$ such that $c \Longrightarrow_i^+ \{p_1\omega_1, \ldots, p_m\omega_m\} \rhd I_0$ and $D = I_0 \cup \bigcup_{k=1}^{m} D_k$.

Algorithm 2: computation of $Y_{\mathcal{BP}'_i}$.

```
Input  : An ABDPDS $\mathcal{BP}'_i = (P'_i, \Gamma_i, \Delta'_i, F_i)$;
Output: An AMA $\mathcal{A}'_i = (Q_i, \Gamma_i, \delta_i, I_i, \{q_f\})$ such that $L(\mathcal{A}'_i) = Y_{\mathcal{BP}'_i}$;
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      $k := k + 1$;
4      Add a new transition rule $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$ in $\delta_i$ for every $p \in F_i$;
5      repeat we call this loop $loop_2$
6          For every $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd D$ in $\Delta'_i$,
7          and every case $p_j^k \xrightarrow{\omega_j/D_j}{}^*_i R_j$ for all $j,\ 1 \le j \le h$;
8              $p^k \xrightarrow{\gamma/D \cup \bigcup_{j=1}^h D_j}_i \bigcup_{j=1}^h R_j$ in $\delta_i$
9      until No new transition rule can be added;
10     Remove from $\delta_i$ the transition rules $p^k \xrightarrow{\epsilon/\emptyset}_i \{p^{k-1}\}$, $\forall\, p \in F_i$;
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 $\forall\, p \in P'_i, \gamma \in \Gamma_i, R \subseteq P'_i \times \{k\} \cup \{q_f\}, D \subseteq \mathcal{D}_i$, $p^k \xrightarrow{\gamma/D}_i R \in \delta_i$ iff $p^{k-1} \xrightarrow{\gamma/D}_i \pi^{-1}(R) \in \delta_i$;
```

Since $\{(p_1\omega_1, D_1), \ldots, (p_m\omega_m, D_m)\} \subseteq Y_{\mathcal{BP}'_i} \cap F_i \times \Gamma_i^* \times 2^{\mathcal{D}_i}$, we obtain that $(p_k\omega_k, D_k) \in Y_{\mathcal{BP}'_i}$ and $p_k \in F_i$ for every k, $1 \le k \le m$. Let us construct a finite tree (run) ρ with root $p\omega$, the leaves of ρ are $p_1\omega_1, \ldots, p_m\omega_m$, the inner nodes of ρ are the successors during the run $p\omega \Longrightarrow_i^+ \{p_1\omega_1, \ldots, p_m\omega_m\} \rhd I_0$. Each path of ρ can visit some control locations in F_i at least once and I_0 is the set of DCLICs created during the run ρ.

Since $p_k\omega_k \in Y_{\mathcal{BP}'_i}$ for every k, $1 \le k \le m$, we can repeatedly construct a finite tree ρ_k for the local configuration $p_k\omega_k$ such that ρ_k has the same properties as ρ. Let us replace each leaf $p_k\omega_k$ in ρ by the tree ρ_k and obtain a new tree ρ such that each path of the new tree ρ can visit some control locations in F_i at least twice.

Now we infinitely repeat this procedure to the leaves of the latest tree ρ. Finally, we can obtain an infinite run such that each path of this run visits some control locations in F_i and D is the set of DCLICs created during this run. □

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_js. 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 $S \subseteq Q_i$,

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

$$\pi^k(S) = \{q^k \mid \exists\, j,\ 1 \le j \le k \text{ such that } q^j \in S\} \cup \{q_f \mid q_f \in S\}.$$

Algorithm 2 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 5.1 *Algorithm 1 always terminates and produces $Y_{\mathcal{BP}'_i}$.*

Proof. The proof follows the proof of [ST11]. Algorithm 1 follows the idea of the algorithm of [ST11]. computing an AMA recognizing the language of an ABDPDS when transition rules are in the form of $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\}$, 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 [BEM97] that computes reachable configurations of *Alternating* PDSs:

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

We use the following saturation procedure:

If $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd D \in \Delta'_i$ and $p_j^k \xrightarrow{\omega_j/D_j}{}^*_i R_j$ for j, $1 \le j \le h$, 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 $p\gamma \hookrightarrow \{p_1\omega_1, \ldots, p_h\omega_h\} \rhd D \in \Delta'_i$ and for every j, $1 \le j \le h$, $(p_j\omega_j\omega', D_j)$ 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., $(p\gamma\omega', D \cup \bigcup_{j=1}^h D_j) \in pre^*_{\mathcal{BP}'_i}(\{(p_1\omega_1\omega', D_1), \ldots, (p_j\omega_j\omega', D_j)\})$. □

Complexity. Following [ST11], we can show that $loop_2$ can be done in time $O(\mid P'_i \mid \cdot \mid \Delta'_i \mid \cdot 2^{4|P'_i|+|\mathcal{D}_i|})$. The substitution (Line 11) and termination condition (Line 12) can be done in time $O(\mid \Gamma_i \mid \cdot \mid P'_i \mid \cdot 2^{2|P'_i|+|\mathcal{D}_i|})$ and $O(\mid \Gamma_i \mid \cdot \mid P'_i \mid \cdot 2^{|P'_i|+|\mathcal{D}_i|})$, respectively. Putting all these estimations together, the global complexity of Algorithm 2 is $O(\mid P'_i \mid^2 \cdot \mid \Delta'_i \mid \cdot \mid \Gamma_i \mid \cdot 2^{5|P'_i|+|\mathcal{D}_i|})$.

By Proposition 5.1 and Theorem 5.1, we get:

Lemma 5.2 *Given an ABDPDS $\mathcal{BP}'_i$, we can construct an AMA $\mathcal{A}'_i$ with $O(\mid \Gamma_i \mid \cdot \mid P'_i \mid \cdot 2^{|P'_i|+|\mathcal{D}_i|})$ transitions and $O(\mid P'_i \mid)$ states in time $O(\mid P'_i \mid^2 \cdot \mid \Delta'_i \mid \cdot \mid \Gamma_i \mid \cdot 2^{5|P'_i|+|\mathcal{D}_i|})$ such that $L(\mathcal{BP}'_i) = L(\mathcal{A}'_i)$.*

From Lemmas 5.1 and 5.2, we get:

Lemma 5.3 *We can compute AMAs $\mathcal{A}'_1, \ldots, \mathcal{A}'_n$ in time $O(\sum_{i=1}^n((\mid P_i \mid \cdot \mid f_i \mid +k)^2 \cdot ((\mid P_i \mid \cdot \mid \Gamma_i \mid + \mid \Delta_i \mid) \mid f_i \mid +d) \cdot \mid \Gamma_i \mid \cdot 2^{5(|P_i|\cdot|f_i|+k)+|\mathcal{D}_i|}))$ such that for every i, $1 \le i \le n$, $p\omega \in P_i \times \Gamma_i^*$, $p\omega \models_D f_i$ iff $([p, f_i], D) \in L(\mathcal{A}'_i)$, where $k = \sum_{a\in At(f_i)}(\mid Q_a \mid + \mid Q_{\neg a} \mid)$ and $d = \sum_{a\in At(f_i)}(\mid \delta_a \mid + \mid \delta_{\neg a} \mid)$.*

5.5. CTL Model checking for DPNs with regular valuations

By Lemma 5.3, we obtain a set of AMAs $\{\mathcal{A}'_1, \ldots, \mathcal{A}'_n\}$ such that for every i, $1 \le i \le n$ 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 such that 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 $D \subseteq \mathcal{D}'_{fp}$ such that $([p, f_{\wp(p)}]\omega, D) \in L(\mathcal{A}'_{\wp(p)})$. $\mathcal{D}'_{fp}$ can be computed as done in Sect. 3.2. We can show that:

Theorem 5.2 *We can compute AMAs $\mathcal{A}'_1, \ldots, \mathcal{A}'_n$ in time $O(\sum_{i=1}^n((|P_i| \cdot \mid f_i \mid +k)^2 \cdot ((|P_i| \cdot |\Gamma_i| + \mid \Delta_i \mid) \mid f_i \mid +d) \cdot \mid \Gamma_i \mid \cdot 2^{5(|P_i|\cdot|f_i|+k)+|\mathcal{D}_i|}))$ such that 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 $k = \sum_{a\in At(f_i)}(\mid Q_a \mid + |Q_{\neg a}|)$ and $d = \sum_{a\in At(f_i)}(\mid \delta_a \mid + \mid \delta_{\neg a} \mid)$.*

6. Conclusion and future work

In this work, we show that the model checking problems for DPNs against single-indexed LTL and CTL with simple and regular valuations are decidable. We present automata-based algorithms for these problems which construct a set of MAs (resp. AMAs) to finitely represent all the global configurations of a DPN that satisfy a given LTL (resp. CTL) formula.

In DPNs, instances of DPDSs do not allow to communicate with each other, therefore DPNs cannot model multi-threaded programs that have synchronization such as message passing, global variables or lock/unlock mechanism. Omitting the synchronization in multi-threaded programs, the DPN models will not precise. For lock/unlock synchronized programs, the resulting models could be an over-approximation of the programs. Extending of DPNs with synchronization is non-trivial. The pairwise reachability problem of DPNs extended with a simple synchronization mechanism, lock/unlock, will be undecidable even for the DPNs without dynamic thread creation [KG06]. It is show that the problem will be decidable if DPNs are extended with well-nested lock/unlock is decidable [LMOW09]. We plan to investigate the single-indexed LTL/CTL model-checking problem for DPNs with well-nested lock/unlock, so that the DPNs can model programs that synchronized via well-nested lock/unlock mechanism.

Acknowledgements

We gratefully acknowledge the editor and anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was partially supported by the Pujiang Talent Project of the Shanghai Science and Technology Committee (No. 14PJ1403200), NSFC Project (No. 61402179, No. 91418203), the ChenGuang Project of the Shanghai Municipal Education Commission and the Shanghai Education Development Foundation (No. 13CG21), the Open Project of Shanghai Key Laboratory of Trustworthy Computing (No. 07dz22304201301), ANR Grant (No. ANR-08-SEGI-006).

References

[ABT08] Atig MF, Bouajjani A, Touili T (2008) On the reachability analysis of acyclic networks of pushdown systems. In *CONCUR*, pp 356–371

[BEM97] Bouajjani A, Esparza J, Maler O (1997) Reachability analysis of pushdown automata: application to model checking. In CONCUR'97. LNCS 1243

[BET03] Bouajjani A, Esparza J, Touili T. (2003) A generic approach to the static analysis of concurrent programs with procedures. In *POPL*, pp 62–73

[BKRS09] Bozzelli L, Kretínský M, Rehák V, Strejcek J (2009) On decidability of LTL model checking for process rewrite systems. Acta Inf, 46(1)

[BMOT05] Bouajjani A, Müller-Olm M, Touili T (2005) Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR*, pp 473–487

[CCK+06] Chaki S, Clarke EM, Kidd N, Reps TW, Touili T (2006) Verifying concurrent message-passing c programs with recursive calls. In *TACAS*, pp 334–349

[EHRS00] Esparza J, Hansel D, Rossmanith P, Schwoon S (2000) Efficient algorithm for model checking pushdown systems. In *CAV'00*, volume 1885 of *LNCS*

[EKS03] Esparza J, Kucera A, Schwoon S (2003) Model checking LTL with regular valuations for pushdown systems. *Inf Comput*, 186(2):355–376

[GL11] Göller S, Lin AW (2011) The complexity of verifying ground tree rewrite systems. In *LICS*, pp 279–288

[GLMO+11] Gawlitza TM, Lammich P, Müller-Olm M, Seidl H, Wenner A (2011) Join-lock-sensitive forward reachability analysis for concurrent programs with dynamic process creation. In *VMCAI*, pp 199–213

[KG06] Kahlon V, Gupta A (2006) An automata-theoretic approach for model checking threads for LTL properties. In *LICS*, pp 101–110

[KG07] Kahlon V, Gupta A (2007) On the analysis of interacting pushdown systems. In *POPL*, pp 303–314

[KIG05] Kahlon V, Ivancic F, Gupta A (2005) Reasoning about threads communicating via locks. In *CAV*, pp 505–518

[KLTR09] Kidd N, Lammich P, Touili T, Reps TW (2009) A decision procedure for detecting atomicity violations for communicating processes with locks. In *SPIN*, pp 125–142

[LMO07] Lammich P, Müller-Olm M (2007) Precise fixpoint-based analysis of programs with thread-creation and procedures. In *CONCUR*, pp 287–302

[LMO08] Lammich P, Müller-Olm M (2008) Conflict analysis of programs with procedures, dynamic thread creation, and monitors. In *SAS*, pp 205–220

[LMOW09] Lammich P, Müller-Olm M, Wenner A (2009) Predecessor sets of dynamic pushdown networks with tree-regular constraints. In *CAV*, pp 525–539

[Lug11] Lugiez D (2011) Forward analysis of dynamic network of pushdown systems is easier without order. Int J Found Comput Sci, 22(4):843–862

[May00] Mayr R (2000) Process rewrite systems. Inf Comput, 156(1–2):264–286

[Sch02] Schwoon S (2002) Model-Checking Pushdown Systems. PhD thesis, Technische Universität München

[ST11] Song F, Touili T (2011). Efficient CTL model-checking for pushdown systems. In *CONCUR*

[ST13] Song F, Touili T (2013) Model checking dynamic pushdown networks. In *APLAS*, pp 33–49

[TA10] Touili T, Atig MF (2010) Verifying parallel programs with dynamic communication structures. Theor Comput Sci, 411(38–39):3460–3468

[VW86] Vardi MY, Wolper P (1986) Automata-theoretic techniques for modal logics of programs. J Comput Syst Sci, 32(2):183–221
[Wen10] Wenner A (2010) Weighted dynamic pushdown networks. In *ESOP*, pp 590–609
[Yah01] Yahav E (2001) Verifying safety properties of concurrent java programs using 3-valued logic. In *POPL*, pp 27–40

Received 3 June 2014
Revised 31 August 2014
Accepted 26 November 2014 by Jin Song Dong
Published online 8 January 2015