MSO decidability of Multi-Pushdown Systems via Split-Width*

Aiswarya Cyriac[1], Paul Gastin[1], and K. Narayan Kumar[2]

[1] LSV, ENS Cachan, CNRS & INRIA, France
{cyriac,gastin}@lsv.ens-cachan.fr
[2] Chennai Mathematical Institute, India
kumar@cmi.ac.in

Abstract. Multi-threaded programs with recursion are naturally modeled as multi-pushdown systems. The behaviors are represented as multiply nested words (MNWs), which are words enriched with additional binary relations for each stack matching a push operation with the corresponding pop operation. Any MNW can be decomposed by two basic and natural operations: shuffle of two sequences of factors and merge of consecutive factors of a sequence. We say that the split-width of a MNW is k if it admits a decomposition where the number of factors in each sequence is at most k. The MSO theory of MNWs with split-width k is decidable. We introduce two very general classes of MNWs that strictly generalize known decidable classes and prove their MSO decidability via their split-width and obtain comparable or better bounds of tree-width of known classes.

1 Introduction

Multi-pushdown systems (MPDS) — finite state systems with several stacks — are natural abstractions of concurrent programs. Verification of multi-pushdown systems is undecidable in general. However concurrency is indispensable for many critical systems. Hence, several behavioral restrictions have been proposed and employed for their under-approximate verification [10, 13, 16, 17, 19].

The first behavioral restriction shown to have a decidable reachability problem was bounded context switching [19] in which the control can switch from one stack to another only a fixed number of times [13, 16, 17]. This was followed by ordered MPDS where the stacks have a priority ordering between them [2, 3], and a stack could pop only when all higher priority stacks are empty. Another restriction is allowing only a fixed number of phases [12], where in one phase only one stack was allowed to return. Later bounded scope MPDS [14], where there are at most k context switches between any push and the corresponding pop, were also shown to have a decidable emptiness. In [18], Madhusudan and Parlato give a unified proof of decidability of emptiness of all but the last, by showing that these restrictions impose bounds on the *tree-width* of the underlying runs.

* Supported by LIA InForMel, and DIGITEO LoCoReP.

As more general classes are desirable in the under-approximate verification, we propose a bigger and natural class of MPDS which is a generalization of ordered and scope bounded MPDS. We freely allow pops of both kinds in this restriction. This can be thought of as the fair runs which comply to the following scheduling policy. There is no restriction on pushes. But the corresponding pop a) has to be within fixed number of context switches from then (analogous to time-out) or b) if a) fails, then all such events will be ordered on a priority basis (assuming a total order on the priorities of different stacks). This class is called scope bounded or ordered return (SBO) in the paper. Thus under-approximate verification wrt. SBO is a kind of fair model checking, in which at least those runs which comply to the fair scheduling policy can be verified against some specification. A similar generalization can be thought of when the ordering policy is replaced by a bounded phase restriction. These two general classes are shown to be decidable. Note that, however, a joint generalization of ordered and phase bounded yields undecidablity.

The decidability proofs for the above classes are done by showing that these classes have bounded split-width. The behaviors of a multi-pushdown system as a graph are called multiply-nested words (MNWs). These are words enriched with additional binary relations matching a push on a stack with the corresponding pop. Split-width is a measure on MNWs which is comparable to tree-width (or *clique-width*) [7, 11]. This, particularly since the latter was used in [18], calls for a comparison of split-width to tree-width.

Split-width has a simpler definition. It is defined in terms of two basic and natural operations — *shuffle* of two sequences of factors and *merge* of consecutive factors in a sequence. Thus split-width is easier to handle as these are well-tuned for MNWs, where as tree-width is defined for general graphs. This gives easier and simpler proofs.

Bound on split-width can be translated (up to a constant factor) to bound on tree-width (or clique-width). MNWs with split-width at most k have tree-width at most $2k-1$ and clique-width at most $2k+1$. For the other direction, MNWs with clique-width at most k have split-width at most $2k$. Thus we do not yet know whether we have an "equivalence" between split-width and tree-width (or clique-width).

Even though the class of bounded split-width MNWs is not known to be MSO definable, they enjoy a decidable MSO theory. Furthermore, split-width is general enough to capture all classes of MNWs with a decidable MSO theory, thanks to the translation from clique-width to split-width.

Thus split-width should be seen as a complementary approach which gives more insight into the structure of the MNWs which have bounded tree-width (or clique-width). The advantages of split-width are reflected in the fact that it helped in improving bounds for tree-width of known classes, and lifting up proofs from different classes to get proofs for joint generalizations.

To summarize, the contributions of this paper are manyfold. On one hand it introduces more general classes of MNWs for more accurate under-approximate verification of MPDS. It introduces the notion of split-width, a measure of com-

plexity of MNWs, which is easier than, yet as general as tree-width or clique-width. It significantly improves the known bounds on tree-width for ordered MPDS and scope bounded MPDS.

The paper is organized as follows. Section 2 recalls some preliminary notions. Section 3 gives the definition of split-width and compares it to tree-width and clique-width. It also shows the MSO decidability of bounded split-width. In Section 4 various decidable classes of MNWs are formally defined, and proof of their decidability is given by showing a bound on split-width of these classes. Some proofs are omitted due to lack of space. These can be found in [9].

2 Preliminaries

$\mathbb{N}$ denotes the set of natural numbers. For $n \in \mathbb{N}$, by $[n]$ we denote the set $\{1, \ldots, n\}$. Let S be a set. For a binary relation $\mathcal{R} \subseteq S \times S$, we define *support* of $\mathcal{R}$, denoted $\mathsf{supp}(\mathcal{R})$, to be $\{x \in S \mid$ there is some $y \in S$ such that $(x, y) \in \mathcal{R}$ or $(y, x) \in \mathcal{R}\}$.

Multi-pushdown systems (MPDS) are finite state systems with a finite number of stacks. A transition may push onto a stack (push transitions), pop from a stack (pop transitions) or leave the stacks untouched. However, in one transition a MPDS can touch at most one stack. Moreover the push transitions and pop transitions are disjoint. Let Σ be the finite alphabet and $s \in \mathbb{N}$ be the number of stacks. We fix the finite alphabet Σ and the set of stacks $[s]$ for the rest of this paper. The behaviors of a multi-pushdown system are represented as multiply-nested words (MNWs).

Multiply-Nested Words (MNWs) A multiply-nested word (MNW) w over Σ is a structure $w = (\mathsf{dom}(w), \lambda, \lessdot, \nearrow^1, \ldots, \nearrow^s)$ where

- $\mathsf{dom}(w)$ is the set of positions
- $\lambda : \mathsf{dom}(w) \mapsto \Sigma$ is the node labeling function
- $\lessdot$ is the successor relation of a total order on $\mathsf{dom}(w)$. We denote this total order by $<$. That is, $<= \lessdot^+$.
- For each $i \in [s]$, $\nearrow^i \subseteq <$ is a binary relation such that
  1. For $i \neq j$, $\mathsf{supp}(\nearrow^i) \cap \mathsf{supp}(\nearrow^j) = \emptyset$
  2. For all $i \in [s]$, $x \nearrow^i y \implies (\forall z\, (x \nearrow^i z \implies z = y) \wedge (z \nearrow^i y \rightarrow z = x))$
  3. For all $i \in [s]$, there do not exist $x < x' < y < y'$ such that $x \nearrow^i y$ and $x' \nearrow^i y'$

We may think of this structure as a graph whose vertices are labelled by the function λ and edges are labelled using the symbols $\Gamma = \{\lessdot, \nearrow^1, \nearrow^2, \ldots, \nearrow^s)\}$. We refer to the edges labelled by $\lessdot$ as *linear edges* and those labelled by $\nearrow^i$ as *nesting edges*. If $s = 1$, a MNW is simply called a *nested word* in the literature [1].

MSO over MNWs We assume that we have an infinite supply of first-order variables $x, y, \ldots$ and second-order variables $X, Y, \ldots$. First order variables vary over positions of an MNW while second order variables vary over subsets of positions. The syntax of the monadic second order logic over MNWs is as follows:

$$\varphi ::= a(x) \mid x \in X \mid x \nearrow^i y \mid x < y \mid x = y \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$ and $i \in [s]$. We assume familiarity with logic and hence omit the obvious semantics associated with this logic.

Remark 1. The language of a Multi-pushdown system as a set of MNWs can be described in MSO.

3 Split-width of MNWs

Given a MNW $w = (\mathsf{dom}(w), \lambda, \lessdot, \nearrow^1, \ldots, \nearrow^s)$, an *m-split of* w is a structure $\overline{w} = (\mathsf{dom}(w), \lambda, \rightarrow, \dashrightarrow, \nearrow^1, \ldots, \nearrow^s)$ where $\rightarrow \cap \dashrightarrow = \emptyset$, $\rightarrow \cup \dashrightarrow = \lessdot$ and $|\dashrightarrow| = m-1$. The intuition is that the $\dashrightarrow$-edges are *missing* and these missing edges divide the linear order into m linear components (though there may be nesting edges connecting these different components).

A *split multiply nested word (SMNW)* is an m-split $\overline{w}$ of some MNW w for some m. We say that $\overline{w}$ is an m-SMNW. The entire multiply nested word is always a 1-SMNW. Notice that SMNWs continue to have the well nesting property for each $\nearrow^i$ w.r.t. the linear order generated by $\rightarrow \cup \dashrightarrow$.

Let $\overline{u} = (\mathsf{dom}(u), \lambda_u, \rightarrow_u, \dashrightarrow_u, \nearrow^1{}_u, \ldots, \nearrow^s{}_u)$ be an m-SMNW and let $\overline{v} = (\mathsf{dom}(v), \lambda_v, \rightarrow_v, \dashrightarrow_v, \nearrow^1{}_v, \ldots, \nearrow^s{}_v)$ be an n-SMNW. The *shuffle* of $\overline{u}$ and $\overline{v}$, denoted $\overline{u} \sqcup\!\sqcup \overline{v}$ is a set of $(m+n)$-SMNWs. A $(m+n)$-SMNW $\overline{w} = (\mathsf{dom}(w), \lambda_w, \rightarrow_w, \dashrightarrow_w, \nearrow^1{}_w, \ldots, \nearrow^s{}_w) \in \overline{u} \sqcup\!\sqcup \overline{v}$ if and only if:

- $\mathsf{dom}(w) = \mathsf{dom}(u) \uplus \mathsf{dom}(v)$
- $\lambda_w = \lambda_u \uplus \lambda_v$
- $\rightarrow_w = (\rightarrow_u \cup \rightarrow_v)$
- $\nearrow^i_w = \nearrow^i_u \cup \nearrow^i_v$

Note that, by explicitly stating that $\overline{w}$ is an $(m+n)$-SMNW, we have ensured that the nesting edges in $\overline{w}$ are well nested w.r.t. the linear order generated by $\dashrightarrow_w \cup \rightarrow_w$. Note also that, $\dashrightarrow_w \not\supseteq \dashrightarrow_u \cup \dashrightarrow_v$. In fact, by alternately choosing components from $\overline{u}$ and $\overline{v}$, we can have $\dashrightarrow_w \cap (\dashrightarrow_u \cup \dashrightarrow_v) = \emptyset$.

Let $\overline{u} = (\mathsf{dom}(u), \lambda_u, \rightarrow_u, \dashrightarrow_u, \nearrow^1{}_u, \ldots, \nearrow^s{}_u)$ be an m-SMNW. The *merge* of $\overline{u}$, denoted $\mathsf{merge}(\overline{u})$, is a set of n-SMNWs for $1 \leq n < m$, obtained by replacing some $\dashrightarrow$ by $\rightarrow$ in $\overline{u}$.

Let $k \geq 2$. We define the class k-BS (for *k-bounded splits*) to be the smallest set of SMNWs closed under the following operations

- $a \in k$-BS. That is, a single node labelled a is in k-BS.
- $a \overset{i}{\dashrightarrow} b \in k$-BS. That is, two nodes labelled a and b, connected by a $\nearrow^i$-edge is in k-BS.

– if $\overline{u}$ is an m-SMNW in k-BS, $\overline{v}$ is an n-SMNW in k-BS and if $m+n \leq k$, then $\overline{u} \sqcup\!\sqcup \overline{v} \subseteq k$-BS.
– if $\overline{u}$ is in k-BS, then $\mathsf{merge}(\overline{u}) \subseteq k$-BS

For any SMNW $\overline{w}$, if $\overline{w} \in k$-BS we say that the *split-width* of $\overline{w}$ is at most k.

3.1 Split-width, Tree-width and Clique-width of MNWs

Split-width compares well to the usual measures of graph complexity: *tree-width* and *clique-width* [5, 11, 20]. This relation is stated in the following theorem:

Theorem 2. *1. The tree-width of a MNW of split-width k is at most $2k-1$.*
2. The clique-width of a MNW with split-width k is at most $2k+1$.
3. The split-width of a MNW with clique-width k is at most $2k$.

It is known that any class of graphs with tree-width bounded by k has clique-width bounded by $2^{k-1}-1$ [8]. However, Item 2 gives a better bound on clique-width. We give only the proof of Item 1 in this paper. The proof of the other two items can be found in [9].

We use the algebraic characterization of tree-width as in [4]. For this we define a syntax for generating graphs.[1] Let C be a finite set of colors. Then C-expressions are given by:

$$e \ ::= \ x \ \mid \ x \ \mathsf{E} \ y \ \mid \ e_1 \parallel e_2 \ \mid \ \mathsf{rnm}_{x \leftrightarrow y}(e) \ \mid \ \mathsf{fg}_x(e)$$

where $x, y \in C$ and E is an edge relation. In particular for nested words $x \to y$, $x \overset{i}{\curvearrowright} y$ are C-expressions. Each expression defines an edge labelled graph (up to isomorphism) as described below:

– The expression x denotes the graph with a single vertex colored x.
– The expression $x \ \mathsf{E} \ y$ denotes the graph with two vertices colored x and y and these vertices are connected by an edge E.
– The expression $e_1 \parallel e_2$ (parallel composition) denotes the disjoint union of the graphs defined by the expressions e_1 and e_2, where the nodes with the same labels are fused.
– The expression $\mathsf{rnm}_{x \leftrightarrow y}(e)$ (renaming) denotes the graph obtained by recoloring the vertices colored x and y in the graph denoted by e with y and x.
– The expression $\mathsf{fg}_x(e)$ (forget color) denotes the graph obtained by removing the color of the vertices colored x in the graph denoted by e.

Notice that there can be at most one vertex colored x for each color x, since the parallel composition fuses nodes with the same color. Also once the color of a vertex is forgotten, that vertex cannot be colored later. Notice that we have ignored the node labels in this definition, as these are not the most interesting. However, one could easily include them.

The tree-width of a graph is at most $|C|-1$ if there is a C-expression denoting it [4]. Using this we will now prove Item 1 of Theorem 2

[1] This is F_C^{HR} in [4]

Proof (of Item 1 of Theorem 2). There are at most k components in any SMNW of split-width at most k. We use $2k$ colors of the form b_i, e_i for $1 \leq i \leq k$. That is we fix $C = \{b_1, e_1, \ldots, b_k, e_k\}$. We maintain the invariant `INV1`:*Color the first node and the last node of factor i by b_i and e_i respectively. If a factor has only one node, its color is b_i.* We show how to obtain a SMNW of split-width at most k using C-expressions inductively. The base cases are the basic splits: The expression for an internal node is b_1, and that for a nesting edge on stack i is $b_1 \overset{i}{\curvearrowright} b_2$.

For $w \in u \sqcup\!\sqcup v$: We identify the index in w of each factor in u and v. Then we do a sequence of renamings in u and v such that each node gets its intended label in w. This is followed by a simple parallel composition. Note that this parallel composition does not result in the fusion of any nodes, as the colors are disjoint. For example, consider $w = (n_1, n_2n_3, n_4, n_5)$ and $u = (n_1, n_5)$ and $v = (n_2n_3, n_4)$. Since u and v satisfies the invariant `INV1`, n_1 and n_2 are colored b_1; n_5 and n_4 are colored b_2; and n_3 is colored e_1. Let e_u, e_v denote the expressions for u and v respectively. Then $e_w = (\mathsf{rnm}_{b_2 \leftrightarrow b_4}(e_u)) \parallel (\mathsf{rnm}_{b_1 \leftrightarrow b_2}(\mathsf{rnm}_{e_1 \leftrightarrow e_2}(\mathsf{rnm}_{b_2 \leftrightarrow b_3}(e_v))))$.

For $w \in \mathsf{merge}(u)$: If w contains a linear edge from factor i in u to factor $i+1$ in u, we do a parallel composition with $(e_i \to b_{i+1})$ (If the factor i is singleton, we do a parallel composition with $(b_i \to b_{i+1})$). The graph $(e_i \to b_{i+1})$ is represented by $\mathsf{rnm}_{b_1 \leftrightarrow e_i}(\mathsf{rnm}_{b_2 \leftrightarrow b_{i+1}}(b_1 \to b_2))$. We do this for each linear edge added in w. Finally, in order to maintain the invariant `INV1`, we do a sequence of forgets and renamings. □

A theorem by Courcelle [6] says that if MSO is decidable for a class $\mathcal{C}$ of graphs with bounded degree, then $\mathcal{C}$ has bounded clique-width. This theorem, along with Item 3, says that any class of MNWs with decidable MSO theory indeed has bounded split-width.

Corollary 3. *Let $\mathcal{C}$ be a class of MNWs. If $\mathcal{C}$ has a decidable MSO theory, then $\mathcal{C}$ has bounded split-width.*

3.2 MSO is decidable over bounded split-width MNWs

An MSO definable class with bounded tree-width (or clique-width) has a decidable MSO theory. However, we do not know whether the class of k-BS MNWs is MSO-definable. Thus Theorem 2 does not imply MSO decidability for k-BS MNWs. Nevertheless, we have the following theorem:

Theorem 4. *Let $k \in \mathbb{N}$. The class of MNWs with split-width at most k has a decidable MSO theory.*

The proof is via a tree interpretation along the lines of the proof of MSO decidability over bounded clique-width graphs [7,11]. Let $\overline{w}$ be a SMNW in k-BS. By definition, the proof of the membership of $\overline{w}$ in k-BS is a tree whose nodes are labelled by elements of k-BS and whose degree is bounded by 2 such that

1. the root is labelled by $\overline{w}$.

2. leaves are labelled by atomic SMNWs.
3. if an internal node labelled $\overline{u}$ has only one child labelled $\overline{v}$ then $\overline{u} \in \mathsf{merge}(\overline{v})$.
4. if an internal node labelled $\overline{u}$ has two children labelled $\overline{x}$ and $\overline{y}$ then $\overline{u} \in \overline{x} \sqcup\!\sqcup \overline{y}$.

We abstract such a proof as a finitely labelled tree, called a proof tree. We can show that the set of valid proof-trees (of membership of SMNWs in k-BS) is accepted by a tree automaton of size exponential in k and s. Then we give a translation from any MSO formula Φ over MNWs to an "equivalent" formula Φ' over proof-trees. The detailed proof is given in [9] where this technique is extended to also show

Theorem 5. *Given a MPDS $\mathcal{M}$ and an integer k, we can construct a tree automaton $\mathcal{A}$ over the proof trees for k-BS, such that $\mathcal{A}$ accepts all the valid proof trees of MNWs in k-BS which have an accepting run in $\mathcal{M}$. The size of $\mathcal{A}$ is exponential in k and the number of stacks s, but is polynomial (with exponent $\mathcal{O}(k)$) in the number of states of $\mathcal{M}$.*

The above theorem allows us to derive several corollaries. *Emptiness checking* of a multi-pushdown system restricted to bounded split-width behaviors is ExpTime. In fact, this allows *MSO-model checking* of a MPDS restricted to k-BS. Given a multi-pushdown system $\mathcal{M}$, an integer k and an MSO formula φ over MNWs, it decidable to check whether all MNWs of split-width at most k generated by $\mathcal{M}$ satisfy φ in time non-elementary in $|\varphi|$, exponential in k and the number of stack s, and polynomial in the number of states of $\mathcal{M}$. *Inclusion checking* of two MPDS wrt. k-BS is 2ExpTime. As the set of all valid proof trees is recognizable, *universality checking* of a MPDS wrt. k-BS is also 2ExpTime.

4 Classes of MNWs

Let w be a MNW. A *factor* u of w is defined to be a sequence of consecutive positions of w. We say that a position $x \in \mathsf{dom}(u)$ is an *i-pending call* in u if there exists $y \in \mathsf{dom}(w) \setminus \mathsf{dom}(u)$ such that $x \nearrow^i y$. Similarly, x is an *i-pending return* in u if there exists $y \in \mathsf{dom}(w) \setminus \mathsf{dom}(u)$ such that $y \nearrow^i x$. We say that u is complete for i if there are no i-pending calls or i-pending returns in u. This notion is lifted naturally to sequences of factors as well. A *context* is a set of consecutive positions which involves at most one stack.

We recall the definitions of three classes of MNWs for which MSO theory is known to be decidable and follow it with definitions of two new classes we propose.

Bounded Scope MNWs [14] We fix a parameter $m \in \mathbb{N}$. We say that a MNW is m-scope bounded if for all nesting edges, there are *no* more than m different contexts between its source and target.

Bounded Phase MNWs [12] A *phase* is a factor of a MNW in which at most one stack is allowed to return. We fix a parameter $p \in \mathbb{N}$. We say that a MNW is p-phase bounded if it can be partitioned into p phases.

Ordered MNWs [2,3] Let $[s]$ be the set of stacks with the natural ordering on them. We say that a MNW is ordered if for all stacks $i \in [s]$, there are no pending calls of any stack $j > i$ at the target of a $\curvearrowright^i$ edge. In other words, if there are many pending calls at any instant, the pending calls of the highest stack will return first, then the second highest and so on. This means that, when stack i is returning, all stacks higher than i are empty.

Scope Bounded or Ordered Returns MNWs (SBO) Let $[s]$ be the set of stacks with the natural ordering on them. We fix a parameter $m \in \mathbb{N}$. Given a MNW and the parameter m, we classify the nesting edges into *long* and *short*. A nesting edge is long if there are more than m different contexts between its source and target. It is short otherwise. We say that a MNW is SBO MNW if for all stacks $i \in [s]$, there are no pending long nesting edges of any stack $j > i$ at the target of a long nesting edge of i. In other words, if there are many pending long nesting edges at any instant, the pending long nesting edges of the highest stack will return first, then the second highest and so on. That is to say that, with respect to the long nesting edges, a SBO MNW behaves exactly like an ordered MNW.

Scope or Phase Bounded Returns MNWs (SPB) Given a MNW and the parameters m and p, as in the case of SBO we classify the nesting edges into *long* and *short* (wrt. the parameter m). We say that a MNW is (m,p)-SPB if it can be partitioned into p phases wrt. the long returns.

Proposition 6. *The classes Bounded Scope, Bounded Phase, Ordered,* SBO, SPB *are MSO definable.*

Proof. All the returns of a MNW have to satisfy certain conditions to belong to a class. These conditions are easily MSO-definable. □

All the above classes have bounded split-width.

Theorem 7. *1. m-Bounded scope MNWs have split-width at most $m+2$.*
2. p-Bounded phase MNWs have split-width at most 2^p.
3. Ordered MNWs have split-width at most 2^s.
4. m-SBO have split-width at most $2^s(2m+1)$.
5. (m,p)-SPB have split-width at most $2^p(2m+1))$.

The proof is given in Section 4.1 below.

Theorem 4 along with Proposition 6 and Theorem 7 gives us the MSO decidability of the classes defined in Section 4:

Corollary 8. *The classes Bounded Scope, Bounded Phase, Ordered,* SBO, SPB *have a decidable MSO theory.*

Theorem 2 along with Theorem 7 gives us new bounds of tree-width of the different classes of MNWs. We improve the $s2^{s-1}$ bound on tree-width of ordered MNWs obtained in [18] to 2^{s+1}. We also improve the $2ms$ bound on tree-width for bounded scope MNWs obtained in [15] to $2(m+2)$.

Corollary 9. *1. m-Bounded scope MNWs have tree-width at most $2(m+2)$.*
2. p-Bounded phase MNWs have tree-width at most 2^{p+1}.
3. Ordered MNWs have tree-width at most 2^{s+1}.
4. m-SBO have tree-width at most $2^{s+1}(2m+1)$.
5. (m,p)-SPB have tree-width at most $2^{p+1}(2m+1)$.

4.1 Bounded split-width

Proof of Bounded Split-Width of Bounded Scope MNWs Our idea is to split the first $m-1$ contexts of a bounded scope MNW into different components.

We write w_i to denote the ith component of a SMNW $\overline{w}$. Given an m-scope bounded MNW w, we repeatedly decompose it using the shuffle and merge operations till we are left with atomic SMNWs, ensuring that we stay within $(m+2)$-BS in this process. We maintain the invariant `INV2`: *All but the last component of the SMNWs are single contexts.* To begin, observe that any m-scope bounded MNW w is the merge of a SMNW $\overline{w}$ with at most m components, where the first $m-1$ components are the first $m-1$ contexts of w. We continue by applying the following rules:

1. If some component w_i is a complete MNW, let $\overline{v} = w_i$ and $\overline{u}$ be $\overline{w}$ without w_i. Clearly $\overline{w} \in \overline{u} \sqcup\!\sqcup \overline{v}$.
2. If some component w_i has a non trivial prefix or suffix which is a complete MNW, we split w_i into $u_i v_i$ (both nonempty) such that one of them, say v_i is a complete MNW. Let $\overline{v}$ be v_i and $\overline{u}$ be $\overline{w}$ without v_i. Clearly $\overline{w} \in \mathsf{merge}(\overline{u} \sqcup\!\sqcup \overline{v})$.
3. If there is a $\overset{i}{\curvearrowright}$-edge e whose source, labelled a, is the first node or last node of w_k and whose target, labelled b, is the first node or last node of w_ℓ, then $\overline{w} \in \mathsf{merge}(\overline{u} \sqcup\!\sqcup a \overset{i}{\dashrightarrow} b)$ where $\overline{u}$ is $\overline{w}$ without the edge e and its source and target nodes.
4. If the last component is w_j with $j < m$ and has more than one context, then we split the first context of the last component into a separate component. Repeated application of this rule yields as many components (but at most m) as possible.

Observe that if the invariant holds for $\overline{w}$ then the same holds for the two SMNWs obtained by the application of any of these four rules, thus the invariant `INV2` is maintained. Observe that the rules preserve another invariant `INV3`: *If there is a position x in ith component and a position y in jth component, then there are at least $|i-j|+1$ different contexts between x and y in the original MNW we started with.*

We will now argue that the above operations decompose the SMNW to base cases. Suppose, for the sake of contradiction, that a non-atomic SMNW $\overline{u}$ is obtained by the above operations from $\overline{w}$ and none of the above operations are applicable.

If for any stack there is a pending return in the first $m-1$ components, consider the first pending return which is in w_j. Let the corresponding call be in

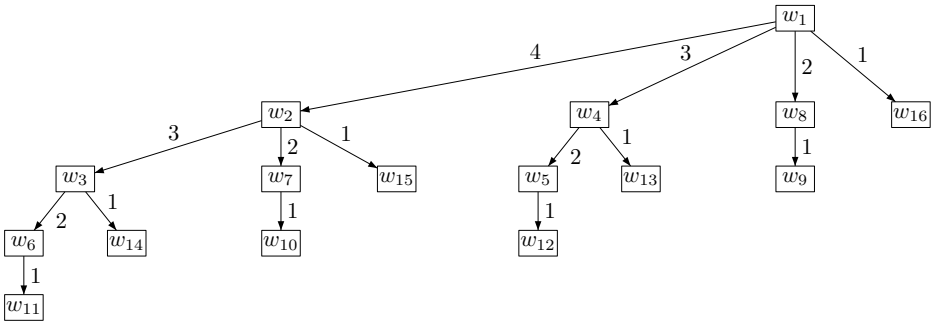

Fig. 1. A binomial tree of rank 5

w_i ($i < j$). Since we are not in case 2, the component w_i, which is single context, ends with this pending call and similarly w_j begins with this pending return, making case 3 applicable. Thus we may assume that in $\overline{w}$ there are no pending returns in any of the first $m-1$ components, and there are m components if the last component has at least two contexts. Since the first $m-1$ components cannot be complete MNWs (case 1 is not applicable) they must involve pending calls. Since they do not have complete MNWs as prefixes or suffices and are single context, each of them must begin and end with pending calls with the corresponding returns in w_m.

Claim: The first node of w_m necessarily has to be a pending return of the stack of w_1. The claim holds since a) the first context of w_m belong to the same stack as that of w_1 and also contains the pending returns called in w_1 (Otherwise there are more than m contexts switches between the first pending call and its corresponding return, thanks to invariant `INV3`). b) w_m cannot have a complete MNW as a prefix, as case 4 was not applicable. This makes case 3 applicable, contradicting the assumption that none of the above cases are applicable.

Notice that, just before any merge, the SMNW contains at most $m+2$ components. □

Proof of Bounded Split-Width of Ordered MNWs We show that any ordered MNW admits a decomposition in which the SMNWs have at most 2^s components. For that, we restrict the number of components of each SMNW to 2^{s-1} before any shuffle operation. A shuffle is followed by a few merge operations so that the bound of 2^{s-1} is maintained before the next shuffle.

The (2^{s-1})-SMNWs we obtain in the decomposition have some nice properties which let us embed them in a binomial tree of size 2^{s-1}. Each node in the binomial tree is a single component of the SMNW. The structure of the binomial tree is given in Figure 1 and is defined below.

A binomial tree is an edge labelled tree where each node has a rank. A node of rank i will have $i-1$ outgoing edges labelled with $i-1,\ldots,1$, and the j-child (child along the edge labeled j) will be a node of rank j. The rank of a binomial tree is the rank of its root. A binomial tree with rank k has height $k-1$ and has 2^{k-1} nodes. We identify a node by the path to that node from the root. In the

figure, root is identified by ε, the leftmost node by 4321 and the rightmost node by 1. The i-child of node x is xi. Note that the rank as well as the labels along any path from the root to a leaf are decreasing.

We say that a SMNW $\overline{w}$ has a *k-binomial embedding* if every component w_i of the SMNW can be assigned a node $\mathsf{node}(i)$ of a binomial tree of rank k such that no two components are assigned to the same node. We will shortly show that a SMNW $\overline{w}$ obtained from the decomposition of an ordered MNW has an s-binomial embedding, satisfying the following properties. We denote the s-binomial embedding of $\overline{w}$ by W. If $\mathsf{node}(i) = x$ under W, then we denote w_i by W_x in the following.

P1 There is a $\overset{i}{\curvearrowright}$ edge from a component w_k to another component w_l only if $\mathsf{node}(l)$ is the i-child of $\mathsf{node}(k)$.

P2 Let x be a node of rank i. All the returns in W_x are on a stack which is at least i.

If $s = 4$, and $\overline{w}$ has 16 nonempty components, a binomial embedding satisfying the above properties may assign nodes of the binomial tree to components as shown in Figure 1. One can verify that it is in fact the only possible binomial embedding satisfying the stack policy and the ordering policy.

Any ordered MNW w is a 1-SMNW. The binomial tree embedding embeds this only component at its leftmost child (node with id $(s-1)(s-2)\cdots 1$). That is, $\overline{w} = w = W_{(s-1)(s-2)\cdots 1}$. Clearly it satisfies the properties P1 and P2.

We show the decomposition by induction. Let $\overline{w}$ be a SMNW with a s-binomial embedding satisfying the properties P1 and P2. We do the following case splittings in a greedy manner (we will go to a case only if it is not possible to match any of the previous cases).

1. If there is a nesting edge $\overset{i}{\curvearrowright}$ whose source, labeled a, is the first or the last position of w_k and whose target, labeled b, is the first or the last position of w_l, then $\overline{w} \in \mathsf{merge}(\overline{u} \sqcup a \overset{i}{\dashrightarrow} b)$ where $\overline{u}$ is $\overline{w}$ without the nesting edge and its source and target nodes. Clearly $\overline{u}$ has a s-binomial embedding inherited from that of $\overline{w}$, satisfying properties P1 and P2.

2. If some w_i is of the form $u_i v_i$ where v_i is complete (there are no pending calls or returns in v_i) and u_i and v_i are nonempty, then $\overline{w} \in \mathsf{merge}(\overline{u} \sqcup \overline{v})$ where $\overline{u}$ is $\overline{w}$ minus v_i and $\overline{v}$ is v_i. Also, $\overline{u}$ has a binomial embedding U inherited from $\overline{w}$ and $\overline{v}$ has a binomial embedding V which embeds its only component at its leftmost child. We have a symmetric dual case when u_i is complete. Note that $\overline{u}$ and U as well as $\overline{v}$ and V satisfies the properties P1 and P2.

3. If W has two nonempty nodes x and y both containing no pending returns: Wlog. let y be of smaller rank if the ranks are different. Due to property P1, we can conclude that the subtree rooted at y is disconnected from the rest. $\overline{v}$ is obtained by projecting $\overline{w}$ to those components whose embedding is in the subtree rooted at y and $\overline{u}$ is $\overline{w}$ without $\overline{v}$. Let U be a binomial embedding identical to W on the subtree rooted at y and empty elsewhere, and V be identical to W everywhere, except on the subtree rooted at y where it is empty. Clearly $\overline{w} \in \overline{u} \sqcup \overline{v}$. Moreover, $\overline{u}$ and U as well as $\overline{v}$ and V satisfies the properties P1 and P2.

4. This splitting in this case is depicted in Figure 2. Let x be a non-empty node such that W_x is of the form U_xV_x where U_x and V_x are non-empty, and V_x does not have any pending return. We will split its children W_{xi} as $W_{xi} = V_{xi}U_{xi}$ such that all pending returns of U_{xi} are called in U_x and those of V_{xi} are called in V_x and there are no nesting edges between U_{xi} and V_{xi}. For this we can take U_{xi} to be the shortest suffix containing all the pending returns from U_x. Note that U_x is a prefix and U_{xi} is a suffix. This is because among all the nesting edges between W_x and W_{xi} (all of them belong to stack i, thanks to property P1), the first pending call will be returned last and the last pending call will be returned first. All the pending returns of U_{xi} should be called in U_x or V_{xi}. Since U_{xi} starts with a pending return of stack i whose call is in U_x , there are no pending returns of stack i in U_{xi} which is called in V_{xi}. Since the ordering policy on stacks is followed, there cannot be any pending returns of stack $j > i$ in U_{xi} which is called in V_{xi}. Due to property P2, there cannot be any returns of stacks $j < i$ in U_{xi}. Thus we can split its children W_{xi} as $W_{xi} = V_{xi}U_{xi}$. SImilarly, we split recursively all nodes in the subtree of x. For all y, $W_{xy} \in \mathsf{merge}(U_{xy} \sqcup\!\sqcup V_{xy})$ (In fact $W_{xy} = U_{xy}V_{xy}$ if $|y|$ is even, $W_{xy} = V_{xy}U_{xy}$ otherwise. For the nodes y which are not split by the above procedure, let $U_y = W_y$ and $V_y = \varepsilon$. Clearly $\overline{w} \in \mathsf{merge}(\overline{u} \sqcup\!\sqcup \overline{v})$ where $\overline{u}, \overline{v}$ are such that U and V are the binomial embeddings of $\overline{u}$ and $\overline{v}$. Once again, $\overline{u}$ and U as well as $\overline{v}$ and V satisfies the properties P1 and P2.

In fact if root of W (node ε) is non empty, then one of the above cases apply. We argue why. Let $w_1 = W_\varepsilon \neq \varepsilon$. If w_1 starts with an internal action, then it is a base case or case 2 or case 3 applies. If w_1 starts with a call to stack $j < s$, thanks to property P2, it is either a base case or case 1 or case 2 or case 3 is applicable. If it is a call to stack s, either case 1 or case 2 or case 4 is applicable.

5. From the above remark, the only remaining case is when root is empty. Let xi be the nonempty node of W with the highest rank (which is i). If W_{xi} does not contain any returns of stack i then we shift node xi to x followed by a shift of nodes xiy to xy. It can be verified that shifting of the nodes gives a binomial embedding satisfying the properties P1 and P2 . Hence we can safely assume that W is a binomial embedding and xi is the nonempty node of W with highest rank and that it contains a return of stack i. Consider the first return of stack i. We split W_{xi} into $W'_xW'_{xi}$ such that W'_{xi} is the shortest suffix containing all the returns of stack i. This will result in the splitting of the children of W_{xi} which are attached to W'_x or W'_{xi} similar to that in case 4. One can verify that $\overline{w'} \in \mathsf{merge}(\overline{w'})$. Once again $\overline{w}$ and its binomial embedding W' satisfies the properties P1 and P2. The splitting in this case is illustrated in Figure 3.

Notice that in each of the above cases, the length of the SMNW decreases, or the number of components increases (it is bounded by 2^{s-1}). Thus by induction, the proof follows. □

Proof of Bounded Split-Width of Bounded Phase The proof for this case is very similar to that of Ordered MNWs. We will only mention the main differences from that of ordered. For the sake of easiness, we will identify the

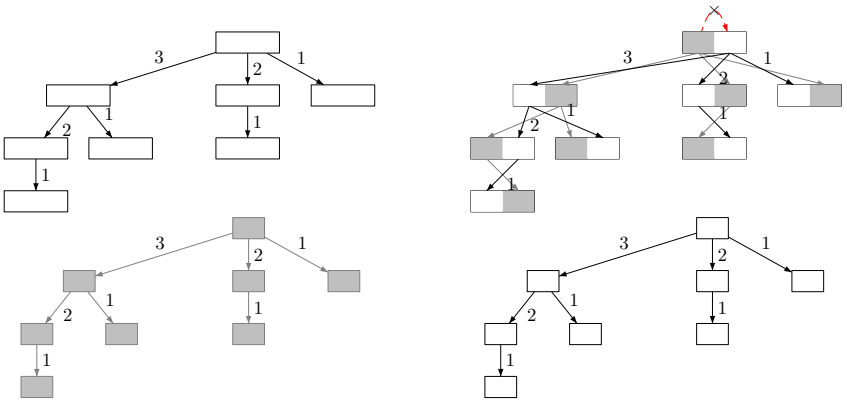

Fig. 2. Splitting of a binomial tree in case 4. Note the left/right alternation of gray (denotes U) and white (denotes V) parts along levels. This is needed since the stacks impose LIFO policy.

phases in the decreasing order. That is, the first phase is called phase_p, second phase is called phase_{p-1} and so on and the last phase is called phase_1.

As in the case of ordered MNWs, our SMNWs $\overline{w}$ will have a p-binomial embedding W satisfying the properties P1' and P2':

P1' There is a $\curvearrowright$edge from a component w_k to a component w_l only if $\mathsf{node}(l)$ is the i-child of $\mathsf{node}(k)$ and the return is in phase_i.

P2' If rank of x is i, then all the returns in W_x are in phase_j where $j \geq i$.

For the inductive decomposition, all the cases remain the same except for case 5. Let W_{xi} be the nonempty node of W with highest rank and assume that it contains at least one return from phase_i. We split W_{xi} into $W'_x W'_{xi}$ such that W'_{xi} is the shortest suffix containing all the returns of phase_i. The figures for ordered MNWs explains the splits for bounded phase as well, except that the edge labels of the binomial tree indicates the phase number of its children rather than the stack to which it belong. The bound follows. □

Proof of Bounded Split-Width of SBO and SPB The proof for this case is a joint generalization of the proof of bounded scope MNWs and that of ordered (resp. bounded phase) MNWs. We first split according to the long nesting edges and obtain a binomial embedding. In order to handle the short edges, we separate the outermost m contexts of this component so that a decomposition similar to that for bounded scope goes through. Thus we have a binomial tree embedding where instead of having a single component in a node of the binomial tree, we have $2m + 1$ components. The details can be found in [9].

5 Discussion and Perspectives

We have introduced and studied a new metric on MNWs called split-width and its relationship with clique-width and tree-width. Using split-width as a tool,

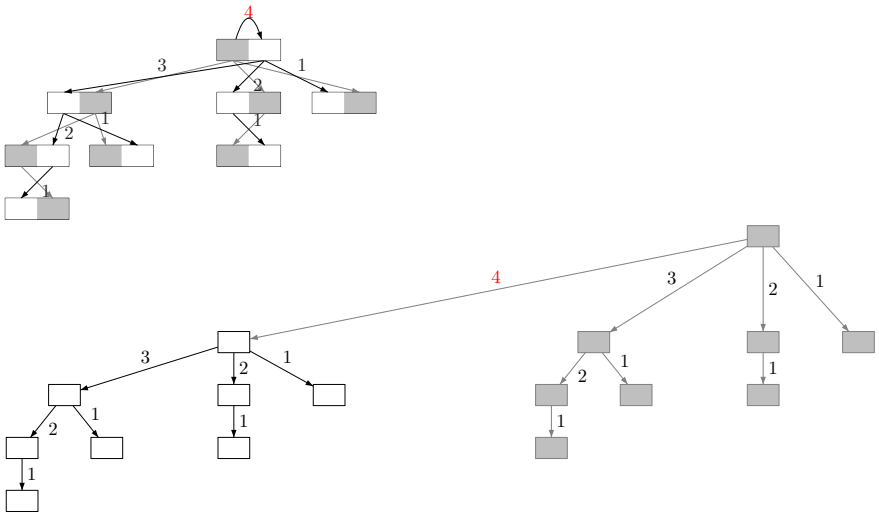

Fig. 3. Splitting of a binomial tree in case 5.

decidability of MSO for several existing as well as new classes of MPDS have been shown. We can even extend the decidable classes further.

An *i-pending-call-context* of a MNW w is a factor u of w in which there are no j-pending calls for $j \neq i$. A *pending-call-context* is an i-pending-call-context for some i.

The proof of bounded scope goes through to show that the same split-width bound of $m + 2$ holds for a generalization of bounded scope. The generalization allows at most m pending-call-contexts at every return. The classes SBO and SPB could be generalized further to replace bounded scope constraint on short returns by the generalization. These generalizations are MSO definable and the split-width remains unchanged.

A next step is to bridge the gap in the translations between split-width and tree-width (or clique-width). Is it possible to obtain a linear translation from tree-width to split-width? Is it possible to close the gap in the back and forth translations between split-width and tree-width (or clique-width)? In other words, is split-width another characterization of tree-width (or clique-width) of MNWs?

Another interesting question is whether MPDS with k bounded split-width restriction are closed under complementation. That is, given a MPDS $\mathcal{M}$ and k, is there another MPDS $\mathcal{M}'$ such that for all k-bounded split-width MNWS w, w is accepted by $\mathcal{M}$ if and only if w is not accepted by $\mathcal{M}'$?

It is interesting to know whether one could employ temporal logics instead of MSO for model checking MPDS wrt. k-split-width-bounded runs, and get a reasonable complexity.

Another important direction is to find notions similar to split-width for other domains like message sequence charts, data words etc.

References

1. R. Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3), 2009.
2. M. F. Atig, B. Bollig, and P. Habermehl. Emptiness of multi-pushdown automata is 2ETIME-Complete. In *Developments in Language Theory*, volume 5257 of *LNCS*, pages 121–133. Springer, 2008.
3. L. Breveglieri, A. Cherubini, C. Citrini, and S. Crespi-Reghizzi. Multi-pushdown languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996.
4. B. Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In *Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1993.
5. B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars*, pages 313–400. World Scientific, 1997.
6. B. Courcelle. The monadic second-order logic of graphs xv: On a conjecture by D. Seese. *Journal of Applied Logic*, 8:1–40, 2006.
7. B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. Syst. Sci.*, 46(2):218–270, 1993.
8. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.
9. A. Cyriac, P. Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. Research Report LSV-12-11, Laboratoire Spécification et Vérification, ENS Cachan, France, June 2012.
10. A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. In C.-H. L. Ong, editor, *FOSSACS'10*, volume 6014 of *LNCS*, pages 267–281. Springer, 2010.
11. S. Kreutzer. Algorithmic meta-theorems. *CoRR*, abs/0902.3616, 2009.
12. S. La Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *LICS'07*, pages 161–170. IEEE Computer Society, 2007.
13. S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In C. R. Ramakrishnan and J. Rehof, editors, *TACAS'08*, volume 4963 of *LNCS*, pages 299–314. Springer, 2008.
14. S. La Torre and M. Napoli. Reachability of multistack pushdown systems with scope-bounded matching relations. In J.-P. Katoen and B. König, editors, *CONCUR'11*, volume 6901 of *LNCS*, pages 203–218. Springer, 2011.
15. S. La Torre and G. Parlato. Scope-bounded multistack pushdown systems: fixed-point, sequentialization, and tree-width. Technical report, University of Southampton, February 2012.
16. A. Lal and T. W. Reps. Reducing concurrent analysis under a context bound to sequential analysis. *Formal Methods in System Design*, 35(1):73–97, 2009.
17. A. Lal, T. Touili, N. Kidd, and T. W. Reps. Interprocedural analysis of concurrent programs under a context bound. In C. R. Ramakrishnan and J. Rehof, editors, *TACAS'08*, volume 4963 of *LNCS*, pages 282–298. Springer, 2008.
18. P. Madhusudan and G. Parlato. The tree width of auxiliary storage. In T. Ball and M. Sagiv, editors, *POPL'11*, pages 283–294. ACM, 2011.
19. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In N. Halbwachs and L. D. Zuck, editors, *TACAS'05*, volume 3440 of *LNCS*, pages 93–107. Springer, 2005.
20. D. Seese. The structure of models of decidable monadic theories of graphs. *Ann. Pure Appl. Logic*, 53(2):169–195, 1991.