On the Analysis of Interacting Pushdown Systems

Vineet Kahlon
NEC Labs, Princeton, USA.
kahlon@nec-labs.com

Aarti Gupta
NEC Labs, Princeton, USA.
agupta@nec-labs.com

Abstract

Pushdown Systems (PDSs) have become an important paradigm for program analysis. Indeed, recent work has shown a deep connection between inter-procedural dataflow analysis for sequential programs and the model checking problem for PDSs. A natural extension of this framework to the concurrent domain hinges on the, somewhat less studied, problem of model checking Interacting Pushdown Systems. In this paper, we therefore focus on the model checking of Interacting Pushdown Systems synchronizing via the standard primitives - locks, rendezvous and broadcasts, for rich classes of temporal properties - both linear and branching time. We formulate new algorithms for model checking interacting PDSs for important fragments of LTL and the Mu-Calculus. Additionally, we also delineate precisely the decidability boundary for each of the standard synchronization primitives.

Categories and Subject Descriptors F [*3*]: 1

General Terms Verification, Theory

Keywords Concurrency, Dataflow Analysis, Pushdown Systems, Model Checking, LTL, Mu-Calculus

1. Introduction

In recent years, Pushdown Systems (PDSs) have emerged as a powerful, unifying framework for efficiently encoding inter-procedural dataflow analysis. Given a sequential program, abstract interpretation is first used to get a finite representation of the control part of the program while recursion is modeled using a stack. Pushdown systems then provide a natural framework to model such abstractly interpreted structures. A PDS has a finite control part corresponding to the valuation of the variables of the program and a stack which provides a means to model recursion. Dataflow analysis then exploits the fact that the model checking problem for PDSs is decidable for very expressive classes of properties - both linear and branching time (cf. [1, 17]). Not only has this powerful framework been useful in encoding many different dataflow analyses but has, in many cases, led to strictly more expressive dataflow frameworks than those provided by classical inter-procedural dataflow analysis. Many variants of the basic PDS model like Weighted Pushdown System [16], Extended Weighted Pushdown System [11], etc., have been proposed and applied to various application domains thus highlighting (i) the deep connection between dataflow analysis and the model checking problem for PDSs, and (ii) the usefulness of PDSs as a natural model for program analysis.

However, most of this work has focused only on the analysis of sequential programs. Analogous to the sequential case, inter-procedural dataflow analysis for concurrent multi-threaded programs can be formulated as a model checking problem for interacting PDSs on which, however, only very limited work exists. Early work focused on the model checking of PDSs interacting via pairwise rendezvous. While for a single PDS the model checking problem is efficiently decidable for very expressive logics, it was shown in [15] that even simple properties like reachability become undecidable for systems with only two threads but where the threads synchronize using CCS-style pairwise rendezvous. In [10], undecidability was shown to hold even for PDSs communicating via locks. It was proved that the model checking problem for pairwise reachability, and hence multi-indexed LTL formulae, is undecidable, in general, even for systems with just two PDSs synchronizing via locks.

However, the more important contribution of [10] was that the problem of deciding simple pairwise reachability for the practically important paradigm of PDSs interacting via nested locks was shown to be efficiently decidable. In [9], the decidability result for PDSs synchronizing via nested locks was further extended to single-index LTL properties. These results demonstrate that there are important fragment of temporal logics and useful models of interacting PDSs for which efficient decidability results can be obtained. It is important that such fragments be identified for each of the standard synchronization primitives. Indeed, formulating efficient algorithms for model checking interacting Pushdown Systems lies at the core of scalable data flow analysis for concurrent programs. Furthermore, of fundamental importance also is the need to delineate precisely the decidability/undecidability boundary of the model checking problem for PDSs interacting via the standard synchronization primitives. Indeed, there is currently little work on understanding exactly where the decidability/undecidability boundary for the problem lies. An insight into the causes of undecidability often plays a key role in devising effective techniques to surmounting the undecidability barrier in practice.

In this paper, we study the problem of model checking PDSs interacting via the standard communication primitives - locks, pairwise and asynchronous rendezvous, and broadcasts. Locks are primitives commonly used to enforce mutual exclusion. Asynchronous Rendezvous and Broadcasts model, respectively, the `Wait()\Notify()` and `Wait()\NotifyAll()` constructs of Java, while Pairwise Rendezvous are inspired by the CCS process algebra. Moreover, we also consider the practically important paradigm of PDSs communicating via nested locks. Most real-world concurrent programs use locks in a nested fashion, viz., each thread can only release the lock that it acquired last and that has not yet been released. Indeed, practical programming guidelines used by software developers often require that locks be used in a nested fashion.

POPL'07 January17–19, 2007, Nice, France.
Copyright © 2007 ACM 1-59593-575-4/07/0001. . . $5.00

In fact, in Java (version 1.4) and C# locking is syntactically guaranteed to be nested.

As is usual when model checking concurrent systems, we consider correctness properties expressed as multi-index temporal logic formulae, where in a k-index formula, the atomic propositions are interpreted over the local control states of k PDSs. It turns out that most interesting properties about concurrent programs can be expressed as single or double-index properties. As part of previous work [9], it has been shown that the model checking problem is efficiently decidable for single-index LTL properties for Dual-PDS systems interacting via nested locks. However, a number of interesting properties about concurrent programs, like data races, can only be expressed as double-indexed properties. In this paper, we therefore consider double-indexed LTL properties. Furthermore, most of the work on the model checking of concurrent programs has focused on safety and linear-time properties with little work addressing branching-time properties. Hence from the branching-time spectrum, we consider Alternation-free Mu-Calculus properties.

It turns out that unlike single-index LTL properties, the decidability scenario for double-indexed LTL properties is more interesting. While the model checking problem for single-index LTL properties is efficiently decidable, it is, in general, not decidable for the full-blown double-index LTL, but only for certain fragments. Undecidability of a sub-logic of double-indexed LTL hinges on whether it is expressive enough to encode the disjointness of the context-free languages accepted by the PDSs in the given Multi-PDS system as a model checking problem. This, in turn, depends on the temporal operators allowed by the logic thereby providing a natural way to characterize fragments of double-indexed LTL for which the model checking problem is decidable. We use $L(Op_1, ..., Op_k)$, where $Op_i \in \{\mathsf{X}, \mathsf{F}, \mathsf{U}, \mathsf{G}, \overset{\infty}{\mathsf{F}}\}$, to denote the fragment comprised of formulae of the form $\mathsf{E}f$, where f is double-indexed LTL formula in positive normal form (PNF), viz., only atomic propositions are negated, built using the operators $Op_1, ..., Op_k$ and the Boolean connectives $\vee$ and $\wedge$. Here X "next-time", F "sometimes", U, "until", G "always", and $\overset{\infty}{\mathsf{F}}$ "infinitely-often" denote the standard temporal operators and E is the "existential path quantifier". Obviously, $L(\mathsf{X}, \mathsf{U}, \mathsf{G})$ is the full-blown double-indexed LTL.

In this paper, we not only formulate efficient procedures for fragments of double-indexed LTL for which the model checking for Dual-PDS system is decidable but also delineate precisely the decidability/undecidability boundary for each of the standard synchronization primitives. Specifically, we show the following.

- The model checking problems for $L(\mathsf{F}, \mathsf{G})$ and $L(\mathsf{U})$, viz., formulae in PNF allowing (i) only the "until" U temporal operator, or (ii) only the "always" G and the "eventual" F temporal operators are, in general, undecidable even for Dual-PDS systems wherein the PDSs *do not interact at all* with each other. The above results imply that in order to get decidability for Dual-PDS systems, interacting or not, we have to restrict ourselves to either the sub-logic $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ or the sub-logic $L(\mathsf{G}, \mathsf{X})$. For these sub-logics, the decidability of model checking depends on the synchronization primitive used by the PDSs.
- For PDSs interacting via pairwise rendezvous we get the surprising result that model checking problem is decidable for the sub-logic $L(\mathsf{X}, \mathsf{G})$. In fact, we show that the decidability result extends to PDS interacting via asynchronous rendezvous and broadcasts. Regarding the other fragment, viz., $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, it is already known that the model checking problem is undecidable for both the sub-logics $L(\mathsf{F})$ and $L(\overset{\infty}{\mathsf{F}})$ (and hence for $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$) for PDSs interacting using either non-nested locks [10] or pairwise rendezvous [15]. The undecidability result for broadcasts and asynchronous rendezvous, both of which are more expressive than pairwise rendezvous, then follows. This settles the model checking problem for all the standard synchronization primitives.
- Finally, for the practically important paradigm of PDSs interacting via nested locks, we show that the model checking problem is efficiently decidable for both the sub-logics $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ and $L(\mathsf{X}, \mathsf{G})$.

The fact that the undecidability results hold even for systems with non-interacting PDSs may seem surprising at first. However, we note that allowing doubly-indexed properties (wherein atomic propositions are interpreted over pairs of control states of the PDSs comprising the given Dual-PDS system) allows us to explore precisely that portion of the state space of the given Dual-PDS system where the PDSs are coupled tightly enough to accept the intersection of the context-free languages accepted by them, thereby yielding undecidability. This is the key reason why the model checking problem for single-indexed LTL properties is robustly decidable for PDSs interacting via nested locks, while for doubly indexed properties it is decidable only for very restricted fragments that do not allow this strong coupling.

The procedure for single-index LTL properties for PDSs synchronizing via nested locks, given in [9], involves reducing the model checking problem to the computation of pre^*-closures of regular sets of configurations of the given Dual-PDS system. For single-index properties, this was accomplished via a Dual Pumping Lemma, which, unfortunately, does not hold for the double-indexed case. In fact, the undecidability of the model checking problem for doubly indexed formulae shows that such a reduction cannot exist in general. Thus model checking double-indexed LTL properties requires a different approach.

To get a model checking procedure for $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, given an automaton $\mathcal{R}_f$ accepting the set of configurations satisfying a formula f of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, we first formulate efficient procedures for computing an automaton $\mathcal{R}_{\mathsf{Op}f}$ accepting the set of all configurations that satisfy $\mathsf{Op}f$, where $\mathsf{Op} \in \{\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}}\}$ is a temporal operator that can be used in a formula of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$. Recursively applying these procedures starting from the atomic propositions and proceeding 'outwards' in the given formula f then gives us the desired model checking algorithm.

A natural question that arises is that if $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ model checking is decidable then why not full-blown double-indexed LTL model checking by reduction to the former via the automata-theoretic paradigm. The technical reason for this is discussed in the paper. However, the broad intuition is that the key obstacle to decidability of the model checking problem for systems comprised of two PDSs is the problem of deciding the disjointness of the context-free languages accepted by the PDSs. If the PDSs can be coupled together tightly enough to accept the intersection of the context free languages accepted by the PDSs, we get undecidability of the model checking problem. This tight coupling can be achieved either by making the synchronization primitive strong enough, e.g., broadcasts, or the property being model checked expressive enough, e.g., $\mathsf{L}(\mathsf{U})$. In fact, it turns out that in order to ensure decidability of the model checking problem for Dual-PDS systems, we have to restrict ourselves to properties that encode "simple" reachability or those for which the model checking problem can be reduced to such properties.

Guided by the above observations, for model checking $L(\mathsf{X}, \mathsf{G})$ we reduce the problem to a set of simple reachability problems. Towards that end, given a formula f of $L(\mathsf{X}, \mathsf{G})$, we consider the equivalent problem of model checking for $g = \neg f$. Then the positive normal form of g is a formula built using the temporal operators AX and AF, where A is the "universal path quantifier". Since AF is a 'simple reachability' property the problem is decidable, even though constructing an automaton accepting $\mathcal{R}_{\mathsf{AF}f}$ from the automaton $\mathcal{R}_f$ is more complicated due to the branching nature of the property.

For the branching-time spectrum, we consider the model checking problem for Alternation-free Mu-Calculus formulae. For lack of space, we focus only on single-index properties. For such properties, we first show that the model checking problem for PDSs communicating via nested locks is efficiently decidable. Given a Multi-PDS system $\mathcal{DP}$ comprised of the PDSs P_1,...,P_n, and a formula $\phi = \bigwedge \phi_i$, where ϕ_i is an alternation-fee Mu Calculus formula interpreted over the control states of P_i, we start by constructing the product $\mathcal{P}_i$ of P_i and $\mathcal{A}_{\phi_i}$, the Alternating Automaton for ϕ_i. Each such product $\mathcal{P}_i$ is represented as an *Alternating Pushdown System* (APDS) [1] which incorporates the branching structure of the original PDS P_i. For model checking the Multi-PDS program $\mathcal{DP}$ for ϕ, we need to compute the pre^*-closure of regular sets of global configurations of the system comprised of all the APDSs $\mathcal{P}_1, ..., \mathcal{P}_n$. The main complexity here lies in the fact that we have to reason about lock interactions along *all* paths of tree-like models of APDSs $\mathcal{P}_1, ..., \mathcal{P}_n$ each having potentially infinitely many states.

This complexity is overcome by our contribution showing how to decompose the computation of the pre^*-closure of a regular set of configurations of a Dual-PDS system $\mathcal{DP}$ synchronizing via nested locks to that of its constituent PDSs. This decomposition allows us to avoid the state explosion problem. To achieve the decomposition, we leverage the new concept of *Lock-Constrained Alternating Multi-Automata Pairs (LAMAPs)* which is used to capture regular sets of configurations of a given Multi-PDS system with nested locks. An LAMAP $\mathcal{A}$ accepting a regular set of configurations C of a Dual-PDS system $\mathcal{DP}$ comprised of PDSs P_1 and P_2 is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_i$ is an *Alternating Multi-Automata (AMA)* (see [1]) accepting the regular set of local configurations of APDS $\mathcal{P}_i$ corresponding to thread P_i occurring in the global configurations of $\mathcal{DP}$ in C.

The lock interaction among threads is encoded in the acceptance criterion for an LAMAP which filters out those pairs of local configurations of $\mathcal{P}_1$ and $\mathcal{P}_2$ which are not simultaneously reachable due to lock interaction. Indeed, for a pair of tree-like models w_1 and w_2 for ϕ_1 and ϕ_2 in the individual APDS $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, to act as a witness for $\phi = \phi_1 \wedge \phi_2$ in the Dual-PDS system $\mathcal{DP}$, they need to be *reconcilable* with respect to each other. Reconcilability means that for each path x in w_1 there must exist a path y in w_2 such that the local computations of P_1 and P_2 corresponding to x and y, respectively, can be executed in an interleaved fashion in $\mathcal{DP}$, and vice versa. For two individual paths x and y reconcilability can be decided by tracking patterns of lock acquisition along x and y. To check reconcilability of the trees w_1 and w_2, however, we need to track lock acquisition patterns along all paths of w_i in APDS $\mathcal{P}_i$. A key difficulty here is that since the depth of the tree w_i could be unbounded, the number of local paths of P_i in w_i could be unbounded forcing us to potentially track an unbounded number of acquisition lock acquisition patterns. However, the crucial observation is that since the number of locks in the Dual-PDS system $\mathcal{DP}$ is fixed, so is the number of all possible acquisition patterns. An important consequence is that instead of storing the lock acquisition patterns for each path of tree w_i, we need only store the different patterns encountered along all paths of the tree. This ensures that the set of patterns that need be tracked is finite and bounded which can therefore be carried out as part of the control state of PDS P_i. Decomposition is then achieved by showing that given an LAMAP $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, if $\mathcal{B}_i$ is an AMA accepting the pre^*-closure of the configurations of the *individual thread* P_i accepted by $\mathcal{A}_i$, then, the LAMAP $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ accepts the pre^*-closure of the regular set of configurations of the Dual-PDS system $\mathcal{DP}$ accepted by $\mathcal{A}$. Thus, broadly speaking, the decomposition results from maintaining the local configurations of the constituent PDSs separately as AMAs and computing the pre^*-closures on these AMAs individually for each PDS for which existing efficient techniques can be leveraged. This yields decidability for PDSs interacting via nested locks. For PDSs communicating via rendezvous and broadcasts, we show the decidability does not hold for the full-blown single-indexed Alternation-free Mu-Calculus but only for certain fragments.

2. System Model

In this paper, we consider multi-threaded programs wherein threads synchronize using the standard primitives - locks, pairwise rendezvous, asynchronous rendezvous and broadcasts. Each thread is modeled as a *Pushdown System (PDS)* [1]. A PDS has a finite control part corresponding to the valuation of the variables of the thread it represents and a stack which models recursion. Formally, a PDS is a five-tuple $P = (Q, Act, \Gamma, c_0, \Delta)$, where Q is a finite set of *control locations*, *Act* is a finite set of *actions*, Γ is a finite *stack alphabet*, and $\Delta \subseteq (Q \times \Gamma) \times Act \times (Q \times \Gamma^*)$ is a finite set of *transition rules*. If $((p, \gamma), a, (p', w)) \in \Delta$ then we write $\langle p, \gamma \rangle \overset{a}{\hookrightarrow} \langle p', w \rangle$. A *configuration* of P is a pair $\langle p, w \rangle$, where $p \in Q$ denotes the control location and $w \in \Gamma^*$ the *stack content*. We call $\mathbf{c}_0$ the *initial configuration* of $\mathcal{P}$. The set of all configurations of P is denoted by $\mathcal{C}$. For each action a, we define a relation $\xrightarrow{a} \subseteq \mathcal{C} \times \mathcal{C}$ as follows: if $\langle q, \gamma \rangle \overset{a}{\hookrightarrow} \langle q', w \rangle$, then $\langle q, \gamma v \rangle \xrightarrow{a} \langle q', wv \rangle$ for every $v \in \Gamma^*$.

Let $\mathcal{DP}$ be a multi-PDS system comprised of the PDSs P_1,...,P_n, where $P_i = (Q_i, Act_i, \Gamma_i, \mathbf{c}_i, \Delta_i)$. In addition to Act_i, we assume that each P_i has special actions symbols labeling transitions *implementing* synchronization primitives. These synchronizing action symbols are shared commonly across all PDSs. In this paper, we consider the following standard primitives:

- *Locks*: Locks are used to enforce mutual exclusion. Transitions acquiring and releasing lock l are labeled with $acquire(l)$ and $release(l)$, respectively.
- *Rendezvous (Wait-Notify)*: We consider two notions of rendezvous: CCS-style *Pairwise Rendezvous* and the more expressive *Asynchronous Rendezvous* motivated by the `Wait()` and `Notify()` primitives of Java. Pairwise send and receive rendezvous are labeled with $a!$ and $a?$, respectively. If $c_{11} \xrightarrow{a!} c_{12}$ and $c_{21} \xrightarrow{a?} c_{22}$ are pairwise send and receive transitions of P_1 and P_2, respectively, then for the rendezvous to be enabled both P_1 and P_2 have to simultaneously be in local control states c_{11} and c_{21}. In that case both the send and receive transitions are fired synchronously in one execution step. If P_1 is in c_{11} but P_2 is not in c_{12} then P_1 cannot execute the send transition, and vice versa. Asynchronous Rendezvous send and receive transitions, on the other hand, are labeled with $a\uparrow$ and $a\downarrow$, respectively. The difference between pairwise rendezvous and asynchronous rendezvous, is that while in the former case the send transition is blocking, in the latter it is non-blocking. Thus a transition of the form $c_{11} \xrightarrow{a\uparrow} c_{12}$ can be executed irrespective of whether a matching receive transition of the form $c_{21} \xrightarrow{a\downarrow} c_{22}$ is currently enabled or not. On the other hand, the execution of a receive transition requires a matching send transition to be enabled with both the sender and receiver then being executed synchronously.

- *Broadcasts (Notify-All)*: Broadcast send and receive rendezvous, motivated by the `Wait()` and `NotifyAll()` primitives of Java, are labeled with $a!!$ and $a??$, respectively. If $b_{11} \overset{a!!}{\rightarrow} b_{12}$ is a broadcast send transition and $b_{21} \overset{a??}{\rightarrow} b_{22}, ..., b_{n1} \overset{a??}{\rightarrow} b_{n2}$ are the matching broadcast receives, then the receive transitions block pending the enabling of the send transition. The send transitions, on the other hand, is non-blocking and can always be executed and its execution is carried out synchronously with *all* the currently enabled receive transitions labeled with $a??$.

A concurrent program with n PDSs and m locks $l_1, ..., l_m$ is formally defined as a tuple of the form $\mathcal{DP} = (P_1, ..., P_n, L_1, ..., L_m)$, where for each i, $P_i = (Q_i, Act_i, \Gamma_i, c_i, \Delta_i)$ is a pushdown system (thread), and for each j, $L_j \subseteq \{\perp, P_1, ..., P_n\}$ is the possible set of values that lock l_j can be assigned. A global configuration of $\mathcal{DP}$ is a tuple $c = (t_1, ..., t_n, l_1, ..., l_m)$ where $t_1, ..., t_n$ are, respectively, the configurations of PDSs $P_1, ..., P_n$ and $l_1, ..., l_m$ the values of the locks. If no thread holds lock l_i in configuration c, then $l_i = \perp$, else l_i is the thread currently holding it. The initial global configuration of $\mathcal{DP}$ is $(c_1, ..., c_n, \perp, ..., \perp)$, where c_i is the initial configuration of PDS P_i. Thus all locks are *free* to start with. We extend the relation $\xrightarrow{a}$ to global global configurations of $\mathcal{DP}$ in the usual way.

The reachability relation $\Rightarrow$ is the reflexive and transitive closure of the successor relation $\rightarrow$ defined above. A sequence $x = x_0, x_1, ...$ of global configurations of $\mathcal{DP}$ is a *computation* if x_0 is the initial global configuration of $\mathcal{DP}$ and for each i, $x_i \overset{a}{\rightarrow} x_{i+1}$, where either for some j, $a \in Act_j$, or for some k, $a = release(l_k)$ or $a = acquire(l_k)$ or pairwise rendezvous send $a = b!$ or receive $a = b?$, or asynchronous rendezvous send $a = b\uparrow$ or receive $a = b\downarrow$, or broadcast send $a = b!!$ or receive $a = b??$. Given a thread T_i and a reachable global configuration $\mathbf{c} = (c_1, ..., c_n, l_1, ..., l_m)$ of $\mathcal{DP}$, we use *Lock-Set*$(T_i, \mathbf{c})$ to denote the set of locks held by T_i in $\mathbf{c}$, viz., the set $\{l_j \mid l_j = T_i\}$. Also, given a thread T_i and a reachable global configuration $\mathbf{c} = (c_1, ..., c_n, l_1, ..., l_m)$ of $\mathcal{DP}$, the *projection* of $\mathbf{c}$ onto T_i, denoted by $c \downarrow T_i$, is defined to be the configuration $(c_i, l'_1, ..., l'_m)$ of the concurrent program comprised solely of the thread T_i, where $l'_i = T_i$ if $l_i = T_i$ and $\perp$, otherwise (locks not held by T_i are freed).

Multi-Automata Let $\mathcal{P} = (P, Act, \Gamma, \mathbf{c}_0, \Delta)$ be a pushdown system where $P = \{p_1, ..., p_m\}$. A $\mathcal{P}$*-multi-automaton* ($\mathcal{P}$-MA for short) is a tuple $\mathcal{A} = (\Gamma, Q, \delta, I, F)$ where Q is a finite set of states, $\delta \subseteq Q \times \Gamma \times Q$ is a set of transitions, $I = \{s_1, ..., s_m\} \subseteq Q$ is a set of initial states and $F \subseteq Q$ is a set of final states. Each initial state s_i corresponds to the control state p_i of $\mathcal{P}$.

We define the transition relation $\longrightarrow \subseteq Q \times \Gamma^* \times Q$ as the smallest relation satisfying the following:

- if $(q, \gamma, q') \in \delta$ then $q \xrightarrow{\gamma} q'$,
- $q \xrightarrow{\epsilon} q$ for every $q \in Q$, and
- if $q \xrightarrow{w} q''$ and $q'' \xrightarrow{\gamma} q'$ then $q \xrightarrow{w\gamma} q'$.

A multi-automaton can be thought of as a *data structure* that is used to succinctly represent (potentially infinite) regular sets of configurations of a given PDS. Towards that end, we say that multi-automaton $\mathcal{A}$ accepts a configuration $\langle p_i, w \rangle$ if $s_i \xrightarrow{w} q$ for some $q \in F$. The set of configurations recognized by $\mathcal{A}$ is denoted by *Conf*$(\mathcal{A})$. A set of configurations is *regular* if it is recognized by some MA.

Relative Expressive Power of Synchronization Primitives. In proving the decidability results, we will exploit the following expressiveness relations: *Pairwise Rendezvous* $<$ *Asynchronous Rendezvous* $<$ *Broadcasts*, where $<$ stands for the relation *can be simulated by* (see [7] for details).

```
Locks: a,b,c

nested() {          bar(){              non_nested(){
   acquire(a);        release(b);          acquire(b);
   acquire(b);        release(a);          acquire(a);
   bar();                                  bar();
                      acquire(c);
   release(c);                             release(c);
}                   }                   }
```

Figure 1. Nested vs. Non-nested lock access

Nested Lock Access. Additionally, we also consider the practically important case of PDSs with nested access to locks. Indeed, standard programming practice guidelines typically recommend that programs use locks in a nested fashion. In fact, in languages like Java (version 1.4) and C# locks are guaranteed to be nested. We say that a concurrent program accesses locks in a nested fashion iff along each computation of the program a thread can only release the last lock that it acquired along that computation and that has not yet been released.

As an example in figure 1, the thread comprised of procedures `nested` and `bar` accesses locks `a,b,` and `c` in a nested fashion whereas the thread comprised of procedures `non_nested` and `bar` does not. This is because calling `bar` from `non_nested` releases lock `b` before lock `a` even though lock `a` was the last one to be acquired.

Correctness Properties. The problem of model checking Dual-PDS systems for the full-blown single-index LTL was shown to be efficiently decidable in [9]. In this paper, we consider double-indexed Linear Temporal Logic (LTL) formulae. Here atomic propositions are interpreted over pairs of control states of different PDSs in the given multi-PDS system. Note that our properties do not take the stack contents of PDSs into account. This is because in dataflow analysis, the dataflow facts being tracked are usually modified only by the program statements at individual control locations. The stack is merely used to track the context, viz., the order in which functions are called in reaching the current control location.

Conventionally, $\mathcal{DP} \models f$ for a given LTL formula f if and only if f is satisfied along all paths starting at the initial state of $\mathcal{DP}$. Using path quantifiers, we may write this as $\mathcal{DP} \models \mathsf{A}f$. Equivalently, we can model check for the dual property $\neg\mathsf{A}f = \mathsf{E}\neg f = \mathsf{E}g$. Furthermore, we can assume that g is in *positive normal form (PNF)*, viz., the negations are pushed inwards as far as possible using DeMorgan's Laws: $(\neg(p \vee q)) = \neg p \wedge \neg q$, $\neg(p \vee q) = \neg p \wedge \neg q$, $\neg Fp \equiv Gq$, $\neg(pUq) \equiv G\neg q \vee \neg qU(\neg p \wedge \neg q)$.

For Dual-PDS systems, it turns out that the model checking problem is not decidable for the full-blown double-indexed LTL but only for certain fragments. Decidability hinges on the set of temporal operators that are allowed in the given property which, in turn, provides a natural way to characterize such fragments. We use $L(Op_1, ..., Op_k)$, where $Op_i \in \{\mathsf{X}, \mathsf{F}, \mathsf{U}, \mathsf{G}, \overset{\infty}{\mathsf{F}}\}$, to denote the fragment of double-indexed LTL comprised of formulae in positive normal form (where only atomic propositions are negated) built using the operators $Op_1, ..., Op_k$ and the Boolean connectives $\vee$ and $\wedge$. Here X "next-time", F "sometimes", U, "until", G "always", and $\overset{\infty}{\mathsf{F}}$ "infinitely-often" denote the standard temporal operators (see [6]). Obviously, $L(\mathsf{X}, \mathsf{U}, \mathsf{G})$ is the full-blown double-indexed LTL.

```
thread_one(){                thread_two(){

1a: lock(p);                 1b: lock(q);
2a: lock(q);                 2b: lock(r);
3a: unlock(q);               3b: unlock(r);
4a: ------ ;                 4b: ------- ;
5a: lock(r);                 5b: lock(p);
6a: unlock(r);               6b: unlock(p);
7a: ------ ;                 7b: ------- ;
8a: unlock(p);               8b: unlock(q) ;
9a: ------ ;                 9b: ------- ;

}                            }
      (a)                          (b)
```

Figure 2. Program $\mathcal{DP}$ with threads P_1(a) and P_2(b).

3. A Review of Acquisition Histories and LMAPs

The core of our decision procedure revolves around manipulating regular sets of configurations of the given Dual-PDS system $\mathcal{DP}$. A natural way to represent regular sets of configurations of a Dual-PDS system with PDSs interacting via nested locks is by using the concept of *Lock-Constrained Multi-Automata Pairs (LMAP)* introduced in [9] which we briefly review next. LMAPs allow us to not only succinctly represent potentially infinite sets of regular configurations of $\mathcal{DP}$ but, in addition, enable us to decompose pre^*-closure computations of regular sets of a Dual-PDS system $\mathcal{DP}$ to its individual PDSs thereby avoiding the state explosion problem. This is accomplished via a *Decomposition Result* that generalizes both the Forward and Backward Decomposition results as presented in [9]. Essentially, the Decomposition Result enables us to reduce the problem of deciding the reachability of one global configuration of $\mathcal{DP}$ from another to reachability problems for local configurations of the individual PDSs.

Lock-Constrained Multi-Automata. The main motivation behind defining a Lock-Constrained Multi-Automaton (LMAP) is to decompose the representation of a regular set of configurations of a Dual-PDS system $\mathcal{DP}$ comprised of PDSs P_1 and P_2 into a pair of regular sets of configurations of the individual PDSs P_1 and P_2. An LMAP accepting a regular set R of configurations of $\mathcal{DP}$ is a pair of Multi-Automata $M = (M_1, M_2)$, where M_i is a multi-automaton accepting the regular set of local configurations R_i of P_i in R. A key advantage of this decomposition is that performing operations on M, for instance computing the pre^*-closure of R reduces to performing the same operations on the individual MAs M_i. This avoids the state explosion problem thereby making our procedure efficient. The lock interaction among the PDSs is captured in the acceptance criterion for the LMAP via the concept of *Backward* and *Forward Acquisition Histories* [9] which we briefly recall next, followed by a formulation of the Decomposition Result.

Consider a concurrent program $\mathcal{DP}$ comprised of the two threads shown in figure 2. Suppose that we are interested in deciding whether a pair of control locations of the two threads are simultaneously reachable. We show that reasoning about reachability for the concurrent program, can be reduced to reasoning about reachability for the individual threads. Observe that $\mathcal{DP} \models \mathsf{EF}(\mathtt{4a} \wedge \mathtt{4b})$ but $\mathcal{DP} \not\models \mathsf{EF}(\mathtt{4a} \wedge \mathtt{7b})$ even though disjoint sets of locks, viz., $\{p\}$ and $\{q\}$, are held at `4a` and `7b`, respectively. The key point is that the simultaneous reachability of two control locations of P_1 and P_2 depends not merely on the locksets held at these locations but also on patterns of lock acquisition along the computation paths of $\mathcal{DP}$ leading to these control locations. These patterns are captured using the notions of backward and forward acquisition histories.

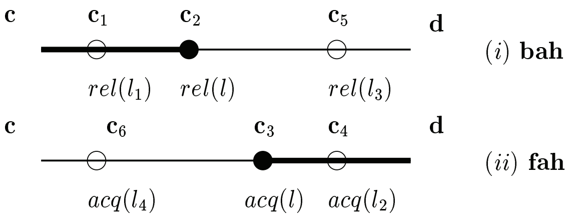

Figure 3. Forward vs. Backward Acquisition History

Indeed, if P_1 executes first it acquires `p` and does not release it along any path leading to `4a`. This prevents P_2 from acquiring `p` which it requires in order to transit from `1b` to `7b`. Similarly if P_2 executes first, it acquires `q` thereby preventing P_1 from transiting from `1a` to `4a` which requires it to acquire and release lock `q`. This creates an unresolvable cyclic dependency. These dependencies can be formally captured using the notions of backward and forward acquisition histories.

Definition (Forward Acquisition History) *For a lock l held by P_i at a control location d_i, the forward acquisition history of l along a local computation x_i of P_i leading from c_i to d_i, denoted by $\mathsf{fah}(P_i, c_i, l, x_i)$, is the set of locks that have been acquired (and possibly released) by P_i since the last acquisition of l by P_i in traversing forward along x_i from c_i to d_i. In case l is not acquired but held in each state along x_i then $\mathsf{fah}(P_i, c_i, l, x_i)$, is simply the set of locks that have been acquired (and possibly released) by P_i along x_i.*

Observe that along any local computations x_1 and x_2 of P_1 and P_2 leading to control locations `4a` and `7b`, respectively, $\mathsf{fah}(P_1, \mathtt{4a}, p, x_1) = \{q\}$ and $\mathsf{fah}(P_2, \mathtt{7b}, q, x_2) = \{p, r\}$. Also, along any local computations x_1' and x_2' of P_1 and P_2 leading to control locations `4a` and `4b`, respectively, $\mathsf{fah}(P_1, \mathtt{4a}, p, x_1') = \{q\}$ and $\mathsf{fah}(P_2, \mathtt{4b}, q, x_2') = \{r\}$. The reason $\mathsf{EF}(\mathtt{4a} \wedge \mathtt{7b})$ does not hold but $\mathsf{EF}(\mathtt{4a} \wedge \mathtt{4b})$ does is because of the existence of the cyclic dependency that $p \in \mathsf{fah}(P_2, \mathtt{7b}, q, x_2)$ and $q \in \mathsf{fah}(P_2, \mathtt{4a}, p, x_1)$ whereas no such dependency exists for the second case.

Definition (Backward Acquisition History). *For a lock l held by P_i at a control location c_i, the backward acquisition history of l along a local computation x_i of P_i leading from c_i to d_i, denoted by $\mathsf{bah}(P_i, c_i, l, x_i)$, is the set of locks that were released (and possibly acquired) by P_i since the last release of l by P_i in traversing backwards along x_i from d_i to c_i. In case l is not released but held in each state along x_i then $\mathsf{bah}(P_i, c_i, l, x_i)$, is simply the set of locks that have been released (and possibly acquired) by P_i along x_i.*

In [9], the concepts of backward and forward acquisition histories were used to decide whether given two global configurations **c** and **d** of $\mathcal{DP}$, whether **d** is reachable from **c**. The notion of forward acquisition history was used in the case where no locks are held in **c** and that of backward acquisition history in the case where no locks are held in **d**. This is illustrated in figure 3 where we want to decide whether **c** is backward reachable from **d**. First, we assume that all locks are free in **d** (case (i) in figure 3). In that case, we track the `bah` of each lock. In our example, lock l, initially held at **c**, is first released at $\mathbf{c}_2$. Then all locks released before the first release of l belongs to the `bah` of l. Thus, l_1 belongs to the `bah` of l but l_3 does not. On other hand, if in **c** all locks are free (case (ii) in figure 3), then we track the `fah` of each lock. If a lock l held at **d** is last acquired at $\mathbf{c}_3$ then all locks acquired since the last acquisition of l belong to the `fah` of l. Thus in our example, l_2 belongs to the forward acquisition history of l but l_4 does not.

When testing for backward reachability of $\mathbf{c}$ from $\mathbf{d}$ in $\mathcal{DP}$, it suffices to test whether there exist local paths x and y of the individual PDSs from states $\mathbf{c}_1 = \mathbf{c} \downarrow P_1$ to $\mathbf{d}_1 = \mathbf{d} \downarrow P_1$ and from $\mathbf{c}_2 = \mathbf{c} \downarrow P_2$ to $\mathbf{d}_2 = \mathbf{d} \downarrow P_2$, respectively, such that along x and y locks operations can be executed in a acquisition history compatible fashion as formulated in the Decomposition Result below.

Theorem 1 (Decomposition Result). *Let $\mathcal{DP}$ be a Dual-PDS system comprised of the two PDSs P_1 and P_2 with nested locks. Then configuration $\mathbf{c}$ of $\mathcal{DP}$ is backward reachable from configuration $\mathbf{d}$ iff configurations $\mathbf{c}_1 = \mathbf{c} \downarrow P_1$ of P_1 and $\mathbf{c}_2 = \mathbf{c} \downarrow P_2$ of P_2 are backward reachable from configurations $\mathbf{d}_1 = \mathbf{d} \downarrow P_1$ and $\mathbf{d}_2 = \mathbf{d} \downarrow P_2$, respectively, via local computation paths x and y of PDSs P_1 and P_2, respectively, such that*

1. *Lock-Set$(P_1, \mathbf{c}_1) \cap$ Lock-Set$(P_2, \mathbf{c}_2) = \emptyset$*
2. *Lock-Set$(P_1, \mathbf{d}_1) \cap$ Lock-Set$(P_2, \mathbf{d}_2) = \emptyset$*
3. *Locks-Acq(x) $\cap$ Locks-Held(y)$= \emptyset$ and Locks-Acq(y) $\cap$ Locks-Held(x)$= \emptyset$, where for path z, Locks-Acq(z) is the set of locks that are acquired (and possibly released) along z and Locks-Held(z) is the set of locks that are held in all states along z.*
4. *there do not exist locks $l \in$ Lock-Set$(P_1, \mathbf{c}_1) \backslash$Locks-Held(x) and $l' \in$ Lock-Set$(P_2, \mathbf{c}_2) \backslash$Locks-Held(y) such that $l \in \mathsf{bah}(P_2, \mathbf{c}_2, l', y)$ and $l' \in \mathsf{bah}(P_1, \mathbf{c}_1, l, x)$.*
5. *there do not exist locks $l \in$ Lock-Set$(P_1, \mathbf{d}_1) \backslash$Locks-Held(x) and $l' \in$ Lock-Set$(P_2, \mathbf{d}_2) \backslash$Locks-Held(y) such that $l \in \mathsf{fah}(P_2, \mathbf{c}_2, l', y)$ and $l' \in \mathsf{fah}(P_1, \mathbf{c}_1, l, x)$.*

Intuitively, conditions 1 and 2 ensure that the locks held by P_1 and P_2 in a global configuration of $\mathcal{DP}$ must be disjoint; condition 3 ensures that if a lock held by a PDS, say P_1, is not released along the entire local computation x, then it cannot be acquired by the other PDS P_2 all along its local computation y, and vice versa; and conditions 4 and 5 ensure compatibility of the acquisition histories, viz., the absence of cyclic dependencies as discussed above.

We now demonstrate that the Decomposition Result allows us to reduce the pre^*-closure computation of a regular set of configurations of a Dual-PDS system to that of its individual acquisition history augmented PDSs. Towards that end, we first need to extend existing pre^*-closure computation procedures for regular sets of configurations of a single PDS to handle regular sets of *acquisition history augmented* (ah-augmented) configurations. An acquisition history augmented configurations $\mathbf{c}_i$ of P_i is of the form $(\langle p_i, w\rangle$, $l_1, ..., l_m$, $\mathsf{bah}_1, ..., \mathsf{bah}_m$, $\mathsf{fah}_1, ..., \mathsf{fah}_m)$ where for each i, fah_i and bah_i are lock sets storing, respectively, the forward and backward acquisition history of lock l_i. Since the procedure is similar to the ones for fah and bah-augmented configurations given in [9], its formal description is omitted. The key result is the following:

Theorem 2 (ah-enhanced pre^*-computation). *Given a PDS P, and a regular set of ah-augmented configurations accepted by a multi-automaton $\mathcal{A}$, we can construct a multi-automaton $\mathcal{A}_{pre^*}$ recognizing $pre^*(Conf(\ \mathcal{A}\ ))$ in time polynomial in the sizes of $\mathcal{A}$ and the control states of P and exponential in the number of locks of P.*

Acceptance Criterion for LMAPs. The absence of cyclic dependencies encoded using bahs and fahs are used in the acceptance criterion for LMAPs to factor in lock interaction among the PDSs that prevents them from simultaneously reaching certain pairs of local configurations. Motivated by the Decomposition Theorem, we say that augmented configurations $\mathbf{c}_1 = (\langle c, w\rangle, l_1, \ldots, l_m, \mathsf{bah}_1, ..., \mathsf{bah}_m, \mathsf{fah}_1, ..., \mathsf{fah}_m)$ and $\mathbf{c}_2 = (\langle c', w'\rangle, l'_1, ..., l'_m, \mathsf{bah}'_1, ..., \mathsf{bah}'_m, \mathsf{fah}'_1, ..., \mathsf{fah}'_m)$ of P_1 and P_2, respectively, are fah-*compatible* iff there do not exist locks l_i and l_j such that $l_i = P_1$, $l'_j = P_2$, $l_i \in \mathsf{fah}'_j$ and $l_j \in \mathsf{fah}_i$. Analogously, we say that $\mathbf{c}_1$ and $\mathbf{c}_2$ are bah-*compatible* iff there do not exist locks l_i and l_j such that $l_i = P_1$, $l'_j = P_2$, $l_i \in \mathsf{bah}'_j$ and $l_j \in \mathsf{bah}_i$.

Definition 3 (LMAP Acceptance Criterion). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an* LMAP, *where $\mathcal{A}_i$ is a multi-automaton accepting ah-augmented configurations of P_i. We say that $\mathcal{A}$ accepts global configuration $(\langle p_i, w\rangle, \langle q_j, v\rangle, l_1, ..., l_m)$ of $\mathcal{DP}$ iff there exist there exist augmented local configurations $\mathbf{c}_1 = (\langle p_i, w\rangle, l'_1, ..., l'_m, \mathsf{bah}_1, ..., \mathsf{bah}_m, \mathsf{fah}_1, ..., \mathsf{fah}_m)$ and $\mathbf{c}_2 = (\langle q_j, v\rangle, l''_1, ..., l''_m, \mathsf{bah}'_1, ..., \mathsf{bah}'_m, \mathsf{fah}'_1, ..., \mathsf{fah}'_m)$, where $l'_i = P_1$ if $l_i = P_1$ and $\bot$ otherwise and $l''_i = P_2$ if $l_i = P_2$ and $\bot$ otherwise, such that*

1. *$\mathcal{A}_i$ accepts $\mathbf{c}_i$, and*
2. Lock-Set$(P_1, \mathbf{c}_1) \cap$ Lock-Set$(P_2, \mathbf{c}_2) = \emptyset$*, and*
3. *$\mathbf{c}_1$ and $\mathbf{c}_2$ are bah-compatible and fah-compatible.*

Given an *LMAP* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we use $Conf(\mathcal{A})$ to denote the set of configurations of $\mathcal{DP}$ accepted by $\mathcal{A}$. Let LMAP $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, where $\mathcal{B}_i$ is the MA accepting $pre^*(Conf(\mathcal{A}_i))$ constructed from $\mathcal{A}_i$ using an ah-augmented pre^*-closure computation procedure similar to the one for bah and fah-augmented configurations given in [9]. Then corollary 4 follows easily from the Decomposition Result which when combined with theorem 2 leads to the efficient pre^*-closure computation result formulated in theorem 5 below.

Corollary 4 (pre^*-closure Decomposition). $pre^*(Conf(\mathcal{A})) = Conf(\mathcal{B})$.

Theorem 5 (pre^*-closure Computation). *Let $\mathcal{A}$ be an LMAP. Then we can construct an LMAP accepting $pre^*(Conf(\mathcal{A})$ in polynomial time in the size of $\mathcal{DP}$ and exponential time in the number of locks of $\mathcal{DP}$.*

4. Nested Locks: The Model Checking Procedure for $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$

We start by presenting the decision procedures for model checking $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ and $L(\mathsf{X}, \mathsf{G})$ for PDSs interacting via nested locks. The model checking problem for single-indexed LTL\X formulae interpreted over finite paths was considered in [10] and over infinite paths in [9]. The procedures for single-index properties involves reducing the model checking problem to the computation of pre^*-closures of regular sets of configurations of the given Dual-PDS system. For single-index properties, this was accomplished via a Dual Pumping Lemma, which, unfortunately, does not hold for the double-indexed case. Therefore, as discussed in the introduction, model checking multi-indexed formulae requires a different approach.

Given an LMAP $\mathcal{R}_f$ accepting the set of configurations satisfying a formula f of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, we give for each temporal operator $\mathsf{Op} \in \{\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}}\}$, a procedure for computing an LMAP $\mathcal{R}_{\mathsf{Op}f}$ accepting the set of all configurations that satisfy $\mathsf{Op}f$. Recursively applying these procedures starting from the atomic propositions and proceeding 'outwards' in the given formula f then gives us the desired model checking algorithm.

The construction of LMAP $\mathcal{R}_{\mathsf{Op}f}$ for the case where $\mathsf{Op} = \mathsf{F}$, was given in [9] as were the constructions for the boolean connectives $\wedge$ and $\vee$. We now show how to handle the cases where $\mathsf{Op} \in \{\overset{\infty}{\mathsf{F}}, \mathsf{X}\}$ starting with the case where $\mathsf{Op} = \overset{\infty}{\mathsf{F}}$. Given an LMAP $\mathcal{R}_f$, constructing the LMAP $\mathcal{R}_{\overset{\infty}{\mathsf{F}}f}$ accepting the set of configurations of $\mathcal{DP}$ satisfying $\overset{\infty}{\mathsf{F}}\ f$ is, in general, not possible. Indeed, that would make the model checking problem for the full-blown doubly-indexed LTL decidable, contrary to the undecidability results in section 7 which show that $\mathcal{R}_{\overset{\infty}{\mathsf{F}}f}$ can be constructed only for the case where f is a $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ formula. However, even here the construction becomes intricate when f is an arbitrary $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ formula. To simplify the procedure, we first show that given a formula f of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, we can drive down the $\overset{\infty}{\mathsf{F}}$ operator so that it quantifies only over atomic propositions or negations thereof. Then it suffices to construct an LMAP $\mathcal{R}_{\overset{\infty}{\mathsf{F}} f}$ only for the case where f is a (doubly-indexed) atomic proposition or negation thereof which can be accomplished elegantly. Formally, we show the following.

Proposition 6 (Driving down the $\overset{\infty}{\mathsf{F}}$ operator). *For any formula f of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$, we can construct a formula f' where the temporal operator $\overset{\infty}{\mathsf{F}}$ quantifies only over atomic propositions or negations thereof such that $\mathcal{DP} \models f$ iff $\mathcal{DP} \models f'$.*

Constructing an LMAP for $\overset{\infty}{\mathsf{F}}$ f. Note that if f is an atomic proposition or negation thereof, $\mathcal{R}_f$ accepts the set $\{c_1\} \times \Gamma_1^* \times \{c_2\} \times \Gamma_2^*$, where $(c_1, c_2) \in C_f \subseteq Q_1 \times Q_2$ is the finite set of pairs of control states of P_1 and P_2 satisfying f.

We start by proving an $\overset{\infty}{\mathsf{F}}$-Reduction Result that allows us to reduce the construction of an LMAP accepting $\mathcal{R}_{\overset{\infty}{\mathsf{F}}}$ to multiple instances of pre^*-closure computations for which theorem 5 can be leveraged. Let $\mathcal{DP}$ be a Dual-PDS system comprised of PDSs P_1 and P_2 and f an atomic proposition, or negation thereof, over the control states of P_1 and P_2. The key idea is to show that a configuration $\mathbf{c} \in \mathcal{R}_{\overset{\infty}{\mathsf{F}} f}$ iff there is a finite path leading to a pumpable cycle containing a configuration $\mathbf{g}$ satisfying f. For a finite state system, a pumpable cycle is a finite path starting and ending in the same state with $\mathbf{g}$ occurring along it. For a PDS which has infinitely many states due to the presence of an unbounded stack, the notion of pump-ability is different. We say that a PDS is pumpable along a finite path x if executing x returns the PDS to the same control location and the same symbol on top of its stack as it started with, without popping any symbol not at the top of the stack to start with. This allows us to execute the sequence of transitions along x back-to-back indefinitely. In a Dual-PDS system, since we have two stacks, the pumping sequence of the individual PDSs can be staggered with respect to each other. More formally, let $\mathcal{DP}$ be a Dual-PDS system comprised of the PDSs $P_1 = (Q_1, Act_1, \Gamma_1, \mathbf{c}_1, \Delta_1)$ and $P_2 = (Q_2, Act_2, \Gamma_2, \mathbf{c}_2, \Delta_2)$ and f an atomic proposition or negation thereof. Then we can show the following.

Theorem 7 ($\overset{\infty}{\mathsf{F}}$-Reduction Result) *Dual-PDS system $\mathcal{DP}$ has a run satisfying $\overset{\infty}{\mathsf{F}}$ f starting from an initial configuration c if and only if there exist $\alpha \in \Gamma_1, \beta \in \Gamma_2$; $u \in \Gamma_1^*, v \in \Gamma_2^*$; a configuration g satisfying f; configurations lf_1 and lf_2 in which all locks are free; lock values $l_1, ..., l_m, l_1', ..., l_m'$; control states $p', p''' \in P_1$, $q', q'' \in P_2$; $u', u'', u''' \in \Gamma_1^*$; and $v', v'', v''' \in \Gamma_2^*$ satisfying the following conditions*

1. $c \Rightarrow (\langle p, \alpha u\rangle, \langle q', v'\rangle, l_1, ..., l_m)$
2. $(\langle p, \alpha\rangle, \langle q', v'\rangle, l_1, ..., l_m) \Rightarrow (\langle p', u'\rangle, \langle q, \beta v\rangle, l_1', ..., l_m')$
3. $(\langle p', u'\rangle, \langle q, \beta\rangle, l_1', ..., l_m')$
   $\Rightarrow lf_1 \Rightarrow g \Rightarrow lf_2$
   $\Rightarrow (\langle p, \alpha u''\rangle, \langle q'', v''\rangle, l_1, ..., l_m)$
   $\Rightarrow (\langle p''', u'''\rangle, \langle q, \beta v'''\rangle, l_1', ..., l_m')$

Intuitively, PDS P_1 can be pumped by executing the stem, viz., the sequence $c \Rightarrow (\langle p, \alpha u\rangle, \langle q', v'\rangle, l_1, ..., l_m)$ followed by repeatedly executing the *pumpable control cycle*, viz., the local sequence x_{11} of P_1 along $(\langle p, \alpha\rangle, \langle q', v'\rangle, l_1, ..., l_m) \Rightarrow (\langle p', u'\rangle, \langle q, \beta v\rangle, l_1', ..., l_m')$ followed by the local sequence x_{12} of P_1 along $(\langle p', u'\rangle, \langle q, \beta\rangle, l_1', ..., l_m') \Rightarrow lf_1 \Rightarrow g \Rightarrow lf_2 \Rightarrow (\langle p, \alpha u''\rangle, \langle q'', v''\rangle, l_1, ..., l_m)$ back-to-back indefinitely. Note that executing the transitions along $x_{11}x_{12}$ returns P_1 to control location p with α at the top of its stack so that it can be executed again and again. Analogously, PDS P_2 can be pumped by repeatedly executing the local sequences x_{21} of P_2 along $(\langle p', u'\rangle, \langle q, \beta\rangle, l_1', ..., l_m') \Rightarrow lf_1 \Rightarrow g \Rightarrow lf_2 \Rightarrow (\langle p, \alpha u''\rangle, \langle q'', v''\rangle, l_1, ..., l_m) \Rightarrow (\langle p''', u'''\rangle, \langle q, \beta v'''\rangle, l_1', ..., l_m')$ back-to-back. Note that executing the transitions along $x_{21}x_{22}$ returns P_2 to control location q with β at the top of its stack so that it can be pumped. We need to ensure, however, that the transitions of P_1 along $x_{11}x_{12}$ can executed in an interleaved fashion with the transitions of P_2 along $x_{21}x_{22}$. The lock-free states lf_1 and lf_2 states ensure that such a scheduling exists. Finally, the sequence $(\langle p, \alpha\rangle, \langle q', v'\rangle, l_1, ..., l_m) \Rightarrow (\langle p', u'\rangle, \langle q, \beta v\rangle, l_1', ..., l_m')$ represents the stagger between the pumping sequences of the two PDSs.

Why the Reduction to pre^*-closure Computations does not work in general. A question that arises here is why can we not reduce the model checking problem for an arbitrary double-indexed LTL formula f to the computation of pre^*-closures as above. Indeed, using the automata theoretic approach to model checking, one can first construct the product $\mathcal{BP}$ of $\mathcal{DP}$ and the Büchi Automaton for f. Then model checking for f simply reduces to checking whether there exists a path along which a final state of $\mathcal{BP}$ occurs infinitely often, viz, $g = \overset{\infty}{\mathsf{F}}$ $green$ holds, where $green$ is an atomic proposition characterizing the set of final states of $\mathcal{BP}$. Note that g is a formula of $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$. It turns out that for the general case, the $\Rightarrow$ direction of the above result holds but not the $\Leftarrow$ direction. Indeed, in order for $(\Leftarrow)$ to hold, we have to decide whether we can construct an accepting sequence of $\mathcal{BP}$ by appropriately scheduling the local transitions of PDSs P_1 and P_2 occurring along the finite sequences satisfying conditions 1,2, and 3 of the $\overset{\infty}{\mathsf{F}}$-Reduction result. However, the key point is that this scheduling must respect the constraints imposed by the Büchi Automaton for f. All that the Decomposition result allows us to decide is whether a global configuration $\mathbf{c}$ is reachable from another global configuration $\mathbf{d}$ of $\mathcal{DP}$, viz., whether a scheduling of the local transitions exists that enables $\mathcal{DP}$ to reach $\mathbf{d}$ from $\mathbf{c}$. However it does not guarantee that this scheduling will not violate the Büchi constraints for f. Indeed, as was discussed in the introduction, it is, in general, undecidable whether a scheduling satisfying given Büchi constraints exists.

Reduction to the Computation of pre^*-closures. The $\overset{\infty}{\mathsf{F}}$-Reduction result allows us to reduce the computation of an LMAP accepting $\overset{\infty}{\mathsf{F}}$ f to the computation of pre^*-closures as given by the following encoding of the conditions of the above theorem.

Let $R_0 = pre^*(\{p\} \times \alpha\Gamma_1^* \times P_2 \times \Gamma_2^* \times \{(l_1, ..., l_m)\})$
Then condition 1 can be re-written as $c \in R_0$. Similarly, if
$R_1 = P_1 \times \Gamma_1^* \times \{q\} \times \beta\Gamma_2^* \times \{(l_1', ..., l_m')\}$
$R_2 = pre^*(R_1) \cap \{p\} \times \{\alpha\} \times P_2 \times \Gamma_2^* \times \{(l_1, ..., l_m)\}$
then condition 2 can be captured as $R_2 \neq \emptyset$. Finally, let
$R_3 = P_1 \times \Gamma_1^* \times \{q\} \times \beta\Gamma_2^* \times \{(l_1', ..., l_m')\}$
$R_4 = pre^*(R_3) \cap \{p\} \times \alpha\Gamma_1^* \times P_2 \times \Gamma_2^* \times \{(l_1, ..., l_m)\}$
$R_5 = pre^*(R_4) \cap P_1 \times \Gamma_1^* \times P_2 \times \Gamma_2^* \times \{(\bot, ..., \bot)\}$,
$R_6 = pre^*(R_5) \cap G \times L_1 \times ... \times L_m$, *where* $G = \bigcup_{(g_1, g_2)}(\{g_1\} \times \Gamma_1^* \times \{g_2\} \times \Gamma_2^*)$ *with* (g_1, g_2) *being a control state pair of* $\mathcal{DP}$ *satisfying* f,
$R_7 = pre^*(R_6) \cap P_1 \times \Gamma_1^* \times P_2 \times \Gamma_2^* \times \{(\bot, ..., \bot)\}$,
$R_8 = pre^*(R_7) \cap P_1 \times \Gamma_1^* \times \{q\} \times \{\beta\} \times \{l_1', ..., l_m'\}$.

Then condition 3 is equivalent to $R_8 \neq \emptyset$. Thus the LMAP $R_0 \cap R_2 \cap R_8$ accepts $\overset{\infty}{\mathsf{F}}$ f. Note that we need take the union of this

LMAP for each possible value of the tuple $(\alpha, \beta, l_1, ..., l_m)$. As a consequence of the above encoding and the the fact that pre^*-closures, unions and intersections of LMAPs can be computed efficiently (theorem 5), we have the following.

Theorem 8. *Let $\mathcal{DP}$ be a Dual-PDS system synchronizing via nested locks and f a boolean combination of atomic propositions. Then given an LMAP $\mathcal{R}_f$ accepting a regular set of configurations of $\mathcal{DP}$, we can construct an LMAP $\mathcal{R}_{\overset{\infty}{\mathsf{F}} f}$ accepting the set of configurations satisfying $\overset{\infty}{\mathsf{F}}$ f in polynomial time in the size control states of $\mathcal{DP}$ and exponential time in the number of locks.*

Constructing an LMAP for $\mathsf{X}f$. Given an LMAP $\mathcal{R}_f$ accepting the set of configurations satisfying f, the LMAP $\mathcal{R}_{\mathsf{X}f}$, due to interleaving semantics, is the disjunction over i of the LMAPs $\mathcal{R}_{\mathsf{X}_i f}$, where $\mathcal{R}_{\mathsf{X}_i f}$, is the LMAP accepting the pre-image of $Conf(\mathcal{R}_f)$ in PDS P_i. The construction of LMAP $\mathcal{R}_{\mathsf{X}_i f}$ is the same as the one carried out in each step of the pre^*-closure computation procedure for an individual ah-enhanced PDS.

This completes the construction of the LMAP $\mathcal{R}_{Opf}$ for each of the operators $\mathsf{Op} \in \{\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}}\}$ leading to the following decidability result.

Theorem 9 ($L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$-decidability). *The model checking problem for $L(\mathsf{X}, \mathsf{F}, \overset{\infty}{\mathsf{F}})$ is decidable for PDSs interacting via nested lock in time polynomial in the sets of control states of the given Dual-PDS system and exponential time in the number of locks.*

5. Nested Locks: The Model Checking Procedure for $L(\mathsf{X}, \mathsf{G})$.

Let f be a formula of $L(\mathsf{X}, \mathsf{G})$. We formulate a decision procedure for the equivalent problem of model checking $\mathcal{DP}$ for $\neg f$. Since f is a formula of $L(\mathsf{X}, \mathsf{G})$, the positive normal form of its negation $\neg f$ is a formula built using the operations AF, AX, $\vee$, $\wedge$ and atomic propositions or negations thereof interpreted over the control states of the PDSs constituting the given Dual-PDS system. Given the formula $g = \neg f$, we proceed as follows:

1. For each sub-formula p of g that is either an atomic proposition, or negation thereof, we construct an LMAP representing the set of regular configurations satisfying p.

2. Next, given an LMAP accepting the set of configurations satisfying a sub-formula h of g, we give procedures for computing LMAPs accepting the regular set of configurations satisfying $\mathsf{AX}h$ and $\mathsf{AF}h$. Leveraging these procedures and the closure properties of LMAPs under $\wedge$ and $\vee$ then gives us a procedure to construct an LMAP $\mathcal{M}_{\neg f}$ accepting the set of configurations satisfying $\neg f$.

3. In the final step, all we do is check whether the initial configuration of $\mathcal{DP}$ is accepted by $\mathcal{M}_{\neg f}$.

Computing the LMAP accepting $\mathsf{AF}g$. We next present a novel way to represent regular sets of configurations of a Dual-PDS system that enables us to compute the regular set of configurations accepting $\mathsf{AF}g$. To motivate our procedure, we recall the one for model checking $\mathsf{EF}g$. Our over-arching goal there was to reduce global reasoning about a Dual-PDS system $\mathcal{DP}$ for $\mathsf{EF}g$ (and other linear-time temporal properties) to local reasoning about the individual PDSs. This was accomplished by using the machinery of acquisition histories. Here, in order to test whether $\mathbf{c} \models \mathsf{EF}g$, viz., there is a global path of $\mathcal{DP}$ starting at $\mathbf{c}$ and leading to a configuration $\mathbf{d}$ satisfying g, it is sufficient to test, whether for each i, there exists a local path x_i leading from $\mathbf{c}_i$ to $\mathbf{d}_i$, the local configurations of $\mathbf{c}$ and $\mathbf{d}$ in P_i, respectively, such that x_1 and x_2 are *acquisition-history compatible*. Towards that end, we augmented the local configuration of each individual PDS P_i with acquisition history information with respect to path x_i and represented regular sets of configurations of $\mathcal{DP}$ as a single LMAP $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$, where each $\mathcal{M}_i$ accepts only those pair of augmented configurations of P_1 and P_2 that are acquisition history compatible. Then, the Decomposition Result allowed us to reduce the problem of deciding whether $\mathbf{c} \models \mathsf{EF}g$, to deciding, whether there existed ah-augmented configurations accepted by $\mathcal{M}_1$ and $\mathcal{M}_2$ with P_1 and P_2 in local configurations $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively, that are acquisition history compatible.

When deciding whether $\mathbf{c} \models \mathsf{AF}g$, our goal remains the same, i.e., to reduce reasoning about the given Dual-PDS system to its individual constituent PDSs. In this case, however, we have to check whether *all* paths starting at $\mathbf{c}$ lead to a configuration satisfying g. The set of all such paths is now a tree T rooted at $\mathbf{c}$ with all its leaves comprised of configurations satisfying g. Analogous to the $\mathsf{EF}g$ case, we consider for each i, the local tree T_i resulting from the projection of each global path x of T onto the local computation of P_i along x. Note that due to lock interaction not all paths from the root to a leaf of T_1 can be executed in an interleaved fashion with a similar path of T_2, and vice versa. Enumerating all pairs of compatible local paths of T_1 and T_2 and executing them in all allowed interleavings will give us back the tree T. But from the Decomposition Result, we have that a pair of local paths of T_1 and T_2 can executed in an interleaved fashion if and only if they are acquisition history compatible. In other words, to reconstruct T from T_1 and T_2, we have to track the acquisition history along each path starting from the root of T_i to its leaves.

A key difficulty is that since the depth of tree T_i could be unbounded, the number of local paths of P_i from $\mathbf{c}_i$ in T_i could be unbounded forcing us to potentially track an unbounded number of acquisition histories. However, the crucial observation is that since the number of locks in the Dual-PDS system $\mathcal{DP}$ is fixed, viz., m, so is the number of all possible acquisition histories ($2^{m(2m+3)}$ in fact). An important consequence is that instead of storing the acquisition history for each path of tree T_i, we need only store the different acquisition histories encountered along all paths of the tree. This ensures that the set of acquisition histories that need be tracked is finite and bounded and can therefore be tracked as part of the control state of PDS P_i. Thus an augmented control state of P_i is now of the form $(c_i, \mathcal{AH})$, where $\mathcal{AH} = \{\mathsf{ah}_1, ..., \mathsf{ah}_k\}$ is a set of acquisition history tuples and c_i is a control state of P_i. Each acquisition history tuple ah_j is of the form $(\mathsf{bah}_1, ..., \mathsf{bah}_m, \mathsf{fah}_1, ..., \mathsf{fah}_m)$, where bah_j and fah_j track, respectively, the backward and forward acquisition history tuples of lock l_j.

For model checking $\mathsf{AF}p$, we make use of a pre^*-closure algorithm based on the fixpoint characterization $AFf = \mu Y.f \vee AXY$. Note that since the given Dual-PDS system has infinitely many states, by naively applying the above procedure we are not guaranteed to terminate. We now propose a novel way to perform pre^*-closures over regular sets of configurations augmented with acquisition history sets. These regular sets are now represented as LMAPs accepting configurations augmented with acquisition history sets instead of merely acquisition histories and are thus referred to as *Augmented LMAPS (A-LMAPs)*. Using A-LMAPs enables us to carry out this pre^*-closure efficiently.

Computing $\mathsf{AF}f$. Let $\mathcal{M}_0 = (\mathcal{M}_{01}, \mathcal{M}_{02})$ be an A-LMAP accepting the set of regular configurations satisfying f. To start with, the fah and bah entries of each acquisition tuple are set to $\emptyset$. Starting at $\mathcal{M}_0$, we construct a finite series of A-LMAPs $\mathcal{M}_0, ..., \mathcal{M}_p$ resulting in the A-LMAP $\mathcal{M}_p$ accepting the set of configurations satisfying $\mathsf{AF}f$. We denote by $\rightarrow_k$ the transition relation of $\mathcal{M}_k$. Then for every $k \geq 0$, $\mathcal{M}_{k+1}$ is obtained from $\mathcal{M}_k$ by conserving the set of states and adding new transitions as follows: Let $\mathbf{c_1} = (c_1, \mathcal{AH}_1)$ and $\mathbf{c_2} = (c_2, \mathcal{AH}_2)$ be a pair of acquisition-history compatible augmented control configurations of P_1 and P_2, respectively. We need to check whether all enabled successors from the augmented control state $(\mathbf{c_1}, \mathbf{c_2})$ of $\mathcal{DP}$ are accepted by $\mathcal{M}_k$. Let tr_{i1},..., tr_{il_i} be all the local transitions of P_i that can be fired by $\mathcal{DP}$ from $(\mathbf{c_1}, \mathbf{c_2})$. We check for each transition $tr_{ij} : c_i \rightarrow c_{ij}$, whether one of the following conditions holds

- if tr_{ij} is an internal transition of P_i of the form $\langle c_i, \gamma\rangle \hookrightarrow \langle c_{ij}, u\rangle$ then in $\mathcal{M}_k = (\mathcal{M}_{k1}, \mathcal{M}_{k2})$, for some $w \in \Gamma_i^*$, the augmented configuration $\langle (c_{ij}, \mathcal{AH}_i), uw\rangle$ is accepted by $\mathcal{M}_{ki}$, viz., there is a path x of $\mathcal{M}_{ki}$ starting at $(c_{ij}, \mathcal{AH}_i)$ and leading to a final state of $\mathcal{M}_{ki}$ such that there is a prefix x' of x leading from $(c_{ij}, \mathcal{AH}_i)$ to a state $(c'_{ij}, \mathcal{AH}_i)$, say, such that x' is labeled with u.
- if tr_{ij} is the locking transition $c_i \overset{acquire(l)}{\hookrightarrow} c_{ij}$, then there is an accepting path in $\mathcal{M}_{ki}$ starting at $(c_{ij}, \mathcal{AH}'_i)$, where $\mathcal{AH}'_i = \{\mathsf{ah}'_{im} | \mathsf{ah}_{im} \in \mathcal{AH}_i\}$ with ah_{im} obtained from ah'_{im} by factoring in the acquisition of l. We remove the fah for lock l (since in the backward step l has now been released) and add l to the fah of every other lock. Similarly l is added to the bah of every other lock initially held by P_i when starting from $\mathcal{M}_0$ and the bah of l is dropped if it was not held initially when starting from $\mathcal{M}_0$.
- if tr_{ij} is the unlocking transition $c_i \overset{release(l)}{\hookrightarrow} c_{ij}$, then the there is an accepting path in $\mathcal{M}_{ki}$ starting at $(c_{ij}, \mathcal{AH}'_i)$, where $\mathcal{AH}'_i = \{\mathsf{ah}'_{im} | \mathsf{ah}_{im} \in \mathcal{AH}_i\}$ with ah_{im} obtained from ah'_{im} by adding an empty FAH-entry, for l (since in the backward step l has now been acquired).

If for each i, at least one of the above conditions holds for each of the transitions $tr_{i1}, ..., tr_{il_i}$, then in $\mathcal{M}_{(k+1)i}$, for each tr_{ij}, we add an internal transition from state $\mathbf{c_i}$ in $\mathcal{M}_i$ to $\mathbf{c'_i} = (\mathbf{c_{ij}}, \mathcal{AH}'_i)$ or a transition labeled with γ to the state $\mathbf{c'_i} = (\mathbf{c'_{ij}}, \mathcal{AH}_\mathbf{i})$ accordingly as tr_{ij} is a locking\unlocking transition or an internal transition, respectively. Since in constructing $\mathcal{M}_{k+1}$ from $\mathcal{M}_k$, we add a new transition to $\mathcal{M}_k$ but conserve the state set, we are guaranteed to reach a fixpoint. Then the resulting A-LMAP accepts $\mathsf{AF}f$.

Computing the LMAP accepting $\mathsf{AX}f$. The construction is similar to the one carried out in each step of the above procedure for computing $\mathsf{AF}f$ and is therefore omitted.

This completes the formulation of procedures for computing A-LMAPs accepting $\mathsf{AX}f$ and $\mathsf{AF}f$ leading to the following result.

Theorem 10 ($L(\mathsf{X}, \mathsf{G})$ Decidability). *The model checking problem for Dual-PDS system $\mathcal{DP}$ synchronizing via nested locks is decidable for $L(\mathsf{X}, \mathsf{G})$ in polynomial time in the size of the control state of $\mathcal{DP}$ and exponential time in the number of locks.*

6. Rendezvous and Broadcasts: Decidability of $L(\mathsf{X}, \mathsf{G})$

In order to get decidability of $L(\mathsf{X}, \mathsf{G})$ for PDSs interacting via asynchronous rendezvous, we reduce the problem to the model checking problem for non-interacting Dual-PDS systems for $L(\mathsf{X}, \mathsf{G})$ which by theorem 10 is decidable.

Let $\mathcal{DP}$ be a given Dual-PDS system comprised of PDSs P_1 and P_2 interacting via asynchronous rendezvous. Let P'_i be the PDS we get from P_i by replacing each asynchronous send transition of the form $c_{11} \overset{a\uparrow}{\rightarrow} c_{12}$ by the sequence of internal transitions $c_{11} \rightarrow c_{a\uparrow} \rightarrow c_{12}$. Similarly, each asynchronous receive transition of the form $c_{21} \overset{a\downarrow}{\rightarrow} c_{22}$ is replaced by the sequence of internal transitions $c_{21} \rightarrow c_{a\downarrow} \rightarrow c_{22}$. Note that P'_i has no rendezvous transitions. Let $\mathcal{DP}'$ be the Dual-PDS systems comprised of the non-interacting PDSs P'_1 and P'_2. The key idea is to construct a formula f_{AR} of $L(\mathsf{X}, \mathsf{G})$ such that $\mathcal{DP} \models \mathsf{E}f$ iff $\mathcal{DP}' \models \mathsf{E}(f \wedge f_{AR})$. The decidability of model checking for $L(\mathsf{X}, \mathsf{G})$ then follows from theorem 10.

In order to simulate asynchronous rendezvous, we have to ensure that (i) whenever P'_1 and P'_2 are in control states c_{11} and c_{21}, respectively, then we execute the transitions $c_{21} \rightarrow c_{a\downarrow}$, $c_{11} \rightarrow c_{a\uparrow}$, $c_{a\downarrow} \rightarrow c_{22}$ and $c_{a\uparrow} \rightarrow c_{12}$ back-to-back, and (ii) whenever P'_2 is in local state c_{21} but P'_1 is not in c_{11}, PDS P'_2 blocks. This can be accomplished by ensuring that each state satisfies $f'_{AR} = ((c_{21} \wedge \neg c_{11} \Rightarrow \mathsf{X}c_{21}) \wedge ((c_{11} \wedge c_{12}) \Rightarrow \mathsf{X}(c_{11} \wedge a_\downarrow) \wedge \mathsf{XX}(a_\uparrow \wedge a_\downarrow) \wedge \mathsf{XXX})(c_{22} \wedge a_\uparrow) \wedge \mathsf{XXXX}(c_{22} \wedge c_{12}))$. Thus we set $f_{AR} = \mathsf{G}f'_{AR}$. The case for broadcasts can be handled similarly. Since pairwise rendezvous is less expressive than asynchronous rendezvous, the decidability of $L(\mathsf{X}, \mathsf{G})$ for pairwise rendezvous follows.

Theorem 11 ($L(\mathsf{X}, \mathsf{G})$ Decidability). *The model checking problem for Dual-PDS system interacting via rendezvous and broadcasts are decidable for $L(\mathsf{X}, \mathsf{G})$.*

7. Decidability Limits for Model Checking Multi-PDS Systems

While it was shown in [9], that the problem of model checking Multi-PDS systems interacting via nested locks for single-index LTL\X properties is efficiently decidable, the problem is not as robustly decidable for double-index properties. In turns out that even when the PDSs in a Dual-PDS system do not interact with each other, the model checking problem for double-indexed LTL becomes undecidable. In fact, we show the much stronger results that the problem is undecidable even for the following restricted fragments of double-indexed LTL: (i) $L(\mathsf{G}, \mathsf{F})$ and (ii) $L(\mathsf{U})$.

The Undecidability Results. We show both the undecidability results by reduction from the problem of deciding the disjointness of the context-free languages accepted by two given Pushdown Automata (PDA). Recall that the language accepted by a PDA $P = (P, Act, \Gamma, c_0, \Delta)$, denoted by $L(P)$, is the set of all words $w \in \Gamma^*$ such that there is a valid path of P labeled with w leading from the initial to a final control state of P. In order to encode the testing of $L(P_1) \cap L(P_2) = \emptyset$, as a model checking problem, we need to make sure that every execution of a transition tr_1 of P_1 labeled with an action symbol a is matched by an execution of a transition tr_2 of P_2 also labeled with a that immediately follows the execution of tr_1. Then testing whether $L(P_1) \cap L(P_2) = \emptyset$, reduces to deciding whether there exists a reachable global configuration of $\mathcal{DP}$ with both P_1 and P_2 in final local states.

Let $\mathcal{DP}$ be the Dual-PDS system comprised of PDSs P_1 and P_2. We construct a formula f of $L(\mathsf{G}, \mathsf{F})$ such that $\mathcal{DP} \models f$ iff $L(P_1) \cap L(P_2) = \emptyset$. Since testing the disjointness of the context-free languages accepted by two PDSs is undecidable, the undecidability of the model checking problem for $L(G, F)$ follows.

Undecidability of Model Checking $L(\mathsf{G}, \mathsf{F})$. We construct a formula f_I comprised of the "always" operator G such that $\mathcal{DP}$ satisfies f_I only along those paths where execution of transitions of the two PDSs P_1 and P_2 labeled with the same symbol a can only execute back-to-back.

Towards that end, let $tr_1 : c_1 \xrightarrow{a} d_1$ and $tr_2 : c_2 \xrightarrow{a} d_2$ be a pair of transitions of P_1 and P_2, respectively, both labeled with a. In each PDS P_i, we introduce new local control states c^a_{i1} and c^a_{i2} and the new transitions $tr_{i1} : c_i \xrightarrow{\epsilon} \mathsf{c}^a_{i1}$, $tr_{i2} : \mathsf{c}^a_{i1} \xrightarrow{a} \mathsf{c}^a_{i2}$ and $tr_{i3} : \mathsf{c}^a_{i2} \xrightarrow{\epsilon} d_i$. Then to simulate the synchronization of the firing of tr_1 and tr_2, we impose the condition that the transitions tr_{12} and tr_{22} are always fired back-to-back. This can be ensured by requiring that the formula $f_I = (c^a_{21} \Rightarrow c_1 \vee c^a_{11} \vee c^a_{12}) \wedge (c^a_{11} \Rightarrow$

$(c_2 \vee c^a_{21})) \wedge (c^a_{12} \Rightarrow c^a_{21} \vee c^a_{22}) \wedge (d_1 \Rightarrow \neg c^a_{21})$ is satisfied in each global configuration along a computation. Thus along every path satisfying $\mathsf{G}f_I$ the execution of each transition of one PDS is matched by the execution of a transition of the other PDS labeled with the same action. Then $L(P_i) \cap L(P_2) = \emptyset$ iff $\mathcal{DP} \models f_I \bigwedge f_F$, where $f_F = \vee_{(f_i,f_j)\in F_1\times F_2}(\mathsf{F}(f_i \wedge f_j))$, ensures that both PDSs simultaneously reach a final state. This gives us the following result.

Theorem 12. *The model checking problem for a system comprised of two non-interacting PDS is undecidable for the logic $L(\mathsf{G},\mathsf{F})$.*

Disjointness of Context-Free Languages via $L(\mathsf{U})$ formulae. In this case, all we need to ensure is that the formula f_I holds only at each state along a path leading to a global configuration with both P_1 and P_2 in final states. Note that once P_1 and P_2 are both in final states, we need no longer check that f_I is satisfied. These condition can be captured by the formula $f_I \mathsf{U} f_F$. Thus, $L(P_1) \cap L(P_2) = \emptyset$ iff $\mathcal{DP} \models f_I \mathsf{U} f_F$. This gives us the following result.

Theorem 13. *The model checking problem for a system comprised of two non-interacting PDS is undecidable for the logic $L(\mathsf{U})$.*

8. Branching Time Properties

While LTL is a very expressive linear time logic, there are some critical properties like deadlockability which are inherently branching time and therefore cannot be expressed in a linear-time framework. In this section, we focus our attention on branching-time temporal logics. Specifically, we consider the model checking problem for Dual-PDS systems for Alternation Free Mu-Calculus formulae. For lack of space, we only consider single-index properties.

For Dual-PDS systems interacting via nested locks, we show that the problem of model checking single-index alternation free Mu-Calculus is decidable. For PDSs interacting via rendezvous and broadcasts, however, it remains undecidable in general.

8.1 Single Index Branching-Time Properties

We begin by formulating the model checking procedure for PDSs interacting via nested locks. As for the linear-time case, our goal is to reduce the model checking problem for Alternation-free Mu-Calculus properties for a Dual-PDS system to its individual PDSs. In order to accomplish that we introduce the new concept of a *Lock-Constrained Alternating Multi-Automata Pair (LAMAP)* that allows us to reduce the pre^*-closure computation for a regular set of configurations for a Dual-PDS systems to its individual PDSs. Thus LAMAPs are the branching-time analogue of LMAPs used previously for model checking linear-time properties.

Lock-Constrained Alternating Multi-Automata Pair (LAMAP). To motivate the concept of an LAMAP, we first re-visit the notions of Alternating Pushdown Systems (APDS) and Alternating Multi-Automata (AMA) used in the model checking of branching-time properties of individual PDSs (see [1]). Model checking branching-time temporal logics, in general, require us to reason about all successors of a global configuration of the given PDS. This branching nature of properties is typically captured by building an alternating multi-automaton (AMA) or a tableau for the given property and then taking the product of the state space of the given PDS with this AMA. Such products can be modeled in a natural fashion by using the concept of an APDS [1] wherein by executing a transition a PDS can transit from a single configuration to a set of configurations, instead of just a single configuration as for the linear time case. Model Checking then reduces to computing pre^*-closures of regular sets of configurations of APDSs. Regular sets of configurations of APDSs can be captured succinctly using the concept of AMAs [1] which are the branching-time analogue of Multi-Automata (MA) the difference again being that in AMAs each transition from a state can have multiple successors.

An LAMAP plays the same role in model checking Dual-PDS systems for branching time properties as an LMAP does in model checking for linear time properties by allowing us to finitely represent (potentially infinite) regular sets of configurations of the given concurrent program in a way that enables us to compute their pre^*-closures efficiently. Whereas, an LMAP is a pair of MAs, an LAMAP $\mathcal{M}$ is a pair $(\mathcal{M}_1, \mathcal{M}_2)$ of AMAs with AMA $\mathcal{M}_i$ representing regular sets of configurations of APDS $\mathcal{P}_i$ corresponding to PDS P_i. However, the key difficulty in extending the concept of an LMAP to that of an LAMAP lies in capturing the lock interaction among the PDSs. For an LMAP, the lock interaction is captured by tracking acquisition histories along individual runs of the two PDSs. For a PDS every run corresponds to a unique path of a PDS. The acquisition history of a given lock in a global state along a path is unique and can be tracked by augmenting the configurations of each PDS. However, a run of an APDS has a tree-like structure and therefore corresponds to multiple paths of the PDS from which the APDS is derived via the product construction. To ensure that tree-like runs run_1 and run_2 of P_1 and P_2, respectively, are reconcilable we need to check that for each local path of P_1 along run_1 there exists a local path y of P_2 along run_2 such that x and y can be fired in an interleaved fashion, and vice versa. But this is precisely the same problem that we faced when building an LMAP for $\mathsf{AF}f$ in the previous section. As before, the solution is to track in each local configuration $\mathbf{c}_i$ of AMA $\mathcal{M}_i$ *all* the different acquisition histories encountered along *all* paths of wit_i starting at $\mathbf{c}_i$. Thus an ah-*augmented configuration* of $\mathcal{P}_i$ is of the form $\mathbf{c}_i = \langle (c_i, \mathcal{AH}_i), u_i\rangle$. In order to check that augmented configurations $\mathbf{c}_1 = \langle (c_1, \mathcal{AH}_1), u_1\rangle$ and $\mathbf{c}_2 = \langle (c_2, \mathcal{AH}_2), u_2\rangle$ accepted by $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, are compatible, one need merely check that for each acquisition history $\mathsf{ah} \in \mathcal{AH}_1$ there is a compatible acquisition history $\mathsf{ah}' \in \mathcal{AH}_2$, and vice versa. In other words, we let LAMAP $\mathcal{A}$ accept $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ iff $\mathbf{c}_1$ and $\mathbf{c}_2$ is accepted by $\mathcal{P}_1$ and $\mathcal{P}_2$ via witnesses wit_1 and wit_2 where wit_1 and wit_2 are *reconcilable* with respect to each other, i.e., for each path x in wit_1 there must be a path y in wit_2 such that the local computations of P_1 and P_2 corresponding to x and y, respectively, can be executed by $\mathcal{DP}$ in an interleaved fashion starting at $\mathbf{c}$, and vice versa. This ensures that for each i, wit_i is indeed a legitimate witness for $\mathbf{c_i}$ in the Dual-PDS system $\mathcal{DP}$.

We show that the model checking problem for PDSs interacting via nested locks can be reduced to the computation of pre^*-closures of regular sets of configurations accepted by LAMAPs. A key property of LAMAPs is that not only are they closed under the computation of pre^*-closures but that the pre^*-closure computation for a given LAMAP can be reduced to pre^*-closure computations for regular sets of configurations of the *individual* PDSs thus avoiding the state explosion problem. We start by formally defining the notion of an LAMAP.

8.1.1 Lock Constrained Alternating Multi-Automata Pair

Let $\mathcal{DP}$ be a given Dual-PDS system comprised of the two PDSs $P_1 = (Q_1, Act_1, \Gamma_1, \mathbf{c}_1, \Delta_1)$ and $P_2 = (Q_2, Act_2, \Gamma_2, \mathbf{c}_2, \Delta_2)$, and $f = \bigwedge_i f_i$ a single index alternation-free Mu-Calculus formula. A *Lock-Constrained Alternating Multi-Automata Pair (LAMAP)* for $\mathcal{DP}$, denoted by $\mathcal{DP}$-LAMAP, is a pair $(\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_i = (\Gamma_i, Q_i, \delta_i, I_i, F_i)$ is an AMA accepting a (regular) set of configurations of the APDS $\mathcal{P}_i$ obtained from P_i by taking the product of P_i with the alternating automaton for f_i. Let $\mathbf{c}_1 = \langle (c_1, \mathcal{AH}_1), u_1\rangle$, and $\mathbf{c}_2 = \langle (c_2, \mathcal{AH}_2), u_2\rangle$ be ah-augmented configurations of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Recall that by our construction, $\mathcal{AH}_i$ tracks the set of acquisition histories encountered along all paths of a tree-like run w_i of $\mathcal{P}_i$ starting at $\langle c_i, u_i\rangle$ with each acquisition history tuple $\mathsf{ah}_{ij} \in \mathcal{AH}_i$ tracking the acquisition history of some path(s) of P_1 along w_1. Motivated by the Decomposition Result, we say that acquisition history tuples $\mathsf{ah}_1 = (\mathsf{lh}, \mathsf{bah}_1, ..., \mathsf{bah}_m, \mathsf{fah}_1, ..., \mathsf{fah}_m)$ and $\mathsf{ah}_2 = (\mathsf{lh}', \mathsf{bah}'_1, ..., \mathsf{bah}'_m, \mathsf{fah}'_1, ..., \mathsf{fah}'_m)$ are *compatible* iff the following conditions are satisfied (i) fah-*compatibility*: there do not exist locks l_i and l_j such that $l_i = P_1$, $l'_j = P_2$, $l_i \in \mathsf{fah}'_j$ and $l_j \in \mathsf{fah}_i$, (ii) bah-*compatibility*: there do not exist locks l_i and l_j such that $l_i = P_1$, $l'_j = P_2$, $l_i \in \mathsf{bah}'_j$ and $l_j \in \mathsf{bah}_i$, and (iii) *Disjointness of Locksets*: $\mathsf{lh} \cap \mathsf{lh}' = \emptyset$. Then for each local path of P_1 along w_1 starting at $\mathbf{c}_1$ to be executable in an interleaved fashion with some local path of P_2 starting at $\mathbf{c}_2$, for each $\mathsf{ah}_{1j} \in \mathcal{AH}_1$ there must exist $\mathsf{ah}_{2j'} \in \mathcal{AH}_2$ such that ah_{1j} and $\mathsf{ah}_{2j'}$ are compatible, and vice versa, thereby leading us to the following definition.

Definition 14 (LAMAP Acceptance). *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a* $\mathcal{DP}$-LAMAP. *We say that* $\mathcal{A}$ *accepts configuration* $(\langle p_1, u_1\rangle, \langle p_2, u_2\rangle, l_1, ..., l_m)$ *of* $\mathcal{DP}$ *iff there exist acquisition history tuples* $\mathcal{AH}_1$ *and* $\mathcal{AH}_2$*, with the same locksets as those held by* P_1 *and* P_2*, respectively, in* $\mathbf{c}$*, such that the following holds*

1. $\mathcal{A}_i$ *accepts* $\langle(p_i, \mathcal{AH}_i), u_i\rangle$*, and*
2. $\mathcal{AH}_1$ *and* $\mathcal{AH}_2$ *are compatible, viz., for each tuple* $\mathsf{ah}_1 \in \mathcal{AH}_1$ *there exists a tuple* $\mathsf{ah}_2 \in \mathcal{AH}_2$ *such that* ah_1 *and* ah_2 *are compatible, and vice versa.*

Given a $\mathcal{DP}$*-LAMAP* $\mathcal{A}$, we use $Conf(\mathcal{A})$ to denote the set of configurations of $\mathcal{DP}$ accepted by $\mathcal{A}$. Our broad goal is the reduction of Model Checking of a Dual-PDS system to its individual PDSs. This is accomplished by (i) reducing the model checking problem for single-index alternation free Mu-calculus formulae to the computation of pre^*-closures for LAMAPs, and (ii) reducing the pre^*-closure for LAMAPs to that for the individual AMAs constituting the LAMAPs. We first show how to compute pre^*-closure for a regular set of ah-enhanced configurations accepted by a given AMA. This will immediately lead to a procedure for pre^*-closure computation for LAMAPs.

ah-enhanced pre^*-computation. We outline only the broad steps of the procedure. We start with an AMA $\mathcal{A}$ accepting a regular set C of acquisition history augmented configurations of an APDS $\mathcal{P}$. Corresponding to each augmented control state $(c_j, \mathcal{AH})$ we have, as in [1], an *initial* state $(s_j, \mathcal{AH})$ of the multi-automaton $\mathcal{A}$, and vice versa. We set $\mathcal{A}_0 = \mathcal{A}$ and construct a finite sequence of AMAs $\mathcal{A}_0, ..., \mathcal{A}_p$ resulting in the AMA $\mathcal{A}_p$ such that the set of ah-augmented configurations accepted by $\mathcal{A}_p$ is the pre^*-closure of the set of AH-augmented configurations accepted by $\mathcal{A}$. We denote by $\rightarrow_i$ the transition relation of $\mathcal{A}_i$. For every $i \geq 0$, $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by conserving the sets of states and transitions of $\mathcal{A}_i$ and adding new transitions as follows. The key difference from the linear time case is that now we need to compute a pre-image with respect to AX instead of EX. Thus for every internal transition $\langle p_j, \gamma\rangle \hookrightarrow \{\langle p_{k_1}, w_1\rangle, ..., \langle p_{k_n}, w_n\rangle\}$ and every set $(s_{k_1}, \mathcal{AH}_1) \xrightarrow{w_1}_i Q_1$, ..., $(s_{k_n}, \mathcal{AH}_n) \xrightarrow{w_n}_i Q_n$, we add the new transition $(s_j, \mathcal{AH}_1 \cup ... \cup \mathcal{AH}_n) \xrightarrow{\gamma}_{i+1} (Q_1 \cup ... \cup Q_n)$. For every backward execution of a lock/unlock transition from a configuration $\langle(c_i, \mathcal{AH}_i, u_i\rangle$ we now need to update every acquisition history in the set $\mathcal{AH}_i$, instead of merely one. Then we have the following result.

Proposition 15. *Given an APDS* $\mathcal{P}$*, and a regular set of* ah-*augmented configurations of* $\mathcal{P}$ *accepted by an AMA* $\mathcal{A}$*, we can construct an AMA* $\mathcal{A}_{pre^*}$ *recognizing* $pre^*(Conf(\mathcal{A}))$ *in time polynomial in the size of the control set of* $\mathcal{P}$ *and exponential the size of* $\mathcal{A}$ *and the number of locks of* $\mathcal{P}$.

Computing the pre*-closure of a LAMAP. Let LC be a regular set accepted by a $\mathcal{DP}$-LAMAP $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. In this section, we show that we can construct a $\mathcal{DP}$-LAMAP $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ accepting $pre^*(LC)$.

The Procedure. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are AMAs accepting regular sets of configurations of the individual APDSs $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, we can construct, using the technique presented above, AMAs $\mathcal{B}_1$ and $\mathcal{B}_2$, accepting, respectively, the pre^*-closures, $pre^*_{\mathcal{P}_1}(Conf(\mathcal{A}_1))$ and $pre^*_{\mathcal{P}_2}(Conf(\mathcal{A}_2))$. Then the following result effectively formulates the decomposition of pre^*-closures for a Dual-PDS system with nested locks to its constituent PDSs.

Theorem 16. *Let* R *be a regular set of configurations of* $\mathcal{DP}$ *accepted by the* $\mathcal{DP}$*-LAMAP* $\mathcal{A}$*. If* $\mathcal{B}$ *is the* $\mathcal{DP}$*-LAMAP constructed from* $\mathcal{A}$ *as above, then* $Conf(\mathcal{B}) = pre^*(R)$.

Complexity Analysis. Note that the computation of an LAMAP $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ accepting the pre^*-closure of a given LAMAP $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ reduces to the computation of AMAs $\mathcal{B}_i$ accepting the pre^*-closure of $Conf(\mathcal{A}_i)$ for each individual PDS P_i, instead of the entire system $\mathcal{DP}$. From proposition 15, $\mathcal{B}_i$ can be computed in polynomial time in the size of the control set of P_i and exponential time in the size of $\mathcal{A}_i$ and number of locks of P_i. Thus we have the following

Theorem 17. *Given a Dual-PDS system* $\mathcal{DP}$ *comprised of PDSs* P_1 *and* P_2 *interacting via nested locks, and a* $\mathcal{DP}$*-LAMAP* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we can construct a* $\mathcal{DP}$*-LAMAP* $\mathcal{A}_{pre^*}$ *recognizing* $pre^*(Conf(\mathcal{A}))$ *in time polynomial in the size of the control set of each* P_i *exponential time in the size of* $\mathcal{A}$ *and the number of locks of* $\mathcal{DP}$.

8.2 The Model Checking Procedure

We now show how to reduce the model checking of Dual-PDS systems with nested locks for single-index alternation-free mu-calculus formulas to it individual PDSs.

Let $\mathcal{DP}$ be a concurrent program comprised of the PDSs $P_1 = (Q_1, Act_1, \Gamma_1, \mathbf{c}_1, \Delta_1)$ and $P_2 = (Q_2, Act_2, \Gamma_2, \mathbf{c}_2, \Delta_2)$ and a labeling function $\Lambda_i : Q_i \rightarrow 2^{Prop_i}$. Let $h = \bigwedge h_i$, where h_i is an alternation-free weak μ-calculus formula interpreted over the control states of thread P_i and let $\mathcal{V}_i$ be a valuation of the free variables in h_i. As in [1], we begin by constructing an APDS $\mathcal{P}_i$ that represents the product of P_i and an alternating automaton for h_i. Set $\phi = h_i$. We start by considering the case where all the σ-subformulas of ϕ are μ-formulas. The product APDS $\mathcal{P}_i = (Q_i^{\phi}, \Gamma_i, \Delta_i^{\phi})$ of P_i and the alternating automaton for h_i, is straightforward to define and is omitted. In order to decide whether two runs of $\mathcal{P}_1$ and $\mathcal{P}_2$ are reconcilable, we need to augment each configuration of $\mathcal{P}_i$ with acquisition history set information as discussed in section 9.1. With this in mind, let $\mathcal{C}_t^i$ be the subset of configurations of $\mathcal{P}_i$ containing all augmented configurations of the form

- $([(p, \mathcal{AH}_{\emptyset}), \pi], u)$,
- $([(p, \mathcal{AH}_{\emptyset}), \neg\pi], u)$,
- $([(p, \mathcal{AH}_{\emptyset}), X], u)$, where X is free in ϕ and $\langle p, u\rangle \in \mathcal{V}_i(X)$.

where $\mathcal{AH}_{\emptyset}$ is the $(2m + 1)$-tuple $\{(\emptyset, ..., \emptyset)\}$ - all acquisition histories are set to the empty set to start with. Clearly, if $\mathcal{V}_i$ is a regular set of configurations for every variable X free in ϕ_i, then $\mathcal{C}_t^i$ is also a regular set of configurations. Then using the concept of signatures for mu-calculus sentences, as in [1], we have.

Proposition 18 ([1]) *Let* $\mathcal{P}_i$ *be the APDS obtained from* P_i *and* h_i *using the construction above. A configuration* $\langle(p, \mathcal{AH}), u\rangle$ *of* $\mathcal{P}_i$ *belongs to* $[\![\phi_i]\!]$ *iff the configuration* $\langle[(p, \mathcal{AH})\ , \phi_i], u\rangle$ *of* $\mathcal{P}_i$ *belongs to* $pre^*_{\mathcal{P}_i}(\mathcal{C}_t^i)$

Furthermore, as in [1], the case where all the σ-subformulas of ϕ_i are ν-subformulas can now be tackled by (i) noting that the negation of ϕ_i is equivalent to a formula ϕ_i' in positive normal form whose σ-subformulas are all μ-subformulas (ii) applying proposition 18, to construct an AMA which accepts the configurations of $\mathcal{P}_i$ that satisfy ϕ_i', and finally (iii) using the fact that AMAs are closed under complementation. Then the general case for the alternation-free mu calculus can be handled by recursively applying the procedure for the above two cases (see [1] for details) giving us the following result analogous to that in [1], but for PDSs with ah-augmented control states.

Theorem 19 ([1]) *Let $\mathcal{P}_i$ be the APDS corresponding to thread P_i as constructed above, and let h_i a formula of the alternation-free mu-calculus interpreted over the local configurations of P_i, and let $\mathcal{V}_i$ be a valuation of the free variables of h_i. We can construct an AMA $\mathcal{A}_{h_i}$ such that $Conf(\mathcal{A}_{h_i}) = [\![h_i]\!]_{\mathcal{T}_i}(\mathcal{V}_i)$.*

The key reduction result is formulated below.

Theorem 20 (Reduction Result) *A configuration $\mathbf{c} = (\langle p_1, u_1\rangle, \langle p_2, u_2\rangle, l_1, ..., l_m)$ of $\mathcal{DP}$ belongs to $[\![h]\!]$ iff there exists a pair of compatible acquisition history sets $\mathcal{AH}_1$ and $\mathcal{AH}_2$ such that the configuration $(\langle[(p_1, \mathcal{AH}_1), h_1], u_1\rangle, \langle [(p_2, \mathcal{AH}_2), h_2], u_2\rangle, l_1, ..., l_m)$ is accepted by the $\mathcal{DP}$-LAMAP $\mathcal{A}_h = (\mathcal{A}_{h_1}, \mathcal{A}_{h_2})$.*

This reduces the model checking problem to the construction of the AMAs $\mathcal{A}_{h_i}$. Then using theorem 17 and the above construction we have the following.

Theorem 21 (Single-index Mu-Calculus Decidability) *The model checking problem for single index alternation-free weak Mu-Calculus formulas for Dual-PDS systems synchronizing via nested locks is decidable in time exponential in the sizes of control states of the individual PDSs and the number of locks.*

Rendezvous and Broadcasts. For PDS interacting via rendezvous and broadcasts, even single-index reachability is undecidable [15] which implies undecidability of the model checking problem for single-index Alternation-free Mu-Calculus.

9. Concluding Remarks

Among prior work on the verification for concurrent programs, [3] attempts to generalize the techniques given in [1] to handle pushdown systems communicating via CCS-style pairwise rendezvous. However since even reachability is undecidable for such a framework, the procedures are not guaranteed to terminate in general but only for certain special cases, some of which the authors identify. The key idea here is to restrict interaction among the threads so as to bypass the undecidability barrier. Another natural way to obtain decidability is to explore the state space of the given concurrent multi-threaded program for a bounded number of context switches among the threads [14]. For PDSs interacting via rendezvous, over-approximation techniques to achieve termination while performing reachability are considered in [5].

Other related interesting work includes the use of tree automata [12] and logic programs [8] for model checking the processes algebra PA which allows modeling of non-determinism, sequential and parallel composition and recursion. The reachability analysis of Constrained Dynamic Pushdown Networks which extend the PA framework by allowing PDSs that can spawn new PDSs to model fork operations, was considered in [4]. However, neither model allows communication among processes. To model interaction among threads, Asynchronous Dynamic Pushdown Network has been proposed recently [2]. This model allows communication via shared variables which however makes the model checking problem undecidable. This problem is bypassed by considering the restricted *bounded model checking* problem wherein only those computations of the given program are explored where each thread is only allowed a bounded number of updates to the shared variables. Another approach that has been explored is to extend the classical procedure-summary based inter-procedural dataflow analysis for sequential programs to concurrent programs via the use of transactions [13].

In this paper, we have focused on the model checking of Interacting Pushdown Systems synchronizing via the standard primitives - locks, rendezvous and broadcasts, for rich classes of temporal properties - both linear and branching time. Since interprocedural dataflow analysis for concurrent programs hinges on an efficient model checking algorithms for interacting PDS systems, it is important we identify temporal logic fragments and useful models of interacting PDSs for which the problem is decidable. In this paper, we have accomplished precisely this. Specifically, we have formulated new efficient algorithms for model checking interacting PDSs for important fragments of LTL and Mu-Calculus. Additionally, we also delineate precisely the decidability boundary for each of the standard synchronization primitives.

References

[1] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, LNCS 1243, pages 135–150, 1997.

[2] A. Bouajjani, J. Esparza, S. Schwoon, and J. Strejcek. Reachability analysis of multithreaded software with asynchronous communication. In *FSTTCS*, 2005.

[3] A. Bouajjani, J. Esparza, and T. Touili. A generic approach to the static analysis of concurrent programs with procedures. In *IJFCS*, volume 14(4), pages 551–, 2003.

[4] A. Bouajjani, M. Olm, and T. Touili. Regular Symbolic Analysis of Dynamic Networks of Pushdown Systems. In *CONCUR*, 2005.

[5] S. Chaki, E. Clarke, N. Kidd, T.Reps, and T.Touili. Verifying concurrent message-passing c programs with recursive calls. In *TACAS*, 2006.

[6] E. A. Emerson. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science, Volume B*, pages 997–1072, 1998.

[7] E.A. Emerson and V. Kahlon. Model checking guarded protocols. In *LICS*, 2003.

[8] J. Esparza and A. Podelski. Efficient Algorithms for pre^* and $post^*$ on Interprocedural Parallel Flow Graphs. In *POPL*, 2000.

[9] V. Kahlon and A. Gupta. An Automata-Theoretic Approach for Model Checking Threads for LTL Properties. In *LICS*, 2006.

[10] V. Kahlon, F. Ivančić, and A. Gupta. Reasoning about threads communicating via locks. In *CAV*, 2005.

[11] A. Lal, G. Balakrishnan, and T. Reps. Extended weighted pushdown systems. In *CAV*, 2005.

[12] D. Lugiez and Ph. Schnoebelen. The Regular Viewpoint on PA-Processes. In *Theor. Comput. Sci.*, volume 274(1-2), 2002.

[13] S. Qadeer, S. K. Rajamani, and J. Rehof. Summarizing procedures in concurrent programs. In *POPL*, pages 245–255, 2004.

[14] S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS*, 2005.

[15] G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. In *ACM Trans. Program. Lang. Syst.*, volume 22(2), pages 416–430, 2000.

[16] T. W. Reps, S. Schwoon, S. Jha, and D. Melski. Weighted pushdown systems and their application to interprocedural dataflow analysis. In *Science of Computer Programming*, volume 58(1-2), 2005.

[17] I. Walukeiewicz. Model checking ctl properties of pushdown systems. In *FSTTCS*, LNCS 1974, pages 127–138, 2000.