ON THE EXPRESSIVE POWER OF HIGHER-ORDER PUSHDOWN SYSTEMS

PAWEŁ PARYS

University of Warsaw, ul. Banacha 2, 02-097 Warszawa, Poland
e-mail address: parys@mimuw.edu.pl

Abstract. We show that deterministic collapsible pushdown automata of second order can recognize a language which is not recognizable by any deterministic higher-order pushdown automaton (without collapse) of any order. This implies that there exists a tree generated by a second order collapsible pushdown system (equivalently: by a recursion scheme of second order), which is not generated by any deterministic higher-order pushdown system (without collapse) of any order (equivalently: by any safe recursion scheme of any order). As a side effect, we present a pumping lemma for deterministic higher-order pushdown automata, which potentially can be useful for other applications.

1. Introduction

Already in the 70's, Maslov ([15, 16]) generalized the concept of pushdown automata to higher-order pushdown automata (n-PDA) by allowing the stack to contain other stacks rather than just atomic elements. In the last decade, renewed interest in these automata has arisen. They are now studied not only as acceptors of string languages, but also as generators of graphs and trees. It was an interesting problem whether the class of trees generated by n-PDA coincides with the class of trees generated by order-n recursion schemes. Knapik et al. [11] showed instead that this class coincides with the class of trees generated by *safe* order-n recursion schemes (safety is a syntactic restriction on the recursion scheme), and Caucal [3] gave another characterization: trees of order $n+1$ are obtained from trees of order n by an MSO-interpretation of a graph, followed by application of unfolding.

Driven by the question whether safety implies a semantical restriction to recursion schemes Hague et al. [6] extended the model of n-PDA to order-n collapsible pushdown automata (n-CPDA) by introducing a new stack operation called collapse (earlier, panic automata [12] were introduced for order 2), and proved that trees generated by n-CPDA coincide with trees generated by all order-n recursion schemes. Let us mention that these trees have decidable MSO theory [17], and that higher-order recursion schemes have close

1998 ACM Subject Classification: F.1.1. Models of computation—Relations between models.
Key words and phrases: Higher-order pushdown systems, collapse, higher-order recursion schemes.
This is the full version of the conference paper [20].
Work supported by the National Science Center (decision DEC-2012/07/D/ST6/02443). The author holds a post-doctoral position supported by Warsaw Center of Mathematics and Computer Science.

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

connections with verification of some real life higher-order programs [13]. Intuitively, the collapse operation allows the removal of all stacks on which a copy of the currently topmost stack symbol is present.

Nevertheless, it was still an open question whether these two hierarchies of trees are possibly the same hierarchy? This problem was stated in [11] and repeated in other papers concerning higher-order pushdown automata [12, 1, 17, 6]. A partial answer to this question was given in [18]: there is a tree generated by a 2-CPDA which is not generated by any 2-PDA. We prove the following stronger property.

Theorem 1.1. *There is a tree generated by a 2-CPDA (equivalently, by a recursion scheme of order 2) which is not generated by any n-PDA, for any n (equivalently, by any safe recursion scheme of any order).*

This confirms that the correspondence between higher-order recursion schemes and higher-order pushdown automata is not perfect. The tree used in Theorem 1.1 (after some adaptations) comes from [11] and from that time was conjectured to be a good example.

In this paper we work with PDA which recognize words instead of generating trees. While in general PDA used to recognize word languages can be nondeterministic, trees generated by PDA closely correspond to word languages recognized by deterministic PDA. Technically, we prove the following theorem, from which Theorem 1.1 follows.

Theorem 1.2. *There is a language recognized by a deterministic 2-CPDA which is not recognized by any deterministic n-PDA, for any n.*

As a side effect, in Section 9 we present a pumping lemma for higher-order pushdown automata. Although its formulation is not very natural, we believe it may be useful for some other applications. Our lemma is similar to the pumping lemma from [21] (see Section 9 for some comments). Earlier, several pumping lemmas related to the second order of the pushdown hierarchy were proposed [7, 5, 9].

This paper is an extended version of the conference paper [20]. The proof of Theorem 1.1 goes along the same line, but with essential differences in details. The part about types (Section 7) was simplified slightly, in the cost of complicating other parts (which was necessary since Theorem 7.3 is proven in a weaker form than in [20]).

1.1. **Related Work.** One may ask a similar question for word languages instead of trees: is there a language recognized by a CPDA which is not recognized by any (nondeterministic) PDA? This is an independent problem. The answer is known only for order 2 and is opposite: one can see that in 2-CPDA the collapse operation can be simulated by nondeterminism, hence 2-PDA and 2-CPDA recognize the same languages [1]. It is also an open question whether all word languages recognized by CPDA are context-sensitive.

In [19] we prove that the collapse operation increases the expressive power of deterministic higher-order pushdown automata with data. In this model of automata each letter from the input word is equipped by a data value, which comes from an infinite set; these data values can be stored on the stack and compared with other data values. In such setting the proof becomes easier than in the no-data case considered in this paper.

One can consider configuration graphs of n-PDA and n-CPDA, and their ε-closures. We know [6] that there is a 2-CPDA whose configuration graph has undecidable MSO theory, hence which is not a configuration graph of an n-PDA, nor an ε-closure of such, as they all have decidable MSO theory.

Engelfriet [4] showed that the hierarchies of word languages and of trees generated by PDA are strict (that is, for each n there is a language recognized by an n-PDA which is not recognized by any $(n-1)$-PDA, and similarly for trees). In fact his proof works equally well for these hierarchies for CPDA, once we know that the reachability problem for n-CPDA is $(n-1)$-EXPTIME complete (which follows from [14]).

2. Preliminaries

For natural numbers a, b, where $b \geq a-1$, by $[a,b]$ we denote the set $\{a,\dots,b\}$ (which is empty if $b = a-1$).

For any alphabet Γ (of stack symbols) we define a *stack of order k* (k*-stack* for short) as an element of the following set Γ^k_*:

$$\Gamma^0_* = \Gamma, \qquad \Gamma^0_+ = \Gamma,$$
$$\Gamma^k_* = (\Gamma^{k-1}_+)^*, \qquad \Gamma^k_+ = (\Gamma^{k-1}_+)^+ \qquad \text{for } k \geq 1.$$

In other words, a 0-stack is just a single symbol, and a k-stack for $k \geq 1$ is a (possibly empty) sequence of nonempty $(k-1)$-stacks. Top of a stack is on the right. The *size* of a k-stack is just the number of $(k-1)$-stacks it contains. For any $s^k \in \Gamma^k_*$ and $s^{k-1} \in \Gamma^{k-1}_+$ we write $s^k : s^{k-1}$ for the k-stack obtained from s^k by placing s^{k-1} at its end. The operator ":" is assumed to be right associative, i.e. $s^2 : s^1 : s^0 = s^2 : (s^1 : s^0)$.

When Γ is fixed, the *stack operations* of order $k \geq 1$ are pop^k and push^k_γ for each $\gamma \in \Gamma$. We can apply them to a nonempty r-stack for $r \geq k$, which gives:

- $\mathsf{pop}^k(s^r : s^{r-1} : \cdots : s^k : s^{k-1}) = s^r : s^{r-1} : \cdots : s^k$, i.e. we remove the topmost $(k-1)$-stack; it is defined only when the topmost k-stack contains at least two $(k-1)$-stacks;
- $\mathsf{push}^k_\gamma(s^r : s^{r-1} : \cdots : s^0) = s^r : s^{r-1} : \cdots : s^{k+1} : (s^k : s^{k-1} : \cdots : s^0) : s^{k-1} : s^{k-2} : \cdots : s^1 : \gamma$, i.e. we duplicate the topmost $(k-1)$-stack, and then we replace the topmost 0-stack by γ.[1]

A *deterministic word-recognizing pushdown automaton of order n* (n*-DPDA* for short) is a tuple $(A, \Gamma, \gamma_I, Q, q_I, F, \delta)$ where A is an input alphabet, Γ is a stack alphabet, $\gamma_I \in \Gamma$ is an initial stack symbol, Q is a set of states, $q_I \in Q$ is an initial state, $F \subseteq Q$ is a set of accepting states, and δ is a transition function which maps every element of $Q \times \Gamma$ into one of the following objects:

- $\mathsf{read}(\vec{q})$, where $\vec{q} : A \to Q$ is an injective function,
- (q, op) where $q \in Q$ and op is a stack operation of order at most n.

The letter n is used exclusively for the order of pushdown automata.

A *configuration* of $\mathcal{A}$ consists of a state and of a nonempty n-stack, i.e. is an element of $Q \times \Gamma^n_+$. The *initial* configuration consists of the initial state q_I and of the n-stack containing only one 0-stack, which is the initial stack symbol γ_I. We use the notation $\pi_i((p_1,\dots,p_k)) = p_i$; in particular for a configuration c, $\pi_1(c)$ denotes its state, and $\pi_2(c)$ its stack. Additionally, for a set X of tuples we define $\pi_i(X)$ to be $\{\pi_i(p) \colon p \in X\}$.

We use a shorthand $\delta(c)$ for a configuration c to denote $\delta(\pi_1(c), s^0)$, where s^0 is the topmost 0-stack of c. A configuration d is a *successor* of a configuration c, if:

[1] In the classical definition the topmost symbol can be changed only when $k = 1$ (for $k \geq 2$ it required that $\gamma = s^0$). We make this (not important) extension to have a uniform definition of push^k for all k.

- $\delta(c) = \mathsf{read}(\vec{q})$, and $d = (\vec{q}(a), \pi_2(c))$ for some $a \in A$, or
- $\delta(c) = (q, op)$, and $d = (q, op(\pi_2(c)))$.

Typically a configuration has exactly one successor. However when the transition is $\mathsf{read}(\vec{q})$, there are $|A|$ successors. It is also possible that there are no successors: when the operation is pop^k but there is only one $(k-1)$-stack on the topmost k-stack.

Next, we define a *run* of $\mathcal{A}$. For $0 \le i \le m$, let c_i be a configuration. A run R from c_0 to c_m is a sequence $c_0, c_1, \dots, c_m$ such that, for each $i \in [1, m]$, c_i is a successor of c_{i-1}. We set $R(i) = c_i$ and call $|R| = m$ the *length of* R. The *subrun* $R{\restriction}_{i,j}$ is $c_i, c_{i+1}, \dots, c_j$. For runs R, S with $R(|R|) = S(0)$, we write $R \circ S$ for the *composition* of R and S which is defined as expected. Sometimes we also consider infinite runs, such that the sequence $c_0, c_1, c_2, \dots$ is infinite. However, unless stated explicitly, a run is finite.

The *word read by a run* is a word over the input alphabet A. For a run from a configuration c to its successor d, it is the empty word if the transition between them is of the form (q, op). If the transition is $\mathsf{read}(\vec{q})$, this is the one-letter word consisting of the letter a for which $\pi_1(d) = \vec{q}(a)$ (this letter is determined uniquely, as f is injective). For a longer run R this is defined as the concatenation of the words read by the subruns $R{\restriction}_{i-1,i}$ for $i \in [1, |R|]$. A run is *accepting* if it ends in a configuration whose state is accepting. A word w is *accepted by* $\mathcal{A}$ if it is read by some accepting run starting in the initial configuration. The *language recognized by* $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$.

2.1. **Collapsible 2-DPDA.** In Section 4 we also use deterministic collapsible pushdown automata of order 2 (2-DCPDA for short). Such automata are defined like 2-DPDA, with the following differences. A 0-stack contains now two parts: a symbol from Γ, and a natural number, but still only the symbol (together with a state) is used to determine which transition is performed from a configuration. The push^1_γ operation sets the number in the topmost 0-stack to the current size of the 2-stack (while push^2_γ does not modify these numbers). We have a new stack operation $\mathsf{collapse}$. Its result $\mathsf{collapse}(s)$ is obtained from s by removing its topmost 1-stacks, so that only $k-1$ of them is left, where k is the number stored in the topmost 0-stack of s (intuitively, we remove all 1-stacks on which the topmost 0-stack is present).

3. Relation between Word Languages and Trees

In this section we describe how word languages recognized by DPDA are related to trees generated by PDA. Before seeing how Theorem 1.2 it implies Theorem 1.1, we need to define how n-PDA are used to generate trees. We consider ranked, potentially infinite trees. Beside of the input alphabet A we have a function $rank\colon A \to \mathbb{N}$; a tree node labelled by some $a \in A$ has always $rank(a)$ children.

Automata used to generate trees are defined like DPDA or DCPDA (in particular they are deterministic as well), with the difference that they do not have the set of accepting states, and that instead of the $\mathsf{read}(\vec{q})$ transitions, there are $\mathsf{branch}(a, q_1, q_2, \dots, q_{rank(a)})$ transitions, for $a \in A$, and for pairwise distinct states $q_1, q_2, \dots, q_{rank(a)} \in Q$. If the transition from c is $\delta(c) = \mathsf{branch}(a, q_1, q_2, \dots, q_{rank(a)})$, in a successor d of c we have $\pi_2(d) = \pi_2(c)$ and $\pi_1(d) = q_i$ for some $i \in [1, rank(a)]$ (in particular c has no successors if $rank(a) = 0$).

Let $T(\mathcal{A})$ be the set of all configurations c of $\mathcal{A}$ reachable from the initial one, such that a branch transition should be performed from c. If there is a configuration of $\mathcal{A}$ reachable

from the initial one, from which there is no run to a configuration from $T(\mathcal{A})$, by definition $\mathcal{A}$ does not generate any tree. Otherwise, a *tree generated by* $\mathcal{A}$ has runs from the initial configuration to a configuration from $T(\mathcal{A})$ as its nodes. A node R is labelled by $a \in A$ such that $\delta(R(|R|)) = \mathsf{branch}(a, q_1, q_2, \ldots, q_{rank(a)})$. A node S is its i-th child ($1 \leq i \leq rank(a)$), if S is the composition of R and a run S' which uses a branch transition only in its first transition, and $\pi_1(S'(1)) = q_i$. Notice that the graph obtained this way is really an A-labelled ranked tree.

We will now see how Theorem 1.1 follows from Theorem 1.2. Let $L \subseteq A^*$ be the language recognized by a 2-DCPDA $\mathcal{A}$ which is not recognized by any n-DPDA, for any n (L exists by Theorem 1.2). First, we transform $\mathcal{A}$ into a 2-DCPDA $\mathcal{B}$, recognizing L as well, such that each configuration of $\mathcal{B}$ reachable from the initial one has a successor. Observe that the only reason why in $\mathcal{A}$ there may be configurations with no successors is that it wants to empty a stack using a pop operation. To avoid such situations, $\mathcal{B}$ should have some bottom-of-stack marker on the bottom of each 1-stack, and on the bottom of the 2-stack (a 1-stack containing only the marker). Thus $\mathcal{B}$ starts with the marker as the initial stack symbol, performs push^2 and push^1, placing the original initial stack symbol. Then whenever $\mathcal{A}$ blocks because it wants to empty a stack, in $\mathcal{B}$ the bottom-of-stack marker is uncovered; in such situation $\mathcal{B}$ starts some loop with no accepting state. There is also a technical detail, that a pop operation which would block $\mathcal{A}$, in $\mathcal{B}$ can enter an accepting state; to overcome this problem, every pop operation ending in an accepting state should first end in some auxiliary, not accepting state, from which (if the bottom-of-stack marker is not seen) an accepting state is reached.

Next, we create a tree-generating 2-CPDA $\mathcal{C}$, which generates a tree over the alphabet $B = \{X, Y, Z\}$, where $rank(X) = |A|$ and $rank(Y) = rank(Z) = 1$. It is obtained from $\mathcal{B}$ in two steps. First, we replace each transition $\mathsf{read}(\vec{q})$ of $\mathcal{B}$ by the transition $\mathsf{branch}(X, \vec{q}(a_1), \vec{q}(a_2), \ldots, \vec{q}(a_{|A|}))$, where $A = \{a_1, \ldots, a_{|A|}\}$. Then, in each transition we replace the resulting state q by its auxiliary copy $\overline{q}$, and from $\overline{q}$ (for any topmost stack symbol) we perform transition $\mathsf{branch}(Y, q)$ if q was accepting, or transition $\mathsf{branch}(Z, q)$ if q was not accepting (this way, after each step of the original automaton, we perform a transition $\mathsf{branch}(Y, \cdot)$ or $\mathsf{branch}(Z, \cdot)$). Notice that from each configuration of $\mathcal{C}$ reachable from the initial one, there exists a run to a configuration from $T(\mathcal{C})$, as required by the definition of a tree-generating CPDA. Let $t_{\mathcal{C}}$ be the tree generated by $\mathcal{C}$.

Finally, suppose that $t_{\mathcal{C}}$ can also be generated by some n-PDA $\mathcal{D}$ (without collapse). From $\mathcal{D}$ we create a word-recognizing n-DPDA $\mathcal{E}$. We replace each transition of the form $\mathsf{branch}(X, q_1, q_2, \ldots, q_{|A|})$ of $\mathcal{D}$ by the transition $\mathsf{read}(\vec{q})$, where $\vec{q}(a_i) = q_i$. We replace each transition $\mathsf{branch}(Y, q)$ of $\mathcal{D}$ by the transition $(p, \mathsf{push}^1_\gamma)$ for a fresh accepting state p and some stack symbol γ; from (p, γ) we perform the transition (q, pop^1) (thus we replace $\mathsf{branch}(Y, q)$ by a pass through an accepting state). The same for a $\mathsf{branch}(Z, q)$ transition, but the fresh state p is not accepting.

Notice that $\mathcal{E}$ recognizes L; this contradicts our assumptions about L, so $t_{\mathcal{C}}$ is not generated by any n-PDA. Indeed, take any word $w \in L$. We have an accepting run of $\mathcal{B}$ which reads w and starts in the initial configuration. This run corresponds to a run of $\mathcal{C}$, that is to a path p in $t_{\mathcal{C}}$ from the root to a Y-labelled node. Letters of w tell us which child the path p chooses in X-labelled nodes: if i-th letter of w is a_j, then from the i-th X-labelled node of p, the path continues to the j-th child. This path p corresponds also to a run of $\mathcal{D}$, so to a run of $\mathcal{E}$. This run starts in the initial configuration, ends with an

accepting state, and reads w; thus $\mathcal{E}$ accepts w. Similarly, each word accepted by $\mathcal{E}$ is also accepted by $\mathcal{B}$.

We also recall that a tree is generated by a recursion scheme of order 2 if and only if it is generated by a 2-CPDA [6], and that a tree is generated by a safe recursion scheme of order n if and only if it is generated by an n-PDA [11]; this implies the "equivalently" parts of Theorem 1.1.

4. The Separating Language

In this section we define a language U which can be recognized by a 2-CPDA, but not by any n-DPDA, for any n. It is a language over the alphabet $A = \{[,], \star, \sharp\}$. For a word $w \in \{[,], \star\}^*$ we define $\mathit{stars}(w)$. Whenever in some prefix of w there are more closing brackets than opening brackets, $\mathit{stars}(w) = 0$. Also when in the whole w we have the same number of opening and closing brackets, $\mathit{stars}(w) = 0$. Otherwise, let $\mathit{stars}(w)$ be the number of stars in w before the last opening bracket which is not closed. Let U be the set of words $w\sharp^{\mathit{stars}(w)+1}$, for any $w \in \{[,], \star\}^*$ (i.e. these are words w consisting of brackets and stars, followed by $\mathit{stars}(w) + 1$ sharp symbols).

It is known that languages similar to U can be recognized by a 2-CPDA (e.g. [1]), but for completeness we briefly show it below. The collapsible 2-CPDA will use three stack symbols: X (used to mark the bottom of 1-stacks), Y (used to count brackets), Z (used to mark the bottommost 1-stack). The initial symbol is X. The automaton first pushes Z, makes a copy of the 1-stack (i.e. push^2), and pops Z (hence the first 1-stack is marked with Z, unlike any other 1-stack used later). Then, for an opening bracket we push Y, for a closing bracket we pop Y, and for a star we perform push^2. Hence for each star we have a 1-stack and on the last 1-stack we have as many Y symbols as the number of currently open brackets. If for a closing bracket the topmost symbol is X, it means that in the word read so far we have more closing brackets than opening brackets; in this case we should accept suffixes of the form $\{[,], \star\}^*\sharp$, which is easy.

Finally the $\sharp$ symbol is read. If the topmost symbol is X, we have read as many opening brackets as closing brackets, hence we should accept one $\sharp$ symbol. Otherwise, the topmost Y symbol corresponds to the last opening bracket which is not closed. We execute the $\mathsf{collapse}$ operation. It leaves the 1-stacks created by the stars read before this bracket, except one (plus the first 1-stack). Thus the number of 1-stacks is precisely equal to $\mathit{stars}(w)$. Now we should read as many $\sharp$ symbols as we have 1-stacks, plus one (after each $\sharp$ symbol we make pop^2), and then accept.

In the remaining part of the paper we prove that any n-DPDA cannot recognize U; in particular all automata appearing in the following sections does not use collapse.

5. Overview of the Proof

Before we start the real proof, in this section we present its general structure, on the intuitive level. Let us first see why U cannot be recognized by any 1-DPDA $\mathcal{A}$. Consider the input word

$$w_1 = [\star^{n_1}[\star^{n_2} \dots [\star^{n_N}[\star^{m_{N+1}}]\star^{m_N}] \cdots \star^{m_1}]\star^{m_0}[$$

(where each bracket is matched, except the last opening bracket). Notice that $\mathit{stars}(w_1)$ equals to the sum of all n_1 and m_i, so $\mathcal{A}$ after reading w_1 has to store all these numbers in

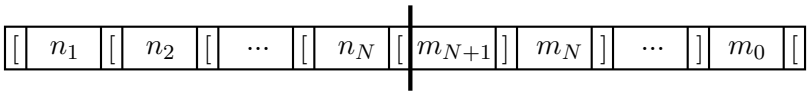

Figure 1: The stack of a 1-DPDA after reading word w_1

its stack. Thus it first stores the number n_1 on the stack (by repeating some stack symbol n_1 times), then it can mark that there was an opening bracket, then it stores n_2, and so on (see Figure 1); none of these numbers can be removed later. Now consider the prefix $w_{1,i}$ of w_1, which is cut just after the i-th closing bracket. Since $\mathcal{A}$ is deterministic, the stack at the end of $w_{1,i}$ looks similarly: it is just shorter, but for sure it ends to the right of the vertical line, which denotes the stack size after the last opening bracket. We see that $\mathit{stars}(w_{1,i}) = n_1 + \cdots + n_{N-i}$. Thus when $\mathcal{A}$ sees a $\sharp$ after $w_{1,i}$, it has to remove (ignore) the numbers above n_{N-i}, and sum the rest. In particular it passes the vertical line in some state q_i. We see that for each i, at the moment of crossing this line the stack is the same (everything to the right of the line is removed), only the state q_i can differ. So in fact each q_i has to be different, since for each i we expect a different behavior. This is a contradiction when N is greater than the number of states.

It follows that $\mathcal{A}$ is of order at least 2, and while reading w_1 at some moment a push^2 has to be performed, where in the topmost 1-stack we don't remember some of the numbers n_i or m_i (for example to recognize w_1 after each] we can perform a push^2 and remove a fragment of the 1-stack so that the matching opening bracket is on the top). But now we can consider the word

$$w_2 = w_1 \star^{n'_1} w_1 \star^{n'_2} \dots w_1 \star^{n'_N} w_1 \star^{m'_{N+1}}] \star^{m'_N}] \cdots \star^{m'_1}] \star^{m'_0} [\,,$$

where the numbers n_i, m_i in each copy of w_1 are independent (so in fact each w_1 is a different word). Notice that each w_1 ends by an unmatched opening bracket; they are matched by the closing brackets at the end of w_2. We can now almost repeat the previous reasoning. First, $\mathit{stars}(w_2)$ equals to the sum of all numbers so they all have to be kept on the stack. Then, we draw a line after reading the last w_1 (that is, separating the 1-stacks created before that moment from those created later). By the order-1 argument, some number from each w_1 is not present in the topmost 1-stack after reading this w_1, so it cannot be present above the line. Next, for each i we try to end the word already after the i-th closing bracket (among those at the end of w_2, not those inside words w_1). When we have a $\sharp$ after each of these prefixes, we have to go below the line and behave differently (include a different subset of those values which are not present above the line), so we have to cross the line in different states. This is again a contradiction when N is greater than the number of states. By induction we can continue like this, and nesting the words w_n again we can show that for each order of the DPDA there is a problem.

Although the above idea of the proof looks simple, formalizing it is not straightforward. We have to deal with the following issues:

(1) Above we have argued why an 1-DPDA cannot deal correctly with the word w_1. But in fact we should consider any n-DPDA, and prove that it is impossible that it stores all numbers from w_1 inside one 1-stack. Then there arises a problem that when crossing "the line" it is no longer true that the stack can only be of one form. Indeed, the topmost 1-stack has one fixed form, but we can cross the line in a copy of this 1-stack, with anything below this 1-stack. And even we can cross the line

multiple times, in several copies of the 1-stack. Thus it is no longer true that the number of states gives the number of ways in which we can visit a substack. The ways of visiting a substack will be described by types of stacks, defined in Section 7. The key point is that there will be finitely many types for a fixed DPDA.

(2) Where exactly is a number stored in a stack? And where exactly "the line" should be placed? This is not sharp, since a DPDA may delay some stack operations by keeping information in its state, as well as it may temporarily create some fancy redundant structures on the stack, which are removed later in the run. To deal with this issue, in Section 8 we define milestone configurations. Intuitively, these are configurations in which no additional garbage is present on the stack.

(3) Finally, why it will be wrong when, while reading the $\sharp$ symbols, the automaton will not visit a place where there is stored a number which is a part of $\mathit{stars}(\cdot)$? Maybe, accidentally, this number is equal to some other amount in the stack. Or maybe it was propagated to some other region on the stack by some involved manipulations. To overcome this difficulty, in Section 9 we prove a pumping lemma. It allows to change any of the numbers in the input word, without altering too much the whole stack. If some number (included in $\mathit{stars}(\cdot)$) is changed, the DPDA has to enter the part of the stack changed by the pumping lemma; otherwise it will incorrectly accept after the same number of the $\sharp$ symbols for two words with different $\mathit{stars}(\cdot)$.

6. The History Function, and Special Runs

We begin this section by defining positions and the history function. Then we define two classes of runs which are particularly interesting for us, namely k-upper runs, and k-returns.

A *position* is a vector $x = (x_n, x_{n-1}, \dots, x_1)$ of n positive integers. The symbol at position x in a configuration c (which is an element of the stack alphabet) is defined in the natural way (we take the x_n-th $(n-1)$-stack of $\pi_2(c)$, then its x_{n-1}-th $(n-2)$-stack, and so on; elements of stacks are numbered from bottom to top). We say that x is a position of c if at position x there is a symbol in c.

For any run R and any position y of $R(|R|)$, we define a position $\mathsf{hist}(R, y)$. Intuitively, $\mathsf{hist}(R, y)$ is the (unique) position of $R(0)$, from which the symbol was copied to y in $R(|R|)$. Precisely, $\mathsf{hist}(R, y) = y$ when $|R| = 0$. For a longer run $R = S \circ T$ with $|T| = 1$ we define it by induction. We take $\mathsf{hist}(R, y) = \mathsf{hist}(S, y)$ if the last transition of R is read or performs pop, as well as if the transition performs push^k and y is not in the topmost $(k-1)$-stack of $R(|R|)$. If the last transition of R performs push^k and y is in the topmost $(k-1)$-stack of $R(|R|)$, then $\mathsf{hist}(R, y) = \mathsf{hist}(S, z)$, where z is equal to y with the $(n-k+1)$-th coordinate decreased by 1 (i.e., z is the position of $T(0)$ from which a symbol was copied to y). Notice that (for technical convenience) hist works in this way also for the topmost position.

We observe that if two positions x and y of $R(|R|)$ are in the same k-stack, for some k, then their histories $\mathsf{hist}(R, x)$ and $\mathsf{hist}(R, y)$ are also in the same k-stack. Moreover, if x is a position of the bottommost symbol in some k-stack, then $\mathsf{hist}(R, x)$ as well. Thus to trace a history of a k-stack it is convenient to look at the history of its bottommost element. For $k \in [0, n]$, by $\mathsf{top}^k(c)$ we denote the position of the bottommost symbol of the topmost k-stack of c. In particular $\mathsf{top}^0(c)$ is the topmost position of c.

For $k \in [0, n]$, we say that a run R is *k-upper* if $\mathsf{hist}(R, \mathsf{top}^k(R(|R|))) = \mathsf{top}^k(R(0))$; let up^k be the set of all such runs. Intuitively, a run R is k-upper when the topmost k-stack of $R(|R|)$ is a copy of the topmost k-stack of $R(0)$, but possibly some changes were made to

Table 1: Stack contents of the example run, and subruns being k-upper runs and k-returns

j	$\pi_2(R(j))$	$\{i\colon R\restriction_{i,j} \in \mathsf{up}^0\}$	$\{i\colon R\restriction_{i,j} \in \mathsf{up}^1\}$	$\{i\colon R\restriction_{i,j} \in \mathsf{ret}^1\}$	$\{i\colon R\restriction_{i,j} \in \mathsf{ret}^2\}$
0	$[ab][cd]$	$\{0\}$	$\{0\}$	$\emptyset$	$\emptyset$
1	$[ab][cd][ce]$	$\{0,1\}$	$\{0,1\}$	$\emptyset$	$\emptyset$
2	$[ab][cd][c]$	$\{2\}$	$\{0,1,2\}$	$\{0,1\}$	$\emptyset$
3	$[ab][cd]$	$\{0,3\}$	$\{0,3\}$	$\emptyset$	$\{1,2\}$
4	$[ab][c]$	$\{4\}$	$\{0,3,4\}$	$\{0,3\}$	$\emptyset$
5	$[ab][cd]$	$\{4,5\}$	$\{0,3,4,5\}$	$\emptyset$	$\emptyset$
6	$[ab][c]$	$\{4,6\}$	$\{0,3,4,5,6\}$	$\{5\}$	$\emptyset$

it. Notice that up^n contains all runs, $\mathsf{up}^k \subseteq \mathsf{up}^l$ for $k \leq l$, and for a run $R \circ S$ with $S \in \mathsf{up}^k$ it holds $R \in \mathsf{up}^k \iff R \circ S \in \mathsf{up}^k$.

For $k \in [1,n]$, a run R is a *k-return* if

- $\mathsf{hist}(R, \mathsf{top}^{k-1}(R(|R|)))$ is the bottommost position of the second topmost $(k-1)$-stack[2] of $R(0)$, and
- $R\restriction_{i,|R|} \notin \mathsf{up}^{k-1}$ for all $i \in [0, |R|-1]$.

Let ret^k be the set of k-returns. Observe that $\mathsf{ret}^k \subseteq \mathsf{up}^k$. Intuitively, R is an r-return when the topmost r-stack of $R(|R|)$ is obtained from the topmost r-stack of $R(0)$ by removing its topmost $(r-1)$-stack (but not only in the sense of contents, but we require that really it was obtained this way).

Example 6.1. Consider a DPDA of order 2. Below, brackets are used to group symbols in one 1-stack. Consider a run R of length 6 in which $\pi_2(R(0)) = [ab][cd]$, and the operations between consecutive configurations are:

$$\mathsf{push}^2_e,\ \mathsf{pop}^1,\ \mathsf{pop}^2,\ \mathsf{pop}^1,\ \mathsf{push}^1_d,\ \mathsf{pop}^1.$$

Recall that our definition is that a push of any order can change the topmost stack symbol. The contents of the stacks of the configurations in the run, and subruns being k-upper runs and k-returns are presented in Table 1. Notice that R is not a 1-return. In configuration $R(0)$ symbol a is at position $(1,1)$ and symbol b is at position $(1,2)$. We have $\mathsf{hist}(R\restriction_{0,5}, (2,2)) = (2,1)$. Notice that positions y in $S(|S|)$ and $\mathsf{hist}(S,y)$ in $S(0)$ do not necessarily contain the same symbol, as for example at position $(2,2)$ in $R(5)$ we have d and at position $(2,1)$ in $R(0)$ we have c.

6.1. **Basic Properties of Runs.** Next we state several easy propositions, which are useful later, and also give more intuition about the above definitions.

Proposition 6.2. *Let R be a k-upper run (where $k \in [0,n]$) such that $R\restriction_{i,|R|} \notin \mathsf{up}^k$ for each $i \in [1, |R|-1]$. Then either*

- *the topmost k-stacks of $R(0)$ and $R(|R|)$ are equal; additionally for every position x in the topmost k-stack of $R(|R|)$, $\mathsf{hist}(R,x)$ is the corresponding position in the topmost k-stack of $R(0)$, or*
- *$|R| = 1$ and the only transition of R performs pop^r for $r \leq k$, or push^r_γ for $r \leq k$.*

[2]By the second topmost $(k-1)$-stack we always mean the $(k-1)$-stack just below the topmost $(k-1)$-stack, in the same k-stack; in particular we require that the topmost k-stack has size at least 2.

Proof. For $|R| \leq 1$ we immediately fall into one of the possibilities. Otherwise, we look at the history of the topmost k-stack of $R(|R|)$. It is covered by the first operation of R, and then it is not the topmost k-stack until $R(|R|)$. Thus it remains unchanged (we have the first possibility). □

Next, we give four propositions about k-upper runs and k-returns.

Proposition 6.3. *Let R be a k-upper run, where $k \in [1, n]$. Then R is $(k-1)$-upper if and only if the size of the topmost k-stack of $R(0)$ is not greater than the size of the topmost k-stack of $R(i)$ for each $i \in [0, |R|]$ such that $R{\restriction_{i,|R|}} \in \mathsf{up}^k$.*

Proposition 6.4. *Let $S \circ T$ be a $(k-1)$-upper run in which T is k-upper, where $k \in [1, n]$. Then S is $(k-1)$-upper.*

Proposition 6.5. *Let R be a run such that $R{\restriction_{0,|R|-1}} \in \mathsf{up}^{k-1}$ and $R{\restriction_{|R|-1,|R|}} \in \mathsf{up}^k$, but $R \notin \mathsf{up}^{k-1}$ (where $k \in [1, n]$). Then R is a k-return.*

Proposition 6.6. *Let R be a k-return, where $k \in [1, n]$. Then the topmost k-stack of $R(0)$ after removing its topmost $(k-1)$-stack is equal to the topmost k-stack of $R(|R|)$. Additionally for every position x in the topmost k-stack of $R(|R|)$, $\mathsf{hist}(R, x)$ is the corresponding position in the topmost k-stack of $R(0)$.*

Proof of Propositions 6.3-6.6. In all four propositions we have an k-upper run R, where $k \in [1, n]$ (where for Proposition 6.4 we take $R = S \circ T$). Let X denote the set of those indices $i \in [0, |R|]$ for which $R{\restriction_{i,|R|}}$ is k-upper. For $i \in X$, let r_i be the size of the topmost k-stack of $R(i)$, and let $y_i(r)$ be the bottommost position of the r-th $(k-1)$-stack in the topmost k-stack of $R(i)$ (for $r \in [1, r_i]$). We will see that for each $i \in X$ and each $r \in [1, r_i]$ it holds

$$\mathsf{hist}(R{\restriction_{0,i}}, y_i(r)) = y_0\big(\min(\{r\} \cup \{r_l \colon l \in X \wedge l < i\})\big). \tag{6.1}$$

We prove (6.1) by induction on i. For $i = 0$ it is true. For the induction step consider the smallest $j \in X$ which is greater than i. The subrun $R{\restriction_{i,j}}$ is in one of the forms described by Proposition 6.2. For both of them we see that for each $r \in [1, r_j]$ it holds

$$\mathsf{hist}(R{\restriction_{i,j}}, y_j(r)) = y_i(\min\{r, r_i\}).$$

Together with the induction assumption for i, this implies equality (6.1) for j.

Observe that R is $(k-1)$-upper if and only if $\mathsf{hist}(R, y_{|R|}(r_{|R|})) = y_0(r_0)$. As we see from (6.1), the right side holds if and only if $r_0 \leq r_i$ for each $i \in X$. This proves Proposition 6.3.

In order to prove Proposition 6.4, we suppose that R is $(k-1)$-upper. Using the above property, $r_0 \leq r_i$ for each $i \in X$. But $R{\restriction_{i,|R|}}$ can be k-upper only for $i \in X$. Thus the same property for S implies that S is $(k-1)$-upper, as required.

As above, the assumptions of Proposition 6.5 imply that $r_0 \leq r_i$ for each $i \in X \setminus \{|R|\}$, but not for each $i \in X$. It follows that $r_0 = r_{|R|-1} = r_{|R|} + 1$, since $|r_{|R|-1} - r_{|R|}| \leq 1$. From (6.1) we deduce that R is a k-return.

Finally suppose that R is a k-return. By the above, if $R{\restriction_{i,|R|}}$ is not $(k-1)$-upper for some $i \in X$, then $r_i > r_j$ for some $j \in X$ such that $j > i$. By definition $R{\restriction_{i,|R|}}$ is not $(k-1)$-upper for each $i \in X \setminus \{|R|\}$, thus $r_i > r_{|R|}$ for each $i \in X \setminus \{|R|\}$. Thus equation (6.1) implies that $\mathsf{hist}(R, y_{|R|}(r)) = y_0(r)$ for each $r \in [1, r_{|R|}]$; moreover the history of $y_{|R|}(r)$ never in the duration of R landed in the topmost $(k-1)$-stack. It means that the

topmost k-stack of $R(|R|)$ consists of the $r_{|R|}$ bottommost $(k-1)$-stacks of the topmost k-stack of $R(0)$, also in the sense of the history function. By the definition of a k-return, $\mathsf{hist}(R, y_{|R|}(r_{|R|})) = y_0(r_0 - 1)$, so $r_{|R|} = r_0 - 1$. □

6.2. **Characterization of Returns and Upper Runs.** Next we give two lemmas, which describe possible forms of upper runs and returns.

Proposition 6.7. *A run R is k-upper (where $k \in [0, n]$) if and only if*

(1) $|R| = 0$, *or*
(2) $|R| = 1$, *and the only transition of R is* read, *or performs* push^r_γ *for any r, or* pop^r *for $r \leq k$, or*
(3) *the first transition of R performs* push^r_γ *for $r \geq k+1$, and $R{\restriction}_{1,|R|}$ is an r-return, or*
(4) *R is a composition of two nonempty k-upper runs.*

Proof. The right-to-left implication is almost immediate; in Case (3) we use Proposition 6.6.

Concentrate on the left-to-right implication. If $|R| = 0$, then we have Case (1). Suppose that $|R| \geq 1$. Notice that the first transition, between $R(0)$ and $R(1)$, cannot perform pop^r for $r \geq k+1$, as such operation removes the topmost k-stack of $R(0)$, which contradicts with the assumption that R is k-upper. Thus, if $|R| = 1$, then we have Case (2). Suppose that $|R| \geq 2$. If the first transition is read, or performs pop^r for $r \leq k$, or push^r_γ for $r \leq k$, then both $R{\restriction}_{0,1}$ and $R{\restriction}_{1,|R|}$ are k-upper; we have Case (4). We can do the same when the operation is push^r_γ for $r \geq k+1$ and $R{\restriction}_{1,|R|}$ is k-upper.

The remaining case is that the first operation is push^r_γ for $r \geq k+1$ and $R{\restriction}_{1,|R|}$ is not k-upper. Notice that $\mathsf{hist}(R{\restriction}_{0,1}, y) = \mathsf{top}^k(R(0))$ holds only for $y = \mathsf{top}^k(R(1))$ and $y = \mathsf{top}^k(R(0))$. So, because R is k-upper and $R{\restriction}_{1,|R|}$ is not k-upper, which by definition means that $\mathsf{hist}(R, \mathsf{top}^k(R(|R|))) = \mathsf{top}^k(R(0))$ and $\mathsf{hist}(R{\restriction}_{1,R}, \mathsf{top}^k(R(|R|))) \neq \mathsf{top}^k(R(1))$, it has to be $\mathsf{hist}(R{\restriction}_{1,R}, \mathsf{top}^k(R(|R|))) = \mathsf{top}^k(R(0))$. Thus also $\mathsf{hist}(R{\restriction}_{1,R}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(R(0))$, which is the bottommost position of the second topmost $(r-1)$-stack of $R(1)$. Let j be the smallest positive index for which $R{\restriction}_{j,|R|}$ is $(r-1)$-upper. Then $\mathsf{hist}(R{\restriction}_{1,j}, \mathsf{top}^{r-1}(R(j))) = \mathsf{top}^{r-1}(R(0))$ and there is no $i \in [1, j-1]$ such that $R{\restriction}_{i,j}$ is $(r-1)$-upper. Thus $R{\restriction}_{1,j}$ is an r-return. If $j = |R|$, then we have Case (3). Suppose that $j < |R|$. By Proposition 6.6, the only position y in the topmost $(r-1)$-stack (even in the topmost r-stack) of $R(j)$ for which $\mathsf{hist}(R{\restriction}_{0,j}, y) = \mathsf{top}^k(R(0))$ is $y = \mathsf{top}^k(R(j))$. Thus $R \in \mathsf{up}^k$, and $R{\restriction}_{j,|R|} \in \mathsf{up}^{r-1}$ implies that $R{\restriction}_{j,|R|} \in \mathsf{up}^k$; we have Case (4). □

Proposition 6.8. *A run R is an r-return (where $r \in [1, n]$) if and only if*

(1) $|R| = 1$, *and the only transition of R performs* pop^r, *or*
(2) *the first transition of R is* read, *or performs* pop^k *for $k < r$, or* push^k_γ *for $k \neq r$, and $R{\restriction}_{1,|R|}$ is an r-return, or*
(3) *the first transition of R performs* push^k_γ *for $k \geq r$, and $R{\restriction}_{1,|R|}$ is a composition of a k-return and an r-return.*

Proof. Let us analyze the right-to-left implication, which is easier. Case (1) is trivial. In Case (2) we observe that for y being the bottommost position of the second topmost $(r-1)$-stack of $R(1)$, also $\mathsf{hist}(R{\restriction}_{0,1}, y)$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$ (it is important that $k \neq r$ in the case of push^k_γ). Thus $\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|)))$

is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$. Of course such R is not $(r-1)$-upper. Thus R is an r-return. In Case (3), let i be the length of the first return, plus one (so the k-return ends in $R(i)$). Recall that $k \geq r$. By Proposition 6.6, for y being the bottommost position of the second topmost $(r-1)$-stack of $R(i)$, also $\mathsf{hist}(R{\restriction}_{0,i}, y)$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$. Thus $\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|)))$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$. Of course R us not $(r-1)$-upper. If $R{\restriction}_{j,|R|}$ is $(r-1)$-upper for some $j \in [1,i]$, then $\mathsf{hist}(R{\restriction}_{j,i}, y) = \mathsf{top}^{r-1}(R(j))$ for y being the bottommost position of the second topmost $(r-1)$-stack of $R(i)$. This implies that $R{\restriction}_{j,i}$ is $(k-1)$-upper (both for $k > r$ and $k = r$), which is impossible, as the $R{\restriction}_{1,i}$ is a k-return. We conclude that R is an r-return.

Concentrate now on the left-to-right implication. Of course $|R| \geq 1$. Notice that the first operation, between $R(0)$ and $R(1)$, cannot be pop^k for $k \geq r+1$, as such operation removes the topmost r-stack of $R(0)$, which contradicts with the assumption that $\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|)))$ is in the topmost r-stack of $R(0)$.

Suppose that the first operation of R is pop^r. In this situation $\mathsf{hist}(R{\restriction}_{0,1}, y)$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$ only if $y = \mathsf{top}^{r-1}(R(1))$. Hence, because R is an r-return, $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(R(1))$, which means that $R{\restriction}_{1,|R|}$ is $(r-1)$-upper. Thus $|R| = 1$; we have Case (1).

Next, suppose that the first operation is read, or pop^k for $k \leq r-1$, or push^k_γ for $k \leq r-1$. In this situation $\mathsf{hist}(R{\restriction}_{0,1}, y)$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$ only if y is the bottommost position of the second topmost $(r-1)$-stack of $R(1)$. Thus, because R is an r-return, $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ is the bottommost position of the second topmost $(r-1)$-stack of $R(1)$, so $R{\restriction}_{1,|R|}$ is an r-return; we have Case (2).

Finally, suppose that the first operation of R is push^k_γ for $k \geq r$. Let s be the size of the topmost k-stack of $R(0)$. For each $i \in [1, |R|]$ we look at the size of the k-stack containing $\mathsf{hist}(R{\restriction}_{i,|R|}, \mathsf{top}^{r-1}(R(|R|)))$. Recall that $\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|)))$ is in the topmost k-stack of $R(0)$, so for $i = 1$ this is also the topmost k-stack and its size is $s+1$. Suppose first that this size is at least $s+1$ for each i. Then $R{\restriction}_{1,|R|}$ is $(k-1)$-upper (Proposition 6.3). Because R is an r-return, we know that $R{\restriction}_{1,|R|}$ is not $(r-1)$-upper (of course $|R| > 1$), so $k \neq r$ (in fact $k > r$). As $R{\restriction}_{1,|R|}$ is $(k-1)$-upper, we know that $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ is in the topmost $(k-1)$-stack of $R(1)$, so it is the bottommost position of the second topmost $(r-1)$-stack of $R(1)$. It follows that $R{\restriction}_{1,|R|}$ is an r-return, and $k \neq r$ (Case (2)). The opposite possibility is that for some $i \in [1, |R|]$, the size of the k-stack containing $\mathsf{hist}(R{\restriction}_{i,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ becomes s. Fix the first such i. Then $R{\restriction}_{j,i}$ is not $(k-1)$-upper for each $j \in [1, i-1]$ (Proposition 6.3), and $\mathsf{hist}(R{\restriction}_{1,i}, \mathsf{top}^{k-1}(R(i)))$ is the bottommost position of the second topmost $(k-1)$-stack of $R(1)$, thus the $R{\restriction}_{1,i}$ is a k-return. By Proposition 6.8, if $\mathsf{hist}(R{\restriction}_{0,i}, y)$ is the bottommost position of the second topmost $(r-1)$-stack of $R(0)$, and if y is in the topmost k-stack, then it y has to be the bottommost position of the second topmost $(r-1)$-stack of $R(i)$. Because the size of the k-stack containing $\mathsf{hist}(R{\restriction}_{i,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ has changed size between $R(i-1)$ and $R(i)$, it has to be the topmost k-stack, so necessarily $\mathsf{hist}(R{\restriction}_{i,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ is the bottommost position of the second topmost $(r-1)$-stack of $R(i)$. Thus $R{\restriction}_{i,|R|}$ is an r-return. □

7. Types and Sequence Equivalence

In this section we assign to each configuration a type from a finite set. The slogan is that configurations with the same topmost k-stacks and the same type are starting points of similar k-upper runs. We start by an example.

Example 7.1. Consider a 3-DPDA that (while being in some state) can perform the following 1-upper run: it executes pop^1, push^3, and then it starts analyzing the topmost 2-stack using pop^1 and pop^2; when a 0-stack containing a fixed stack symbol a is found, the automaton performs pop^3; the run ends in the same state as it begins. As an effect of this run, the topmost 0-stack is removed, so this is indeed an 1-upper run. Notice that it can be executed only when the topmost 2-stack contains the a symbol, and can be repeated as long as the topmost 1-stack is nonempty. Consider now two configuration of this 3-DPDA, having the same topmost 1-stack. If additionally the topmost 2-stack of both configuration contains the a symbol, then from each of them we can start the 1-upper run described above, and repeat it the same number of times.

Because a 1-upper run can arbitrarily modify the topmost 1-stack, we consider configurations having the same topmost 1-stack. The rest of the stack will be summed up in a small piece of information, called type. In this example we only need to know whether there is the a symbol in the topmost 2-stack (below the topmost 1-stack). In general, whenever a 3-DPDA removes the topmost 1-stack and starts analyzing the stack below, next it has to remove the whole topmost 2-stack (since we consider a 1-upper run). Thus for each entering state (i.e. the state when removing the topmost 1-stack) we only need to know the exit state (i.e. the state when removing the topmost 2-stack). For higher orders the situation is slightly more complicated, but similar.

There is also a second goal of this section. Suppose that we have a sequence of configurations, all having the same topmost k-stack and the same type. Then, as said above, from each of then we can execute a similar k-upper run. But, typically, these k-upper runs will be prefixes of some accepting runs. We want to determine whether such accepting runs can read an unbounded number of $\sharp$ symbols, or not. (For technical reasons, we consider n-returns instead of accepting runs.)

For this section we fix an n-DPDA $\mathcal{A}$ with stack alphabet Γ, state set Q, and input alphabet A that contains a distinguished symbol $\sharp$. Moreover we fix a morphism $\varphi\colon A^* \to M$ into a finite monoid M. For a run R reading a word w, by $\varphi(R)$ we denote $\varphi(w)$, and by $\sharp(R)$ we denote the number of sharps in w. The goal of the morphism is to describe when two upper runs read a similar word: we want to distinguish input words evaluating to different elements of M.

Recall that when both $R \circ S$ and S are k-upper runs, then R is k-upper as well. It follows that any nonempty k-upper run R can be uniquely represented as a composition of the maximal number of nonempty k-upper runs $R_1 \circ \cdots \circ R_r$: we keep on cutting off suffixes which are k-upper (notice that infixes or even prefixes of R_i can be k-upper, but suffixes do not). We will be comparing k-upper runs using the following definition of being (k, φ)-parallel.

Definition 7.2. Let $R = R_1 \circ \cdots \circ R_r$ and $S = S_1 \circ \cdots \circ S_s$ be k-upper runs decomposed into the maximal number of nonempty k-upper runs. We say that R and S are *(k, φ)-parallel* when $r = s$, and for each $i \in [1, r]$ it holds $\varphi(R_i) = \varphi(S_i)$ and the topmost k-stacks of $R_i(0)$ and $S_i(0)$ are equal, as well as the topmost k-stacks of $R_r(|R_r|)$ and $S_r(|S_r|)$. Additionally

two runs R, S of length 0 are (k,φ)-parallel when the topmost k-stacks of $R(0)$ and $S(0)$ are equal. When saying that two runs are (k,φ)-parallel we implicitly mean that they are k-upper.

Notice that if runs R and S are (k,φ)-parallel, and R is divided in any way into k-upper runs $R = R'_1 \circ \dots \circ R'_m$, then S can be as well divided into k-upper runs $S = S'_1 \circ \dots \circ S'_m$ such that for each $i \in [1,m]$ it holds $\varphi(R'_i) = \varphi(S'_i)$ and the topmost k-stacks of $R'_i(0)$ and $S'_i(0)$ are equal, as well as the topmost k-stacks of $R'_m(|R_m|)$ and $S_m(|S_m|)$. Indeed, on one hand, each nonempty R'_i can be further subdivided into k-upper runs of the finest decomposition. On the other hand, for each empty R'_i we can insert an empty S'_i into the sequence for S.

As already mentioned, to each configuration c we will assign its $(\mathcal{A},\varphi)$-*type* (simply called *type* when $\mathcal{A}$ and φ are fixed), that comes from a finite set. Before giving a definition, we state two theorems which describe required properties of our types.

Theorem 7.3. *Let R be a k-upper run, where $k \in [0,n]$, and let c be a configuration having the same $(\mathcal{A},\varphi)$-type and the same topmost k-stack as $R(0)$. Then from c we can start a run that is (k,φ)-parallel to R.*

Beside of types, we will also define an equivalence relation over infinite sequences of configurations of $\mathcal{A}$, called $(\mathcal{A},\varphi)$-sequence equivalence, which has finitely many equivalence classes. The goal is to specify whether the number of $\sharp$ symbols read by a run constructed in Theorem 7.3 is big or small. However instead of having "big" and "small" numbers, we say whether their sequence is bounded or unbounded. This is made precise in the following theorem.

Theorem 7.4. *Let $R \circ R'$ be a run in which R is k-upper and R' is an n-return, where $k \in [0,n]$. Let $c_1, c_2, \dots$ and $d_1, d_2, \dots$ be infinite sequences of configurations which are $(\mathcal{A},\varphi)$-sequence equivalent, and in which all configurations have the same $(\mathcal{A},\varphi)$-type and the same topmost k-stack as $R(0)$. Then for each i there exist runs $S_i \circ S'_i$ from c_i, and $T_i \circ T'_i$ from d_i in which S_i and T_i are (k,φ)-parallel to R, and S'_i and T'_i are n-returns such that $\varphi(S'_i) = \varphi(T'_i) = \varphi(R')$, and such that either the sequences $\sharp(S_1 \circ S'_1), \sharp(S_2 \circ S'_2), \dots$ and $\sharp(T_1 \circ T'_1), \sharp(T_2 \circ T'_2), \dots$ are both bounded, or both unbounded.*

The n-returns in Theorem 7.4 should be understood as accepting runs. Indeed, in Section 10 we increase by 1 the order of an arbitrary $(n-1)$-DPDA, and we add a pop^n operation just before reaching an accepting state; after such modification, a run is accepting if and only if it is an n-return. This trick is performed only for uniformity of presentation: instead of considering accepting runs as a separate concept, we see them as a special case of returns (and returns are used anyway).

One may be puzzled by the fact that Theorem 7.3 talks about a k-upper run, while Theorem 7.4 about a k-upper run composed with an n-return. This difference is application-driven: the first theorem needs to be used without an n-return, while the second one with an n-return. In fact Theorem 7.3 is true also with an n-return at the end, and Theorem 7.4 also without an n-return.

Example 7.5. Consider the 3-DPDA and the 1-upper run from Example 7.1, with the difference that now whenever a b symbol is removed from the stack during the analyzes of the topmost 2-stack, the DPDA reads the $\sharp$ symbol from the input. Additionally suppose that when the topmost 1-stack becomes empty (a bottom-of-stack symbol is uncovered), the DPDA performs pop^3; this pop^3 will serve as the 3-return R'. Then basically we need two

equivalence classes of sequences of configurations (recall that only for sequences with the same topmost 1-stack the relation is meaningful): such that the number of b symbols in the topmost 2-stacks in the configurations is bounded, and such that this number is unbounded. Depending on this fact, the runs will read a bounded or unbounded number of sharps. Of course in general we need more classes than just two ("bounded" and "unbounded"), because e.g. another 1-upper run (having a different image under φ) might read one sharp per each c symbol found on the stack (instead of the b symbols).

The rest of this section is devoted to defining types and sequence equivalence, and proving Theorems 7.3 and 7.4. This is independent from the rest of the paper.

7.1. **Definition of Types.** The types considered here are similar to stack automata from e.g. [2], as well as to intersection types of Kobayashi (e.g. [13]). Notice however that we extend them by a productive/nonproductive flag, which is not present there. This flag is essential for our proof, since we want to estimate the number of $\sharp$ symbols read by our runs, not just to determine existence of some kind of runs. On the other hand in [20] we were using types that were directly describing returns (while here returns correspond to using an assumption); these types were more complicated.

We will be labeling stacks by *run descriptors*. To label a k-stack s^k, where $k \in [0, n]$, we can use a run descriptor from a set $\mathcal{T}^k$. The sets $\mathcal{T}^k$ are defined inductively as follows:

$$\mathcal{T}^k = Q \times \mathcal{P}(M \times \mathcal{T}^n) \times \mathcal{P}(M \times \mathcal{T}^{n-1}) \times \cdots \times \mathcal{P}(M \times \mathcal{T}^{k+1}) \times \{\mathsf{np}, \mathsf{pr}\},$$

where $\mathcal{P}(X)$ denotes the power set of X. We use lowercase Greek letters $(\sigma, \tau, \dots)$ to denote elements of $\mathcal{T}^k$, uppercase Greek letters $(\Psi, \Phi, \dots)$ to denote subsets of $M \times \mathcal{T}^k$, and uppercase Greek letters with a tilde $(\widetilde{\Psi}, \widetilde{\Phi}, \dots)$ to denote subsets of $\mathcal{T}^k$; to all of them we often attach k in superscript.

A run descriptor in $\mathcal{T}^k$ is of the form $\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, f)$. Such run descriptor σ assigned to some k-stack s^k describes a run which starts in a configuration with state p and topmost k-stack s^k. It "can be used" only when the stack $t^n : t^{n-1} : \cdots : t^{k+1} : s^k$ in this configuration is such that for each $i \in [k+1, n]$ to the i-stack t^i we have assigned $\pi_2(\Psi^i)$. As expected, an assumption $(m, \tau) \in \Psi^i$ will be used when the stack t^i will be uncovered. The run descriptor τ also describes a run from such configuration d; it will be used as a suffix of the run from $c = (p, t^n : t^{n-1} : \cdots : t^{k+1} : s^k)$. The run from c to d which uncovers t^i is an i-return. The monoid element in m describes the word w read by the return: $m = \varphi(w)$.

There is also the f component of σ; it says whether the run descriptor is productive (pr) or nonproductive (np). Intuitively, the run descriptor σ is productive if s^k is itself responsible for reading some $\sharp$ symbols. It means that either some reading of a $\sharp$ symbol is performed "inside s^k", or some productive assumption from some Ψ^i is used at least twice (which also increases the number of $\sharp$ symbols read, since some reading described by this productive assumption will be repeated). Thanks to this flag, we can estimate the number of $\sharp$ symbols read, by calculating the number of productive run descriptors used.

We slowly come to the definition of types, before which we need some auxiliary notions. By $\mathcal{T}_{\mathsf{np}}$ and $\mathcal{T}_{\mathsf{pr}}$ we denote the subsets of $\bigcup_{0 \le k \le n} \mathcal{T}^k$ having respectively np or pr on the last coordinate. For $m \in M$ and $\Psi \subseteq M \times \mathcal{T}^k$ we use the notation $m \circ \Psi$ for $\{(m \cdot m', \sigma) \colon (m', \sigma) \in \Psi\}$. Having a run descriptor $\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, f)$, for $i \in [k+1, n]$ by $\mathsf{ass}^i(\sigma)$ we denote the set of assumptions Ψ^i, and by $\mathsf{red}^i(\sigma)$ we denote the "reduced" run descriptor

$(p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{i+1}, g) \in \mathcal{T}^i$ in which $g = \mathsf{np}$ if and only if $f = \mathsf{np}$ and $\pi_2(\Psi^j) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $j \in [k+1, i]$; to p we refer as the state of σ.

We first define a composer, which is used to compose run descriptors corresponding to smaller stacks into run descriptors corresponding to greater stacks.

Definition 7.6. Consider a tuple $(\Phi^k, \Phi^{k-1} \dots, \Phi^l; \Psi^k; f)$, where $0 \le l < k \le n$, $\Phi^i \subseteq M \times \mathcal{T}^i$ for each $i \in [l, k]$, $\Psi^k \subseteq M \times \mathcal{T}^k$, and $f \in \{\mathsf{np}, \mathsf{pr}\}$. Such a tuple is called a *composer* if:

- $\Phi^i = \bigcup\{m \circ \mathsf{ass}^i(\sigma) \colon (m, \sigma) \in \Phi^l\}$ for each $i \in [l+1, k]$,
- $\Psi^k = \{(m, \mathsf{red}^k(\sigma)) \colon (m, \sigma) \in \Phi^l\}$, and $|\pi_2(\Psi^k)| = |\pi_2(\Phi^l)|$ (which means that each $\sigma \in \pi_2(\Phi^l)$ gives a different $\mathsf{red}^k(\sigma)$), and
- $f = \mathsf{np}$ if and only if $\pi_2(\mathsf{ass}^i(\sigma)) \cap \pi_2(\mathsf{ass}^i(\tau)) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [l+1, k]$ and each $\sigma, \tau \in \pi_2(\Psi^l)$ such that $\sigma \neq \tau$.

Next, we define when a run descriptor σ from $\mathcal{T}^0$ can be assigned to a 0-stack γ. This is the case when we can derive a statement $\gamma \vdash \sigma$, that is when there exists a *derivation tree for* $\gamma \vdash \sigma$. In such situation, $\gamma \vdash \sigma$ is called the *conclusion* of the derivation tree D, and σ is called the run descriptor of D and is denoted $\mathsf{rd}(D)$.

Definition 7.7. We define the set of *derivation trees* as the smallest set satisfying the following conditions. Let p be a state, and γ a 0-stack.

(1) A pair (γ, p) is a derivation tree for $\gamma \vdash (p, \emptyset, \emptyset, \dots, \emptyset, \mathsf{np})$.

(2) Suppose that $\delta(\gamma, p) = \mathsf{read}(\vec{q})$ and that D' is a derivation tree for $\gamma \vdash \tau$, where the state of τ is $\vec{q}(a)$ for some $a \in A$. Denote $\Phi^i = \varphi(a) \circ \mathsf{ass}^i(\tau)$ for $i \in [1, n]$. Then (p, D') is a derivation tree for $\gamma \vdash (p, \Phi^n, \Phi^{n-1}, \dots, \Phi^1, f)$, where $f = \mathsf{np}$ if and only if $\tau \in \mathcal{T}_{\mathsf{np}}$ and $a \neq \sharp$.

(3) Suppose that $\delta(\gamma, p) = (q, \mathsf{pop}^k)$, and that $\tau^k \in \mathcal{T}^k$ is a run descriptor with state q. Then (γ, p, τ^k) is a derivation tree for

$$\gamma \vdash (p, \mathsf{ass}^n(\tau^k), \mathsf{ass}^{n-1}(\tau^k), \dots, \mathsf{ass}^{k+1}(\tau^k), \{(\mathbf{1}_M, \tau^k)\}, \emptyset, \dots, \emptyset, \mathsf{np}).$$

(4) Suppose that $\delta(\gamma, p) = (q, \mathsf{push}^k_\alpha)$, and that D' is a derivation tree for $\alpha \vdash \tau$, where the state of τ is q. Denote $\Psi^i = \mathsf{ass}^i(\tau)$ for $i \in [1, n]$. Suppose also that $(\Phi^k, \Phi^{k-1}, \dots, \Phi^0; \Psi^k; f)$ is a composer, and $\mathfrak{D}$ is a set of derivation trees, all having the stack element γ in their conclusion, and such that $\{\mathsf{rd}(E) \colon E \in \mathfrak{D}\} = \pi_2(\Phi^0)$ and $|\mathfrak{D}| = |\pi_2(\Phi^0)|$. Let

$$\Upsilon^i = \begin{cases} \Psi^i & \text{for } i \in [k+1, n], \\ \Phi^i & \text{for } i = k, \\ \Psi^i \cup \Phi^i & \text{for } i \in [1, k-1]. \end{cases}$$

Then $(\gamma, p, D', \mathfrak{D})$ is a derivation tree for $\gamma \vdash (p, \Upsilon^n, \Upsilon^{n-1}, \dots, \Upsilon^1, g)$, where $g = \mathsf{np}$ if and only if $f = \mathsf{np}$ and $\{\tau\} \cup \pi_2(\Phi^0) \subseteq \mathcal{T}_{\mathsf{np}}$ and $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [1, k-1]$.

The *depth* of a derivation tree D, denoted $\mathsf{depth}(D)$, is defined naturally: it is 0 in Cases (1) and (3), $1 + \mathsf{depth}(D')$ in Case (2), and $1 + \max(\mathsf{depth}(D'), \max_{E \in \mathfrak{D}} \mathsf{depth}(E))$ in Case (4).

Notice that in Case (4), the composer is uniquely determined by the derivation tree. Indeed, Ψ^k and $\pi_2(\Phi^0)$ are fixed by D' and $\mathfrak{D}$; the set Φ^0 has to contain those pairs $(m, \tau) \in M \times \pi_2(\Phi^0)$ for which $(m, \mathsf{red}^k(\tau)) \in \Psi^k$, and Φ^0 fixes all other Φ^i.

We will annotate our stack using sets of derivation trees. An *annotated k-stack* is a k-stack over an extended alphabet, whose elements are pairs $(\gamma, \mathfrak{D})$, where $\gamma \in \Gamma$ and $\mathfrak{D}$ is a set of derivation trees having conclusions with the stack symbol γ, and different run descriptors (that is, $\mathsf{rd}(D) = \mathsf{rd}(E)$ for $D, E \in \mathfrak{D}$ implies $D = E$). Annotated stacks will be denoted using boldface letters, often with their order written in the superscript: $\mathbf{s}^0$, $\mathbf{t}^5$, etc. The projection of each letter in an annotated k-stack $\mathbf{s}^k$ into the Γ coordinate will be denoted by $\mathsf{st}(\mathbf{s}^k)$.

We also define the type of an annotated k-stack, which is a subset of $\mathcal{T}^k$:

$$\mathsf{type}((\gamma, \mathfrak{D})) = \{\mathsf{rd}(D) \colon D \in \mathfrak{D}\}, \qquad \mathsf{type}([\,]) = \emptyset,$$
$$\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^{k-1})\}.$$

We always want to annotate stacks in a consistent way. Intuitively, when a run descriptor assigned to some stack element requires some assumptions, then the part of the stack which is below has to deliver annotations fulfilling this assumption. Moreover, all annotations have to be useful. To formalize this, we define below when an annotated k-stack $\mathbf{s}^k$ is *well-formed* (in the sequel we only consider well-formed annotated stacks).

Definition 7.8. Each annotated 0-stack, and the empty annotated k-stack for each $k \geq 1$, are always well-formed. An annotated k-stack $\mathbf{s}^k : \mathbf{s}^{k-1}$ is well-formed if both $\mathbf{s}^k$ and $\mathbf{s}^{k-1}$ are well-formed, and $\mathsf{type}(\mathbf{s}^k) = \bigcup_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \pi_2(\mathsf{ass}^k(\sigma))$, and $|\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1})| = |\mathsf{type}(\mathbf{s}^{k-1})|$.

An annotated stack $\mathbf{s}$ is called *singular* if $|\mathsf{type}(\mathbf{s})| = 1$. When an annotated n-stack $\mathbf{s}^n$ is singular, we define $\mathsf{conf}(\mathbf{s}^n)$ to be the configuration $(q, \mathsf{st}(\mathbf{s}^n))$, where q is the state of the only run descriptor in $\mathsf{type}(\mathbf{s}^n)$.

It is useful to have a notation for the topmost k-stack of an annotated stack $\mathbf{s}$; we denote it as $\mathsf{top}^k(\mathbf{s})$, hoping that it will not be confused with $\mathsf{top}^k(c)$, the bottommost position of the topmost k-stack in a configuration c.

As the type of a configuration c, denoted $\mathsf{type}_{\mathcal{A},\varphi}(c)$, we take the union of $\mathsf{type}(\mathsf{top}^0(\mathbf{s}^n))$ over all well-formed singular annotated n-stacks $\mathbf{s}^n$ such that $\mathsf{conf}(\mathbf{s}^n) = c$,

$$\mathsf{type}_{\mathcal{A},\varphi}(c) = \bigcup \{\mathsf{type}(\mathsf{top}^0(\mathbf{s}^n)) \colon \mathbf{s}^n \text{ well-formed}, \mathsf{conf}(\mathbf{s}^n) = c\}.$$

We see a direct connection between the well-formedness property and composers.

Proposition 7.9. *Let $0 \leq l < k \leq n$, let $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^l$ be an annotated k-stack in which each $\mathbf{s}^i$ is well-formed, and let $\Psi^k \subseteq M \times \mathcal{T}^k$. The following two conditions are equivalent:*

- *there exists a composer $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^l; \Psi^k; f)$ such that $\pi_2(\Phi^i) = \mathsf{type}(\mathbf{s}^i)$ for each $i \in [l, k]$, and*
- *$\mathbf{s}$ is well-formed and $\pi_2(\Psi^k) = \mathsf{type}(\mathbf{s})$.*

Example 7.10. Consider the 1-DPDA $\mathcal{A}$ with stack alphabet $\{a\}$, state set $\{q_1, q_2, q_3\}$, and transitions

$$\delta(q_1, a) = (q_2, \mathsf{pop}^1), \qquad \delta(q_2, a) = (q_3, \mathsf{pop}^1), \qquad \delta(q_3, a) = (q_1, \mathsf{push}^1_a),$$

and the trivial monoid $\{1\}$. Denote

$$\gamma_i = (q_i, \mathsf{np}), \qquad \sigma_i = (q_i, \emptyset, \mathsf{np}) \qquad \text{for } i \in \{1, 2, 3\},$$
$$\rho_3 = (q_3, \{(1, \gamma_3)\}, \mathsf{np}), \qquad \tau_i = (q_i, \{(1, \gamma_{i+1})\}, \mathsf{np}) \qquad \text{for } i \in \{1, 2\}.$$

Then $D_i = (a, q_i)$ for $i \in \{1,2,3\}$ is a derivation tree for $a \vdash \sigma_i$, and $E_i = (a, q_i, \gamma_{i+1})$ for $i \in \{1,2\}$ is a derivation tree for $a \vdash \tau_i$, and $F_3 = (a, q_3, E_1, \{E_2\})$ is a derivation tree for $a \vdash \rho_3$. Notice that $a \vdash \sigma_3$ can be also derived using derivation trees $(a, q_3, D_1, \emptyset)$ and $(a, q_3, E_1, \{D_2\})$. For this 1-DPDA we can derive only nonproductive run descriptors, so also types of annotated stacks contain only nonproductive run descriptors. For this reason we cannot use a derivation tree for $a \vdash (q_2, \{(1, (q_3, \mathsf{pr}))\}, \mathsf{np})$ in any well-formed annotated 1-stack, although such derivation tree exists. An example well-formed annotated 1-stack is $[(a,\emptyset), (a, \{D_3\}), (a, \{F_3\}), (a, \{E_2\})]$, having type $\{\gamma_2\}$.

To each nonempty 1-stack s we can add annotations in a well-formed way, so that to the topmost 0-stack we assign $\{D_1\}$ or $\{D_2\}$ or $\{D_3\}$, whichever we want. If $|s| \geq 2$, then we can also have there $\{E_1\}$ or $\{E_2\}$ or $\{F_3\}$. Thus $\mathsf{type}_{\mathcal{A},\varphi}((q_i, [a])) = \{\sigma_i\}$ for $i \in \{1,2,3\}$, and for any 1-stack s having at least two a's, $\mathsf{type}_{\mathcal{A},\varphi}((q_i, s)) = \{\sigma_i, \tau_i\}$ for $i \in \{1,2\}$ and $\mathsf{type}_{\mathcal{A},\varphi}((q_3, s)) = \{\sigma_3, \rho_3\}$.

When $\widetilde{\Psi}$ is a subset of the type of a well-formed annotated k-stack $\mathbf{s}$, we can remove some of the annotations in $\mathbf{s}$ in order to obtain a well-formed annotated k-stack $\mathbf{s}{\restriction_{\widetilde{\Psi}}}$ whose type is $\widetilde{\Psi}$. We do this by induction:

- For $k = 0$, we restrict the set of derivation trees in $\mathbf{s}$ to those trees whose run descriptor is in $\widetilde{\Psi}$.
- The type of the empty stack is empty, so we need not to restrict it in any way.
- For $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$, we restrict $\mathbf{s}^{k-1}$ to the set $\widetilde{\Phi}$ containing those $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$ for which $\mathsf{red}^k(\sigma) \in \widetilde{\Psi}$, and we restrict $\mathbf{s}^k$ to $\bigcup_{\sigma\in\widetilde{\Phi}} \pi_2(\mathsf{ass}^k(\sigma))$.

7.2. **Annotated runs.** In this subsection we describe how run descriptors are connected with runs.

Definition 7.11. Let $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^0$ be a well-formed singular annotated n-stack, where $\mathbf{s}^0 = (\gamma, \{D\})$. We define the *successor* of $\mathbf{s}$.

(1) If $D = (\gamma, p)$, then $\mathbf{s}$ has no successor.
(2) If $D = (p, D')$, then the successor is $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\gamma, \{D'\})$.
(3) If $D = (\gamma, p, \tau^k)$, then the successor is $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$.
(4) Suppose that $D = (\gamma, p, D', \mathfrak{D})$. Let α, k, Ψ^i, Φ^i be as in Definition 7.7(4). In this situation, the successor of $\mathbf{s}$ is

$$\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{t}^k : \mathbf{s}^{k-1}{\restriction_{\pi_2(\Psi^{k-1})}} : \mathbf{s}^{k-2}{\restriction_{\pi_2(\Psi^{k-2})}} : \cdots : \mathbf{s}^1{\restriction_{\pi_2(\Psi^1)}} : (\alpha, \{D'\}),$$

where $\mathbf{t}^k = \mathbf{s}^k{\restriction_{\pi_2(\Phi^k)}} : \mathbf{s}^{k-1}{\restriction_{\pi_2(\Phi^{k-1})}} : \cdots : \mathbf{s}^1{\restriction_{\pi_2(\Phi^1)}} : (\gamma, \mathfrak{D})$.

Comparing this definition with Definitions 7.7 (and using Proposition 7.9 in Case (4)), we directly see that the successor $\mathbf{t}$ of $\mathbf{s}$, if exists, is well-formed, and that $\mathsf{conf}(\mathbf{t})$ is a successor of $\mathsf{conf}(\mathbf{s})$ (in the considered automaton). An *annotated run* $\mathbf{R}$ is a sequence $\mathbf{s}_0, \ldots, \mathbf{s}_m$ of well-formed singular n-stacks in which $\mathbf{s}_i$ is the successor of $\mathbf{s}_{i-1}$ for each $i \in [1, m]$. By replacing each $\mathbf{s}_i$ by $\mathsf{conf}(\mathbf{s}_i)$ we obtain a run $\mathsf{st}(\mathbf{R})$.

Notice that an annotated stack $\mathbf{s}$ may have less successors than $\mathsf{conf}(\mathbf{s})$. Even more: not every run is of the form $\mathsf{st}(\mathbf{R})$ for some annotated run $\mathbf{R}$ (this is because the push in Case (4) leaves the same annotations in the original substack as in the copied substack, up to a restriction, but later different annotations may be needed in these substacks).

A priori there might exist an infinite annotated run. As we will see below, this is impossible: always after some number of steps we reach an annotated stack with no successors (Case (1)). Moreover we show that the number of $\sharp$ symbols read by the run can be estimated by the number of productive run descriptors in the annotations of $\mathbf{s}$. For this reason, to each well-formed annotated stack $\mathbf{s}$ we assign three natural numbers: $\mathsf{low}(\mathbf{s})$, $\mathsf{high}(\mathbf{s})$, and $\mathsf{len}(\mathbf{s})$. The first two of them will give a lower and an upper bound on the number of $\sharp$ symbols read by our run, and the last will give an upper bound on the length of the run.

Definition 7.12. For positive integers $m_1, \dots, m_k$ we define $\mathsf{pow}(m_1, \dots, m_k)$ by induction on k:

$$\mathsf{pow}() = 1, \qquad \text{and} \qquad \mathsf{pow}(m_1, m_2, \dots, m_k) = (1 + m_1)^{\mathsf{pow}(m_2,\dots,m_k)} - 1.$$

Definition 7.13. For a well-formed annotated k-stack $\mathbf{s}$ we define natural numbers $\mathsf{low}(\mathbf{s})$, $\mathsf{high}(\mathbf{s})$, and $\mathsf{len}(\mathbf{s})$ by induction on the structure of $\mathbf{s}$.

- If $\mathbf{s} = (\gamma, \mathfrak{D})$, we take
$$\begin{aligned} \mathsf{low}(\mathbf{s}) &= |\mathsf{type}(\mathbf{s}) \cap \mathcal{T}_{\mathsf{pr}}|, \\ \mathsf{high}(\mathbf{s}) &= \prod_{D \in \mathfrak{D}\colon\ \mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}} C_{\mathsf{depth}(D)}, \qquad \text{and} \\ \mathsf{len}(\mathbf{s}) &= \prod_{D \in \mathfrak{D}} C_{\mathsf{depth}(D)}, \end{aligned}$$
where C_z is defined inductively:
$$C_0 = 2, \qquad \text{and} \qquad C_{z+1} = (2|\mathcal{T}^0|)^n \cdot (C_z)^{|\mathcal{T}^0|+1}.$$
- We take $\mathsf{low}([\,]) = 0$ and $\mathsf{high}([\,]) = \mathsf{len}([\,]) = 1$.
- If $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$, we take
$$\begin{aligned} \mathsf{low}(\mathbf{s}) &= \sum_{\sigma \in \mathsf{type}(s^{k-1})} \big(\mathsf{low}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}) + \mathsf{low}(\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}})\big), \\ \mathsf{high}(\mathbf{s}) &= \prod_{\sigma \in \mathsf{type}(s^{k-1})} \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{high}(\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}})\big), \qquad \text{and} \\ \mathsf{len}(\mathbf{s}) &= \prod_{\sigma \in \mathsf{type}(s^{k-1})} \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{len}(\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}})\big). \end{aligned}$$

Example 7.14. Recall from Example 7.10 the 1-DPDA and the annotated stack $\mathbf{s}_1 = [(a, \emptyset), (a, \{D_3\}), (a, \{F_3\}), (a, \{E_2\})]$. Its successors are consecutively:
$$\begin{aligned} &[(a, \emptyset), (a, \{D_3\}), (a, \{F_3\})], \\ &[(a, \emptyset), (a, \{D_3\}), (a, \{E_2\}), (a, \{E_1\})], \\ &[(a, \emptyset), (a, \{D_3\}), (a, \{E_2\})], \\ &[(a, \emptyset), (a, \{D_3\})], \end{aligned}$$
the last of which has no more successors. It holds $\mathsf{low}(\mathbf{s}_1) = 0$, $\mathsf{high}(\mathbf{s}_1) = 1$ (no productive run descriptors at all), and
$$\mathsf{len}(\mathbf{s}_1) = \mathsf{pow}(\mathsf{pow}(\mathsf{pow}(1, C_0), C_1), C_0) = 2^{C_0 \cdot C_1 \cdot C_0} - 1.$$

Example 7.15. Consider the 1-DPDA $\mathcal{A}$ with input alphabet $\{\sharp\}$, stack alphabet $\{a\}$, state set $\{q_1, q_2\}$, and transitions

$$\delta(q_1, a) = \mathsf{read}(\vec{q}) \text{ for } \vec{q}(\sharp) = q_2, \qquad \delta(q_2, a) = \mathsf{pop}^1(q_1),$$

and the trivial monoid $\{1\}$. Then we have four well-formed annotated stacks with $\mathsf{conf}(\mathbf{s}_i) = (q_1, [aa])$:

$$\begin{aligned}
\mathbf{s}_0 &= [(a, \emptyset), (a, \{D_0\})] && \text{where } D_0 = (a, q_1),\\
\mathbf{s}_1 &= [(a, \emptyset), (a, \{D_1\})] && \text{where } D_1 = (q_1, (a, q_2)),\\
\mathbf{s}_2 &= [(a, \{D_0\}), (a, \{D_2\})] && \text{where } D_2 = (q_1, (a, q_2, (q_1, \mathsf{np}))),\\
\mathbf{s}_3 &= [(a, \{D_1\}), (a, \{D_3\})] && \text{where } D_3 = (q_1, (a, q_2, (q_1, \mathsf{pr}))).
\end{aligned}$$

We have $\mathsf{type}(\mathbf{s}_0) = \{(q_1, \mathsf{np})\}$ and $\mathsf{type}(\mathbf{s}_i) = \{(q_1, \mathsf{pr})\}$ for $i \in \{1, 2, 3\}$. The maximal annotated runs starting in these stacks are of length zero, one (only read), two (read and pop^1) and three (read, pop^1 and read). We see that

$$\begin{aligned}
&\mathsf{low}(\mathbf{s}_0) = 0, && \mathsf{low}(\mathbf{s}_1) = \mathsf{low}(\mathbf{s}_2) = 1, && \mathsf{low}(\mathbf{s}_3) = 2,\\
&\mathsf{high}(\mathbf{s}_0) = 1, && \mathsf{high}(\mathbf{s}_1) = \mathsf{high}(\mathbf{s}_2) = \mathsf{pow}(1, C_1), && \mathsf{high}(\mathbf{s}_3) = \mathsf{pow}(C_1, C_1).
\end{aligned}$$

The required properties of the three numbers are described by the following lemma.

Lemma 7.16. *Let* $\mathbf{R}$ *be an annotated run. It holds*

$$\begin{aligned}
&\mathsf{low}(\mathbf{R}(0)) \le \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}(\mathbf{R}(|\mathbf{R}|)),\\
&\mathsf{high}(\mathbf{R}(0)) \ge \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(|\mathbf{R}|)), && \textit{and}\\
&\mathsf{len}(\mathbf{R}(0)) \ge |\mathbf{R}| + \mathsf{len}(\mathbf{R}(|\mathbf{R}|)).
\end{aligned}$$

In our proofs we need some (in)equalities regarding the pow function.

Proposition 7.17. *The following is true for all positive integers:*

$$\mathsf{pow}(a_1, \dots, a_k, \mathsf{pow}(b_1, \dots, b_l)) = \mathsf{pow}(a_1, \dots, a_k, b_1, \dots, b_l), \tag{7.1}$$

$$\mathsf{pow}(a_1, \dots, a_k, \mathsf{pow}(c_0, c_1, \dots, c_l), b_1, \dots, b_l) \le \le \mathsf{pow}(a_1, \dots, a_k, c_0, b_1c_1, \dots, b_lc_l), \tag{7.2}$$

$$\mathsf{pow}(a_1, \dots, a_{i-1}, a_i^x, a_{i+1}, \dots, a_{k-1}, a_k) \le \mathsf{pow}(a_1, \dots, a_{k-1}, xa_k) \qquad \textit{for } i < k, \tag{7.3}$$

$$\mathsf{pow}(a_1, \dots, a_{k-1}, a_k) + 1 \le \mathsf{pow}(a_1, \dots, a_{k-1}, a_k + 1), \tag{7.4}$$

$$\mathsf{pow}(a_1, \dots, a_k) \cdot \mathsf{pow}(b_1, \dots, b_k) \le \mathsf{pow}(a_1b_1, \dots, a_kb_k). \tag{7.5}$$

□

Heading toward the proof of Lemma 7.16, we first observe an auxiliary property.

Proposition 7.18. *Let* $\mathbf{s}$ *be a well-formed annotated stack. If* $\mathsf{type}(\mathbf{s}) \subseteq \mathcal{T}_{\mathsf{np}}$ *then* $\mathsf{low}(\mathbf{s}) = 0$ *and* $\mathsf{high}(\mathbf{s}) = 1$*; otherwise* $\mathsf{low}(\mathbf{s}) \ge 1$ *and* $\mathsf{high}(\mathbf{s}) \ge 2$*.*

Proof. By induction on the structure of $\mathbf{s}$; we analyze Definition 7.13. Notice that $\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1}) \subseteq \mathcal{T}_{\mathsf{np}}$ if and only if $\mathsf{type}(\mathbf{s}^k) \cup \mathsf{type}(\mathbf{s}^{k-1}) \subseteq \mathcal{T}_{\mathsf{np}}$, assuming that $\mathbf{s}^k : \mathbf{s}^{k-1}$ is well-formed. □

Next, we observe how the functions interplay with composing annotated stacks.

Lemma 7.19. *Let $0 \leq l < k \leq n$, let $(\Phi^k, \Phi^{k-1}, \dots, \Phi^l; \Psi^k; f)$ be a composer, and let $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^l$ be a well-formed annotated k-stack such that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Phi^i)$ for each $i \in [l,k]$. In this situation*

$$\sum_{i=l}^{k} \mathsf{low}(\mathbf{s}^i) \leq \mathsf{low}(\mathbf{s}), \tag{7.6}$$

$$\sum_{i=l}^{k} \mathsf{low}(\mathbf{s}^i) < \mathsf{low}(\mathbf{s}) \qquad \text{if } f = \mathsf{pr}, \tag{7.7}$$

$$\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \dots, \mathsf{high}(\mathbf{s}^{l+1}), \big|\mathcal{T}^0\big|^n \cdot \mathsf{high}(\mathbf{s}^l)\big) \geq \mathsf{high}(\mathbf{s}), \tag{7.8}$$

$$\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \dots, \mathsf{high}(\mathbf{s}^{l+1}), \mathsf{high}(\mathbf{s}^l)\big) \geq \mathsf{high}(\mathbf{s}) \qquad \text{if } f = \mathsf{np}, \tag{7.9}$$

$$\mathsf{pow}\big(\mathsf{len}(\mathbf{s}^k), \mathsf{len}(\mathbf{s}^{k-1}), \dots, \mathsf{len}(\mathbf{s}^{l+1}), \big|\mathcal{T}^0\big|^n \cdot \mathsf{len}(\mathbf{s}^l)\big) \geq \mathsf{len}(\mathbf{s}). \tag{7.10}$$

Proof. Directly from Definition 7.13 it follows that for any well-formed annotated stack $\mathbf{t}$ it holds

$$\mathsf{low}(\mathbf{t}) = \sum_{\sigma \in \mathsf{type}(\mathbf{t})} \mathsf{low}(\mathbf{t}{\restriction_{\{\sigma\}}}),$$

$$\mathsf{high}(\mathbf{t}) = \prod_{\sigma \in \mathsf{type}(\mathbf{t})} \mathsf{high}(\mathbf{t}{\restriction_{\{\sigma\}}}), \qquad \text{and}$$

$$\mathsf{len}(\mathbf{t}) = \prod_{\sigma \in \mathsf{type}(\mathbf{t})} \mathsf{len}(\mathbf{t}{\restriction_{\{\sigma\}}}).$$

Expanding the definition of low, we see that

$$\mathsf{low}(\mathbf{s}) = \mathsf{low}(\mathbf{s}^l) + \sum_{i=l+1}^{k} \sum_{\sigma \in \pi_2(\Phi^l)} \mathsf{low}(\mathbf{s}^i{\restriction_{\pi_2(\mathsf{ass}^i(\sigma))}}).$$

For each $i \in [l+1,k]$ it holds $\mathsf{type}(\mathbf{s}^i) = \bigcup\{\pi_2(\mathsf{ass}^i(\sigma)) \colon \sigma \in \pi_2(\Phi^l)\}$, so

$$\mathsf{low}(\mathbf{s}^i) = \sum_{\tau \in \mathsf{type}(\mathbf{s}^i)} \mathsf{low}(\mathbf{s}^i{\restriction_{\{\tau\}}}) \leq \sum_{\sigma \in \pi_2(\Phi^l)} \sum_{\tau \in \pi_2(\mathsf{ass}^i(\sigma))} \mathsf{low}(\mathbf{s}^i{\restriction_{\{\tau\}}}) = \sum_{\sigma \in \pi_2(\Phi^l)} \mathsf{low}(\mathbf{s}^i{\restriction_{\pi_2(\mathsf{ass}^i(\sigma))}}).$$

Altogether it gives Inequality (7.6).

For Inequality (7.7) observe that if $f = \mathsf{pr}$, then for some $i \in [l+1,k]$, some $\tau \in \mathcal{T}_{\mathsf{pr}}$ appears in $\pi_2(\mathsf{ass}^i(\sigma))$ for two different $\sigma \in \pi_2(\Phi^l)$. Thus some positive component $\mathsf{low}(\mathbf{s}^i{\restriction_{\{\tau\}}})$ appears in two sums $\sum_{\tau \in \pi_2(\mathsf{ass}^i(\sigma))} \mathsf{low}(\mathbf{s}^i{\restriction_{\{\tau\}}})$, so the inequality becomes strict.

For $i \in [l+1,k]$ and $\sigma \in \pi_2(\Phi^l)$ denote $H^i_\sigma = \mathsf{high}(\mathbf{s}^k{\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}})$. Using Inequality (7.5) we obtain

$$\begin{aligned}\mathsf{high}(\mathbf{s}) &= \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{pow}\big(H^k_\sigma, H^{k-1}_\sigma, \dots, H^{l+1}_\sigma, \mathsf{high}(\mathbf{s}^l{\restriction_{\{\sigma\}}})\big) \leq \\ &\leq \mathsf{pow}\Big(\prod_{\sigma \in \pi_2(\Phi^l)} H^k_\sigma, \prod_{\sigma \in \pi_2(\Phi^l)} H^{k-1}_\sigma, \dots, \prod_{\sigma \in \pi_2(\Phi^l)} H^{l+1}_\sigma, \mathsf{high}(\mathbf{s}^l)\Big).\end{aligned} \tag{7.11}$$

Now observe for each $i \in [l+1, k]$ that

$$\prod_{\sigma \in \pi_2(\Phi^l)} H^i_\sigma \leq \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{high}(\mathbf{s}^i) = (\mathsf{high}(\mathbf{s}^i))^{|\pi_2(\Psi^k)|} \leq (\mathsf{high}(\mathbf{s}^i))^{|\mathcal{T}^0|}.$$

The last inequality is true, because $|\pi_2(\Psi^k)| \leq |\mathcal{T}^k| \leq |\mathcal{T}^0|$. Using Inequality (7.3) we move the $|\mathcal{T}^0|$ exponents (there is at most n of them) into the last argument of pow and we obtain Inequality (7.8):

$$\begin{aligned}\mathsf{high}(\mathbf{s}) &\leq \mathsf{pow}\big((\mathsf{high}(\mathbf{s}^k))^{|\mathcal{T}^0|}, (\mathsf{high}(\mathbf{s}^{k-1}))^{|\mathcal{T}^0|}, \dots, (\mathsf{high}(\mathbf{s}^{l+1}))^{|\mathcal{T}^0|}, \mathsf{high}(\mathbf{s}^l)\big) \leq \\ &\leq \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \dots, \mathsf{high}(\mathbf{s}^{l+1}), |\mathcal{T}^0|^n \cdot \mathsf{high}(\mathbf{s}^l)\big).\end{aligned}$$

Now suppose that $f = \mathsf{np}$. It implies, for each $i \in [l+1, k]$, that each $\tau \in \pi_2(\Phi^i) \cap \mathcal{T}_{\mathsf{pr}}$ belongs to the set $\pi_2(\mathsf{ass}^i(\sigma))$ only for one $\sigma \in \pi_2(\Phi^l)$, so all the common factors are equal to 1:

$$\prod_{\sigma \in \pi_2(\Phi^l)} H^i_\sigma = \prod_{\sigma \in \pi_2(\Phi^l)} \prod_{\tau \in \pi_2(\mathsf{ass}^i(\sigma))} \mathsf{high}(\mathbf{s}^i{\restriction}_{\{\tau\}}) = \prod_{\tau \in \pi_2(\Phi^i)} \mathsf{high}(\mathbf{s}^i{\restriction}_{\{\tau\}}) = \mathsf{high}(\mathbf{s}^i).$$

By substituting this to Inequality (7.11) we obtain Inequality (7.9).

Inequality (7.10) is obtained in the same way as Inequality (7.8), as the definitions of len and high differ only in the base case. □

Proof of Lemma 7.16. It is enough to prove the lemma for annotated runs of length 1. Then the result for longer runs follow by an immediate induction. Thus assume that $|\mathbf{R}| = 1$, and denote $\mathbf{R}(0) = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^0$ with $\mathbf{s}^0 = (\gamma, \{D\})$. We have four cases, depending on the shape of D.

Case 1. It is impossible that D is of the form (γ, p), since then $\mathbf{R}(0)$ would not have a successor.

Case 2. Suppose that $D = (p, D')$. Then $\mathbf{R}(1) = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^1 : (\gamma, \{D'\})$. The difference between $\mathsf{low}(\mathbf{R}(0))$ and $\mathsf{low}(\mathbf{R}(1))$ is that in the former we add $\mathsf{low}(\mathbf{s}^0)$ where in the latter $\mathsf{low}((\gamma, \{D'\}))$. Thus the required inequality about low can be restated as

$$\mathsf{low}(\mathbf{s}^0) \leq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}((\gamma, \{D'\})).$$

It holds when $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$ (we have $\mathsf{low}(\mathbf{s}^0) = 0$). If $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$, then $\mathsf{low}(\mathbf{s}^0) = 1$, and either $\mathsf{rd}(D') \in \mathcal{T}_{\mathsf{pr}}$ or the letter read by $\mathsf{st}(\mathbf{R})$ is $\sharp$, so the right side is positive.

If $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$, then $\sharp(\mathsf{st}(\mathbf{R})) = 0$ and $\mathsf{rd}(D') \in \mathcal{T}_{\mathsf{np}}$. In this case

$$\mathsf{high}(\mathbf{R}(0)) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), 1) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)).$$

If $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$, then using Inequality (7.4) we obtain

$$\begin{aligned}\mathsf{high}(\mathbf{R}(0)) &= \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), C_{\mathsf{depth}(D)}) \geq \\ &\geq \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), C_{\mathsf{depth}(D')} + 1) \geq \\ &\geq \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), C_{\mathsf{depth}(D')}) + 1 \geq \\ &\geq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)).\end{aligned}$$

In the same way we obtain the required inequality for len.

Case 3. Suppose that $D = (\gamma, p, \tau^k)$. Then $\mathbf{R}(1) = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$. Recall that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D))) = \emptyset$ for $i \in [1, k-1]$, and $|\mathsf{type}(\mathbf{s}^k)| = 1$, and $\sharp(\mathsf{st}(\mathbf{R})) = 0$. Because $\mathsf{low}(\mathbf{s}^0) = 0$, and $\mathsf{high}(\mathbf{s}^0) = 1$, and $\mathsf{len}(\mathbf{s}^0) = C_0 = 2$, it holds

$$\begin{aligned}
\mathsf{low}(\mathbf{R}(0)) &= \sum_{i=0}^{n} \mathsf{low}(\mathbf{s}^i) = \sum_{i=k}^{n} \mathsf{low}(\mathbf{s}^i) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}(\mathbf{R}(1)),\\
\mathsf{high}(\mathbf{R}(0)) &= \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \ldots, \mathsf{high}(\mathbf{s}^k), 1, \ldots, 1\big) =\\
&= \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \ldots, \mathsf{high}(\mathbf{s}^k)\big) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)),\\
\mathsf{len}(\mathbf{R}(0)) &= \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^n), \mathsf{len}(\mathbf{s}^{n-1}), \ldots, \mathsf{len}(\mathbf{s}^k), 1, \ldots, 1, 2\big) \geq\\
&\geq \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^n), \mathsf{len}(\mathbf{s}^{n-1}), \ldots, \mathsf{len}(\mathbf{s}^k), 1, \ldots, 1, 1\big) + 1 = 1 + \mathsf{len}(\mathbf{R}(1)).
\end{aligned}$$

as required.

Case 4. Suppose that $D = (\gamma, p, D', \mathfrak{D})$. Let α and k be such that $\delta(\gamma, p)$ performs push^k_α. By Definition 7.7(4) we have a composer $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$ such that $\pi_2(\Phi^0) = \{\mathsf{rd}(E) \colon E \in \mathfrak{D}\}$. It holds $\Psi^k = \mathsf{ass}^k(\mathsf{rd}(D'))$, and $\pi_2(\Phi^i) = \bigcup_{E \in \mathfrak{D}} \pi_2(\mathsf{ass}^i(\mathsf{rd}(E)))$ for $i \in [1, k]$, and $\sharp(\mathsf{st}(\mathbf{R})) = 0$. Denote also $\Psi^i = \mathsf{ass}^i(\mathsf{rd}(D'))$ for $i \in [1, k-1]$. It holds

$$\mathbf{R}(1) = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{t}^k : \mathbf{s}^{k-1}{\restriction}_{\pi_2(\Psi^{k-1})} : \mathbf{s}^{k-2}{\restriction}_{\pi_2(\Psi^{k-2})} : \cdots : \mathbf{s}^1{\restriction}_{\pi_2(\Psi^1)} : (\alpha, \{D'\}),$$

where $\mathbf{t}^k = \mathbf{s}^k{\restriction}_{\pi_2(\Phi^k)} : \mathbf{s}^{k-1}{\restriction}_{\pi_2(\Phi^{k-1})} : \cdots : \mathbf{s}^1{\restriction}_{\pi_2(\Phi^1)} : (\gamma, \mathfrak{D})$.

From Lemma 7.19 we obtain the following inequality, that is strict if $f = \mathsf{pr}$:

$$\sum_{i=1}^{k} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Phi^i)}) + \mathsf{low}((\gamma, \mathfrak{D})) \leq \mathsf{low}(\mathbf{t}^k), \tag{7.12}$$

Because $\mathsf{type}(\mathbf{s}^k) = \pi_2(\Phi^k)$ and $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i) \cup \pi_2(\Phi^i)$ for $i \in [1, k-1]$, it holds

$$\begin{aligned}
&\mathsf{low}(\mathbf{s}^k) = \mathsf{low}(\mathbf{s}^k{\restriction}_{\pi_2(\Phi^k)}), \qquad \text{and}\\
&\mathsf{low}(\mathbf{s}^i) \leq \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Psi^i)}) + \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Phi^i)}) \qquad \text{for } i \in [1, k-1].
\end{aligned}$$

Moreover, if for some $i \in [1, k-1]$ we have $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \not\subseteq \mathcal{T}_{\mathsf{np}}$, then the appropriate inequality is strict (since the positive component corresponding to $\tau \in \pi_2(\Psi^i) \cap \pi_2(\Phi^i) \cap \mathcal{T}_{\mathsf{pr}}$ appears in both low's on the right side, and only once on the left side). We apply these inequalities to the definition of $\mathsf{low}(\mathbf{R}(0))$; next we substitute Inequality (7.12); we obtain

$$\begin{aligned}
\mathsf{low}(\mathbf{R}(0)) &\leq \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) + \sum_{i=1}^{k-1} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Psi^i)}) + \sum_{i=1}^{k} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Phi^i)}) + \mathsf{low}(\mathbf{s}^0) \leq\\
&\leq \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) + \sum_{i=1}^{k-1} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\Psi^i)}) + \mathsf{low}(\mathbf{t}^k) - \mathsf{low}((\gamma, \mathfrak{D})) + \mathsf{low}(\mathbf{s}^0) =\\
&= \mathsf{low}(\mathbf{R}(1)) - \mathsf{low}((\alpha, \{D'\})) - \mathsf{low}((\gamma, \mathfrak{D})) + \mathsf{low}(\mathbf{s}^0) \leq \mathsf{low}(\mathbf{R}(1)) + \mathsf{low}(\mathbf{s}^0).
\end{aligned}$$

If $\{\mathsf{rd}(D')\} \cup \pi_2(\Phi^0) \not\subseteq \mathcal{T}_{\mathsf{np}}$, the last inequality is strict, as we have removed negative components. Because $\mathsf{low}(\mathbf{s}^0) \leq 1$, if some on the above inequalities was strict, we can remove $\mathsf{low}(\mathbf{s}^0)$, and we obtain $\mathsf{low}(\mathbf{R}(0)) \leq \mathsf{low}(\mathbf{R}(1))$, as required. On the other hand, if none of these inequalities was strict, we have $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [1, k-1]$, and

$f = \mathsf{np}$, and $\{\mathsf{rd}(D')\} \cup \pi_2(\Phi^0) \subseteq \mathcal{T}_{\mathsf{np}}$; from Definition 7.7(4) it follows that in this case $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$, so $\mathsf{low}(\mathbf{s}^0) = 0$ and we obtain the required inequality as well.

Next, we prove the inequality for high. Denote

$$\begin{aligned} a_i &= \mathsf{high}(\mathbf{s}^i) && \text{for } i \in [k+1, n],\\ a_i &= \mathsf{high}(\mathbf{s}^i\restriction_{\pi_2(\Psi^i)}) && \text{for } i \in [1, k-1],\\ b_i &= \mathsf{high}(\mathbf{s}^i\restriction_{\pi_2(\Phi^i)}) && \text{for } i \in [1, k]. \end{aligned}$$

Suppose first that $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$. Then $\{\mathsf{rd}(D')\} \cup \pi_2(\Phi^0) \subseteq \mathcal{T}_{\mathsf{np}}$ and $f = \mathsf{np}$; we have $\mathsf{high}(\mathbf{s}^0) = \mathsf{high}((\gamma, \mathfrak{D})) = \mathsf{high}((\alpha, \{D'\})) = 1$. Moreover $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [1, k-1]$; thus we have

$$\mathsf{high}(\mathbf{s}^i) = a_i \cdot b_i \qquad \text{for } i \in [1, k-1]. \tag{7.13}$$

Because $f = \mathsf{np}$, from Lemma 7.19 we know that

$$\mathsf{pow}(b_k, b_{k-1}, \dots, b_1, \mathsf{high}((\gamma, \mathfrak{D}))) \geq \mathsf{high}(\mathbf{t}^k).$$

Using Equalities (7.13), then Inequality (7.2), and then the above inequality, we obtain

$$\begin{aligned} \mathsf{high}(\mathbf{R}(0)) &= \mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, b_k, a_{k-1}b_{k-1}, a_{k-2}b_{k-2}, \dots, a_1b_1, 1) \geq\\ &\geq \mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, \mathsf{pow}(b_k, b_{k-1}, \dots, b_1, 1), a_{k-1}, a_{k-2} \dots, a_1, 1) \geq\\ &\geq \mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, \mathsf{high}(\mathbf{t}^k), a_{k-1}, a_{k-2}, \dots, a_1, 1) = \mathsf{high}(\mathbf{R}(1)). \end{aligned}$$

Next, suppose that $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$. Then Lemma 7.19 gives us the inequality

$$\mathsf{pow}\big(b_k, b_{k-1}, \dots, b_1, |\mathcal{T}^0|^n \cdot \mathsf{high}((\gamma, \mathfrak{D}))\big) \geq \mathsf{high}(\mathbf{t}^k). \tag{7.14}$$

By definition it holds

$$\begin{aligned} \mathsf{high}(\mathbf{s}^0) = C_{\mathsf{depth}(D)} &= (2|\mathcal{T}^0|)^n \cdot (C_{\mathsf{depth}(D)-1})^{|\mathcal{T}^0|+1} \geq\\ &\geq 2^{k-1} \cdot |\mathcal{T}^0|^n \cdot C_{\mathsf{depth}(D')} \cdot \prod_{E \in \mathfrak{D}} C_{\mathsf{depth}(E)} \geq\\ &\geq 2^{k-1} \cdot |\mathcal{T}^0|^n \cdot \mathsf{high}((\alpha, \{D'\})) \cdot \mathsf{high}((\gamma, \mathfrak{D})). \end{aligned}$$

Using Inequality (7.3) we replace 2^{k-1} in the last argument of pow by 2 in the $k-1$ previous arguments; then we observe that for each $i \in [1, k-1]$ we have $(\mathsf{high}(\mathbf{s}^i))^2 \geq a_i b_i$; then we use Inequality (7.2), and finally Inequality (7.14):

$$\begin{aligned} \mathsf{high}(\mathbf{R}(0)) &\geq \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^k), (\mathsf{high}(\mathbf{s}^{k-1}))^2, (\mathsf{high}(\mathbf{s}^{k-2}))^2, \dots,\\ &\qquad (\mathsf{high}(\mathbf{s}^1))^2, |\mathcal{T}^0|^n \cdot \mathsf{high}((\alpha, \{D'\})) \cdot \mathsf{high}((\gamma, \mathfrak{D}))\big) \geq\\ &\geq \mathsf{pow}\big(a_n, a_{n-1}, \dots, a_{k+1}, b_k, a_{k-1}b_{k-1}, a_{k-2}b_{k-2}, \dots, a_1b_1,\\ &\qquad |\mathcal{T}^0|^n \cdot \mathsf{high}((\alpha, \{D'\})) \cdot \mathsf{high}((\gamma, \mathfrak{D}))\big) \geq\\ &\geq \mathsf{pow}\big(a_n, a_{n-1}, \dots, a_{k+1}, \mathsf{pow}\big(b_k, b_{k-1}, \dots, b_1, |\mathcal{T}^0|^n \cdot \mathsf{high}((\gamma, \mathfrak{D}))\big),\\ &\qquad a_{k-1}, a_{k-2}, \dots, a_1, \mathsf{high}((\alpha, \{D'\}))\big) \geq\\ &\geq \mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, \mathsf{high}(\mathbf{t}^k), a_{k-1}, a_{k-2}, \dots, a_1, \mathsf{high}((\alpha, \{D'\}))) =\\ &= \mathsf{high}(\mathbf{R}(1)). \end{aligned}$$

The inequality for len can be proved in a very similar way as that for high in the case $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$. □

7.3. **Relating Upper and Lower Bounds.** It is meaningful to consider the functions low and high because they are closely related: one is bounded if the other is bounded.

Proposition 7.20. *There exists a function $H\colon \mathbb{N} \to \mathbb{N}$ such that for each configuration c and each run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(c)$ there exists a well-formed annotated n-stack $\mathbf{s}$ for which $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$, and $\mathsf{conf}(\mathbf{s}) = c$, and $\mathsf{high}(\mathbf{s}) \leq H(\mathsf{low}(\mathbf{s}))$.*

Proof. Let d be a number such that for each derivation tree there exists a derivation tree with the same conclusion and depth at most d; such a number exists, because there are only finitely many possible conclusions. For each $k \geq 0$ we define a function $N_k\colon \mathbb{N} \to \mathbb{N}$, and we take $H = N_n$. The definition is inductive: $N_k(0) = 1$, and for $L > 0$:

$$N_0(L) = (C_d)^{|\mathcal{T}^0|},$$
$$N_k(L) = \big(\mathsf{pow}(N_k(L-1), N_{k-1}(L))\big)^{|\mathcal{T}^{k-1}|} \qquad \text{for } k > 0,$$

where C_d is the constant from Definition 7.13.

By definition for each configuration c and each run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(c)$ there exists a well-formed annotated stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{conf}(\mathbf{s}) = c$. W.l.o.g. we can assume that all derivation trees in s have depth at most d: we can safely replace each tree by another (smaller) tree having the same conclusion. Thus it is enough to prove that for each well-formed annotated k-stack $\mathbf{s}$, in which all derivation trees have depth at most d, it holds $\mathsf{high}(\mathbf{s}) \leq N_k(\mathsf{low}(\mathbf{s}))$.

Denote $L = \mathsf{low}(\mathbf{s})$. If $L = 0$ then $\mathsf{high}(\mathbf{s}) = 1 = N_k(L)$, thanks to Proposition 7.18. Suppose that $L > 0$. In this case we prove the thesis by induction on the structure of $\mathbf{s}$. For a stack $\mathbf{s} = (\gamma, \mathfrak{D})$ of order 0 it holds

$$\mathsf{high}(\mathbf{s}) \leq \prod_{D \in \mathfrak{D}} C_{\mathsf{depth}(D)} \leq (C_d)^{|\mathcal{T}^0|} = N_0(L).$$

Next, consider a stack $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$. Recall that $\mathsf{low}(\mathbf{s})$ equals to the sum of low for $\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}$ and $\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}}$ over all $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$. We have two cases. One possibility is that $\mathsf{low}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}) = L$ for some $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$. Then low for all other considered stacks is 0, so their high is 1. Using the induction assumption we obtain

$$\mathsf{high}(\mathbf{s}) \leq \mathsf{pow}(N_k(L), 1) \cdot \prod_{\tau \in \mathsf{type}(\mathbf{s}^{k-1}) \setminus \{\sigma\}} \mathsf{pow}(1,1) = N_k(L).$$

The opposite situation is that $\mathsf{low}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}) \leq L-1$ for each $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$. Observing that N_k is monotone, by the induction assumption $\mathsf{high}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}) \leq N_k(L-1)$ and $\mathsf{high}(\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}}) \leq N_{k-1}(L)$ for each $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$, so we obtain

$$\mathsf{high}(\mathbf{s}) \leq \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}(N_k(L-1), N_{k-1}(L)) \leq N_k(L).$$

□

7.4. **Assumptions are used in returns.** Our next goal is to formally prove that whenever an assumption of a run descriptor is used in an annotated run, then we have a return.

Lemma 7.21. *Let* $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^0$ *be a well-formed singular annotated* n*-stack, where* $\mathsf{type}(\mathbf{s}^0) = \{\sigma\}$*. If* $(m, \xi) \in \mathsf{ass}^r(\sigma)$*, then there exists an annotated run* $\mathbf{R}$ *starting in* $\mathbf{s}$ *such that* $\mathsf{st}(\mathbf{R})$ *is an* r*-return,* $\varphi(\mathsf{st}(\mathbf{R})) = m$*, and* $\mathsf{top}^r(\mathbf{R}(|\mathbf{R}|)) = \mathbf{s}^r{\restriction_{\{\xi\}}}$.

Proof. We use induction on $\mathsf{len}(\mathbf{s})$. Thanks to Lemma 7.16 we can always use the induction assumption for the successor of $\mathbf{s}$. We have several cases depending on the shape of the derivation tree D in $\mathbf{s}^0$ (that is, on the first operation in an annotated run starting in $\mathbf{s}$). We use the characterization of returns from Proposition 6.8.

Case 1. If $D = (\gamma, p)$ then $\mathsf{ass}^r(\sigma) = \emptyset$, so the assumptions cannot hold.

Case 2. Suppose that $D = (p, D')$. Then the successor $\mathbf{t}$ of $\mathbf{s}$ differs from $\mathbf{s}$ only in the topmost 0-stack; the new topmost 0-stack has type $\{\tau\}$ such that $\mathsf{ass}^r(\sigma) = \varphi(a) \circ \mathsf{ass}^r(\tau)$, where a is the letter read by the step between $\mathsf{conf}(\mathbf{s})$ and $\mathsf{conf}(\mathbf{t})$. Consider an element m' such that $m = \varphi(a) \cdot m'$ and $(m', \xi) \in \mathsf{ass}^r(\tau)$. By the induction assumption for $\mathbf{t}$, there exists an annotated run $\mathbf{S}$ starting in $\mathbf{t}$ such that $\mathsf{st}(\mathbf{S})$ is an r-return, $\varphi(\mathsf{st}(\mathbf{S})) = m'$, and $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r{\restriction_{\{\xi\}}}$. Together with the step between $\mathbf{s}$ and $\mathbf{t}$, it gives us an annotated run as required.

Case 3. Suppose that $D = (\gamma, p, \tau)$, where $\tau \in \mathcal{T}^k$. The successor of $\mathbf{s}$ is $\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^k$. Recall that $\mathsf{ass}^i(\sigma) = \emptyset$ for $i < k$, so $r \geq k$. If $r = k$, then $(m, \xi) = (\mathbf{1}, \tau)$. In this case the annotated run of length 1 satisfies the thesis. Otherwise $r > k$, and $(m, \xi) \in \mathsf{ass}^r(\tau)$. Then as well $(m, \xi) \in \mathsf{ass}^r(\tau')$, where τ' is the run descriptor in the type of $\mathsf{top}^0(\mathbf{s}^k)$. The induction assumption for $\mathbf{t}$ gives us an annotated run $\mathbf{S}$ starting in $\mathbf{t}$ such that $\mathsf{st}(\mathbf{S})$ is an r-return, $\varphi(\mathsf{st}(\mathbf{S})) = m$, and $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r{\restriction_{\{\xi\}}}$. Together with the step between $\mathbf{s}$ and $\mathbf{t}$, it gives us an annotated run as required.

Case 4. Suppose that $D = (\gamma, p, D', \mathfrak{D})$. Let α, k, Ψ^i, Φ^i be as in Definition 7.7(4). The successor of $\mathbf{s}$ is

$$\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^{k+1} : \mathbf{t}^k : \mathbf{s}^{k-1}{\restriction_{\pi_2(\Psi^{k-1})}} : \mathbf{s}^{k-2}{\restriction_{\pi_2(\Psi^{k-2})}} : \dots : \mathbf{s}^1{\restriction_{\pi_2(\Psi^1)}} : (\alpha, \{D'\}),$$

where $\mathbf{t}^k = \mathbf{s}^k{\restriction_{\pi_2(\Phi^k)}} : \mathbf{s}^{k-1}{\restriction_{\pi_2(\Phi^{k-1})}} : \dots : \mathbf{s}^1{\restriction_{\pi_2(\Phi^1)}} : (\gamma, \mathfrak{D})$. If $(m, \xi) \in \Psi^r$ and $r \neq k$, then the induction assumption for $\mathbf{t}$ gives us an annotated run $\mathbf{S}$ starting in $\mathbf{t}$ such that $\mathsf{st}(\mathbf{S})$ is an r-return, $\varphi(\mathsf{st}(\mathbf{S})) = m$, and $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r{\restriction_{\{\xi\}}}$; together with the step between $\mathbf{s}$ and $\mathbf{t}$, it gives us an annotated run as required. Otherwise $(m, \xi) \in \Phi^r$ and $r \leq k$. Recall that we have a composer $(\Phi^k, \Phi^{k-1}, \dots, \Phi^0; \Psi^k; f)$. The definition of a composer gives us some $(m_1, \tau) \in \Phi^0$ and $m_2 \in M$ such that $m = m_1 \cdot m_2$ and $(m_2, \xi) \in \mathsf{ass}^r(\tau)$. Then we have $(m_1, \mathsf{red}^k(\tau)) \in \Psi^k$. The induction assumption for $\mathbf{t}$ and $(m_1, \mathsf{red}^k(\tau)) \in \Psi^k$ gives us an annotated run $\mathbf{S}$ starting in $\mathbf{t}$ such that $\mathsf{st}(\mathbf{S})$ is a k-return, $\varphi(\mathsf{st}(\mathbf{S})) = m_1$, and $\mathsf{top}^k(\mathbf{S}(|\mathbf{S}|)) = \mathbf{t}^k{\restriction_{\{\mathsf{red}^k(\tau)\}}}$. Notice that

$$\mathbf{t}^k{\restriction_{\{\mathsf{red}^k(\tau)\}}} = \mathbf{s}^k{\restriction_{\pi_2(\mathsf{ass}^k(\tau))}} : \mathbf{s}^{k-1}{\restriction_{\pi_2(\mathsf{ass}^{k-1}(\tau))}} : \dots : \mathbf{s}^1{\restriction_{\pi_2(\mathsf{ass}^1(\tau))}} : (\gamma, \mathfrak{D}){\restriction_{\{\tau\}}}.$$

Recalling that $r \leq k$, the induction assumption for $\mathbf{S}(|\mathbf{S}|)$ and $(m_2, \xi) \in \mathsf{ass}^r(\tau)$ gives us an annotated run $\mathbf{T}$ starting in $\mathbf{S}(|\mathbf{S}|)$ such that $\mathsf{st}(\mathbf{T})$ is an r-return, $\varphi(\mathsf{st}(\mathbf{T})) = m_2$, and $\mathsf{top}^r(\mathbf{T}(|\mathbf{T}|)) = \mathbf{s}^r{\restriction_{\{\xi\}}}$. The step between $\mathbf{s}$ and $\mathbf{t}$ composed with $\mathbf{S}$ and then with $\mathbf{T}$ gives us an annotated run as required. □

7.5. **Completeness of Types.** In the previous subsection we have proved soundness of the type system, which means that if a run descriptor is contained in the type of a configuration then a corresponding run exists from this configuration. As usual, we need the opposite direction (completeness) as well, that is having a run from a configuration, we want to imply that the corresponding run descriptor is in the type of this configuration. While reversing Lemma 7.21 we have to remember that not every run can be extended to an annotated run, so in Lemma 7.22 we need to use standard runs.

Lemma 7.22. *Let R be an r-return, and let $\xi \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$. Then there exists a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $(\varphi(R), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\sigma)$.*

We first state four auxiliary lemmas. These lemmas will be used also in the next subsection.

Lemma 7.23. *Let R be a run of length 1 whose transition is read, and let $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$. Then there exists $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\mathsf{ass}^i(\sigma) = \varphi(R) \circ \mathsf{ass}^i(\tau)$ for each $i \in [1, n]$.*

Proof. By definition of $\mathsf{type}_{\mathcal{A},\varphi}$, there exists a well-formed annotated stack $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\gamma, \{D'\})$ which after removing annotations gives $\pi_2(R(1))$, and such that $\mathsf{rd}(D') = \tau$. Well-formedness implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\tau))$ for each $i \in [1, n]$. When p is the state of $R(0)$, Definition 7.7(2) implies that $D = (p, D')$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where $\sigma = (p, \Phi^n, \Phi^{n-1}, \ldots, \Phi^1, f)$ and $\Phi^i = \varphi(R) \circ \mathsf{ass}^i(\tau)$ for each $i \in [1, n]$. Because $\pi_2(\mathsf{ass}^i(\sigma)) = \pi_2(\mathsf{ass}^i(\tau))$, the annotated stack $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\gamma, \{D\})$ is well-formed, so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$. □

Lemma 7.24. *Let R be a run of length 1 performing pop^k, and let $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$. Then there exists $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\mathsf{ass}^i(\sigma) = \mathsf{ass}^i(\tau)$ for each $i \in [k+1, n]$, and $\mathsf{ass}^k(\sigma) = \{(\mathbf{1}_M, \mathsf{red}^k(\tau))\}$.*

Proof. Denote $R(0) = (p, s^n : s^{n-1} : \cdots : s^0)$; then $\pi_2(R(1)) = s^n : s^{n-1} : \cdots : s^k$. By definition of $\mathsf{type}_{\mathcal{A},\varphi}$, there exists a well-formed annotated stack $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$ such that $\mathsf{type}(\mathbf{s}^k) = \{\mathsf{red}^k(\tau)\}$, and $\mathsf{st}(\mathbf{s}^i) = s^i$ for each $i \in [k, n]$. Well-formedness implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\tau))$ for each $i \in [k+1, n]$. For $i \in [1, k-1]$ let $\mathbf{s}^i$ be the well-formed annotated i-stack such that $\mathsf{type}(\mathbf{s}^i) = \emptyset$ and $\mathsf{st}(\mathbf{s}^i) = s^i$ (we annotate s^i by empty sets). Finally, by Definition 7.7(3), $D = (p, s^0, \mathsf{red}^k(\xi))$ is a derivation tree with conclusion $s^0 \vdash \sigma$, where

$$\sigma = (p, \mathsf{ass}^n(\tau), \mathsf{ass}^{n-1}(\tau), \ldots, \mathsf{ass}^{k+1}(\tau), \{(\mathbf{1}_M, \mathsf{red}^k(\tau)\}, \emptyset, \ldots, \emptyset, \mathsf{np}),$$

so $\mathbf{s}^0 = (s^0, \{D\})$ has type $\{\sigma\}$. We observe that $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^0$ is well-formed, so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$. □

Lemma 7.25. *Let R be a run of length 1 performing push^k_α, and let $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$. Then there exists $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\mathsf{ass}^i(\tau) \subseteq \mathsf{ass}^i(\sigma)$ for each $i \in [1, n] \setminus \{k\}$.*

Proof. Before starting the actual proof, we observe that for each pair of well-formed annotated stacks $\mathbf{s}$, $\mathbf{t}$ such that $\mathsf{st}(\mathbf{s}) = \mathsf{st}(\mathbf{t})$ we can construct a well-formed annotated stack $\mathbf{s} \oplus \mathbf{t}$ whose type is $\mathsf{type}(\mathbf{s}) \cup \mathsf{type}(\mathbf{t})$ and such that $\mathsf{st}(\mathbf{s} \oplus \mathbf{t}) = \mathsf{st}(\mathbf{s})$. We construct $\mathbf{s} \oplus \mathbf{t}$ by induction on the structure of $\mathbf{s}$. Denote $\widetilde{\Psi} = \mathsf{type}(\mathbf{t}) \setminus \mathsf{type}(\mathbf{s})$. If $\mathbf{s}$ is of order 0, then we take $\mathbf{s} \oplus \mathbf{t} = (\gamma, \mathfrak{D} \cup \mathfrak{D}')$, where $\mathbf{s} = (\gamma, \mathfrak{D})$ and $\mathbf{t}{\restriction_{\widetilde{\Psi}}} = (\gamma, \mathfrak{D}')$. If $\mathbf{s} = \mathbf{t} = [\,]$, then $\mathbf{s} \oplus \mathbf{t} = [\,]$ is fine. If $\mathbf{s} = \mathbf{s}^j : \mathbf{s}^{j-1}$ and $\mathbf{t}{\restriction_{\widetilde{\Psi}}} = \mathbf{t}^j : \mathbf{t}^{j-1}$, then as $\mathbf{s} \oplus \mathbf{t}$ we take $(\mathbf{s}^j \oplus \mathbf{t}^j) : (\mathbf{s}^{j-1} \oplus \mathbf{t}^{j-1})$; observe that it is well-formed, because the types of $\mathbf{s}$ and $\mathbf{t}{\restriction_{\widetilde{\Psi}}}$ are disjoint.

Denote $R(0) = (p, s^n : s^{n-1} : \dots : s^1 : \gamma)$; then

$$\pi_2(R(1)) = s^n : s^{n-1} : \dots : s^{k+1} : (s^k : s^{k-1} : \dots : s^1 : \gamma) : s^{k-1} : s^{k-2} : \dots : s^1 : \alpha.$$

By definition of $\mathsf{type}_{\mathcal{A},\varphi}$, there exists a well-formed annotated stack $\mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^1 : (\alpha, \{D'\})$ in which $\mathbf{s}^k = \mathbf{t}^k : \mathbf{t}^{k-1} : \dots : \mathbf{t}^1 : (\gamma, \mathfrak{D})$, such that $\mathsf{rd}(D') = \tau$ and $\mathsf{st}(\mathbf{s}^i) = s^i$ for each $i \in [1,n] \setminus \{k\}$, and $\mathsf{st}(\mathbf{t}^i) = s^i$ for each $i \in [1,k]$. Denote $\Psi^i = \mathsf{ass}^i(\tau)$ for each $i \in [1,n]$. Well-formedness implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i)$ for each $i \in [1,n]$, and, thanks to Proposition 7.9, there exists a composer $(\Phi^k, \Phi^{k-1}, \dots, \Phi^0; \Psi^k; f)$ such that $\mathsf{type}(\mathbf{t}^i) = \pi_2(\Phi^i)$ for each $i \in [1,k]$ and $\{\mathsf{rd}(E)\colon E \in \mathfrak{D}\} = \pi_2(\Phi^0)$. By Definition 7.7(4), $D = (p, \gamma, D', \mathfrak{D})$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where

$$\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, \Phi^k, \Psi^{k-1} \cup \Phi^{k-1}, \Psi^{k-2} \cup \Phi^{k-2}, \dots, \Psi^1 \cup \Phi^1, g).$$

We observe that the annotated stack

$$\mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^{k+1} : \mathbf{t}^k : (\mathbf{s}^{k-1} \oplus \mathbf{t}^{k-1}) : (\mathbf{s}^{k-2} \oplus \mathbf{t}^{k-2}) : \dots : (\mathbf{s}^1 \oplus \mathbf{t}^1) : (\gamma, \{D\})$$

is well-formed, so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$. □

Lemma 7.26. *Let R be a run in which $R\restriction_{0,1}$ performs push^k_α and $R\restriction_{1,|R|}$ is a k-return, let $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$, and let $\rho \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$ be such that $(\varphi(R), \mathsf{red}^k(\tau)) \in \mathsf{ass}^k(\rho)$. Then there exists $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\varphi(R) \circ \mathsf{ass}^i(\tau) \subseteq \mathsf{ass}^i(\sigma)$ for each $i \in [1,k]$.*

Proof. Denote $R(0) = (p, s^n : s^{n-1} : \dots : s^1 : \gamma)$; then $\pi_2(R(1))$ is as in the previous lemma, and the topmost k-stacks of $R(0)$ and of $R(|R|)$ are equal (due to Proposition 6.6). The definition of $\mathsf{type}_{\mathcal{A},\varphi}$ gives us a well-formed annotated k-stack $\mathbf{u}^k$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{u}^k)) = \{\tau\}$ and $\mathsf{st}(\mathbf{u}^k) = s^k : s^{k-1} : \dots : s^1 : \gamma$, and a well-formed annotated stack $\mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^1 : (\alpha, \{D'\})$ such that $\mathsf{rd}(D') = \rho$ and $\mathsf{st}(\mathbf{s}^i) = s^i$ for each $i \in [1,n] \setminus \{k\}$, and $\mathsf{st}(\mathbf{s}^k) = s^k : s^{k-1} : \dots : s^1 : \gamma$. Denote $\Psi^i = \mathsf{ass}^i(\rho)$ for each $i \in [1,n]$. Well-formedness implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i)$ for each $i \in [1,n]$. By assumption $\mathsf{type}(\mathbf{u}^k) = \{\mathsf{red}^k(\tau)\} \subseteq \pi_2(\Psi^k)$, so the annotated stack $\mathbf{u}^k \oplus \mathbf{s}^k$ has type $\pi_2(\Psi^k)$, similarly to $\mathbf{s}^k$, but additionally $\tau \in \mathsf{type}(\mathsf{top}^0(\mathbf{u}^k \oplus \mathbf{s}^k))$ (recalling the construction from the previous subsection, we see that to $\mathbf{u}^k \oplus \mathbf{s}^k$ we take all annotations from $\mathbf{u}^k$ and some annotations from $\mathbf{s}^k$). Denote $\mathbf{u}^k \oplus \mathbf{s}^k = \mathbf{t}^k : \mathbf{t}^{k-1} : \dots : \mathbf{t}^1 : (\gamma, \mathfrak{D})$. By Proposition 7.9 we have a composer $(\Phi^k, \Phi^{k-1}, \dots, \Phi^0; \Psi^k; f)$ such that $\mathsf{type}(\mathbf{t}^i) = \pi_2(\Phi^i)$ for each $i \in [1,k]$ and $\{\mathsf{rd}(E)\colon E \in \mathfrak{D}\} = \pi_2(\Phi^0)$. Because $\tau \in \Phi^0$ and $(\varphi(R), \mathsf{red}^k(\tau)) \in \Psi^k$, it holds $(\varphi(R), \tau) \in \Phi^0$, which implies $\varphi(R) \circ \mathsf{ass}^i(\tau) \subseteq \Phi^i$ for each $i \in [1,k]$. By Definition 7.7(4), $D = (p, \gamma, D', \mathfrak{D})$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where

$$\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, \Phi^k, \Psi^{k-1} \cup \Phi^{k-1}, \Psi^{k-2} \cup \Phi^{k-2}, \dots, \Psi^1 \cup \Phi^1, g).$$

We observe that the annotated stack

$$\mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^{k+1} : \mathbf{t}^k : (\mathbf{s}^{k-1} \oplus \mathbf{t}^{k-1}) : (\mathbf{s}^{k-2} \oplus \mathbf{t}^{k-2}) : \dots : (\mathbf{s}^1 \oplus \mathbf{t}^1) : (\gamma, \{D\})$$

is well-formed, so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$. □

Proof of Lemma 7.22. We use induction on the length of the r-return R. Proposition 6.8 gives us possible forms of R; we analyze these cases.

Suppose first that $|R| = 1$ and the only transition of R is performs pop^r. We take σ from Lemma 7.24, where we take ξ as τ and r as k. By assumption $\varphi(R) = \mathbf{1}_M$, so $(\varphi(R), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\sigma)$.

Next, suppose that $R{\restriction}_{1,|R|}$ is an r-return, and the first transition of R is read, or performs pop^k for $k < r$, or push^k_α for $k \neq r$. The induction assumption for $R{\restriction}_{1,|R|}$ gives us a run descriptor $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$ such that $(\varphi(R{\restriction}_{1,|R|}), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\tau)$, and Lemma 7.23 or 7.24 or 7.25, respectively, used for $R{\restriction}_{0,1}$ gives us a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\varphi(R{\restriction}_{0,1}) \circ \mathsf{ass}^r(\tau) \subseteq \mathsf{ass}^r(\sigma)$ (where $\varphi(R{\restriction}_{0,1})$ may be nontrivial only when the transition is read).

Finally, suppose that the first transition of R performs push^k_α for $k \geq r$ and $R{\restriction}_{1,|R|} = S \circ T$ for some k-return S and r-return T. The induction assumption for T gives us a run descriptor $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(T(0))$ such that $(\varphi(T), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\tau)$, and the induction assumption for S gives us a run descriptor $\rho \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$ such that $(\varphi(S), \mathsf{red}^k(\tau)) \in \mathsf{ass}^k(\rho)$. Using Lemma 7.26 for $R{\restriction}_{0,1} \circ S$ we obtain a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ such that $\varphi(S) \circ \mathsf{ass}^r(\tau) \subseteq \mathsf{ass}^r(\sigma)$ (recalling that $r \leq k$), so $(\varphi(R), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\sigma)$. □

7.6. **Reproducing Upper Runs.** Till now we were using types to describe returns from a configuration, but thanks to the decomposition given by Proposition 6.7 we can also describe r-upper runs. This is stated in the following lemma.

Lemma 7.27. *Let R be an r-upper run (where $r \in [0, n]$), and let $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$. Then there exists a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ and a monotone function $f_R \colon \mathbb{N} \to \mathbb{N}$ such that the following is satisfied. Let $\mathbf{s}$ be a well-formed annotated n-stack such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and the topmost r-stacks of $\mathsf{conf}(\mathbf{s})$ and of $R(0)$ are equal. Then there exists a well-formed annotated n-stack $\mathbf{t}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{t})) = \{\tau\}$, and there exists a run S from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{t})$ which is (r, φ)-parallel to R, and*

$$\mathsf{low}(\mathbf{s}) \leq f_R(\sharp(S) + \mathsf{low}(\mathbf{t})), \qquad f_R(\mathsf{high}(\mathbf{s})) \geq \sharp(S) + \mathsf{high}(\mathbf{t}).$$

Before proving this lemma we observe that Theorem 7.3 follows from it immediately. For this purpose the inequalities regarding low and high are redundant; they will be used later to prove Theorem 7.4.

Proof of Theorem 7.3. Recall that we are given a k-upper run R, and a configuration c having the same $(\mathcal{A}, \varphi)$-type and the same topmost k-stack as $R(0)$. Consider the run descriptor $\tau = (\pi_1(R(|R|)), \emptyset, \dots, \emptyset, \mathsf{np})$ and observe that $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$ (we annotate the topmost 0-stack of $\pi_2(R(|R|))$ by the derivation tree from Definition 7.7(1)). Applying Lemma 7.27 we obtain a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0)) = \mathsf{type}_{\mathcal{A},\varphi}(c)$. Then we take any well-formed annotated n-stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{conf}(\mathbf{s}) = c$, existing by the definition of $\mathsf{type}_{\mathcal{A},\varphi}$. Since the topmost k-stacks of $\mathsf{conf}(\mathbf{s})$ and of $R(0)$ are equal, from the second part of Lemma 7.27 we obtain a run S as required. □

Proof of Lemma 7.27. The proof is by induction on the length of R. Proposition 6.7 gives us possible forms of R; we analyze these cases.

If R has length 0, then we can take τ as σ and the identity function as f_R; given $\mathbf{s}$ we take it as $\mathbf{t}$, and the run of length 0 from $\mathsf{conf}(\mathbf{s})$ as S.

Suppose that R has length 1, and its transition either is read or performs push^k_α. Then we take the identity function as f_R, and we construct σ out of R and τ as in Lemma 7.23 or Lemma 7.25, respectively. Next, we are given a well-formed annotated n-stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and the topmost r-stacks of $\mathsf{conf}(\mathbf{s})$ and of $R(0)$ are equal. As $\mathbf{t}$ we take the successor of $\mathbf{s}$, and as S the one-step run from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{t})$. Comparing the construction of σ in the proof of the mentioned lemmas with Definition 7.11, we see that the successor of $\mathbf{s}$ indeed exists, and that $\mathsf{type}(\mathsf{top}^0(\mathbf{t})) = \{\tau\}$. Moreover, S performs the same transition as R, and in the case of read it reads the same letter, so S is (r,φ)-parallel to R. The inequalities follow from Lemma 7.16.

Next, suppose that $R\restriction_{0,1}$ performs push^k_α and $R\restriction_{1,|R|}$ is a k-return, where $k \geq r+1$. This case is similar, but slightly more complicated. First, using Lemma 7.22, we construct a run descriptor $\rho \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$ such that $(\varphi(R), \mathsf{red}^k(\tau)) \in \mathsf{ass}^k(\rho)$. As f_R we take the identity function, and we construct σ out of R, τ and ρ as in Lemma 7.26. When we are gives $\mathbf{s}$, we proceed as follows. Let $\mathbf{u} = \mathbf{u}^n : \mathbf{u}^{n-1} : \cdots : \mathbf{u}^0$ be the successor of $\mathbf{s}$. Inspecting the proof of Lemma 7.26 we see that $\mathbf{u}$ indeed exists, and $\mathsf{type}(\mathsf{top}^0(\mathbf{u})) = \{\rho\}$, and $\tau \in \mathsf{top}^0(\mathbf{u}^k)$. Lemma 7.21 gives us an annotated run $\mathbf{S}$ starting in $\mathbf{u}$ such that $\mathsf{st}(\mathbf{S})$ is a k-return, $\varphi(\mathsf{st}(\mathbf{S})) = \varphi(R)$, and $\mathsf{top}^k(\mathbf{S}(|\mathbf{S}|)) = \mathbf{u}^k\restriction_{\mathsf{red}^k(\tau)}$. As S we take the composition of the one-step run from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{t})$ with the run $\mathsf{st}(\mathbf{S})$. By Proposition 6.6, the topmost k-stacks of $R(0)$ and of $R(|R|)$ are equal, and the topmost k-stacks of $S(0)$ and of $S(|S|)$ are equal; since $k \geq r+1$, the topmost r-stacks of $R(|R|)$ and $S(|S|)$ are equal as well. Together with $\varphi(S) = \varphi(R)$ this means that R and S are (r,φ)-parallel, because by definition no prefix of a k-return can be $(k-1)$-upper (r-upper). Again, the inequalities follow from Lemma 7.16.

Next, suppose that R has length 1 and performs pop^k for $k \leq r$. We construct σ out of R and τ as in Lemma 7.24. Because $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$, we can construct a well-formed annotated k-stack $\mathbf{u}^k$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{u}^k)) = \{\tau\}$, and $\mathsf{st}(\mathbf{u}^k)$ is the topmost k-stack of $R(1)$. As f_R we take a monotone function such that for each tuple $(a_{k+1}, \dots, a_n)$ of positive integers it holds

$$f_R(\mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, a_k)) \geq \mathsf{pow}(a_n, a_{n-1}, \dots, a_{k+1}, \mathsf{high}(\mathbf{u}^k)),$$

and for each $N \in \mathbb{N}$ and each annotated k-stack $\mathbf{s}^k$ such that $\mathsf{st}(\mathbf{s}^k) = \mathsf{st}(\mathbf{u}^k)$ it holds

$$f_R(N + \mathsf{low}(\mathbf{u}^k)) \geq N + \mathsf{low}(\mathbf{s}^k).$$

Notice that for each $H \in \mathbb{N}$ there are finitely many tuples such that $\mathsf{pow}(a_n, a_{n-1}, \dots, a_k) \leq H$ (in particular none of a_i may be greater than H). Recall also that $\mathsf{low}(\mathbf{s}^k)$ is equal to the number of derivation trees with productive run descriptor annotating some 0-stack in $\mathbf{s}^k$. So, although there are infinitely many annotated k-stacks $\mathbf{s}^k$, the value of $\mathsf{low}(\mathbf{s}^k)$ is bounded by the number of 0-stacks in $\mathsf{st}(\mathbf{s}^k)$ times $|\mathcal{T}^0|$. Thus such a function f_R exists.

Then, we are given an annotated stack $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^0$. The successor of $\mathbf{s}$ is $\mathbf{t}' = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$. It would be incorrect to take $\mathbf{t}'$ as $\mathbf{t}$: we only know that $\mathsf{type}(\mathsf{top}^k(\mathbf{t}')) = \{\mathsf{red}^k(\tau)\}$, but we require that $\mathsf{type}(\mathsf{top}^0(\mathbf{t})) = \{\tau\}$. Instead, we take $\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{u}^k$. The topmost r-stacks of $\mathsf{conf}(\mathbf{s})$ and $R(0)$ are equal, and $k \leq r$, so $\mathsf{st}(\mathbf{u}^k) = \mathsf{st}(\mathbf{s}^k)$, and the topmost r-stacks of $\mathsf{conf}(\mathbf{t})$ and $R(1)$ are equal as well.

As S we take the one-step run from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{t})$; it is (r,φ)-parallel to R. Recalling that $\sharp(S)=0$, and using Lemma 7.16 and the definition of f_R we obtain

$$f_R(\mathsf{low}(\mathbf{t})) = f_R\Big(\sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i)+\mathsf{low}(\mathbf{u}^k)\Big) \geq \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i)+\mathsf{low}(\mathbf{s}^k) = \mathsf{low}(\mathbf{t}') \geq \mathsf{low}(\mathbf{s}),$$

$$f_R(\mathsf{high}(\mathbf{s})) \geq f_R(\mathsf{high}(\mathbf{t}')) = f_R\big(\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n),\mathsf{high}(\mathbf{s}^{n-1}),\ldots,\mathsf{high}(\mathbf{s}^{k+1}),\mathsf{high}(\mathbf{s}^k)\big)\big) \geq$$
$$\geq \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n),\mathsf{high}(\mathbf{s}^{n-1}),\ldots,\mathsf{high}(\mathbf{s}^{k+1}),\mathsf{high}(\mathbf{u}^k)\big) = \mathsf{high}(\mathbf{t}).$$

Finally, suppose that R is a composition of shorter k-upper runs R_1 and R_2. The induction assumption used for R_2 and for τ gives us a run descriptor $\rho \in \mathsf{type}_{\mathcal{A},\varphi}(R_2(0))$ and a function f_2. Then, the induction assumption used for R_1 and for ρ gives us a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ and a function f_1. As f_R we take a monotone function such that for each pair a, b of natural numbers it holds $f_R(a) \geq f_1(a)+f_2(f_1(a))$ and $f_R(a+b) \geq f_1(a+f_2(b))$.

Then, we are given a well-formed annotated n-stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s}))=\{\sigma\}$ and the topmost r-stacks of $\mathsf{conf}(\mathbf{s})$ and of $R(0)$ are equal. From the induction assumption for R_1 we obtain a well-formed annotated n-stack $\mathbf{u}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{u}))=\{\rho\}$, and a run S_1 from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{u})$ being (r,φ)-parallel to R_1. Then, from the induction assumption for R_2 we obtain a well-formed annotated n-stack $\mathbf{t}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{t}))=\{\tau\}$, and a run S_2 from $\mathsf{conf}(\mathbf{u})$ to $\mathsf{conf}(\mathbf{t})$ being (r,φ)-parallel to R_2. As S we take the composition of S_1 and S_2; it is (r,φ)-parallel to R. Using the inequalities from the induction assumption we obtain:

$$f_R(\sharp(S)+\mathsf{low}(\mathbf{t})) \geq f_1(\sharp(S_1)+f_2(\sharp(S_2)+\mathsf{low}(\mathbf{t})) \geq f_1(\sharp(S_1)+\mathsf{low}(\mathbf{u})) \geq \mathsf{low}(\mathbf{s}),$$
$$f_R(\mathsf{high}(\mathbf{s})) \geq f_1(\mathsf{high}(\mathbf{s}))+f_2(f_1(\mathsf{high}(\mathbf{s})) \geq \sharp(S_1)+f_2(\mathsf{high}(\mathbf{u})) \geq \sharp(S)+\mathsf{high}(\mathbf{t}).$$

□

7.7. **Sequence Equivalence.** In the final part of this section we define the sequence equivalence, and we prove Theorem 7.4.

Definition 7.28. Let $(c_i)_{i=1}^{\infty}$ be a sequence of configurations. We define $\mathsf{stype}((c_i)_{i=1}^{\infty}) \subseteq \mathcal{T}^0$ to be the set of such $\sigma \in \mathcal{T}^0$ that there exists a sequence of well-formed annotated n-stacks $(\mathbf{s}_i)_{i=1}^{\infty}$ for which $\mathsf{type}(\mathsf{top}^0(\mathbf{s}_i))=\{\sigma\}$ and $\mathsf{conf}(\mathbf{s}_i)=c_i$ for each i, and the sequence $(\mathsf{high}(\mathbf{s}_i))_{i=1}^{\infty}$ is bounded. We say that two sequences of configurations, $(c_i)_{i=1}^{\infty}$ and $(d_i)_{i=1}^{\infty}$, are $(\mathcal{A},\varphi)$-*sequence equivalent* when it holds $\mathsf{stype}((c_i)_{i=1}^{\infty})=\mathsf{stype}((d_i)_{i=1}^{\infty})$.

Proof of Theorem 7.4. Let $\xi=(\pi_1(R'(|R'|)),\emptyset,\ldots,\emptyset,\mathsf{np})$; we see that $\xi \in \mathsf{type}_{\mathcal{A},\varphi}(R'(|R'|))$. Lemma 7.22 applied to R' and ξ implies that $\mathsf{type}_{\mathcal{A},\varphi}(R'(0))$ contains a run descriptor τ such that $(\varphi(R'),\mathsf{red}^n(\xi)) \in \mathsf{ass}^n(\tau)$. Then, Lemma 7.27 applied to R and τ gives us a run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$ and a function f_R. We have two cases.

Case 1. Suppose first that $\sigma \in \mathsf{stype}((c_i)_{i=0}^{\infty})=\mathsf{stype}((d_i)_{i=0}^{\infty})$. Then we have a sequence of annotated n-stacks $(\mathbf{s}_i)_{i=1}^{\infty}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s}_i))=\{\sigma\}$ and $\mathsf{conf}(\mathbf{s}_i)=c_i$ for each i, and the sequence $(\mathsf{high}(\mathbf{s}_i))_{i=1}^{\infty}$ is bounded. Recall that the topmost k-stacks of c_i and of $R(0)$ are equal, for each i. We use the second part of Lemma 7.27 for the annotated stack $\mathbf{s}_i$. We obtain a well-formed annotated n-stack $\mathbf{t}_i$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{t}_i))=\{\tau\}$, and a run S_i from c_i to $\mathsf{conf}(\mathbf{t}_i)$ being (k,φ)-parallel to R such that $f_R(\mathsf{high}(\mathbf{s}_i)) \geq \sharp(S_i)+\mathsf{high}(\mathbf{t}_i)$. Next, for each i we apply Lemma 7.21 for $\mathbf{t}_i$ and for the pair $(\varphi(R'),\mathsf{red}^n(\xi))$. We obtain

an annotated run $\mathbf{S}'_i$ starting in $\mathbf{t}_i$ such that $\mathsf{st}(\mathbf{S}'_i)$ is an n-return, $\varphi(\mathsf{st}(\mathbf{S}'_i)) = \varphi(R')$, and $\mathsf{type}(\mathbf{S}'_i) = \{\mathsf{red}^n(\xi)\}$. Let $S'_i = \mathsf{st}(\mathbf{S}'_i)$, and $\mathbf{u}_i = \mathbf{S}'_i(|\mathbf{S}'_i|)$. Thanks to Lemma 7.16, $\mathsf{high}(\mathbf{t}_i) \geq \sharp(S'_i) + \mathsf{high}(\mathbf{u}_i)$. Because $(\mathsf{high}(\mathbf{s}_i))_{i=0}^{\infty}$ is bounded, we see that the sequence $(\sharp(S_i \circ S'_i))_{i=0}^{\infty}$ is bounded as well.

We perform the same construction for $(d_i)_{i=0}^{\infty}$, obtaining runs $T_i \circ T'_i$ from d_i, such that $(\sharp(T_i \circ T'_i))_{i=0}^{\infty}$ is bounded.

Case 2. This is the opposite case: we suppose that $\sigma \notin \mathsf{stype}((c_i)_{i=0}^{\infty})$. For each i we have $\sigma \in \mathsf{type}(c_i)$; using Proposition 7.20 we construct a well-formed annotated n-stack $\mathbf{s}_i$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s}_i)) = \{\sigma\}$ and $\mathsf{conf}(\mathbf{s}_i) = c_i$, and $\mathsf{high}(\mathbf{s}_i) \leq H(\mathsf{low}(\mathbf{s}_i))$ for a function H not depending on i. Our assumption ensures that $(\mathsf{high}(\mathbf{s}_i))_{i=1}^{\infty}$ is unbounded, so $(\mathsf{low}(\mathbf{s}_i))_{i=1}^{\infty}$ is unbounded as well. We construct the runs exactly in the same way as in Case 1, but this time we concentrate on the opposite inequalities. For each i it holds $\mathsf{low}(\mathbf{s}_i) \leq f_R(\sharp(S_i) + \mathsf{low}(\mathbf{t}_i)) \leq f_R(\sharp(S_i \circ S'_i) + \mathsf{low}(\mathbf{u}_i))$. Additionally $\mathsf{low}(\mathbf{u}_i) = 0$, because by construction $\mathsf{type}(\mathbf{u}_i) = \{\mathsf{red}^n(\xi)\} \subseteq \mathcal{T}_{\mathsf{np}}$. It follows that $(\sharp(S_i \circ S'_i))_{i=0}^{\infty}$ is unbounded, and similarly $(\sharp(T_i \circ T'_i))_{i=0}^{\infty}$. □

8. Milestone Configurations

In this section we define so-called milestone configurations and we show their basic properties. The intuitions are as follows. Consider a long run reading only stars. Looking globally, the stack grows (or remains unchanged). However locally, some parts of the stack might be constructed, and few steps later removed. In order to handle this behavior, we concentrate on these configurations of the run in which the stack is minimal (in appropriate sense) and will not be destroyed later; they will be called milestone configurations.

The idea of considering milestone configurations comes from [8], but our definition is slightly different (namely, their definition is relative to a run, which can be arbitrary, while our definition is absolute, we always consider the run reading only stars). For this section we fix an n-DPDA $\mathcal{A}$ with stack alphabet Γ and with input alphabet A containing a distinguished symbol denoted $\star$ (star).

Definition 8.1. We say that a configuration c is a *milestone* (or a milestone configuration) if there exists an infinite run R from c reading only stars, and an infinite set I of indices such that $0 \in I$, and $R{\restriction_{i,j}} \in \mathsf{up}^0$ for each $i, j \in I$, $i \leq j$.

Example 8.2. Consider a DPDA of order 3. Suppose that there is a run which begins in a stack $[[aa]]$, and performs forever the following sequence of operations, in a loop:

$$\mathsf{push}^2_a,\ \mathsf{push}^3_a,\ \mathsf{pop}^1,\ \mathsf{push}^3_a,\ \mathsf{pop}^2,\ \mathsf{push}^3_a.$$

Then the topmost 2-stack is alternatively: $[aa]$ or $[aa][aa]$ or $[aa][a]$. This run does not read any symbols, so it is a degenerate case of an infinite run which reads only stars. Configurations with topmost 2-stack $[aa]$ are milestones (and no other configurations in this run). To obtain a less degenerate case, we may consider a loop of transitions as above, but containing additionally a read transition; when a star is read, the loop continues (we do not care what happens when any other symbol is read). Then again configurations having $[aa]$ as the topmost 2-stack are milestones.

If c is a milestone, R the (unique) infinite run from c reading only stars, and I a set like in the definition of a milestone, then for each $i \in I$ the configuration $R(i)$ is a milestone as well. The following lemma shows that in fact the set I can contain all i for which $R(i)$ is a milestone.

Lemma 8.3. *Let R be a run between two milestone configurations which reads only stars. Then R is 0-upper.*

Proof. We prove by induction on $n-k$, where $k \in [0,n]$, that each run R as in the lemma is k-upper. Trivially each run is n-upper. Now suppose that the thesis holds for some $k > 0$, and take a run R between two milestone configurations which reads only stars. Let S be the infinite run staring in $R(0)$ which reads only stars (since $R(0)$ is a milestone, the run is really infinite); R is its prefix. Notice that we can find a milestone $S(i)$ such that $i \geq |R|$ and $S{\restriction_{0,i}}$ is $(k-1)$-upper (it can be even 0-upper): it is enough to take any $i \geq |R|$ from the infinite set I from Definition 8.1. From the induction assumption we know that $S{\restriction_{|R|,i}}$ is k-upper. We conclude that $S{\restriction_{0,|R|}} = R$ is $(k-1)$-upper using Proposition 6.4 for run $S{\restriction_{0,|R|}} \circ S{\restriction_{|R|,i}}$. □

Another important property is that in a very long run reading only stars we can find a milestone configuration. What "very long" means of course depends on the size of the configuration where the run starts.

Lemma 8.4. *Let $l \in [1,n]$. There exists a function $\beta\colon \Gamma_*^l \to \mathbb{N}$, assigning a number to an l-stack, having the following property. Let R be a run which reads only stars, and let y be a position of $R(|R|)$. Let s^l be the l-stack of $R(0)$ containing $\mathsf{hist}(R,y)$. Suppose that there exist at least $\beta(s^l)$ indices i such that position $\mathsf{hist}(R{\restriction_{i,|R|}}, y)$ is in the topmost l-stack of $R(i)$. Then for some i, configuration $R(i)$ is a milestone and position $\mathsf{hist}(R{\restriction_{i,|R|}}, y)$ is in the topmost l-stack of $R(i)$.*

Corollary 8.5. *Let R be an infinite run reading only stars. Then, for infinitely many i the configuration $R(i)$ is a milestone.*

Proof. To obtain a first milestone configuration, it is enough to use Lemma 8.4 for $l := n$ and for the prefix of R of length $\beta(\pi_2(R(0)))$. We repeat this procedure for the remaining suffix of R.[3] □

In the remaining part of the section we prove Lemma 8.4. Our proof strategy is as follows. The indices i for which $\mathsf{hist}(R{\restriction_{i,|R|}}, y)$ is in the topmost l-stack give us a decomposition of an infix of R into many l-upper runs. As a first step, consecutively for $k = l-1, l-2, \ldots, 0$ we construct a decomposition of an infix of R into many k-upper runs. Then, among the borders of the constructed 0-upper runs we find two configurations having the same type. Using Theorem 7.3 we can replicate the 0-upper run between them into arbitrarily many consecutive 0-upper runs, proving that these two configurations are milestones.

The division of an infix of R into a k-upper run will be described using k-advancing sets, defined as follows. Assuming that R is fixed, a set $I^k \subseteq [0,|R|]$ is called *k-advancing* if

$$\emptyset \neq I^k = \{i \in [\min I^k, \max I^k] \colon\ R{\restriction_{i,\max I^k}} \in \mathsf{up}^k\}.$$

Notice that when $\min I^k \leq i \leq j \in I^k$, we have $i \in I^k$ if and only if $R{\restriction_{i,j}}$ is k-upper. In other words, a k-advancing set not only gives us a decomposition into k-upper runs, but also

[3] There exists a direct proof of this corollary, not presented here, which is much easier than the proof of Lemma 8.4.

these k-upper runs cannot be farther subdivided into shorter k-upper runs. The following auxiliary lemma describes our induction step.

Lemma 8.6. *Let $k \in [1,n]$, and $N \in \mathbb{N}$. There exists a function $f_N^k \colon \mathbb{N} \to \mathbb{N}$, having the following properties. Let R be a run which reads only stars, and I^k a k-advancing set. Suppose that $|I^k| \geq f_N^k(r)$, where r is the size of the topmost k-stack of $R(\min I^k)$. Then there exists a $(k-1)$-advancing subset $I^{k-1} \subseteq I^k$ of size at least N, such that the topmost k-stack of $R(\min I^{k-1})$ is one of $(k-1)$-stacks in the topmost k-stack of $R(\min I^k)$.*

Proof. We prove the lemma by induction on N. For $N = 1$ we can take $f_1^k(r) := 1$, and then $I^{k-1} := \{\min I^k\}$. Let now $N \geq 2$. We take

$$f_N^k(r) := 1 + \sum_{m=1}^{r} f_{N-1}^k(m+1).$$

Fix some R and I^k satisfying the assumptions. Let $a := \min I^k$. By r_i we denote the size of the topmost k-stack of $R(i)$ (for each $i \in I^k$).

For each $j \in I^k$ denote

$$m_j := \min\{r_i \colon i \in I^k \wedge i \leq j\}.$$

Notice that $1 \leq m_j \leq r$ (because $r_a = r$) and that $m_j \geq m_{j'}$ for $j \leq j'$. From the formula for $f_N^k(r)$ we see that for some m we have at least $f_{N-1}^k(m+1)+1$ indices $j \in I^k$ such that $m_j = m$. Choose some such m; let b be the first index such that $m_b = m$, and e the last such index. We have $m = r_b$.

Let c be the next index after b which is in I^k (of course $c \leq e$). Notice that $r_c \leq r_b + 1 = m+1$; this follows from Proposition 6.2 used for run $R\restriction_{b,c}$. Thus we have

$$|I^k \cap [c,e]| \geq f_{N-1}^k(m+1) \geq f_{N-1}^k(r_c).$$

We use the induction assumption for $I^k \cap [c,e]$. We obtain a $(k-1)$-advancing subset $J^{k-1} \subseteq I^k \cap [c,e]$ of size at least $N-1$. We take

$$I^{k-1} := \{i \in [b, \min J^{k-1}] \colon R\restriction_{i,\max J^{k-1}} \in \mathsf{up}^{k-1}\} \cup J^{k-1}.$$

We easily see that I^{k-1} is $(k-1)$-advancing (we add to J^{k-1} exactly these indices for which the appropriate run is $(k-1)$-upper). Because $r_i \geq r_b$ for each $i \in I^k \cap [b,e]$, Proposition 6.3 implies that $R\restriction_{b,\max J^{k-1}}$ is $(k-1)$-upper. Thus I^{k-1} in addition to the $N-1$ elements of J^{k-1} contains at least one additional element b.

Finally, we show that the topmost k-stack of $R(b)$ is one of $(k-1)$-stacks in the topmost k-stack of $R(a)$. For each $i \in I^k \cap [a, b-1]$ we have $r_i > r_b$, so, due to Proposition 6.3, run $R\restriction_{i,b}$ is not $(k-1)$-upper. This means that the topmost $(k-1)$-stack of $R(b)$ was not modified since $R(a)$. On the other hand $R\restriction_{a,b}$ is k-upper, thus, indeed, the topmost $(k-1)$-stack of $R(b)$ is one of the $(k-1)$-stacks in the topmost k-stack of $R(a)$. □

While using Theorem 7.3 we need to ensure that the replicated run reads only stars. For this reason, let $\varphi \colon A^* \to M$ be a morphism into a finite monoid, which checks whether a word consists only of stars. Our second auxiliary lemma will be used to conclude the proof of Lemma 8.4.

Lemma 8.7. *Let S be a nonempty 0-upper run reading only stars, in which $S(|S|)$ has the same $(\mathcal{A},\varphi)$-type and the same topmost stack symbol as $S(0)$ (where φ as above). Then S can be extended into a run $S \circ T \circ U$ reading only stars, where T and U are nonempty*

0-upper runs, and $U(|U|)$ has the same $(\mathcal{A},\varphi)$-type and the same topmost stack symbol as $U(0)$. As a consequence, $S(0)$ is a milestone.

Proof. First, we observe that for each $r \in \mathbb{N}$ we can construct a composition $S_1 \circ \cdots \circ S_r$ of r nonempty 0-upper runs, reading only stars, in which $S_1 = S$. For $r = 1$ this is trivially true. Suppose that we have such composition for some r. Then Theorem 7.3 applied to this composition and to $S(|S|)$ gives us a composition $S'_1 \circ \cdots \circ S'_r$ of r nonempty 0-upper runs, reading only stars, starting from $S(|S|)$. Together with S at the beginning, they give a longer composition as required.

Take such a composition for r equal to the number of stack symbols times the number of $(\mathcal{A},\varphi)$-types, plus two. Then we can find two indices $i, j \in [2, r]$ with $i < j$ for which $S_j(|S_j|)$ has the same $(\mathcal{A},\varphi)$-type and the same topmost stack symbol as $S_i(|S_i|)$. Skipping the part after S_j, we obtain a composition $S \circ T \circ U$ as required.

We can repeat the same construction for U, and append two more nonempty 0-upper runs, out of which the second has equal $(\mathcal{A},\varphi)$-type and topmost stack symbol at its two ends. Continuing this forever, we obtain an infinite run reading only stars, divided into 0-upper runs. Since it starts in $S(0)$, this configuration is a milestone. □

Proof of Lemma 8.4. Fix some l-stack s^l. Let N_0 be equal to the number of stack symbols times the number of $(\mathcal{A},\varphi)$-types, plus one, where again φ checks whether a word consists only of stars. For $k \in [1, l]$ we take $N_k = f^k_{N_{k-1}}(r_k)$, where r_k is the maximal size of a k-stack which appears in s^l, and $f^k_{N_{k-1}}$ is the function from Lemma 8.6. We define $\beta(s^k) := N_k$.

Now take a run R and a position y in $R(|R|)$, such that the assumptions of the lemma are satisfied. First, for each $k \in [0, l]$ we want to construct a k-advancing set I^k of size at least N_k, such that the topmost k-stack of $R(\min I^k)$ is one of the k-stacks in s^l.

As I^l we take the set of those indices i for which position $\mathsf{hist}(R{\restriction}_{i,|R|}, y)$ is in the topmost l-stack of $R(i)$. It is immediate from the definitions that I^l is l-advancing (notice that the history of y always lands in the same l-stack as the history of the bottommost position from the l-stack of y). By assumption $|I^l| \geq \beta(s^l) = N_l$. Moreover, the topmost l-stack of $R(\min I^l)$ was not modified from the beginning of the run (as it was not the topmost l-stack), so this l-stack is the same as the l-stack of $R(0)$ containing $\mathsf{hist}(R, y)$, which is s^l.

Then by induction on $l - k$, we construct I^{k-1} out of I^k using Lemma Lemma 8.6. Notice that the size of the topmost k-stack of $R(\min I^k)$ is at most r_k, so we can indeed obtain I^{k-1} of size at least N_{k-1}.

Finally, we observe that in I^0 we can find two indices i, j with $i < j$ such that $R(j)$ has the same $(\mathcal{A},\varphi)$-type and the same topmost stack symbol as $R(i)$. Lemma 8.7 applied to $R{\restriction}_{i,j}$ proves that $R(i)$ is a milestone. By construction $i \in I^0 \subseteq \cdots \subseteq I^l$, so position $\mathsf{hist}(R{\restriction}_{i,|R|}, y)$ is in the topmost l-stack of $R(i)$. □

9. Pumping Lemma

In this section we present a pumping lemma, which can be used to change the number of stars read in some place of a run, without changing too much the rest of the run. For this section we fix an n-DPDA $\mathcal{A}$ with input alphabet A containing a $\star$ symbol. We also fix a morphism $\varphi\colon A^* \to M$ into a finite monoid.

We start by an intuitive explanation of the pumping lemma. In the situation which we consider, we have a milestone configuration from which we start runs that first read

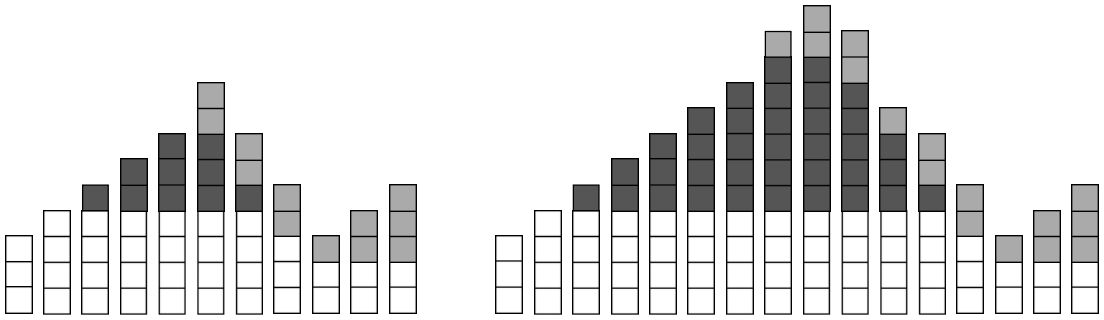

Figure 2: An example configuration at the end of a run of a 2-DPDA, and an analogous configuration after pumping. The 2-stack grows from left to right. White symbols were already present in $R(0)$; dark gray symbols were created while reading stars at the beginning of R; light gray symbols were created later.

some number of stars, and later also other symbols. One possibility is that most of these runs are k-upper (for some k), except maybe some runs reading a small number of stars at the beginning. Then we will be unable to use our pumping lemma, but we gain the knowledge that our run is k-upper. The opposite situation is that there are runs from this configuration which are not k-upper and read arbitrarily many stars at the beginning; our pumping lemma talks about this situation. Consider such a run R of a 2-DPDA, whose last configuration is depicted on the left of Figure 2. It starts in a milestone, so its initial fragment that reads only stars is basically 0-upper. This means that the automaton builds on top of the stack of $R(0)$ (depicted in white), without modifying it; also in the copies of the topmost 1-stack the original part is not modified (the automaton can inspect this part, but then it has to be removed). We consider a run which is not 0-upper, so later, when we start reading other symbols than stars, the "white part" of the topmost 1-stack is uncovered; its content is the same as in $R(0)$. By assumption there exists a run from the same configuration which reads more (arbitrarily many) stars at the beginning, and is not 0-upper. When it uncovers the "white part" of the topmost 1-stack, this part is exactly the same as in the original run, so these runs can continue in the same way. This is depicted on the right of the figure.

Next, we state our pumping lemma. For uniformity of presentation, we refer there to (-1)-upper runs, with the assumption that no run is (-1)-upper.

Theorem 9.1 (Pumping lemma)**.** *For each milestone configuration c there exists a number $\mathsf{pb}(c)$ having the following property. Let $R \circ R'$ be a run starting in c, where R is not $(k-1)$-upper and reads a word beginning with at least $\mathsf{pb}(c)$ stars, and R' is k-upper. In such situation, for each $l \in \mathbb{N}$ there exists a run $S \circ S'$ starting in c, and such that $\varphi(S) = \varphi(R)$, and S reads a word beginning with at least l stars, and S' is (k, φ)-parallel to R'.*

Let us mention that another pumping lemma for higher-order pushdown automata appears in [21]. There are several differences between these two lemmas. An advantage of the lemma in [21] is that it gives a precise value for $\mathsf{pb}(c)$, in terms of the size of c. Moreover, it works not only for deterministic PDA, but also for nondeterministic PDA in which the ε-closure of the configuration graph is finitely-branching. On the other hand, the

pumping lemma in [21] is only given for $k = 0$. Additionally, it just says that the length of the word read by the run increases, not necessarily the number of stars at its beginning. The pumping lemma from [21] was generalized to collapsible pushdown automata in [10].

In the rest of the section we present a proof of Theorem 9.1. Its essence is as described above: we consider the moment when the run stops to be $(k-1)$-upper. On possibility is as depicted in Figure 2: this happens during a pop^k operation; our topmost k-stack becomes equal to the topmost k-stack of $R(0)$ with its topmost $(k-1)$-stack removed. This can also happen during a pop^r operation for some $r \geq k$. Then we can obtain another topmost k-stack, but altogether we have only finitely many possibilities. At least one of these possibilities happens for runs reading arbitrarily large number of stars at the beginning; we can stick to this possibility. Next, when we change the number of stars read at the beginning, we still land in a configuration having the same topmost k-stack as in the original run, when the run stops to be $(k-1)$-upper; from this configuration we can mimic the rest of the original run. When $k < n-1$, the type of the rest of the stack is important as well (the latter fragment of the run can perform returns visiting interiors of our stack; the existence of such returns is described by the type). This is not a problem, since the type comes from a finite set, so we can assume that it is fixed as well.

The most difficult part of the proof is to show that indeed when the run stops to be $(k-1)$-upper, there are only finitely many possible shapes for the topmost k-stack. This is shown in Corollary 9.3. It bases on Lemma 9.2, in which we analyze the situation just after reading the stars.

Lemma 9.2. *For each milestone configuration c, and for $k \in [1, n]$, there exists a finite set $\mathcal{S}^k(c)$ of k-stacks having the following property. Let R be a run starting in c and reading only stars. Let x be the bottommost position of the topmost $(k-1)$-stack in some k-stack of $R(|R|)$. If $\mathsf{hist}(R, x) \neq \mathsf{top}^{k-1}(c)$, then the k-stack of $R(|R|)$ containing x is in $\mathcal{S}^k(c)$.*

Proof. Let $\mathcal{X}(c)$ be the set containing all k-stacks of c, and additionally the topmost k-stack of c with its topmost $(k-1)$-stack removed. Let $\mathcal{S}^k(c)$ contain all k-stack which can be obtained from a k-stack $s^k \in \mathcal{X}(c)$ by applying at most $\beta(s^k)$ of push and pop operations, where β is the function from Lemma 8.4.

Fix a run R from c which reads only stars. We say that a k-stack of some configuration $R(i)$ is *c-clear*, if $\mathsf{hist}(R{\restriction}_{0,i}, x) \neq \mathsf{top}^{k-1}(c)$ for x being the bottommost position of the topmost $(k-1)$-stack in the considered k-stack of $R(i)$. Our goal is to show that every c-clear k-stack of $R(|R|)$ is in $\mathcal{S}^k(c)$.

Let us fix some c-clear k-stack of $R(|R|)$, let y be its bottommost position. Consider the smallest index i for which this k-stack (more precisely the k-stack of $R(i)$ containing $\mathsf{hist}(R{\restriction}_{i,|R|}, y)$) is c-clear in $R(i)$. We claim that this k-stack of $R(i)$ is in $\mathcal{X}(c)$. Indeed, either $i = 0$ and it is one of the k-stacks of c, or the k-stack of $R(i-1)$ containing $\mathsf{hist}(R{\restriction}_{i-1,|R|}, y)$ is not c-clear. In the latter case, this k-stack becomes c-clear in the next configuration, so necessarily this is the topmost k-stack, and the operation between these configurations is pop^k. We see that $R{\restriction}_{0,i-1}$ is $(k-1)$-upper, and $R{\restriction}_{0,i}$ is not $(k-1)$-upper. Proposition 6.5 implies that $R{\restriction}_{0,i}$ is a k-return, and Proposition 6.6 implies that the topmost k-stack of $R(i)$ is equal to the topmost k-stack of c with its topmost $(k-1)$-stack removed; thus it belongs to $\mathcal{X}(c)$.

Let s^k be the k-stack of $R(i)$ containing $\mathsf{hist}(R{\restriction}_{i,|R|}, y)$ (where i as above). Observe that a k-stack can be changed only when it is the topmost k-stack. If there exist at most $\beta(s^k)$ indices $j \in [i, |R|]$ such that position $\mathsf{hist}(R{\restriction}_{j,|R|}, y)$ is in the topmost k-stack of $R(j)$,

then the k-stack of $R(|R|)$ containing y can be obtained from s^k by applying at most $\beta(s^k)$ of push and pop operations, so it is in $\mathcal{S}^k(c)$. Suppose to the contrary that there are more than $\beta(s^k)$ such indices j. Then we can use Lemma 8.4 for $R{\restriction_{i,|R|}}$; it gives us an index j such that configuration $R(j)$ is a milestone and position $\mathsf{hist}(R{\restriction_{j,|R|}}, y)$ is in the topmost k-stack of $R(j)$. Because both c and $R(j)$ are milestones, we know that $R{\restriction_{0,j}}$ is 0-upper, thanks to Lemma 8.3. One case is that $i = 0$; then $\mathsf{hist}(R, y) \neq \mathsf{top}^k(c)$ (since the k-stack of c containing $\mathsf{hist}(R, y)$ is c-clear), so $R{\restriction_{0,j}}$ is not k-upper, thus even more it cannot be 0-upper. Otherwise, as already observed, $R{\restriction_{0,i}}$ is not $(k-1)$-upper and $\mathsf{hist}(R{\restriction_{i,|R|}}, y) = \mathsf{top}^k(R(i))$, which implies that $R{\restriction_{i,j}}$ is k-upper. But $R{\restriction_{0,j}}$ is $(k-1)$-upper, so this contradicts with Proposition 6.4 applied for $R{\restriction_{0,i}} \circ R{\restriction_{i,j}}$. □

Corollary 9.3. *For each milestone configuration c there exists a finite set $\mathcal{S}(c)$ of configurations having the following property. Let $k \in [0, n]$, let R be a run starting in c, and let $r \in [0, |R|]$ be such that $R{\restriction_{0,r}}$ reads only stars. Suppose that R is not $(k-1)$-upper, but for each $i \in [r, |R| - 1]$ either $R{\restriction_{0,i}}$ is $(k-1)$-upper or $R{\restriction_{i,|R|}}$ is not k-upper. Then we can find a configuration $d \in \mathcal{S}(c)$ having the same $(\mathcal{A}, \varphi)$-type and the same topmost k-stack as $R(|R|)$.*

Proof. There are only finitely many possible values of a $(\mathcal{A}, \varphi)$-type of a configuration. Thus it is enough to show, for each k, that there are only finitely many possible topmost k-stacks over all configurations $R(|R|)$ satisfying the assumptions. For $k = 0$ this is trivial as 0-stack contains just one symbol. Suppose that $k \geq 1$. We have two cases.

First suppose that $R{\restriction_{i,|R|}}$ is k-upper for some $i \in [r, |R| - 1]$; fix the greatest such index i. Then by assumption $R{\restriction_{0,i}}$ is $(k-1)$-upper, but R is not. This is possible only when $i = |R| - 1$ (thanks to Proposition 6.2 used for $R{\restriction_{i,|R|}}$). Proposition 6.5 says that R is necessarily a k-return. Thus the topmost k-stack of $R(|R|)$ is equal to the topmost k-stack or $R(0)$ with its topmost $(k-1)$-stack removed (Proposition 6.6). Thus the content of this k-stack is fixed.

The other case is that $R{\restriction_{i,|R|}}$ is not k-upper for every $i \in [r, |R| - 1]$. This means that the topmost k-stack of $R(|R|)$ is an unchanged copy of some k-stack of $R(r)$. As R is not $(k-1)$-upper, this k-stack of $R(r)$ is c-clear, and by Lemma 9.2 it is in the finite set $\mathcal{S}^k(c)$. □

Proof of Theorem 9.1. Consider the infinite run P starting at the milestone configuration c and reading only stars. Consider first the degenerate case when in P only finitely many stars are read. As $\mathsf{pb}(c)$ we take their number, plus one. Then the thesis is satisfied trivially, as there is no run starting in c which reads a word beginning with $\mathsf{pb}(c)$ stars. So for the rest of the proof suppose that P reads infinitely many stars.

Let $\mathcal{S}(c)$ be the set from Corollary 9.3 (used for c). For each $i \geq 1$ we define the set $T_i \subseteq [0, n] \times \mathcal{S}(c) \times M$ as follows. A triple (j, d, m) belongs to T_i if and only if there exists a run R from c such that the word read by R begins with (at least) i stars, and $\varphi(R) = m$, and $R(|R|)$ has the same $(\mathcal{A}, \varphi)$-type and the same topmost j-stack as d. By definition $T_{i+1} \subseteq T_i$ (for each i), and there are only finitely many possible sets, so from some moment every T_i is the same. As $\mathsf{pb}(c)$ we take a positive number such $T_i = T_{\mathsf{pb}(c)}$ for each $i \geq \mathsf{pb}(c)$.

Consider now a run $R \circ R'$ starting in c, where R is not $(k-1)$-upper and reads a word beginning with at least $\mathsf{pb}(c)$ stars, and R' is k-upper, for some $k \in [0, n]$. Consider also a number l. Our goal is to construct a run $S \circ S'$ starting in c and such that $\varphi(S) = \varphi(R)$, and S reads a word beginning with at least l stars, and S' is (k, φ)-parallel to R'. W.l.o.g. we can assume that $l \geq \mathsf{pb}(c)$. Let r be an index such that $R{\restriction_{0,r}}$ reads exactly $\mathsf{pb}(c)$ stars.

W.l.o.g. we can assume that there is no $i \in [r, |R| - 1]$ such that $R{\restriction}_{0,i}$ is not $(k-1)$-upper and $R{\restriction}_{i,|R|}$ is k-upper (if such i exists, we move the subrun $R{\restriction}_{i,|R|}$ to R', that is we use the pumping lemma for $R{\restriction}_{0,i} \circ (R{\restriction}_{i,|R|} \circ R')$, and then in the resulting S' we find the subrun (k, φ)-parallel to $R{\restriction}_{i,|R|}$ and we move it back to S).

We use Corollary 9.3 for R and r; its assumptions are satisfied thanks to our w.l.o.g. assumption. We obtain some $d \in \mathcal{S}(c)$ which has the same $(\mathcal{A}, \varphi)$-type and the same topmost k-stack as $R(|R|)$. It means that $(k, d, \varphi(R)) \in T_{\mathsf{pb}(c)}$. Because $T_{\mathsf{pb}(c)} = T_l$, there exists a run S from c such that the word read by S begins with (at least) l stars, and $\varphi(S) = \varphi(R)$, and $S(|S|)$ has the same $(\mathcal{A}, \varphi)$-type and the same topmost k-stack as $R(|R|)$.

Finally, we use Theorem 7.3 for R' in order to obtain an accepting run S' that starts in $S(|S|)$ and is (k, φ)-parallel to R'. □

10. Why U Cannot Be Recognized?

In this section we prove that language U cannot be recognized by a deterministic higher-order pushdown automaton of any order. Notice that our techniques presented in previous sections were quite general (not too much related to the U language). We believe that they can be useful for other purposes, to analyze behavior of some automata (in particular automata whose main objective is to count and compare the number of times a symbol appears on the input).

Of course our proof goes by contradiction: suppose that for some n we have an $(n-1)$-DPDA recognizing U. We construct an n-DPDA $\mathcal{A}$ which works as follows. First it performs a push^n operation. Then it simulates the $(n-1)$-DPDA (not using the push^n and pop^n operations). When the $(n-1)$-DPDA is going to accept, $\mathcal{A}$ performs a pop^n operation and afterwards accepts. Clearly, $\mathcal{A}$ recognizes U as well. Such normalization allows us to use Theorem 7.4, as in $\mathcal{A}$ every accepting run is an n-return.

Fix a morphism $\lambda\colon A^* \to M$ into a finite monoid M, which checks whether a word is of the form $\sharp^*$ (some number of $\sharp$ symbols), or of the form $\star^*]\star^*$ (a closing bracket surrounded by some number of stars), or of neither of these two forms. This means that for words u, v being of different form we have $\lambda(u) \neq \lambda(v)$. Let N be the number of equivalence classes of the $(\mathcal{A}, \lambda)$-sequence equivalence relation, times the number of $(\mathcal{A}, \lambda)$-types, plus one. Consider the following words:

$$w_0 = [$$
$$w_{k+1} = w_k^N]^N[\qquad \text{for } k \in [0, n-1],$$

where the number in the superscript (in this case N) denotes the number of repetitions of a word. For a word w, its *pattern* is a word obtained from w by removing its letters other than brackets (leaving only brackets). Fix a morphism $\varphi\colon A^* \to M$ such that from its value $\varphi(w)$ we can deduce

- whether the word w contains the $\sharp$ symbol, and
- whether the pattern of w is longer than $|w_n|$, and
- the exact value of the pattern of w, whenever this pattern is not longer than $|w_n|$.

We fix a run R, and an index $z(w)$ for each prefix w of w_n, such that the following holds. Run R begins in the initial configuration. Between $R(0)$ and $R(z(\varepsilon))$ only stars are read. For each prefix w of w_n, the configuration $R(z(w))$ is a milestone. Just after $z(w)$, the run R reads $\mathsf{pb}(R(z(w)))$ stars, where pb is the function from Theorem 9.1 used for

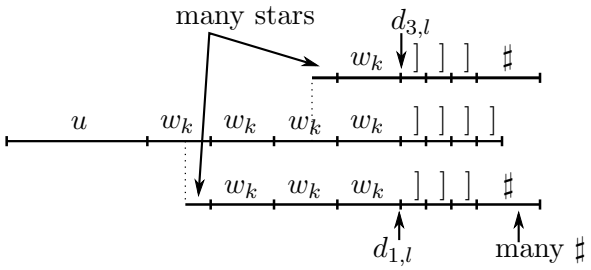

Figure 3: Illustration of runs appearing in the proof (where $N = 4$, $x = 1$, $y = 3$). Recall that stars can appear between letters of the words.

morphism φ. If $w = va$ (where a is a single letter), the word read by R between $R(z(v))$ and $R(z(w))$ consists of a surrounded by some number of stars. Of course such run R exists: we read stars until we reach a milestone (succeeds thanks to Corollary 8.5), then we read as many stars as required by the pumping lemma, then we read the next letter of w_n, and so on (because $\mathcal{A}$ accepts U, it will never block).

It will be important to analyze relations between configurations $R(z(v))$ for some prefixes v of w_n. In order to avoid complicated subscripts, for any prefixes v, w of w_n we denote $\langle v, w\rangle := R{\restriction}_{z(v),z(w)}$.

By construction of $\mathcal{A}$, for every prefix v of w_n the run $\langle v, w_n\rangle$ is $(n-1)$-upper (as we never perform a pop^n operation before reading some ♯ symbol). This contradicts with the following key lemma (taken for $k = n-1$ and $u = \varepsilon$).

Lemma 10.1. *Let $k \in [-1, n-1]$, and let u be a word such that uw_{k+1} is a prefix of w_n. Then there exist a prefix v of w_{k+1} such that $\langle uv, uw_{k+1}\rangle$ is not k-upper.*

Proof. The proof is by induction on k. For $k = -1$ this is obvious, as no run is (-1)-upper.

Let now $k \geq 0$. Figure 3 might be helpful in finding different runs present in the proof below. Suppose that the thesis of the lemma does not hold. Then for each prefix v of w_{k+1} the run $\langle uv, uw_{k+1}\rangle$ is k-upper. From this we get the following property ♡.

Let v' be a prefix of w_{k+1}, and v a prefix of v'. Then $\langle uv, uv'\rangle$ is k-upper.

By the induction assumption (where uw_k^{i-1} is taken as u), for each $i \in [1, N]$ there exist a prefix v_i of w_k such that $\langle uw_k^{i-1}v_i, uw_k^i\rangle$ is not $(k-1)$-upper. As $\langle uw_k^i, uw_k^N\rangle$ is k-upper (property ♡), from Proposition 6.4 we know that $\langle uw_k^{i-1}v_i, uw_k^N\rangle$ cannot be $(k-1)$-upper as well.

Now we are ready to use the pumping lemma (Theorem 9.1). For each $i \in [1, N]$ we use it for $\langle uw_k^{i-1}v_i, uw_k^N\rangle \circ \langle uw_k^N, uw_{k+1}\rangle$. Recall from the definition of R that the word read by $\langle uw_k^{i-1}v_i, uw_k^N\rangle$ begins with such a number of stars that the pumping lemma can be used. For each number l we obtain a run $S_{i,l} \circ S'_{i,l}$, such that $\varphi(S_{i,l}) = \varphi(\langle uw_k^{i-1}v_i, uw_k^N\rangle)$, and $S_{i,l}$ reads a word beginning with at least l stars, and $S'_{i,l}$ is (k, φ)-parallel to $\langle uw_k^N, uw_{k+1}\rangle$; let $d_{i,l} = S_{i,l}(|S_{i,l}|)$. Notice that the run $R{\restriction}_{0,z(uw_k^{i-1}v_i)} \circ S_{i,l}$, starts in the initial configuration, ends in $d_{i,l}$, and reads a word having pattern uw_k^N.

Because there are finitely many possible $(\mathcal{A}, \lambda)$-types, we can assume that $\mathsf{type}_{\mathcal{A},\lambda}(d_{i,l}) = \mathsf{type}_{\mathcal{A},\lambda}(d_{i,j})$ for each $i \in [1, N]$ and each l and j. Indeed, we can choose (for each i separately) some value of $\mathsf{type}_{\mathcal{A},\lambda}(d_{i,l})$ which appears infinitely often, and then we take the subsequence of only these $d_{i,l}$ which give this value.

Since there are more possible indices $i \in [1, N]$ than the number of classes of the $(\mathcal{A}, \lambda)$-sequence equivalence relation, times the number of $(\mathcal{A}, \lambda)$-types, there have to exist two indices x, y with $1 \leq x < y \leq N$ such that $\mathsf{type}_{\mathcal{A},\lambda}(d_{x,1}) = \mathsf{type}_{\mathcal{A},\lambda}(d_{y,1})$, and the sequences $d_{x,1}, d_{x,2}, \dots$ and $d_{y,1}, d_{y,2}, \dots$ are $(\mathcal{A}, \lambda)$-sequence equivalent. From now we fix these two indices x, y. Furthermore, because $S'_{i,l}$ is (k, φ)-parallel to $\langle uw_k^N, uw_{k+1}\rangle$ for each $i \in [1, N]$ and each l, we know that the topmost k-stacks of all $d_{x,l}$ and of all $d_{y,l}$ are equal.

Let R' be a prefix of $S'_{x,1}$ which is (k, φ)-parallel to $\langle uw_k^N, uw_k^N]^{N-x}\rangle$. Notice that R' consists of $N - x$ runs, each of which is k-upper and reads a word of the form $\star^*]\star^*$ (a closing bracket surrounded by some number of stars). Let also R'' be an n-return starting in $R'(|R'|)$ reading only $\sharp$ symbols (because $\mathcal{A}$ recognizes U, there is an accepting run R'' from $R'(|R'|)$ reading only $\sharp$ symbols; by construction of $\mathcal{A}$, it is an n-return).

Finally we use Theorem 7.4 for λ (as φ), sequences $d_{x,1}, d_{x,2}, \dots$ (as $c_1, c_2, \dots$) and $d_{y,1}, d_{y,2}, \dots$ (as $d_1, d_2, \dots$), and for run $R' \circ R''$.[4] As noticed above (in particular because $R'(0) = d_{x,1}$), the configurations $R'(0)$ and $d_{x,l}$ and $d_{y,l}$ for each l all have the same $(\mathcal{A}, \lambda)$-types and topmost k-stacks. Thus the assumptions of the theorem are satisfied. For each l, we obtain runs S_l (from $d_{x,l}$) and T_l (from $d_{y,l}$). The word read by any of these runs contains $N - x$ closing brackets with some number of stars around them, and after them some number of $\sharp$ symbols.

The runs $R{\restriction}_{0,z(uw_k^{x-1}v_x)} \circ S_{x,l} \circ S_l$ and $R{\restriction}_{0,z(uw_k^{y-1}v_y)} \circ S_{y,l} \circ T_l$ for each l have pattern $uw_k^N]^{N-x}$. In this pattern the last opening bracket which is not closed is the last bracket of the x-th w_k after u. Recall that configurations $d_{x,l}$ were obtained by pumping inside the x-th w_k, so before this bracket; for $l \to \infty$ the number of stars inserted there is unbounded. From the definition of the language U it follows that the sequence $\sharp(S_1), \sharp(S_2), \dots$ has to be unbounded. On the other hand, configurations $d_{y,l}$ were obtained by pumping inside the y-th w_k, so after the last opening bracket which was not closed (as $y > x$). For each l the number of stars before this bracket is the same. From the definition of the language U it follows that the sequence $\sharp(T_1), \sharp(T_2), \dots$ has to be constant, hence bounded. This contradicts with the thesis of Theorem 7.4, which says that either both these sequences are bounded or both unbounded. □

Acknowledgment

We thank A. Kartzow and the anonymous reviewers of the conference version for their constructive comments.

References

[1] K. Aehlig, J. G. de Miranda, and C.-H. L. Ong, "Safety is not a restriction at level 2 for string languages," in *FoSSaCS*, ser. Lecture Notes in Computer Science, V. Sassone, Ed., vol. 3441. Springer, 2005, pp. 490–504.

[2] C. H. Broadbent, A. Carayol, M. Hague, and O. Serre, "A saturation method for collapsible pushdown systems," in *ICALP (2)*, ser. Lecture Notes in Computer Science, A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, Eds., vol. 7392. Springer, 2012, pp. 165–176.

[3] D. Caucal, "On infinite terms having a decidable monadic theory," in *MFCS*, ser. Lecture Notes in Computer Science, K. Diks and W. Rytter, Eds., vol. 2420. Springer, 2002, pp. 165–176.

[4] Notice that we cannot use $\langle uw_k^N, uw_k^N]^{N-x}\rangle$ instead of R', because do not know anything about the $(\mathcal{A}, \lambda)$-type of $R(z(uw_k^N)))$.

[4] J. Engelfriet, "Iterated stack automata and complexity classes," *Inf. Comput.*, vol. 95, no. 1, pp. 21–75, 1991.
[5] R. H. Gilman, "A shrinking lemma for indexed languages," *Theor. Comput. Sci.*, vol. 163, no. 1&2, pp. 277–281, 1996.
[6] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre, "Collapsible pushdown automata and recursion schemes," in *LICS*. IEEE Computer Society, 2008, pp. 452–461.
[7] T. Hayashi, "On derivation trees of indexed grammars," *Publ. RIMS, Kyoto Univ.*, vol. 9, pp. 61–92, 1973.
[8] A. Kartzow, "Collapsible pushdown graphs of level 2 are tree-automatic," in *STACS*, ser. LIPIcs, J.-Y. Marion and T. Schwentick, Eds., vol. 5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 501–512.
[9] ——, "A pumping lemma for collapsible pushdown graphs of level 2," in *CSL*, ser. LIPIcs, M. Bezem, Ed., vol. 12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 322–336.
[10] A. Kartzow and P. Parys, "Strictness of the collapsible pushdown hierarchy," in *MFCS*, ser. Lecture Notes in Computer Science, B. Rovan, V. Sassone, and P. Widmayer, Eds., vol. 7464. Springer, 2012, pp. 566–577.
[11] T. Knapik, D. Niwinski, and P. Urzyczyn, "Higher-order pushdown trees are easy," in *FoSSaCS*, ser. Lecture Notes in Computer Science, M. Nielsen and U. Engberg, Eds., vol. 2303. Springer, 2002, pp. 205–222.
[12] T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz, "Unsafe grammars and panic automata," in *ICALP*, ser. Lecture Notes in Computer Science, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580. Springer, 2005, pp. 1450–1461.
[13] N. Kobayashi, "Model-checking higher-order functions," in *PPDP*, A. Porto and F. J. López-Fraguas, Eds. ACM, 2009, pp. 25–36.
[14] N. Kobayashi and C.-H. L. Ong, "Complexity of model checking recursion schemes for fragments of the modal mu-calculus," in *ICALP (2)*, ser. Lecture Notes in Computer Science, S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, Eds., vol. 5556. Springer, 2009, pp. 223–234.
[15] A. N. Maslov, "The hierarchy of indexed languages of an arbitrary level," *Soviet Math. Dokl.*, vol. 15, pp. 1170–1174, 1974.
[16] ——, "Multilevel stack automata," *Problems of Information Transmission*, vol. 12, pp. 38–43, 1976.
[17] C.-H. L. Ong, "On model-checking trees generated by higher-order recursion schemes," in *LICS*. IEEE Computer Society, 2006, pp. 81–90.
[18] P. Parys, "Collapse operation increases expressive power of deterministic higher order pushdown automata," in *STACS*, ser. LIPIcs, T. Schwentick and C. Dürr, Eds., vol. 9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 603–614.
[19] ——, "Higher-order pushdown systems with data," in *GandALF*, ser. EPTCS, M. Faella and A. Murano, Eds., vol. 96, 2012, pp. 210–223.
[20] ——, "On the significance of the collapse operation," in *LICS*. IEEE, 2012, pp. 521–530.
[21] ——, "A pumping lemma for pushdown graphs of any level," in *STACS*, ser. LIPIcs, C. Dürr and T. Wilke, Eds., vol. 14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 54–65.