Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp 37-48

Analyzing Asynchronous Programs with Preemption

Mohamed Faouzi Atig, Ahmed Bouajjani, Tayssir Touili
LIAFA, CNRS and University Paris Diderot
France
{atig,abou,touili}@liafa.jussieu.fr

ABSTRACT. Multiset pushdown systems have been introduced by Sen and Viswanathan as an adequate model for asynchronous programs where some procedure calls can be stored as tasks to be processed later. The model is a pushdown system supplied with a multiset of pending tasks. Tasks may be added to the multiset at each transition, whereas a task is taken from the multiset only when the stack is empty. In this paper, we consider an extension of these models where tasks may be of different priority level, and can be preempted at any point of their execution by tasks of higher priority. We investigate the control point reachability problem for these models. Our main result is that this problem is decidable by reduction to the reachability problem for a decidable class of Petri nets with inhibitor arcs. We also identify two subclasses of these models for which the control point reachability problem is reducible respectively to the reachability problem and to the coverability problem for Petri nets (without inhibitor arcs).

1 Introduction

In the last few year, a lot of effort has been devoted to the verification problem for models of concurrent programs (see, e.g., [17, 7, 15, 19, 4, 3, 2, 23, 13, 1]). Multiset Pushdown Systems (MPDS) have been introduced in [22] as an adequate model for asynchronous programs. These programs constitute an important class of program widely used in the management of concurrent interactions with the environment, e.g., in building networked software systems, distributed systems, etc. In these programs, procedure calls can be either synchronous, which means that the caller waits as usual until the callee returns, or asynchronous, which means that the callee is rather stored as a task to be processed later. Repetitively, pending tasks are chosen and executed until completion, which may generate other pending tasks.

The MPDS model consists of a pushdown system supplied with a multiset store containing pending tasks. When (and only when) the stack is empty, a task is taken from the multiset and put into the stack. Then, the system starts executing the task using pushdown transition rules which, in addition to usual push and pop operations (modeling synchronous procedure calls) can generate new tasks (modeling asynchronous procedure calls). Notice that in this model, both the stack and the multiset store are of unbounded sizes. The control point reachability problem has been proved to be decidable in [22], and an efficient procedure for deciding this problem has been developed in [12].

In this paper, we consider a wider class of programs where tasks may have different priority levels (assuming that there is a finite number of such levels), and that at any point in time tasks are executed according to their priority level ordering. This means that tasks can be preempted by tasks of higher priority level: When a task γ of level i generates a

FSTTCS 2008
IARCS Annual Conference on
Foundations of Software Technology and Theoretical Computer Science
http://drops.dagstuhl.de/opus/volltexte/2008/1739

task γ' of level $j > i$, the task γ is preempted and must wait until the task γ' as well as all its descendants (i.e., tasks it created) of level greater than i are done. We consider that in general the task γ' may also have descendant of level less or equal to i; these tasks are stored among the other pending tasks of their level for later execution.

To reason about this class of programs, we introduce the model of k-MPDS corresponding to MPDS with $k+1$ priority levels and preemption (i.e., 0-MPDS coincides with the model of [22]). We address the control point/configuration reachability problem in these models. Our main result is that both of these problems are decidable. The proof is not trivial. The main difficulty to face is that a preempted task can be resumed only when there are no pending tasks of higher level. This involves a kind of test to 0 of some counters (which count the number of pending tasks at each priority level). We show that in fact these reachability problems can be reduced to the reachability problem in a class of Petri nets with inhibitor arcs shown to be decidable by Reinhardt in [21].

Then, we consider two classes of k-MPDS obtained by introducing some restrictions either on the way priority levels are assigned to newly created tasks, or on the allowed kind of communication through return values between asynchronous calls. The first subclass of models we consider, called hierarchical MPDS, corresponds to systems where each created task is assigned a priority level which is at least as high as the one of its caller. We show that this inheritance-based policy of priority assignment leads to a model for which both the control point and the configuration reachability problem can be reduced to the reachability problem in Petri nets without inhibitor arcs.

The second subclass we consider, called restricted MPDS, corresponds to systems where return values are taken into account (1) for synchronous calls at any level, (2) for asynchronous calls at level 0, and (3) between tasks of different levels when a preemption or a resumption occurs. This means that returns values by asynchronous calls within levels greater than 0 are not taken into account (i.e., they are abstracted away), but these calls may have an influential side effect by creating new tasks at any level, and this is taken into account in our model. We prove that for the corresponding models to this class of programs the control point and the configuration reachability problems are reducible to the corresponding problems in Petri nets using Parikh image computations of context-free languages. This means in particular that the control point problem (which the relevant problem for proving safety properties) for these models can be reduced to the coverability problem in Petri nets. As far as we know, our results are not covered by any existing result in the literature.

2 Preliminaries

Words and Languages. Let Σ be a finite alphabet. We denote by Σ^* (resp. Σ^+) the set of all *words* (resp. non empty words) over Σ, and by ϵ the empty word. A language is a (possibly infinite) set of words. Given two disjoint finite alphabets Σ and Σ' and a language L over $\Sigma \cup \Sigma'$, the projection of L on Σ, denoted L_Σ, is the set of words $a_1 \dots a_n \in \Sigma^*$ such that $\left(\Sigma'^* a_1 \Sigma'^* a_2 \Sigma'^* \cdots \Sigma'^* a_n \Sigma'^*\right) \cap L \neq \emptyset$.

Finite State Automata. A Finite State Automaton (FSA) is a tuple $\mathcal{S} = (S, \Sigma, \delta, s^i, s^f)$ where S is a finite set of states, Σ is a finite alphabet, $\delta \subseteq S \times (\Sigma \cup \{\epsilon\}) \times S$ is a set of rules, $s^i \in S$ is an initial state, and $s^f \in S$ is a final state. Let $L(\mathcal{S})$ denotes the language accepted by $\mathcal{S}$.

Multi-sets. Let Σ be a finite alphabet. A Multi-set over Σ is a function $M : \Sigma \to \mathbb{N}$. We denote by $M[\Sigma]$ the collection of all multi-sets over Σ and by $\emptyset$ the empty multi-set. Given two multi-sets M and M', we write $M' \leq M$ iff $M'(a) \leq M(a)$ for every $a \in \Sigma$; and $M + M'$ (resp. $M - M'$ if $M' \leq M$) to denote the multi-set where $(M + M')(a) = M(a) + M'(a)$ (resp. $(M - M')(a) = M(a) - M'(a)$) for every $a \in \Sigma$. For a word $w \in \Sigma^*$, $\lfloor w \rfloor$ is the multi-set formed by counting the number of symbols occurring in w; and for a language $L \subseteq \Sigma^*$, $\lfloor L \rfloor = \{\lfloor w \rfloor : w \in L\}$. A set $\mathcal{M} \subseteq M[\Sigma]$ is semi-linear iff there is a FSA $\mathcal{S}$ s.t. $\mathcal{M} = \lfloor L(\mathcal{S}) \rfloor$.

Context-Free Grammars. A Context-Free Grammar (CFG) is a tuple $G = (V, \Sigma, R, S)$ where V is a set of non terminal symbols, Σ is an input alphabet, $S \in V$ is the start symbol (called also axiom), and $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of production rules. Given two words $u, v \in (V \cup \Sigma)^*$, we write $u \vdash_G v$ iff $\exists (\alpha, \beta) \in R$ such that $u = u_1 \alpha u_2$ and $v = u_1 \beta u_2$ for some $u_1, u_2 \in (V \cup \Sigma)^*$. We denote by $\vdash_G^*$ the transitive and reflexive closure of $\vdash_G$ and by $L(G) = \{w \in \Sigma^* \mid S \vdash_G^* w\}$ the context free language generated by G.

Labeled Pushdown Systems. A Labeled Pushdown System (LPDS) is a tuple $\mathcal{P} = (Q, \Sigma, \Gamma, \delta)$ where Q is a finite set of states, Σ is an input alphabet, Γ is a stack alphabet, and δ is a finite set of transition rules of the form: $q\gamma \overset{a}{\hookrightarrow} q'w'$ where $q, q' \in Q$, $\gamma \in \Gamma$, $a \in \Sigma \cup \{\epsilon\}$, and $w \in \Gamma^*$. A *configuration* of $\mathcal{P}$ is a tuple (q, σ, w) where $q \in Q$ is a state, $\sigma \in \Sigma^*$ is an input word, and $w \in \Gamma^*$ is a stack content. We define the binary relation $\Rightarrow_{\mathcal{P}}$ between configurations as follows: $(q, a\sigma, w\gamma) \Rightarrow_{\mathcal{P}} (q', \sigma, ww')$ iff $q\gamma \overset{a}{\hookrightarrow} q'w' \in \Delta$. The *transition relation* $\Rightarrow_{\mathcal{P}}^*$ is the reflexive transitive closure of $\Rightarrow_{\mathcal{P}}$.

Given a LPDS $\mathcal{P}$, a pair of states $q_1, q_2 \in Q$, and a stack symbol $\gamma \in \Gamma$, we define $L_{\mathcal{P}}(q_1, q_2, \gamma)$ as the set of words $\{\sigma \in \Sigma^* \mid (q_1, \sigma, \gamma) \Rightarrow_{\mathcal{P}}^* (q_2, \epsilon, \epsilon)\}$. It is well-known that $L_{\mathcal{P}}(q_1, q_2, \gamma)$ is a context-free language, and conversely, every context-free language can be defined as a trace language of some LPDS.

Finally, we recall a result due to Parikh [18] which will be used later in the paper.

PROPOSITION 1. *Given a LPDS $\mathcal{P} = (Q, \Sigma, \Gamma, \delta)$, two states $q_1, q_2 \in Q$, and a stack symbol $\gamma \in \Gamma$, it is possible to construct a FSA $\mathcal{S}$ such that $\lfloor L_{\mathcal{P}}(q_1, q_2, \gamma) \rfloor = \lfloor L(\mathcal{S}) \rfloor$.*

Petri Nets with Inhibitor arcs. A Petri net with inhibitor arcs is a pair $\mathcal{N} = (P, T)$ where P is a finite set of places, and $T \subseteq 2^P \times P^* \times P^*$ is a finite set of transitions. Given a transition $t = (I, w, w')$, we define the relation $\xrightarrow{t} \subseteq M[P] \times M[P]$ as follows: $W \xrightarrow{t} W'$ iff $W \geq \lfloor w \rfloor$, $W' = W + \lfloor w' \rfloor - \lfloor w \rfloor$ and $W(p) = 0$ for every $p \in I$. We define the transition relation $\rightarrow_{\mathcal{N}}$ on multi-sets over P by the union of the $\xrightarrow{t}$, i.e., $\rightarrow_{\mathcal{N}} = \bigcup_{t \in T} \xrightarrow{t}$. The transition relation $\rightarrow_{\mathcal{N}}^*$ is the reflexive transitive closure of $\rightarrow_{\mathcal{N}}$.

A Petri net with *weak* inhibitor arcs is a Petri-net with inhibitor arcs (P, T) such that there is a function $f : P \to \mathbb{N} \setminus \{0\}$ such that $\forall p, p' \in P$, $f(p) \leq f(p') \Rightarrow (\forall (I, w, w') \in T, p' \in I \Rightarrow p \in I)$. A Petri net can be seen as a subclass of Petri nets with inhibitor arcs where all the transitions $(I, w, w') \in T$ are such that $I = \emptyset$. In this case, the transitions T can be described in $P^* \times P^*$.

The reachability (resp. coverability) problem for a Petri net with inhibitor arcs N is the problem of deciding for two given multi-sets W' and W whether $W \rightarrow_{\mathcal{N}}^* W'$ (resp. there is a multi-set $W'' \geq W'$ such that $W \rightarrow_{\mathcal{N}}^* W''$). Reachability and coverability problems for Petri

nets with inhibitor arcs are undecidable [10]. Fortunately, they become decidable for Petri nets Petri nets with weak inhibitor arcs.

THEOREM 2. *Reachability and coverabilty problems for Petri nets with weak inhibitor arcs are decidable* [21]. *Moreover, the reachability (resp. coverability) problem for Petri nets is decidable and EXSPACE-hard (resp. EXSPACE-complete)*[14, 20, 16, 5].

3 Multi-set Pushdown Systems with Preemption

3.1 Definition of the Model

We introduce multiset pushdown systems with preemptive task generation according to a finite number of priority classes. The model of MPDS defined in [22] corresponds to the particular case where all tasks have the same priority (and therefore preemption never occurs).

DEFINITION 3. *Let k be a natural number. A k-Multi-set Pushdown System with Preemption (k-MPDS) is a tuple $\mathcal{A} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \gamma_0)$ where Q is a finite set of states, $\Gamma = \bigcup_{0 \le j \le k} \Gamma_j$ is a finite set of multi-set symbols, $q_0 \in Q$ is the initial state, $\gamma_0 \in \Gamma_0$ is the initial task, and the sets $\Delta \subseteq \bigcup_{j=0}^{k} (Q \times \Gamma_j) \times (Q \times \Gamma_j^* \times (\Gamma \cup \{\epsilon\}))$ and $\Delta' \subseteq Q \times Q \times \Gamma$ form together the transition rules.*

For presentation matter, transitions in Δ (resp. Δ') will be represented respectively by $q\gamma \hookrightarrow q'w' \rhd \gamma'$ (resp. $q \hookrightarrow q' \lhd \gamma'$) with $q, q' \in Q$, $\gamma \in \Gamma_j$, $w' \in \Gamma_j^*$, and $\gamma' \in \Gamma$ for some $j \in \{0, \ldots, k\}$. Intuitively, rules of the form $q\gamma \hookrightarrow q'w' \rhd \gamma'$ correspond, in addition to the usual pushdown operations (popping γ and then pushing w' while changing the control state from q to q'), to generate the task γ'. Rules of the form $q \hookrightarrow q' \lhd \gamma$ correspond to move the control state from q to q' and to start executing the pending task γ if the priority level of the topmost symbol in the stack is strictly less than the priority level of γ.

A configuration of $\mathcal{A}$ is a tuple $(q, w, M_0, \ldots, M_k)$ where $q \in Q$, $w \in \Gamma_0^* \times \cdots \times \Gamma_k^*$, and $M_j \in M[\Gamma_j]$, $0 \le j \le k$, is a multiset representing the waiting tasks of priority j. The content of the stack w is always of the form $w_0 w_1 \ldots w_i$ where for every $j \in \{0, \ldots, i\}$, $w_j \in \Gamma_j^*$ is the tasks of priority j that are waiting in the stack. The initial configuration of $\mathcal{A}$ is $(q_0, \epsilon, \lfloor \gamma_0 \rfloor, \emptyset, \ldots, \emptyset)$. The transition relation $\Rightarrow_{\mathcal{A}}$ is defined as the union of the binary relations $\rightarrow_{0 \le j \le k}$, $\hookrightarrow_{0 \le j \le k}$, and $\rightsquigarrow_{0 \le j < k}$ defined as follows:

- **Move with task creation ($\hookrightarrow_j$):** $(q, w\gamma_j, M_0, \ldots, M_j, \emptyset, \ldots, \emptyset) \hookrightarrow_j (q', ww', M_0, \ldots, M_i + \lfloor \gamma_i \rfloor, \ldots, M_j, \emptyset, \ldots, \emptyset)$ iff $(q\gamma_j \hookrightarrow q'w' \rhd \gamma_i) \in \Delta$, $\gamma_j \in \Gamma_j$, $\gamma_i \in \Gamma_i \cup \{\epsilon\}$, and $i \le j$. Such transitions correspond to move the control state from q to q', pop γ_j from the top of the stack, push w' into the stack, and generate the task γ_i.
- **Move with task preemption ($\rightsquigarrow_j$):** $(q, w\gamma_j, M_0, \ldots, M_j, \emptyset, \ldots, \emptyset) \rightsquigarrow_j (q', ww'\gamma_i, M_0, \ldots, M_j, \emptyset, \ldots, \emptyset)$ iff $(q\gamma_j \hookrightarrow q'w' \rhd \gamma_i) \in \Delta$, $\gamma_j \in \Gamma_j$, $\gamma_i \in \Gamma_i$, and $i > j$. Such transitions correspond to move the control state from q to q', pop γ_j from the top of the stack, push w' into the stack, and to start executing the task γ_i.
- **Treatment of a new task ($\rightarrow_j$):** $(q, w, M_0, \ldots, M_j + \lfloor \gamma_j \rfloor, \emptyset, \ldots, \emptyset) \rightarrow_j (q', w\gamma_j, M_0, \ldots, M_j, \emptyset, \ldots, \emptyset)$ iff $(q \hookrightarrow q' \lhd \gamma_i) \in \Delta'$, $\gamma_j \in \Gamma_j$, and $w \in \Gamma_0^* \times \cdots \times \Gamma_{j-1}^*$. Such transitions correspond to move the control state from q to q' and to start executing the pending task γ_j if its priority level is strictly greater than the priority level of the topmost symbol in the stack.

Finally, let $\Rightarrow^*_{\mathcal{A}}$ denotes the reflexive and transitive closure of the binary relation $\Rightarrow_{\mathcal{A}}$.

3.2 Reachability Problems

The configuration (resp. control state) reachability problem is to determine, given a k-MPDS $\mathcal{A} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \gamma_0)$ and a configuration $(q, w, M_0, \ldots, M_k)$ (resp. a control state q), whether $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset, \ldots, \emptyset) \Rightarrow^*_{\mathcal{A}} (q, w, M_0, \ldots, M_k)$ (resp. $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset, \ldots, \emptyset) \Rightarrow^*_{\mathcal{A}} (q, w', M'_0, \ldots, M'_k)$ for some w' and $M'_0, \ldots, M'_k$). The *empty stack* configuration (resp. control state) reachability problem is to determine, given a k-MPDS $\mathcal{A} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \gamma_0)$ and a control state q, whether $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset, \ldots, \emptyset) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, \emptyset, \ldots, \emptyset)$ (resp. $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset, \ldots, \emptyset) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M'_0, \emptyset, \ldots, \emptyset)$ for some M'_0).

LEMMA 4. *The configuration (resp. control state) reachability problem is polynomially reducible to empty stack configuration (resp. control state) reachability problem for k-MPDSs and vice-versa.*

From now, we sometimes use configuration (resp. control state) reachability problem to denote the empty stack configuration (resp. control state) reachability problem.

3.3 Sub-classes of Multi-set Pushdown Systems with preemption

Two subclasses of our models can be defined by restricting either (1) the way the priorities are assigned to newly created tasks, or (2) the way tasks returns values after their executions.

The first class we define, called Hierarchical k-MPDS, corresponds to systems where created tasks inherit is a priority which at least as high as the one of their parents.

DEFINITION 5. *A* Hierarchical k-MPDS *(k-HMPDS) $\mathcal{A} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \gamma_0)$ is a k-MPDS such that $\Delta \subseteq \bigcup_{j=0}^{k} (Q \times \Gamma_j) \times (Q \times \Gamma_j^* \times (\bigcup_{l \geq j} \Gamma_l \cup \{\epsilon\}))$.*

The second class we consider, called Restricted k-MPDS, corresponds to systems where communication between tasks through shared memory can only happen (1) for tasks of level 0, or (2) between tasks at different levels (at the preemption and resumption operations). In other words, intra-level communication cannot occur between asynchronous tasks of level greater or equal to 1 (but value passing at synchronous procedure calls and returns is not restricted). Formally, the restriction we consider can be modeled by the fact that for each level $j \geq 1$, there is a designated state q_j such that tasks of level j can be treated only if the control state of the system is q_j.

DEFINITION 6. *A* Restricted k-MPDS *(k-RMPDS) is a tuple $\mathcal{R} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \ldots, q_k, \gamma_0)$ where $\mathcal{A} = (Q, \Gamma_0, \ldots, \Gamma_k, \Delta, \Delta', q_0, \gamma_0)$ is an k-MPDS, $q_1, \ldots, q_k \in Q$ is a fixed sequence of states, and $\Delta' \subseteq (Q \times Q \times \Gamma_0) \cup (\bigcup_{j=1}^{k} (\{q_j\} \times \{q_j\} \times \Gamma_j))$.*

4 0-MPDSs vs Petri nets

In the case of 0-MPDS, the decidability of the control state reachability problem has been shown to be decidable in [22]. We present hereafter an alternative proof based on a reduction of this problem to the coverability problem for Petri nets. We show actually that 0-MPDS can be simulated by Petri nets and vice-versa (in some sense that will be made clear later). Our principal aim by showing this relationship between the two models is to introduce smoothly

ideas and constructions which constitute the basis of the constructions presented in the next sections that are our main contributions. Actually, the reduction we show provides also a more robust proof principle, since it allows us to establish the decidability not only for control state reachability problem but also for configuration reachability problem.

4.1 From 0-MPDSs to Petri nets

We prove that every 0-MPDS can be simulated by a Petri net in the following sense:

THEOREM 7. *Given a 0-MPDS $\mathcal{A} = (Q, \Gamma_0, \Delta, \Delta', q_0, \gamma_0)$, it is possible to construct a Petri net $\mathcal{N}$ such that $(q_0, \epsilon, \lfloor \gamma_0 \rfloor) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M)$ iff $\lfloor q_0 \rfloor + \lfloor \gamma_0 \rfloor \rightarrow^*_{\mathcal{N}} M + \lfloor q \rfloor$.*

Proof (Sketch): First, we observe that for any run $(q_0, \epsilon, \{\gamma_0\}) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M)$ of $\mathcal{A}$ there are $q_0, q'_0, \ldots, q_n, q'_n \in Q$, $M_1, \ldots, M_n \in M[\Gamma_0]$, and $\gamma_0, \ldots, \gamma_n \in \Gamma_0$ such that:

$$(q_0, \epsilon, \lfloor \gamma_0 \rfloor) \rightarrow_0 (q'_0, \gamma_0, \epsilon) \hookrightarrow^*_0 (q_1, \epsilon, M_1 + \lfloor \gamma_1 \rfloor) \rightarrow_0 (q'_1, \gamma_1, M_1) \hookrightarrow^*_0 (q_2, \epsilon, M_2 + \lfloor \gamma_2 \rfloor) \rightarrow_0 \cdots \hookrightarrow^*_0 (q_{n-1}, \epsilon, M_{n-1} + \lfloor \gamma_{n-1} \rfloor) \rightarrow_0 (q'_{n-1}, \gamma_{n-1}, M_{n-1}) \hookrightarrow^*_0 (q_n, \epsilon, M_n + \lfloor \gamma_n \rfloor) \rightarrow_0 (q'_n, \gamma_n, M_n) \hookrightarrow^*_0 (q, \epsilon, M)$$

(Notice that $\rightsquigarrow_0$ is never used since there are no preemptions for 0-MPDS models.)

Then, the first step of the reduction is to show using Proposition 1 that, for every $p, p' \in Q$ and $\gamma \in \Gamma_0$, the set $\mathcal{M}(p, p', \gamma) = \{M' \mid (p, \gamma, \emptyset) \hookrightarrow^*_0 (p', \epsilon, M')\}$ is a semi-linear set. In fact, a transition rule $p_1\gamma_1 \hookrightarrow p_2 w \rhd \gamma_2$ of Δ (and therefore of $\hookrightarrow_0$) can be seen as a transition rule $p_1\gamma_1 \overset{\gamma_2}{\hookrightarrow} p_2 w$ of the LPDS $\mathcal{P} = (Q, \Gamma_0, \Gamma_0, \delta)$. Thus, a word in the trace language $L_{\mathcal{P}}(p, p', \gamma)$ corresponds to the set of waiting tasks added to the multi-set during the execution of γ to its completion, i.e. $\lfloor L_{\mathcal{P}}(p, p', \gamma) \rfloor = \mathcal{M}(p, p', \gamma)$. Hence, it is possible to construct a finite state automaton $\mathcal{S}_{(p,p',\gamma)} = (S_{(p,p',\gamma)}, \Gamma_0, \delta_{(p,p',\gamma)}, s^i_{(p,p',\gamma)}, s^f_{(p,p',\gamma)})$ such that $\lfloor L(\mathcal{S}_{(p,p',\gamma)}) \rfloor = \mathcal{M}(p, p', \gamma)$.

In the second step, we prove that every run of $\mathcal{A}$ of the form: $(p_1, \epsilon, \lfloor \gamma \rfloor) \rightarrow_0 (p_2, \gamma, \emptyset) \hookrightarrow^*_0 (p_3, \epsilon, M')$ can be simulated by a computation of the Petri net $\mathcal{N}_{(p_1,p_3,\gamma)} = (P_{(p_1,p_3,\gamma)}, T_{(p_1,p_3,\gamma)})$ such that $P_{(p_1,p_3,\gamma)} = Q \cup (\bigcup_{p_2 \in Q} S_{(p_2,p_3,\gamma)}) \cup \Gamma_0$ and $T_{(p_1,p_3,\gamma)}$ is the smallest set of transitions containing:

- **Initialization:** A transition $(p_1\gamma, s^i_{(p_2,p_3,\gamma)})$ for every transition rule $(p_1 \hookrightarrow p_2 \lhd \gamma)$ in Δ'. Such a transition takes a token from each of the places p_1 and γ and puts a token in the place $s^i_{(p_2,p_3,\gamma)}$. This allows to simulate the move $(p_1, \epsilon, \lfloor \gamma \rfloor) \rightarrow_0 (p_2, \gamma, \emptyset)$.
- **Simulation of $\mathcal{S}_{(p_2,p_3,\gamma)}$:** A transition $(s, s'\gamma')$ (resp. (s, s')) for every (s, γ', s') (resp. (s, ϵ, s')) in $\delta_{(p_2,p_3,\gamma)}$. Such transitions allow the simulation of computation of the form $(p_2, \gamma, \emptyset) \hookrightarrow^*_0 (p_3, \epsilon, M')$.

LEMMA 8. *Given a multi-set $M' \in M[\Gamma_0]$, states $p_1, p_2, p_3 \in Q$, and a task $\gamma \in \Gamma_0$, $(p_1, \epsilon, \lfloor \gamma \rfloor) \rightarrow_0 (p_2, \gamma, \emptyset) \hookrightarrow^*_0 (p_3, \epsilon, M')$ iff $\lfloor p_1 \rfloor + \lfloor \gamma \rfloor \rightarrow^*_{N_{(p_1,p_3,\gamma)}} M' + \lfloor s^f_{(p_2,p_3,\gamma)} \rfloor$.*

Since any run that reaches a configuration (q, ϵ, M) can be decomposed as a sequence of runs of the form: $(p_1, \epsilon, \lfloor \gamma \rfloor) \rightarrow_0 (p_2, \gamma, \emptyset) \hookrightarrow^*_0 (p_3, \epsilon, M')$, then $(q_0, \epsilon, \lfloor \gamma_0 \rfloor) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M)$ can be simulated by the following Petri net $\mathcal{N} = (P, T)$ where: $P = \bigcup_{p_1,p_3 \in Q, \gamma \in \Gamma_0} P_{(p_1,p_3,\gamma)}$ is a finite set of places, and T is the smallest set of transitions such that: $T_{(p_1,p_3,\gamma)} \subseteq T$ and $(s^f_{(p_2,p_3,\gamma)}, p_3)$ is in T for every $p_1, p_2, p_3 \in Q$ and $\gamma \in \Gamma_0$. Hence, Theorem 7 follows immediately from the following lemma:

LEMMA 9. *Given a state $q \in Q$ and a multi-set $M \in M[\Gamma_0]$, $(q_0, \epsilon, \lfloor \gamma_0 \rfloor) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M)$ iff $W' = M + \lfloor q \rfloor$ is reachable by the Petri net $\mathcal{N}$ from $W = \lfloor \gamma_0 \rfloor + \lfloor q_0 \rfloor$.* □

The following fact follows immediately from Proposition 4 and Theorem 7.

COROLLARY 10. *Configuration and control state reachability for 0-MPDSs are decidable.*

4.2 From Petri nets to 0-MPDSs

We show that every Petri net can be simulated by a 0-MPDS in the following sense:

THEOREM 11. *Given a Petri net $\mathcal{N} = (P, T)$, it is possible to construct a 0-MPDS $\mathcal{A}$ with a special state q_0 such that $W \rightarrow^*_{\mathcal{N}} W'$ iff $(q_0, \epsilon, W) \Rightarrow^*_{\mathcal{A}} (q_0, \epsilon, W')$.*

This can be done by adapting the construction given in [22] to prove the lower bound on the complexity for control state reachability problem for 0-MPDS.

By Theorem 11 and the fact that the set of reachable multi-sets for Petri nets is in general not semi-linear [11], it is possible to show that:

COROLLARY 12. *The set of reachable multi-set configurations $\{M \mid (q_0, \epsilon, \lfloor \gamma_0 \rfloor) \Rightarrow^*_{\mathcal{A}} (q, \epsilon, M)\}$ by an 0-MPDS $\mathcal{A} = (Q, \Gamma_0, \Delta, \Delta', q_0, \gamma_0)$ is in general not semi-linear.*

5 Reachability Analysis for k-MPDSs

In this section, we prove that the configuration (resp. the control state) reachability problem for k-MPDSs is decidable by reduction to the reachability (resp. coverability) problem for Petri nets with weak inhibitor arcs.

THEOREM 13. *Configuration and control state reachability are decidable for k-MPDSs.*

Proof (Sketch): We consider here that $k \geq 1$ (since $k = 0$ has been already considered in the previous section). To simplify the presentation of the proof, we consider first the case of $k = 1$. The generalization to any $k \geq 1$ is given later.

Case $k = 1$: Let $\mathcal{A} = (Q, \Gamma_0, \Gamma_1, \Delta, \Delta', q_0, \gamma_0)$ be an 1-MPDS. Let $\Rightarrow_1 = \rightarrow_1 \cup \hookrightarrow_1$ and $\Rightarrow_0 = \rightsquigarrow_0 \cup \Rightarrow_1 \cup \hookrightarrow_0$ be two transition relations. Thanks to Proposition 4, we consider w.l.o.g configurations of the form $(q, \epsilon, M, \emptyset)$ with $M \in M[\Gamma_0]$. We observe that $(q, \epsilon, M, \emptyset)$ is reachable by $\mathcal{A}$ iff there are some $q_0, q'_0, q_1, \ldots, q_n \in Q$, $\gamma_0, \ldots, \gamma_{n-1} \in \Gamma_0$, and $M_1, \ldots, M_n \in M[\Gamma_0]$ such that $q_n = q$, $M_n = M$, and:

$$\begin{aligned}\textit{Path 0: } & (q_0, \epsilon, \lfloor \gamma_0 \rfloor, \emptyset) \rightarrow_0 (q'_0, \gamma_0, \emptyset, \emptyset) \Rightarrow^*_0 (q_1, \epsilon, M_1 + \lfloor \gamma_1 \rfloor, \emptyset) \rightarrow_0 (q'_1, \gamma_1, M_1, \emptyset) \Rightarrow^*_0 \\ & (q_2, \epsilon, M_2 + \lfloor \gamma_2 \rfloor, \epsilon) \rightarrow_0 (q'_2, \gamma_2, M_2, \emptyset) \cdots (q'_{n-1}, \gamma_{n-1}, M_{n-1}, \emptyset) \Rightarrow^*_0 (q_n, \epsilon, M_n, \emptyset)\end{aligned}$$

Indeed, any computation of $\mathcal{A}$ of the form $(p, \gamma, \emptyset, \emptyset) \Rightarrow^*_0 (p', \epsilon, N, \emptyset)$ for some $p, p' \in Q$, $\gamma \in \Gamma_0$, and $N, N' \in M[\Gamma_0]$, there are $p'_0, p_0, p''_1, p'_1, \ldots, p''_m \in Q$, $\gamma'_0, \ldots, \gamma'_{m-1} \in \Gamma_1$, $w'_0, w_1, w'_1, \ldots, w_m \in \Gamma_0^*$, and $N'_0, N_1, N'_1, \ldots, N_m \in M[\Gamma_0]$ such that:

$$\begin{aligned}\textit{Path 1: } & (p, \gamma, \emptyset, \emptyset) \hookrightarrow^*_0 (p'_0, w'_0, N'_0, \emptyset) \rightsquigarrow_0 (p_0, w_1\gamma'_0, N'_0, \emptyset) \Rightarrow^*_1 (p''_1, w_1, N_1, \emptyset) \hookrightarrow^*_0 \\ & (p'_1, w'_1, N'_1, \emptyset) \rightsquigarrow_0 (p_1, w_2\gamma'_1, N'_1, \emptyset) \Rightarrow^*_1 (p''_2, w_2, N_2, \emptyset) \hookrightarrow^*_0 (p'_2, w'_2, N'_2, \emptyset) \rightsquigarrow_0 \\ & (p_2, w_3\gamma'_2, N'_2, \emptyset) \Rightarrow^*_1 (p''_3, w_3, N_3, \emptyset) \hookrightarrow^*_0 \cdots \Rightarrow^*_1 (p''_m, w_m, N_m, \emptyset) \hookrightarrow^*_0 (p', \epsilon, N, \emptyset)\end{aligned}$$

Then the proof is structured as follows:

- For every $g, g' \in Q$ and $\gamma' \in \Gamma_1$, we construct a Petri net $\mathcal{N}'_{(g,g',\gamma')}$ with a special place c counting the number of pending tasks of priority 1, such that the set of reachable multi-sets when the place c is empty is precisely $\mathcal{M}_1(g, g', \gamma') = \{N' \in M[\Gamma_0] \mid (g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)\}$.
- For every $p, p' \in Q$ and $\gamma \in \Gamma_0$, we construct a Petri net $\mathcal{N}_{(p,p',\gamma)}$ with weak inhibitor arcs that characterizes the set $\mathcal{M}_0(p, p', \gamma) = \{N \in M[\Gamma_0] \mid (p, \gamma, \emptyset, \emptyset) \Rightarrow_0^* (p', \epsilon, N, \emptyset)\}$. The Petri net $\mathcal{N}_{(p,p',\gamma)}$ simulates the runs of the form *Path 1* by delegating the $\Rightarrow_1^*$ segments of these runs to the networks $\mathcal{N}'_{(g,g',\gamma')}$ introduced above.
  The difficulties to face in doing that are: (1) transitions $\rightsquigarrow_0$ are not always taken when the stack is empty (since at level 1 the context of the interrrupted task of level 0 is still present in the stack), and (2) the effect of $\Rightarrow_1^*$ computations on the multiset of pending tasks of level 0 must be computed precisely and this should be done only for such computations that reach at their end a configuration where the multiset of pending tasks of level 1 is empty. Since computations at level 1 can be as general as computations of any Petri net, the latter problem needs to be addressed using some notion of place emptyness testing. Then, inhibitor arcs are used to check at the end of $\Rightarrow_1^*$ computations that the place c of $\mathcal{N}'_{(g,g',\gamma')}$ is empty.
  To tackle the first issue, the idea is to reason about the whole computations of level 0 by inserting instead of the level 1 segments a tuple $(g_1, g_2, \gamma') \in Q \times Q \times \Gamma_1$ corresponding to the guess that an interruption by a task γ' of level 1 is able to bring the control state from g_1 to g_2. So, we build a pushdown system labelled by the generated task of level 0 as well as the guessed tuples (g_1, g_2, γ') defined as above. Then, the key observation is that the information represented in the traces of this LPDS can be represented by the traces of a finite state automata $\mathcal{S}$. Indeed, (1) like in the previous section, the ordering between tasks generated by level 0 computations between two given control states does not need to be kept, and (2) it is sufficient to know for each *Path 1* computation how many times each guessing pair (g_1, g_2, γ') occurs; the consistency of these occurrences within the computation (i.e., these guesses can indeed be inserted in the computation) can be checked using a finite control.
  Then, to simulate the computations of the form *Path 1*, the Petri net $\mathcal{N}_{(p,p',\gamma)}$ simulates in parallel the evolution of the control states and the finite-state automaton $\mathcal{S}$ by (1) generating a level 0 task whenever the transition of the automaton is labelled by this task, and (2) simulating the network $\mathcal{N}'(g_1, g_2, \gamma')$ whenever the transition of $\mathcal{S}$ is labelled by (g_1, g_2, γ') where γ' is a level 1 task (which is supposed to be the one which preempts the current level 0 task).
- The collection of all the networks $\mathcal{N}_{(p,p',\gamma)}$ with $p, p' \in Q$ and $\gamma \in \Gamma_0$ are used to build a network $\mathcal{N}_0$ with weak inhibitor arcs that simulates all the runs that reaches a configuration of the form $(q, \epsilon, M, \emptyset)$ (i.e. computations of the form *Path 2*).

Computing $\mathcal{N}'_{(g,g',\gamma')}$: Let $g, g' \in Q$ be a pair of states and $\gamma' \in \Gamma_1$ be a task of priority 1. Then, any computation of the form $(g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', N', M')$ can be seen as a computation of the 0-MPDS $\mathcal{A}_1$ which mimics the execution of $\mathcal{A}$ over tasks of priority 1. Formally, $\mathcal{A}_1$ is defined by the tuple $(Q, \Gamma_0 \cup \Gamma_1, \Delta_1, \Delta'_1, g, \gamma')$ where $\Delta_1 = \Delta$ and $\Delta'_1 = \Delta' \cap (Q \times Q \times \Gamma_1)$. Thus, $(g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', N', M')$ iff $(g, \gamma', \emptyset) \Rightarrow_{\mathcal{A}_1}^* (g', \epsilon, N' + M')$.

Then, by adapting the construction given in the previous section (see Theorem 7) to $\mathcal{A}_1$, we can construct a Petri net $\mathcal{N}'_{(g,g',\gamma')} = (P'_{(g,g',\gamma')}, T'_{(g,g',\gamma')})$ which has two special places c and $t_{g'}$. The place c is used to count the number of pending task of priority 1. A token in the place $t_{g'}$ means that the guessed control state when all tasks of priority level 1 are done is g'. The relation between $\mathcal{A}$ and $\mathcal{N}'_{(g,g',\gamma')}$ is given by the following lemma:

LEMMA 14. *Let $g, g' \in Q$ be a pair of states and $\gamma' \in \Gamma_1$ be a task of priority 1. Then, $(g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)$ iff $\lfloor g \rfloor + \lfloor \gamma' \rfloor + \lfloor t_{g'} \rfloor \rightarrow^*_{\mathcal{N}'_{(g,g',\gamma')}} \lfloor g' \rfloor + \lfloor t_{g'} \rfloor + N'$.*

Computing $\mathcal{N}_{(p,p',\gamma)}$: For every $p, p' \in Q$ and $\gamma \in \Gamma_0$, we construct a Petri net $\mathcal{N}_{(p,p',\gamma)}$ with weak inhibitor arcs that simulates computations of $\mathcal{A}$ of the form: $(p, \epsilon, \lfloor \gamma \rfloor, \emptyset) \rightarrow_0 (p'', \gamma, \emptyset, \emptyset) \Rightarrow_0^* (p', \epsilon, M', \emptyset)$. Then, let $\mathcal{P}' = (Q, \Sigma', \Gamma_0, \delta')$ be a LPDS with $\Sigma' = \{(g, g', \gamma') \mid g, g' \in Q, \gamma' \in \Gamma_1\} \cup \Gamma_0$. For every $g, g' \in Q$ and $\gamma' \in \Gamma_1$, (g, g', γ') is a new symbol which represents the set $\mathcal{M}_1(g, g', \gamma')$ (we have showed in the previous paragraph how these sets can be characterized by the Petri nets $\mathcal{N}'_{(g,g',\gamma')}$). The set δ' is defined as the smallest set of transition rules such that:

- If $g_1\gamma_1 \hookrightarrow g_2 w' \rhd \gamma_2 \in \Delta$, $\gamma_1 \in \Gamma_0$, and $\gamma_2 \in \Gamma_0 \cup \{\epsilon\}$, then $g_1\gamma_1 \overset{\gamma_2}{\hookrightarrow} g_2 w' \in \delta'$;
- If $g_1\gamma_1 \hookrightarrow g_2 w' \rhd \gamma' \in \Delta$, $\gamma_1 \in \Gamma_0$, and $\gamma_2 \in \Gamma_1$, then $g_1\gamma_1 \overset{(g_2,g_3,\gamma_2)}{\hookrightarrow} g_3 w' \in \delta'$ for every $g_3 \in Q$. Such a transition rule records in its label the fact that a guess is made at this point of the computation: The level 1 task γ_2 interrupts the level 0 task γ_0, and then the level 1 computation of the form $(g_2, \gamma_2, \emptyset, \emptyset) \Rightarrow_1^* (g_3, \epsilon, N, \emptyset)$ brings the control state from g_2 to g_3.

Thanks to Proposition 1, it is possible to construct a finite state automaton $\mathcal{S}_{(p'',p',\gamma)} = (S_{(p'',p',\gamma)}, \Sigma', \delta_{(p'',p',\gamma)}, s^i_{(p'',p',\gamma)}, s^f_{(p'',p',\gamma)})$ such that $\lfloor L(\mathcal{S}_{(p'',p',\gamma)}) \rfloor = \lfloor L_{\mathcal{P}'}(p'', p', \gamma) \rfloor$. Then, we can define a Petri net with weak inhibitor arcs $\mathcal{N}_{(p,p',\gamma)} = (P_{(p,p',\gamma)}, T_{(p,p',\gamma)})$ using the set of automata $\mathcal{S}_{(p'',p',\gamma)}$ as follows:

- $P_{(p,p',\gamma)} = \{\top, \bot\} \cup P \cup (\bigcup_{p'' \in Q} S_{(p'',p',\gamma)})$ is a finite set of places where $P = \bigcup_{g,g' \in Q, \gamma' \in \Gamma_1} P'_{(g,g',\gamma')}$. The places $\top$ and $\bot$ are flags that indicate if the simulation of $\mathcal{S}_{(p'',p',\gamma)}$ has been initiated or not. When the simulation starts, the place $\top$ is emptied and a token is put in $\bot$. This allows to ensure that the segments $\rightarrow_0 \circ \Rightarrow_0^*$ are simulated in a serial manner and do not interfer.
- The set of transitions $T_{(p,p',\gamma)}$ is the set of the following transitions:
  - **Initialization:** A transition $(\emptyset, \top p\gamma, \bot s^i_{(p'',p',\gamma)})$ for each transition rule $p \hookrightarrow p'' \lhd \gamma$ in Δ'. This transition simulates the move $(p, \epsilon, \lfloor \gamma \rfloor, \emptyset) \rightarrow_0 (p'', \gamma, \emptyset, \emptyset)$.
  - **Simulation of $\mathcal{S}_{(p'',p',\gamma)}$:** A transition rule $(\emptyset, \bot s, \bot s'\gamma')$ (resp. $(\emptyset, \bot s, \bot s')$) for each each transition (s, γ', s') (resp. (s, ϵ, s')) in $\delta_{(p'',p',\gamma)}$ with $\gamma' \in \Gamma_0$. A transition rule $(\emptyset, \bot s, g\, t_{g'}\, \gamma'\, s')$ for each transition $(s, (g, g', \gamma'), s')$ of $\mathcal{S}_{(p,p',\gamma)}$.
  - **Simulation of $\mathcal{N}'_{(g,g',\gamma')}$:** The set of transitions $T'_{(g,g',\gamma')}$ of the network $\mathcal{N}'_{(g,g',\gamma')}$ defined previously for each $g, g' \in Q$ and $\gamma' \in \Gamma_1$.
  - **Checking the guessed tuple** (g, g', γ')**:** A transition rule $(c, g' t_{g'}, \bot)$ for each $g' \in Q$. This rule checks if there are no more tasks of level 1 (i.e. $\lfloor c \rfloor = \emptyset$) and that the control state at the resumption of the preempted task of level 0 is g'.

LEMMA 15. $(p, \epsilon, \lfloor\gamma\rfloor, \emptyset) \rightarrow_0 (p'', \gamma, \emptyset, \emptyset) \Rightarrow_0^* (p', \epsilon, M', \emptyset)$ *iff the multi-set* $W' = M' + \lfloor s^f_{(p'',p',\gamma)}\rfloor + \lfloor\bot\rfloor$ *is reachable by* $\mathcal{N}_{(p,p',\gamma)}$ *from* $W = \lfloor\top\rfloor + \lfloor\gamma\rfloor + \lfloor p\rfloor$.

Computing the Petri net $\mathcal{N}_0$: We observe that any run $(q_0, \epsilon, \{\gamma_0\}, \emptyset) \Rightarrow_{\mathcal{A}}^* (q, \epsilon, M, \emptyset)$ of $\mathcal{A}$ can be simulated by a sequence of executions of the Petri nets $\mathcal{N}_{(p,p',\gamma)}$ with $p, p' \in Q$ and $\gamma \in \Gamma_0$. This can be obtained by defining the Petri net $\mathcal{N}_0 = (P_0, T_0)$ as follows:

- P_0 is a finite union of places $P_{(p,p',\gamma)}$ for every $p, p' \in Q$ and $\gamma \in \Gamma_0$.
- T_0 is the smallest set of transitions satisfying the following conditions:
  - $T_{(p,p',\gamma)} \subseteq T_0$ for every $p, p' \in Q$ and $\gamma \in \Gamma_0$,
  - $(\emptyset, \bot s^f_{(p'',p',\gamma)}, \top p') \in T_0$ for every $p'', p' \in Q$ and $\gamma \in \Gamma_0$, which makes possible the iteration of the executions of Petri nets of the form $\mathcal{N}_{(p,p',\gamma)}$.

Then, Theorem 13 follows immediately from the following lemma:

LEMMA 16. $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset) \Rightarrow_{\mathcal{A}}^* (q, \epsilon, M, \emptyset)$ *iff* $\lfloor\gamma_0\rfloor + \lfloor q_0\rfloor + \lfloor\top\rfloor \rightarrow_{\mathcal{N}_0}^* M + \lfloor q\rfloor + \lfloor\top\rfloor$.

General Case: This construction can be extended to the case where we have an arbitrary number k of priorities. In this case, we compute a Petri net with *weak inhibitor arcs* as follows: we need k places $c_1, \ldots, c_k$, where c_i counts the number of tasks in the multiset of level i. Then, these places can be ordered as follows: $c_1 > c_2 > \cdots > c_k$. Indeed, in the computed Petri net with inhibitor arcs, we need to check whether $c_i = 0$ only if the other counters $c_j, j \geq i$ are also null. This ensures that the network we construct is a weak inhibitor arcs Petri net. □

6 Reachability Analysis for k-RMPDSs and k-HMPDSs

6.1 Reachability Analysis of Restricted k-MPDSs

In this section, we prove that the configuration and control state reachability problems for k-RMPDSs are decidable and reducible to the same problems for 0-MPDSs. In particular, we show that the control state reachability problem for k-RMPDSs is reducible to the coverability problem for Petri nets. This is based on the fact that it is possible to prove in this case that the sets $\mathcal{M}_1(g, g', \gamma') = \{N' \in M[\Gamma_0] \mid (g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)\}$ are semi-linear and can be computed as Parikh images of context free languages. Notice that for the case of (unrestricted) k-MPDS these sets are not semi-linear in general (see Corrollary 12).

THEOREM 17. *For any $k > 0$, control state and configuration reachability problems for k-RMPDSs are reducible to the same problems for some $(k-1)$-RMPDSs.*

Proof (Sketch): Let us fix a 1-RMPDS $\mathcal{R} = (Q, \Gamma_0, \Gamma_1, \Delta, \Delta', q_0, q_1, \gamma_0)$ and its transition relations $\Rightarrow_1 = \rightarrow_1 \cup \hookrightarrow_1$ and $\Rightarrow_0 = \rightsquigarrow_0 \cup \Rightarrow_1 \hookrightarrow_0$. Given $g, g' \in Q$ and $\gamma' \in \Gamma_1$, we construct a context free grammar $G_{(g,g',\gamma')}$ such that $\lfloor L(G_{(g,g',\gamma')})\rfloor = \mathcal{M}_1(g, g', \gamma') = \{N' \in M[\Gamma_0] \mid (g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)\}$. Indeed, let us observe that such computations can be decomposed as follows:

$$(g, \gamma', \emptyset, \emptyset) \hookrightarrow_1^* (q_1, \epsilon, N_0^1, N_1^1 + \lfloor\gamma_1\rfloor) \rightarrow_1 (q_1, \gamma_1, N_0^1, N_1^1) \hookrightarrow_1^* (q_1, \epsilon, N_0^2, N_1^2 + \lfloor\gamma_2\rfloor) \rightarrow_1 (q_1, \gamma_2, N_0^2, N_1^2) \hookrightarrow_1^* \cdots \hookrightarrow_1^* (q_1, \epsilon, N_0^n, N_1^n + \lfloor\gamma_n\rfloor) \rightarrow_1 (q_1, \gamma_n, N_0^n, N_1^n) \hookrightarrow_1^* (g', \epsilon, N', \emptyset)$$

Then, given $g_1, g_2 \in Q$ and $\gamma \in \Gamma_1$, it is possible to characterize the set of computations $(g_1, \gamma, \emptyset, \emptyset) \hookrightarrow_1^* (g_2, \epsilon, N_0, N_1)$ by a CFG $G'_{\langle g_1, g_2, \gamma\rangle}$ such that $\lfloor L(G'_{\langle g_1, g_2, \gamma\rangle})\rfloor =$

$\{N_0 + N_1 \mid (g_1, \gamma, \emptyset, \emptyset) \hookrightarrow_1^* (g_2, \epsilon, N_0, N_1)\}$ using a similar construction to the characterization of $\hookrightarrow_0^*$ computations in 0-MPDS (see proof of Theorem 7). Now, we use the set of CFGs $G'_{\langle g,g',\gamma'\rangle}$, $G'_{\langle g,q_1,\gamma'\rangle}$, $G'_{\langle q_1,q_1,\gamma\rangle}$, and $G'_{\langle q_1,g',\gamma\rangle}$ to built $G_{\langle g,g',\gamma'\rangle}$ as follows. Every computation $(g_1, \epsilon, \emptyset, \lfloor\gamma\rfloor) \rightarrow_1 (g_1, \gamma, \emptyset, \emptyset) \hookrightarrow_1^* (g_2, \epsilon, N_0, N_1)$ is simulated by $G_{(g,g',\gamma')}$: γ is rewritten by the axiom of $G'_{\langle g_1,g_2,\gamma\rangle}$, and then rules of $G'_{\langle g_1,g_2,\gamma\rangle}$ are applied. This can be done because the processing order of pending tasks of level 1 is not relevant due to the restriction in RMPDS models (two pending tasks of level 1 cannot communicate).

By Proposition 1, we can construct a finite state automaton $\mathcal{S}_{(g,g',\gamma')}$ such that $\lfloor L(\mathcal{S}_{(g,g',\gamma')})\rfloor = \lfloor L(G_{(g,g',\gamma')})\rfloor = \mathcal{M}_1(g, g', \gamma')$. Then, we construct a 0-MPDS $\mathcal{A}'$ over the alphabet Γ_0 that mimics any computation of $\mathcal{R}$ over tasks of priority 0, i.e. $g_1\gamma_1 \hookrightarrow g_2w' \rhd \gamma_2$ (resp. $g_1 \hookrightarrow g_2 \lhd \gamma_1$) is a rule of $\mathcal{A}'$ iff $g_1\gamma_1 \hookrightarrow g_2w' \rhd \gamma_2$ (resp. $g_1 \hookrightarrow g_2 \lhd \gamma_1$) is a transition rule of $\mathcal{R}$ and $\gamma_1, \gamma_2 \in \Gamma_0 \cup \{\epsilon\}$. On the other hand, any computation $(g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)$, with $g, g' \in Q$ and $\gamma' \in \Gamma_1$, can be simulated by a computations of $\mathcal{A}'$ that: (1) moves the control state from g to the initial state of $\mathcal{S}_{(g,g',\gamma')}$; (2) each transition (s, γ'', s') of $\mathcal{S}_{(g,g',\gamma')}$ by rule that moves the control state from s to s' and creates the task γ''; and (3) changes the control state from the final state $\mathcal{S}_{(g,g',\gamma')}$ to g'. Hence, $(q_0, \epsilon, \lfloor\gamma_0\rfloor, \emptyset) \Rightarrow_{\mathcal{R}}^* (q, \epsilon, M, \emptyset)$ iff $(q_0, \epsilon, \lfloor\gamma_0\rfloor) \Rightarrow_{\mathcal{A}'}^* (q, \epsilon, M)$ for any $q \in Q$ and $M \in M[\Gamma_0]$. □

6.2 Reachability Analysis of Hierarchical k-MPDSs

In this section, we study the reachability problem for Hierarchical k-MPDS. We show that control state and configuration reachability problems for k-HMPDSs are decidable using reachability for Petri nets without inhibitor arcs. Here, we use the fact that the set $\mathcal{M}_1(g, g', \gamma') = \{N' \in M[\Gamma_0] \mid (g, \gamma', \emptyset, \emptyset) \Rightarrow_1^* (g', \epsilon, N', \emptyset)\}$ is empty. Indeed, in that case, the only relevant information about level 1 computation segments when simulating *Path 1* computations is whether, given two states g and g′ and a task $\gamma \in \Gamma_1$, it is possible to have a run from $g\gamma$ which reaches a configuration with control state g' where no level 1 tasks are left. This can be solved as a reachability problem in a Petri net simulating the level 1 computations of a configuration where there are no pending tasks of priority 1.

THEOREM 18. *For any $k \geq 0$, the control state and configuration reachability problems for (k)-HMPDSs are reducible to the corresponding problems for $(k-1)$-HMPDSs using the reachability problem for Petri nets.*

7 Conclusion

We have investigated the reachability problem for a model of concurrent programs where tasks (1) can be dynamically created, (2) may have different levels of priority, and (3) may be preempted by tasks of higher priority level. We have shown that this problem is difficult but decidable. Our proof is based on a reduction to the reachability problem in a class of Petri nets with inhibitor arcs. We have also identified a class of models for which the (control point) reachability problem can be reduced to the reachability problem in Petri nets without inhibitor arcs, and another class of models for which the control point reachability problem can be reduced, using Parikh image computations of context-free languages, to the coverability problem in Petri nets. For the latter class, although the problem of solving

the control point reachability problem remains complex (EXPSPACE-hard), we believe that it can be handled in practice using efficient algorithms and tools for (1) computing CFL Parikh images using a Newton method based technique for solving polynomial equations in commutative Kleene algebras [6], and for (2) solving the coverability problem in Petri nets using forward reachability analysis and complete abstraction techniques [8, 9].

References

[1] M. F. Atig, A. Bouajjani, and T. Touili. On the reachability analysis of acyclic networks of pushdown systems. In *CONCUR'08*, LNCS, 2008.
[2] A. Bouajjani and J. Esparza. Rewriting models of boolean programs. In *RTA*, volume 4098 of *LNCS*, pages 136–150. Springer, 2006.
[3] A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR'05*, LNCS, 2005.
[4] A. Bouajjani and T. Touili. On Computing Reachability Sets of Process Rewrite Systems. In *RTA'05*. LNCS, 2005.
[5] J. Esparza. Decidability and complexity of Petri net problems – an introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, number 1491 in Lecture Notes in Computer Science, pages 374–428, 1998.
[6] J. Esparza, S. Kiefer, and M. Luttenberger. On fixed point equations over commutative semirings. In *STACS*, volume 4393 of *LNCS*, pages 296–307. Springer, 2007.
[7] J. Esparza and A. Podelski. Efficient algorithms for pre* and post* on interprocedural parallel flow graphs. In *POPL'00*. ACM, 2000.
[8] G. Geeraerts, J.-F. Raskin, and L. V. Begin. Expand, enlarge, and check: New algorithms for the coverability problem of wsts. In *FSTTCS*, volume 3328 of *LNCS*, 2004.
[9] G. Geeraerts, J.-F. Raskin, and L. V. Begin. Expand, enlarge and check... made efficient. In *CAV*, volume 3576 of *LNCS*, pages 394–407. Springer, 2005.
[10] M. Hack. Decision problems for petri nets and vector addition systems. Technical Report TR 161, 1976.
[11] J. E. Hopcroft and J. J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979.
[12] R. Jhala and R. Majumdar. Interprocedural analysis of asynchronous programs. In *POPL*, pages 339–350, 2007.
[13] V. Kahlon, F. Ivancic, and A. Gupta. Reasoning about threads communicating via locks. In *CAV*, volume 3576 of *LNCS*. Springer, 2005.
[14] R. Lipton. The reachability problem requires exponential time. Technical Report TR 66, 1976.
[15] D. Lugiez and P. Schnoebelen. The regular viewpoint on pa-processes. *Theor. Comput. Sci.*, 274(1-2):89–115, 2002.
[16] E. W. Mayr. An algorithm for the general petri net reachability problem. In *STOC '81*, pages 238–246, New York, NY, USA, 1981. ACM Press.
[17] R. Mayr. Decidability and Complexity of Model Checking Problems for Infinite-State Systems. Phd. thesis, Technical University Munich, 1998.
[18] R. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.
[19] S. Qadeer, S. Rajamani, and J. Rehof. Procedure Summaries for Model Checking Multithreaded Software. In *POPL'04*. ACM, 2004.
[20] C. Rackoff. The covering and boundedness problem for vector addition systems. In *TCS*, 1978.
[21] K. Reinhardt. Reachability in petri nets with inhibitor arcs. Revised manuscript, 2006.
[22] K. Sen and M. Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *CAV*, pages 300–314, 2006.
[23] S. L. Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS*, pages 161–170. IEEE, 2007.