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 that 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) that 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 [Mas74, Mas76] 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, Niwiński, and Urzyczyn [KNU02] showed something similar but different: 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 [Cau02] 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, Murawski, Ong, and Serre [HMOS08] extended the model of n-PDA to order-n collapsible pushdown automata (n-CPDA) by introducing a new stack operation called collapse, and proved that the class of trees generated by n-CPDA coincides with the class of trees generated by order-n recursion schemes (earlier, Knapik, Niwiński, Urzyczyn, and Walukiewicz [KNUW05] introduced panic automata, a model equivalent to 2-CPDA).

Let us mention that these trees have decidable MSO theory [Ong06], and that higher-order recursion schemes have close connections with verification of some real life higher-order programs [Kob09].

Nevertheless, it was still an open question whether these two hierarchies of trees are possibly the same hierarchy? This problem was stated in Knapik et al. [KNU02] and repeated in other papers concerning higher-order pushdown automata [KNUW05, AdMO05, Ong06, HMOS08]. A partial answer to this question was given in our previous paper [Par11]: there is a tree generated by a 2-CPDA that 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) that 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 Knapik et al. [KNU02] and from that time was conjectured to be a good example.

In this paper we work with PDA that 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 (it is shown in Section 3 how these theorems are related).

Theorem 1.2. *There is a language recognized by a deterministic 2-CPDA that 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. The lemma is similar to the pumping lemma from our another paper [Par12c]; see Section 9 for some comments. Earlier, several pumping lemmas related to the second order of the pushdown hierarchy were proposed [Hay73, Gil96, Kar11].

This paper is an extended version of our conference paper [Par12b]. 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 now proven in a weaker form than in the conference paper).

1.1. **Related Work.** One may ask a similar question for word languages instead of trees: is there a language recognized by a CPDA that 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 [AdMO05]. It is also an open question whether all word languages recognized by CPDA are context-sensitive.

We have shown [Par12a] 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 a 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 [HMOS08] 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 theories.

Engelfriet [Eng91] 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 that is not recognized by any $(n-1)$-PDA, and similarly for trees). As observed by Haußner and Kartzow [HK13], 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 Kobayashi and Ong [KO09]).

2. Preliminaries

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

In the whole paper, the letter n is used exclusively for the order of pushdown automata, which is usually assumed to be fixed and known implicitly.

We now define *stacks of order k* (*k-stacks* for short). Traditionally, 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. However, having a k-stack that is a part of an r-stack for $k < r$, it is convenient to know where this k-stack is located in the r-stack. For this reason, we equip every element of a stack by its position, written as a vector of natural numbers. Thus, for a fixed alphabet Γ (of *stack symbols*), a stack of order 0 is a pair (γ, x), where $\gamma \in \Gamma$ and $x = (x_n, x_{n-1}, \ldots, x_1)$ is a vector of n positive integers, called a *position*. Then, for $k \in [1, n]$ we define k-stacks by induction: a k-stack is a list $[s_1, s_2, \ldots, s_m]$ of nonempty $(k-1)$-stacks (where, by convection, all 0-stacks are nonempty) for which there exist numbers $x_n, x_{n-1}, \ldots, x_{k+1}$ such that, for $i \in [1, m]$, all positions in s_i are of the form $(x_n, x_{n-1}, \ldots, x_{k+1}, i, y_{k-1}, y_{k-2}, \ldots, y_1)$. By Γ_*^k and Γ_+^k we denote the the set of order-k stacks, and the set of nonempty order-k stack, respectively, where $k \in [0, n]$. The top of a stack is on the right.

For example, when we have a 3-stack s, and $n = 5$, then the second 0-stack of the third 1-stack (counting from the bottom) of the bottommost 2-stack of s is of the form $(\gamma, (x_5, x_4, 1, 3, 2))$, where x_5 and x_4 say where s is located in an imaginary 5-stack; the numbers x_5 and x_4 should be the same in the whole s.

For a k-stack s^k, where $k \in [0, n-1]$, let $\mathsf{p}_{+1}(s^k)$ be the k-stack obtained from s^k by increasing the $(n-k)$-th coordinate of all its positions by 1. For example $\mathsf{p}_{+1}((\gamma, (2, 3))) = (\gamma, (2, 4))$, and $\mathsf{p}_{+1}([(\gamma, (2, 1)), (\gamma, (2, 2))]) = [(\gamma, (3, 1)), (\gamma, (3, 2))]$.

Let us emphasize that when for two k-stacks s^k, t^k we write $s^k = t^k$, we mean that not only their contents are equal, but also positions contained in their 0-stacks are equal; thus, when s^k and t^k come from the same n-stack, this actually means that s^k and t^k refer to the same k-stack.

While comparing two stacks, we sometimes need to ignore positions contained in their 0-stacks, and compare only their contents. For a k-stack s^k, let *positionless stack* $\mathsf{pos}{\downarrow}(s^k)$ be the list of lists of ... of lists of stack symbols obtained from s^k by removing positions from all 0-stacks. We say that two k-stacks s^k, t^k are *positionless-equal*, denoted $s^k \cong t^k$, when $\mathsf{pos}{\downarrow}(s^k) = \mathsf{pos}{\downarrow}(t^k)$. When s_-^n is a positionless n-stack, there is a unique n-stack s^n such that $s_-^n = \mathsf{pos}{\downarrow}(s^n)$; we write $\mathsf{pos}_+(s_-^n)$ for s^n.

The *size* of a k-stack s^k, denoted $|s^k|$, is the number of $(k-1)$-stacks it contains. When $s^k = [s_1, s_2, \ldots, s_m] \in \Gamma_*^k$, and $s^{k-1} \in \Gamma_+^{k-1}$, and $[s_1, s_2, \ldots, s_m, s^{k-1}]$ is a valid k-stack, we denote this k-stack by $s^k : s^{k-1}$. The operator "$:$" is assumed to be right associative (i.e., e.g., $s^2 : s^1 : s^0 = s^2 : (s^1 : s^0)$). When $0 \leq k \leq r$, and $s^r = t^r : t^{r-1} : \cdots : t^k \in \Gamma_+^r$, by $\mathsf{top}^k(s^r)$ we denote the topmost k-stack of s^r, that is, t^k. We use the name *positionless topmost k-stack* for $\mathsf{pos}{\downarrow}(\mathsf{top}^k(\cdot))$.

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

- $\mathsf{pop}^k(s^r : s^{r-1} : \cdots : s^k : s^{k-1}) = s^r : s^{r-1} : \cdots : s^k$, that is, we remove the topmost $(k-1)$-stack; it is defined only when the topmost k-stack contains at least two $(k-1)$-stacks;
- $\mathsf{push}_\gamma^k(s^r : s^{r-1} : \cdots : s^0) = s^r : s^{r-1} : \cdots : s^{k+1} : (s^k : s^{k-1} : \cdots : s^0) : \mathsf{p}_{+1}(s^{k-1} : s^{k-2} : \cdots : s^1 : (\gamma, x))$ for $s^0 = (\gamma', x)$, that is, we duplicate the topmost $(k-1)$-stack, and then we replace the topmost stack symbol by γ, adjusting appropriately all positions.[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 that maps every element of $Q \times \Gamma$ into one of the following objects:

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

A *configuration* of $\mathcal{A}$ consists of a state and of a nonempty n-stack, that is, it 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, enclosing the initial stack symbol γ_I. We use the notation $\pi_i((p_1, \ldots, p_k)) = p_i$; 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 $\{\pi_i(p) \colon p \in X\}$. In order to shorten the notation, for a configuration c we sometimes write $\mathsf{top}^k(c)$ or $\mathsf{pop}^k(c)$ for $\mathsf{top}^k(\pi_2(c))$ or $\mathsf{pop}^k(\pi_2(c))$, respectively.

We use a shorthand $\delta(c)$ for a configuration c to denote $\delta(\pi_1(c), \mathsf{pos}{\downarrow}(\mathsf{top}^0(c)))$. A configuration d is a *successor* of a configuration c, if

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

Notice that a configuration c has

- $|A|$ successors, if the transition is $\mathsf{read}(\vec{q})$;
- no successors, if the operation is pop^k but there is only one $(k-1)$-stack on the topmost k-stack;
- one successor, otherwise.

Next, we define a *run* of $\mathcal{A}$. For $0 \leq i \leq m$, let c_i be a configuration. A run R from c_0 to c_m is a sequence $c_0, c_1, \ldots, c_m$ 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 $c_i, c_{i+1}, \ldots, c_j$. For runs R, S with $R(|R|) = S(0)$, we write $R \circ S$ for the *composition* of R and S that is defined as expected. Sometimes we also consider infinite runs, such that the sequence $c_0, c_1, c_2, \ldots$ is infinite. However, unless stated explicitly, a run is finite.

---

[1] In the classical definition the topmost symbol can be changed only when $k = 1$ (for $k \geq 2$ it required that $\gamma = \gamma'$). We make this (unimportant) extension to have a uniform definition of push^k for all k.

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 $\vec{q}$ 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 three parts: a symbol from Γ, a position, 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 are left, where k is the number stored in $\mathsf{top}^0(s)$ (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 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 $\mathsf{branch}(a, q_1, q_2, \ldots, q_{rank(a)})$ transitions, for $a \in A$, and for pairwise distinct states $q_1, q_2, \ldots, q_{rank(a)} \in Q$. If the transition from c is $\delta(c) = \mathsf{branch}(a, q_1, q_2, \ldots, q_{rank(a)})$, 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 $(1 \le i \le rank(a))$, if S is the composition of R and a run S' that uses a branch transition only in its first transition, and for which $\pi_1(S'(1)) = q_i$. Notice that the graph obtained this way is really an A-labelled ranked tree.

We now see how Theorem 1.1 follows from Theorem 1.2. Let $L \subseteq A^*$ be the language recognized by a 2-DCPDA $\mathcal{A}$ that 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 $\perp$ on the bottom of each 1-stack, and on the bottom of the 2-stack (a 1-stack containing only the $\perp$ marker). Thus, $\mathcal{B}$ starts with the $\perp$ marker as the initial stack symbol, performs $\mathsf{push}^2_\perp$ and $\mathsf{push}^1_{\gamma_I}$, placing the original initial stack symbol γ_I. Then, whenever $\mathcal{A}$ blocks because it wants to empty a stack, in $\mathcal{B}$ the bottom-of-stack marker is uncovered; in such a situation $\mathcal{B}$ starts some loop with no accepting state. There is also a technical detail, that a pop operation that 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) the 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 a fresh auxiliary state $\overline{q}$, 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 $\mathcal{B}$ that 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 [HMOS08], 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 [KNU02]; this implies the "equivalently" parts of Theorem 1.1.

4. The Separating Language

In this section we define a language U that can be recognized by a 2-DCPDA, 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 $stars(w)$. Whenever in some prefix of w there are more closing brackets than opening brackets, $stars(w) = 0$. Also when in the whole w we have the

| [| n_1 | [| n_2 | [| $\cdots$ | [| n_N | [| m_{N+1} |] | m_N |] | $\cdots$ |] | m_0 | [|
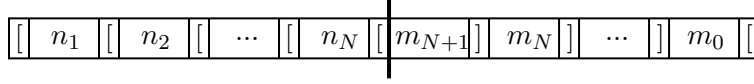
Figure 1: The stack of a 1-DPDA after reading the word w_1

same number of opening and closing brackets, $stars(w) = 0$. Otherwise, let $stars(w)$ be the number of stars in w before the last opening bracket that is not closed. Let U be the set of words $w\sharp^{stars(w)+1}$, for any $w \in \{[,],\star\}^*$ (i.e., these are words w consisting of brackets and stars, followed by $stars(w) + 1$ sharp symbols).

It is known that languages similar to U can be recognized by a 2-DCPDA (cf., e.g., Aehlig, de Miranda, and Ong [AdMO05]), but for completeness we briefly show it below. The 2-DCPDA uses 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., it performs push_Z^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_γ^2 (where γ is the topmost stack symbol). 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 that 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 $stars(w)$. Now we should read as many $\sharp$ symbols as we have 1-stacks, plus one (after each $\sharp$ symbol we perform 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 do 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

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

(where each bracket is matched, except the last opening bracket). Notice that $stars(w_1)$ equals the sum of all n_i and m_i, so $\mathcal{A}$, after reading w_1, has to store all these numbers in 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 that ends just after the i-th closing bracket. Since $\mathcal{A}$ is deterministic, the stack at the end of $w_{1,i}$ looks similar: 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

$stars(w_{1,i}) = n_1 + \cdots + n_{N-i}$. 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 of order 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, in order to recognize w_1, after each $]$ we can copy the topmost 1-stack, and remove a fragment of its copy, so that the matching opening bracket is on the top). But now we can consider the word

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

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, $stars(w_2)$ equals 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 a 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: 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. We can even 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 are described by types of stacks and by types of sequences of configurations, defined in Section 7. The key point is that there are 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 would be wrong when, while reading the $\sharp$ symbols, the automaton did not visit a place where there is stored a number that is a part of $stars(\cdot)$? 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 $stars(\cdot)$) is changed, the DPDA has to enter the part of the stack changed by the pumping lemma; otherwise it would incorrectly accept after the same number of the $\sharp$ symbols for two words with different $stars(\cdot)$.

6. The History Function and Special Runs

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

For any run R and any k-stack s^k of $R(|R|)$, where $k \in [0, n]$, we define a k-stack $\mathsf{hist}(R, s^k)$. Intuitively, $\mathsf{hist}(R, s^k)$ is the (unique) k-stack of $R(0)$, which evolved to the k-stack s^k in $R(|R|)$. Formally, we define $\mathsf{hist}(R, s^k)$ by induction on the length of R, starting with the case of $k = 0$. When $|R| = 0$, we take $\mathsf{hist}(R, s^0) = s^0$. Consider now a longer run $R = S \circ T$ with $|T| = 1$. We take $\mathsf{hist}(R, s^0) = \mathsf{hist}(S, s^0)$ if the last transition of R is read or performs pop, as well as if the transition performs push_γ^r and s^0 is not in the topmost $(r-1)$-stack of $R(|R|)$. If the last transition of R performs push_γ^r and s^0 is in the topmost $(r-1)$-stack of $R(|R|)$, then $\mathsf{hist}(R, s^0) = \mathsf{hist}(S, t^0)$, where t^0 is equal to s^0 with the $(n - r + 1)$-th coordinate of its position decreased by 1 (i.e., t^0 is the 0-stack of $T(0)$ from which s^0 was obtained as a copy). Notice that (for technical convenience) hist works in this way also for the topmost 0-stack, although the content of the topmost 0-stack changes during the push_γ^r operation. For $k > 0$, we define $\mathsf{hist}(R, s^k)$ to be the k-stack of $R(0)$ containing $\mathsf{hist}(R, s^0)$ for all 0-stacks s^0 in s^k (observe that when s^0, t^0 are two 0-stacks in s^k, the 0-stacks $\mathsf{hist}(R, s^0)$ and $\mathsf{hist}(R, t^0)$ are in the same k-stack).

It is important to notice that whenever $R = S \circ T$, then $\mathsf{hist}(S, \mathsf{hist}(T, s^k)) = \mathsf{hist}(R, s^k)$. In the sequel we extensively use this property, which we call *compositionality of histories*.

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 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 $R \in \mathsf{up}^k \iff R \circ S \in \mathsf{up}^k$ (the last property is by compositionality of histories).

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

- $\mathsf{hist}(R, \mathsf{top}^{k-1}(R(|R|))) = \mathsf{top}^{k-1}(\mathsf{pop}^k(R(0)))$, 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 a k-return when the topmost k-stack of $R(|R|)$ is obtained from the topmost k-stack of $R(0)$ by removing its topmost $(k-1)$-stack (but not only in the sense of contents, but we require that really it was obtained this way).

Example 6.1. Consider a 2-DPDA, and its run R of length 6 in which $\mathsf{pos}\!\downarrow(\pi_2(R(0))) = [[a, b], [c, d]]$, and in which the operations between consecutive configurations are

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

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

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

j	$\mathsf{pos}\!\downarrow\!(\pi_2(R(j)))$	$i\colon R{\upharpoonright}_{i,j} \in \mathsf{up}^0$	$i\colon R{\upharpoonright}_{i,j} \in \mathsf{up}^1$	$i\colon R{\upharpoonright}_{i,j} \in \mathsf{ret}^1$	$i\colon R{\upharpoonright}_{i,j} \in \mathsf{ret}^2$
0	$[[a,b],[c,d]]$	0	0	$-$	$-$
1	$[[a,b],[c,d],[c,e]]$	$0,1$	$0,1$	$-$	$-$
2	$[[a,b],[c,d],[c]]$	2	$0,1,2$	$0,1$	$-$
3	$[[a,b],[c,d]]$	$0,3$	$0,3$	$-$	$1,2$
4	$[[a,b],[c]]$	4	$0,3,4$	$0,3$	$-$
5	$[[a,b],[c,d]]$	$4,5$	$0,3,4,5$	$-$	$-$
6	$[[a,b],[c]]$	$4,6$	$0,3,4,5,6$	5	$-$

runs and k-returns are presented in Table 1. Notice that R is not a 1-return. We have $\mathsf{hist}(R{\upharpoonright}_{0,5}, (d, (2,2))) = (c, (2,1))$.

6.1. **Basic Properties of Runs.** We now 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 $R{\upharpoonright}_{i,|R|} \notin \mathsf{up}^k$ for each $i \in [1, |R|-1]$. Then either*
- $\mathsf{top}^k(R(0)) \cong \mathsf{top}^k(R(|R|))$*; additionally for every 0-stack s^0 in $\mathsf{top}^k(R(|R|))$, $\mathsf{hist}(R, s^0)$ is the corresponding 0-stack in $\mathsf{top}^k(R(0))$, or*
- $|R| = 1$ *and the only transition of R performs pop^r for $r \leq k$, or push^r_γ for $r \leq k$.*

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). $\qquad\square$

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 $|\mathsf{top}^k(R(0))| \leq \mathsf{top}^k(R(i))|$ for each $i \in [0,|R|]$ such that $R{\upharpoonright}_{i,|R|} \in \mathsf{up}^k$.*

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 that is not $(k-1)$-upper, where $k \in [1,n]$. Suppose that $R{\upharpoonright}_{0,j}$ is $(k-1)$-upper for the greatest index $j \in [0,|R|-1]$ such that $R{\upharpoonright}_{j,|R|}$ is k-upper (in particular such an index j exists). Then R is a k-return.*

Proposition 6.6. *Let R be a k-return, where $k \in [1,n]$. Then $\mathsf{pop}^k(\mathsf{top}^k(R(0))) \cong \mathsf{top}^k(R(|R|))$. Additionally for every 0-stack s^0 in $\mathsf{top}^k(R(|R|))$, $\mathsf{hist}(R, s^0)$ is the corresponding 0-stack in $\mathsf{pop}^k(\mathsf{top}^k(R(0)))$.*

Proof of Propositions 6.3-6.6. Recall that a $(k-1)$-upper run, a composition of two k-upper runs, and a k-return are special cases of k-upper runs. Thus in all four propositions we have a 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 $R{\upharpoonright}_{i,|R|}$ is k-upper. Notice that $0 \in X$
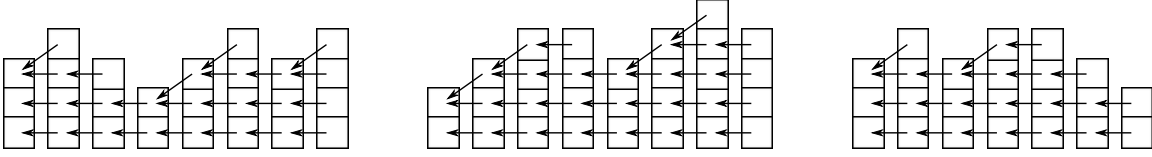
Figure 2: Illustrations to Equality (6.1) and Propositions 6.3-6.6. Particular columns represent a k-stack in consecutive configurations of a run. Arrows show the value of the hist function. The run on the first diagram is not $(k-1)$-upper. The run on the second diagram is $(k-1)$-upper; notice that every its prefix is $(k-1)$-upper, assuming that (in the final configuration of the prefix) the illustrated k-stack is the topmost one (cf. Proposition 6.4). The run on the last diagram is a k-return.

and $|R| \in X$. For $i \in X$, let $r_i = |\mathsf{top}^k(R(i))|$, and let $s_i(r)$ be the r-th $(k-1)$-stack in $\mathsf{top}^k(R(i))$ (for $r \in [1, r_i]$). We claim that for all $b, e \in X$ with $b \leq e$, and for each $r \in [1, r_e]$,

$$\mathsf{hist}(R{\restriction}_{b,e}, s_e(r)) = s_b\big(\min(\{r\} \cup \{r_l \colon l \in X \wedge b \leq l < e\})\big). \qquad (6.1)$$

See Figure 2 for an illustration. We prove Equality (6.1) by induction on $e - b$. For $e = b$ it is true. For the induction step consider the smallest $e' \in X$ that is greater than e. Notice that for all $l \in [e+1, e'-1]$ necessarily $R{\restriction}_{l,e'} \notin \mathsf{up}^k$ (since $R{\restriction}_{l,e'} \in \mathsf{up}^k$ implies that $R{\restriction}_{l,|R|} \in \mathsf{up}^k$, that is, that $l \in X$), so the subrun $R{\restriction}_{e,e'}$ is in one of the forms described by Proposition 6.2. For both of them we see that for each $r \in [1, r_{e'}]$,

$$\mathsf{hist}(R{\restriction}_{e,e'}, s_{e'}(r)) = s_e(\min\{r, r_e\}).$$

Together with the induction assumption for b, e, this implies Equality (6.1) for b, e'.

We also claim that for all $b, e \in X$ with $b \leq e$,

$$R{\restriction}_{b,e} \in \mathsf{up}^{k-1} \quad \Leftrightarrow \quad r_b \leq r_i \text{ for each } i \in X \cap [b, e]. \qquad (6.2)$$

Indeed, $R{\restriction}_{b,e}$ is $(k-1)$-upper if and only if $\mathsf{hist}(R{\restriction}_{b,e}, s_e(r_e)) = s_b(r_b)$, and, as we see from Equality (6.1), the latter holds if and only if $r_b \leq r_i$ for each $i \in X \cap [b, e]$.

Proposition 6.3 follows directly from Equivalence (6.2) used with $b = 0$ and $e = |R|$.

In order to prove Proposition 6.4, we suppose that $R = S \circ T$, that R is $(k-1)$-upper, and that T is k-upper. Using Equivalence (6.2) with $b = 0$ and $e = |R|$, we obtain that $r_0 \leq r_i$ for each $i \in X$. Since T is k-upper, $|S| \in X$. Thus, we can use Equivalence (6.2) with $b = 0$ and $e = |S|$; it tells us that S is $(k-1)$-upper, as required.

Consider now a situation as in Proposition 6.5, namely, let $R \notin \mathsf{up}^{k-1}$, let $j = \max(X \cap [0, |R|-1])$, and let $R{\restriction}_{0,j} \in \mathsf{up}^{k-1}$. If $j < |R| - 1$, then from Proposition 6.2 applied to $R{\restriction}_{j,|R|}$ we obtain that $\mathsf{hist}(R{\restriction}_{j,|R|}, \mathsf{top}^{k-1}(R(|R|))) = \mathsf{top}^{k-1}(R(j))$, that is, that $R{\restriction}_{j,|R|}$ is $(k-1)$-upper. Since a composition of $(k-1)$-upper runs is $(k-1)$-upper, this contradicts the assumptions that $R{\restriction}_{0,j}$ is $(k-1)$-upper but R is not. Thus, $j = |R| - 1$. By Equivalence (6.2), the assumptions $R{\restriction}_{0,|R|-1} \in \mathsf{up}^{k-1}$ and $R \notin \mathsf{up}^{k-1}$ imply that $r_0 \leq r_i$ for each $i \in X \setminus \{|R|\}$, but not for $i = |R|$. It follows that $r_0 = r_{|R|-1} = r_{|R|} + 1$, since $|r_{|R|-1} - r_{|R|}| \leq 1$. From Equality (6.1) we deduce that $\mathsf{hist}(R, \mathsf{top}^{k-1}(R(|R|))) = \mathsf{top}^{k-1}(\mathsf{pop}^k(R(0)))$, and from Equivalence (6.2) that $R{\restriction}_{i,|R|} \notin \mathsf{up}^{k-1}$ for $i \in X \cap [0, |R|-1]$. For $i \notin X$, we also have that $R{\restriction}_{i,|R|} \notin \mathsf{up}^k \supseteq \mathsf{up}^{k-1}$ by definition of X. Thus R is a k-return.

Finally, suppose that R is a k-return. By definition, this implies that $R{\restriction}_{i,|R|}$ is not $(k-1)$-upper for any $i \in X \setminus \{|R|\}$, so, by Equivalence (6.2), for every $i \in X \setminus \{|R|\}$ there is some $j \in X$ such that $j > i$ and $r_i > r_j$. By transitivity, it actually holds that $r_i > r_{|R|}$ for each $i \in X \setminus \{|R|\}$. Thus, Equality (6.1) implies that $\mathsf{hist}(R, s_{|R|}(r)) = s_0(r)$ for each $r \in [1, r_{|R|}]$; moreover, $\mathsf{hist}(R{\restriction}_{i,|R|}, s_{|R|}(r)) \neq \mathsf{top}^{k-1}(R(i))$ for all $i < |R|$, which implies that $s_{|R|}(r)$ is an unmodified copy of $s_0(r)$ (a $(k-1)$-stack can be modified only when it becomes the topmost $(k-1)$-stack). It means that $\mathsf{top}^k(R(|R|))$ consists of the $r_{|R|}$ bottommost $(k-1)$-stacks of $\mathsf{top}^k(R(0))$, also in the sense of the history function. By the definition of a k-return, $\mathsf{hist}(R, s_{|R|}(r_{|R|})) = s_0(r_0 - 1)$, so $r_{|R|} = r_0 - 1$.   □

We now give one more proposition, which is an immediate consequence of Proposition 6.6.

Proposition 6.7. *Let R be a run such that its first transition performs push^k_γ, and $R{\restriction}_{1,|R|}$ is a k-return, where $k \in [1, n]$. Then $\mathsf{top}^k(R(0)) \cong \mathsf{top}^k(R(|R|))$. Additionally, for every 0-stack s^0 in $\mathsf{top}^k(R(|R|))$, $\mathsf{hist}(R, s^0)$ is the corresponding 0-stack in $\mathsf{top}^k(R(0))$.*   □

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

Proposition 6.8. *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 it 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 6.7.

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 an operation removes the topmost k-stack of $R(0)$, which contradicts 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 $\mathsf{hist}(R{\restriction}_{0,1}, s^k) = \mathsf{top}^k(R(0))$ holds only for two k-stack of $R(1)$: for $s^k = \mathsf{top}^k(R(1))$ and for $s^k = \mathsf{top}^k(\mathsf{pop}^r(R(1)))$. 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 $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^k(R(|R|))) = \mathsf{top}^k(\mathsf{pop}^r(R(1)))$. Thus, also $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$. Let x be the smallest positive index for which $R{\restriction}_{x,|R|}$ is $(r-1)$-upper. Then $\mathsf{hist}(R{\restriction}_{1,x}, \mathsf{top}^{r-1}(R(x))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$ (by compositionality of histories, because $\mathsf{hist}(R{\restriction}_{x,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(R(x))$ and $\mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$), and there is no $i \in [1, x - 1]$ such that $R{\restriction}_{i,x}$ is $(r - 1)$-upper (because $R{\restriction}_{i,x} \in \mathsf{up}^{r-1}$ and $R{\restriction}_{x,|R|} \in \mathsf{up}^{r-1}$ would imply that $R{\restriction}_{i,|R|} \in \mathsf{up}^{r-1}$). Thus, $R{\restriction}_{1,x}$ is an r-return. The knowledge at this point of the proof is summarized in Figure 3.
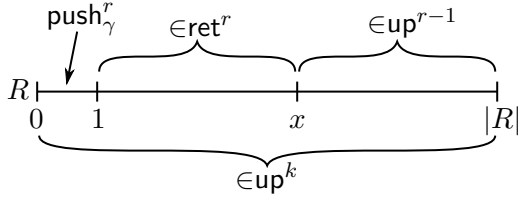
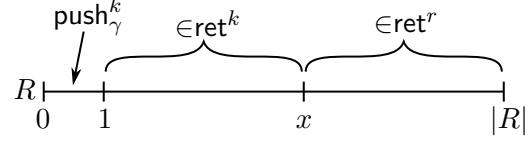Figure 3: Illustration for the proof of Proposition 6.8

Figure 4: Illustration for the proof of Proposition 6.9

If $x = |R|$, then we have Case (3). For the remaining part of the proof suppose that $x < |R|$. Let $s^k = \mathsf{hist}(R{\restriction}_{x,|R|}, \mathsf{top}^k(R(|R|)))$. Because $R{\restriction}_{x,|R|} \in \mathsf{up}^{r-1}$ and $\mathsf{top}^k(R(|R|))$ is in $\mathsf{top}^{r-1}(R(|R|))$ (recall that $k \leq r-1$), we have that s^k is in $\mathsf{top}^{r-1}(R(x))$. On the other hand, because $R \in \mathsf{up}^k$, by compositionality of histories we know that $\mathsf{hist}(R{\restriction}_{0,x}, s^k) = \mathsf{top}^k(R(0))$. Proposition 6.7 applied to $R{\restriction}_{0,x}$ (its first operation is push^r_γ, and $R{\restriction}_{1,x}$ is an r-return) implies that $s^k = \mathsf{top}^k(R(x))$, that is, that $R{\restriction}_{0,|R|}$ and $R{\restriction}_{x,|R|}$ are k-upper. Thus, we have Case (4). □

Proposition 6.9. *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 it 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 $\mathsf{hist}(R{\restriction}_{0,1}, \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$ (it is important that $k \neq r$ in the case of push^k_γ), and hence

$$\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{hist}(R{\restriction}_{0,1}, \mathsf{hist}(R{\restriction}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))))$$
$$= \mathsf{hist}(R{\restriction}_{0,1}, \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))\,.$$

In particular, this implies that R is not $(r-1)$-upper; moreover $R{\restriction}_{i,|R|}$ is not $(r-1)$-upper for $i \in [1, |R|-1]$ because $R{\restriction}_{1,|R|}$ is an r-return. Thus, R is an r-return. In Case (3), let $x-1$ be the length of the first return (so the k-return ends in $R(x)$). The situation is depicted in Figure 4. Recall that $k \geq r$. By Proposition 6.7, $\mathsf{hist}(R{\restriction}_{0,x}, \mathsf{top}^{r-1}(\mathsf{pop}^r(R(x)))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$. Since $\mathsf{hist}(R{\restriction}_{x,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(x)))$, we conclude that $\mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$. This in particular implies that R is not $(r-1)$-upper. Because $R{\restriction}_{x,|R|}$ is an r-return, $R{\restriction}_{i,|R|}$ cannot be $(r-1)$-upper for $i \in [x, |R|-1]$. If $R{\restriction}_{i,|R|}$ was $(r-1)$-upper for some $i \in [1, x-1]$, then $\mathsf{hist}(R{\restriction}_{i,x}, \mathsf{top}^{r-1}(\mathsf{pop}^r(R(x)))) = \mathsf{top}^{r-1}(R(i))$. This would imply that $R{\restriction}_{i,x}$ is $(k-1)$-upper (both for $k > r$ and $k = r$), which is impossible, because $R{\restriction}_{1,x}$ is a k-return. We conclude that R is an r-return.

Concentrate now on the left-to-right implication. Before starting the proof, notice that in order to prove that $R{\restriction}_{x,|R|} \in \mathsf{ret}^r$ for some $x \in [0, |R|]$, it is enough to check that $\mathsf{hist}(R{\restriction}_{x,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(x)))$: the condition that $R{\restriction}_{i,|R|} \notin \mathsf{up}^{r-1}$ for all $i \in [x, |R|-1]$ is ensured by the fact that R itself is an r-return.

Of course $|R| \geq 1$. Because R is an r-return,

$$\mathsf{hist}(R{\upharpoonright}_{0,1}, \mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|)))) = \mathsf{hist}(R, \mathsf{top}^{r-1}(R(|R|)))$$
$$= \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0))) \,. \qquad (6.3)$$

Observe that the first operation, between $R(0)$ and $R(1)$, cannot be pop^k for $k \geq r + 1$, as after such an operation there would be no $(r-1)$-stack s^{r-1} of $R(1)$ such that $\mathsf{hist}(R{\upharpoonright}_{0,1}, s^{r-1}) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$, which contradicts Equality (6.3).

Suppose that the first operation of R is pop^r. In this situation, the only $(r-1)$-stack s^{r-1} of $R(1)$ such that $\mathsf{hist}(R{\upharpoonright}_{0,1}, s^{r-1}) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$ is $s^{r-1} = \mathsf{top}^{r-1}(R(1))$, and thus we have $\mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(R(1))$ by Equality (6.3). This means that $R{\upharpoonright}_{1,|R|}$ is $(r-1)$-upper. A nonempty suffix of an r-return cannot be $(r-1)$-upper, so $|R| = 1$; we have Case (1).

Next, suppose that the first operation is read, or pop^k for $k \leq r-1$, or push_γ^k for $k \leq r-1$. In this situation, the only $(r-1)$-stack s^{r-1} of $R(1)$ such that $\mathsf{hist}(R{\upharpoonright}_{0,1}, s^{r-1}) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$ is $s^{r-1} = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$, and thus $\mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$, by Equality (6.3). In consequence, $R{\upharpoonright}_{1,|R|}$ is an r-return; we have Case (2).

Finally, suppose that the first operation of R is push_γ^k for $k \geq r$. If $k > r$, then there are two $(r-1)$-stacks s^{r-1} of $R(1)$ such that $\mathsf{hist}(R{\upharpoonright}_{0,1}, s^{r-1}) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$, namely $s^{r-1} = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$ and $s^{r-1} = \mathsf{top}^{r-1}(\mathsf{pop}^r(\mathsf{pop}^k(R(1))))$. If $k = r$, only the latter possibility remains: $s^{r-1} = \mathsf{top}^{r-1}(\mathsf{pop}^r(\mathsf{pop}^k(R(1))))$. By Equality (6.3), $\mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|)))$ has to be one of these two $(r-1)$-stacks.

Suppose first that $\mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(1)))$ and $k > r$. Then $R{\upharpoonright}_{1,|R|}$ is an r-return; we have Case (2).

The opposite possibility is that $\mathsf{hist}(R{\upharpoonright}_{1,|R|}, \mathsf{top}^{r-1}(R(|R|))) = \mathsf{top}^{r-1}(\mathsf{pop}^r(\mathsf{pop}^k(R(1))))$. Because $\mathsf{top}^{r-1}(R(|R|))$ and $\mathsf{top}^{r-1}(\mathsf{pop}^r(\mathsf{pop}^k(R(1))))$ are in $\mathsf{top}^k(R(|R|))$ and $\mathsf{top}^k(R(1))$, respectively (recall that $k \geq r$), this implies that $R{\upharpoonright}_{1,|R|} \in \mathsf{up}^k$. Let x be the smallest positive index such that $R{\upharpoonright}_{1,x} \notin \mathsf{up}^{k-1}$ and $R{\upharpoonright}_{x,|R|} \in \mathsf{up}^k$ (it exists: in the worst case we can take $x = |R|$, since $R{\upharpoonright}_{1,|R|} \notin \mathsf{up}^{k-1}$ and $R{\upharpoonright}_{|R|,|R|} \in \mathsf{up}^k$). Because $R{\upharpoonright}_{1,|R|}$ and $R{\upharpoonright}_{x,|R|}$ are k-upper, also $R{\upharpoonright}_{1,x}$ is k-upper. Moreover, because $R{\upharpoonright}_{1,x} \notin \mathsf{up}^{k-1}$ and $R{\upharpoonright}_{1,1} \in \mathsf{up}^{k-1}$, necessarily $x > 1$. Let also j be the greatest index in $[1, x-1]$ such that $R{\upharpoonright}_{j,x} \in \mathsf{up}^k$ (it exists, because $R{\upharpoonright}_{1,x} \in \mathsf{up}^k$ and $1 \in [1, x-1]$). Then $R{\upharpoonright}_{j,|R|}$ (a composition of two k-upper runs) is k-upper, and thus $R{\upharpoonright}_{1,j}$ is not $(k-1)$-upper, by minimality of x. In such a situation, Proposition 6.5 implies that $R{\upharpoonright}_{1,x}$ is a k-return. Let $t^{r-1} = \mathsf{hist}(R{\upharpoonright}_{x,|R|}, \mathsf{top}^{r-1}(R(|R|)))$. Because $R{\upharpoonright}_{x,|R|} \in \mathsf{up}^k$ and $\mathsf{top}^{r-1}(R(|R|))$ is in $\mathsf{top}^k(R(|R|))$ (since $k \geq r$), we have that t^{r-1} is in $\mathsf{top}^k(R(x))$. On the other hand, because $R \in \mathsf{ret}^r$, by compositionality of histories we know that $\mathsf{hist}(R{\upharpoonright}_{0,x}, t^{r-1}) = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(0)))$. Proposition 6.7 applied to $R{\upharpoonright}_{0,x}$ (its first operation is push_γ^k, and $R{\upharpoonright}_{1,x}$ is a k-return) implies that $t^{r-1} = \mathsf{top}^{r-1}(\mathsf{pop}^r(R(x)))$, that is, that $R{\upharpoonright}_{x,|R|} \in \mathsf{ret}^r$; we have Case (3). $\square$

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 positionless 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, only the topmost 0-stack is removed, so this is indeed a 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 positionless topmost 1-stack. If additionally the topmost 2-stacks of both configurations contain 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 positionless topmost 1-stack. On the other hand, we summarize the rest of the stack in a small piece of information, called a 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 positionless topmost k-stack and the same type. Then, as said above, from each of them we can execute a similar k-upper run. Typically, these k-upper runs are 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 minimal suffixes that are k-upper (notice that infixes or even prefixes of R_i can be k-upper, but suffixes are not). We compare 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 that $\varphi(R_i) = \varphi(S_i)$ and $\mathsf{top}^k(R_i(0)) \cong \mathsf{top}^k(S_i(0))$,

as well as $\mathsf{top}^k(R(|R|)) \cong \mathsf{top}^k(S(|S|))$. In particular, two runs R, S of length 0 are (k, φ)-parallel when $\mathsf{top}^k(R(0)) \cong \mathsf{top}^k(S(0))$. When saying that two runs are (k, φ)-parallel we implicitly mean that they are k-upper.

We claim that if runs R and S are (k, φ)-parallel, and R is divided in any way into k-upper runs $R = R'_1 \circ \cdots \circ R'_m$, then S can be as well divided into k-upper runs $S = S'_1 \circ \cdots \circ S'_m$ such that for each $i \in [1, m]$ it holds that $\varphi(R'_i) = \varphi(S'_i)$ and $\mathsf{top}^k(R'_i(0)) \cong \mathsf{top}^k(S'_i(0))$, as well as $\mathsf{top}^k(R(|R|)) \cong \mathsf{top}^k(S(|S|))$. Indeed, on the 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 assign its $(\mathcal{A}, \varphi)$-*type* (simply called *type* when $\mathcal{A}$ and φ are fixed), which 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 positionless topmost k-stack as $R(0)$. Then from c we can start a run that is (k, φ)-parallel to R.*

In addition to types, we 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 $c_1, c_2, \ldots$ and $d_1, d_2, \ldots$ be infinite sequences of configurations that are $(\mathcal{A}, \varphi)$-sequence-equivalent, and in which all configurations have the same $(\mathcal{A}, \varphi)$-type and the same positionless 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 the sequences $\sharp(S_1 \circ S'_1), \sharp(S_2 \circ S'_2), \ldots$ and $\sharp(T_1 \circ T'_1), \sharp(T_2 \circ T'_2), \ldots$ are either both bounded or both unbounded.*

Let us mention briefly how this theorem is used in Section 10. We consider there a configuration c reached after reading a complicated word, containing some blocks of stars, separated by some brackets. Using a pumping lemma developed in Section 9, we increase the number of stars read in one of such blocks, obtaining configurations $c_1, c_2, \ldots$ at the end of the run (where consecutive configurations are reached after reading more and more stars in the considered block). It is ensured that all c_i have the same $(\mathcal{A}, \varphi)$-type, and the same positionless topmost k-stack. Likewise, we increase the number of stars read in some other block of stars, obtaining configurations $d_1, d_2, \ldots$. Having more blocks of stars than classes of the $(\mathcal{A}, \varphi)$-sequence-equivalence relation we can ensure that the sequences $c_1, c_2, \ldots$ and $d_1, d_2, \ldots$ are $(\mathcal{A}, \varphi)$-sequence-equivalent, by the pigeonhole principle. Theorem 7.4 says that the two sequences of configurations cannot be distinguished by runs (of a specific form) starting in these configurations: the runs contain sharps corresponding to stars either from both considered blocks of stars (and then both sequences $\sharp(S_1 \circ S'_1), \sharp(S_2 \circ S'_2), \ldots$ and $\sharp(T_1 \circ T'_1), \sharp(T_2 \circ T'_2), \ldots$ are unbounded) or from none of them (and then both these sequences are bounded).

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 a 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 analysis 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 serves as the 3-return R'. Then basically we need two equivalence classes of sequences of configurations (recall that only for sequences with the same positionless topmost 1-stack the relation is meaningful): one where the number of b symbols in the topmost 2-stacks in the configurations is bounded, and one where this number is unbounded. Depending on this fact, the runs read either a bounded or an unbounded number of sharps. Of course in general we need more classes than just two ("bounded" and "unbounded"), because, for example, 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 of Broadbent, Carayol, Hague, and Serre [BCHS12], as well as to intersection types of Kobayashi [Kob09]. 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 a run, not just to determine existence of some kind of runs. On the other hand, in the conference version of the current paper [Par12b] we were using types that were directly describing returns (while here returns correspond to using an assumption); these types were more complicated.

Run Descriptors. We label 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 $\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, f)$. Its first coordinate, p, is called the state of σ. The sets Ψ^i, for $i \in [k+1, n]$, are called *assumption sets* of σ, and are denoted $\mathsf{ass}^i(\sigma)$. The last coordinate, f, is called a *productivity flag* of σ. When $f = \mathsf{np}$, we say that σ is *nonproductive*; otherwise, it is *productive*. By $\mathcal{T}_{\mathsf{np}}$ and $\mathcal{T}_{\mathsf{pr}}$ we denote

the subsets of $\bigcup_{k\in[0,n]} \mathcal{T}^k$ containing only nonproductive and productive run descriptors, respectively.

A run descriptor σ assigned to some k-stack s^k describes a run that starts in a configuration with state p and topmost k-stack s^k. The run descriptor "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 $\pi_2(\Psi^i)$. An assumption $(m, \tau) \in \Psi^i$ is used when (a copy of) the stack t^i becomes uncovered. The run descriptor τ describes a run from such a configuration d; this run is 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 m describes the word w read by the return: $m = \varphi(w)$.

Beside of the state p, and the assumption sets, in σ we also have a productivity flag. Roughly speaking, 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 run descriptor (coming from some assumption set Ψ^i) is used at least twice as an assumption (the latter also increases the number of $\sharp$ symbols read, since some reading described by this productive assumption is repeated). Thanks to the productivity flag, we can estimate the number of $\sharp$ symbols read, by calculating the number of productive run descriptors used.

One may wonder which runs have a description by a run descriptor. The answer is that all runs: we do not restrict ourselves to any specific kind of runs at this point.

We now give more intuitions on run descriptors, in particular cases. Run descriptors in $\mathcal{T}^n$, assigned to stacks s^n of the maximal order n, are simply of the form (p, f). When the starting state p is fixed, we only have two run descriptors: (p, np) and (p, pr). The former describes runs from (p, s^n) that do not read any $\sharp$ symbols, while the latter those that do read some $\sharp$ symbols.

A run descriptor in $\mathcal{T}^{n-1}$ is of the form (p, Ψ^n, f). When assigned to a stack s^{n-1}, it describes a run R from a configuration of the form $c = (p, t^n : s^{n-1})$. It is possible that R never visits t^n, and only builds on top of s^{n-1} (i.e., R is $(n-1)$-upper). In this situation, the set of assumptions Ψ^n is empty, and the flag f simply says whether R reads some $\sharp$ symbols. The opposite case is that R uncovers the stack t^n in some configuration $d = (q, t^n)$, that is, that some its prefix $R\!\restriction_{0,i}$ is an n-return. In this situation $\Psi^n = \{(\varphi(R\!\restriction_{0,i}), \tau)\}$, where τ describes the suffix $R\!\restriction_{i,|R|}$. Because we are considering the highest order, when t^n is uncovered in $R(i)$ there are no other copies of t^n. This means that only a single assumption may be used for t^n (i.e., $|\Psi^n| \leq 1$), and this assumption is used only once. The flag f simply says whether the prefix $R\!\restriction_{0,i}$ reads some $\sharp$ symbol.

For run descriptors in $\mathcal{T}^{n-2}$ the situation becomes more interesting. We explain this by means of an example.

Example 7.6. Consider a run R such that
- $R(0) = (p_0, t^n : t^{n-1} : s^{n-2})$,
- $R\!\restriction_{0,i}$ is an $(n-1)$-return with $R(i) = (p_1, (t^n : t^{n-1} : u^{n-2}) : \mathsf{p}_{+1}(t^{n-1}))$ (notice that $R\!\restriction_{0,i}$ performs some push^n_γ without a corresponding pop^n);
- $R\!\restriction_{i,j}$ is an n-return with $R(j) = (p_2, t^n : t^{n-1} : u^{n-2})$;
- $R\!\restriction_{j,k}$ is an $(n-1)$-return with $R(k) = (p_3, t^n : t^{n-1})$;
- $R\!\restriction_{k,l}$ is an n-return with $R(l) = (p_4, t^n)$.

Let us see how such a run is described by a run descriptor $\sigma = (p_0, \Psi^n, \Psi^{n-1}, f) \in \mathcal{T}^{n-2}$, which can be assigned to the $(n-2)$-stack s^{n-2}. Necessarily Ψ^n is a singleton containing

$(\varphi(R{\upharpoonright}_{0,l}), \tau_4)$ for τ_4 describing the suffix $R{\upharpoonright}_{l,|R|}$. The set Ψ^{n-1} contains in general two elements: $(\varphi(R{\upharpoonright}_{0,i}), \tau_1)$ for τ_1 describing the suffix $R{\upharpoonright}_{i,|R|}$, and $(\varphi(R{\upharpoonright}_{0,k}), \tau_3)$ for τ_3 describing the suffix $R{\upharpoonright}_{k,|R|}$. If $\tau_1 \neq \tau_3$, then f says whether $R{\upharpoonright}_{0,i}$ or $R{\upharpoonright}_{j,k}$ reads some $\sharp$ symbol (and the flags in τ_1, τ_3, τ_4 are responsible for the subruns $R{\upharpoonright}_{i,k}$, $R{\upharpoonright}_{k,l}$, and $R{\upharpoonright}_{l,|R|}$, respectively). In may also happen that $\tau_1 = \tau_3$. In this situation, we say that the run descriptor τ_1 is used twice as an assumption. Then $f = \mathsf{pr}$ if $R{\upharpoonright}_{0,i}$ or $R{\upharpoonright}_{j,k}$ reads some $\sharp$ symbol, but also when τ_1 is productive (i.e., when a productive run descriptor is used more than once as an assumption). The intuition for this is that now while looking at run descriptors in $\pi_2(\Psi^{n-1})$ it is not visible that there are two subruns $R{\upharpoonright}_{i,j}$ and $R{\upharpoonright}_{k,l}$ reading $\sharp$ symbols, as they both correspond to the same assumption; by setting $f = \mathsf{pr}$ we reflect the fact that R reads more $\sharp$ symbols than in the situation when every assumption would be used only once.[2]

We remark that a run descriptor $\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, f)$ should not be seen as a classical implication of the form "if the stacks below the topmost k-stack satisfy assumptions $\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}$, then there exists a run satisfying some properties". It is much closer to an implication in a linear logic. Indeed, the conclusion of the implication is trivial, as it, basically, says only that there exists some (arbitrary) run. The interesting information about the run is contained in the sets of assumptions: by specifying an assumption, we say that there is a suffix of the run corresponding to this assumption. In particular, it is essential that there are no redundant assumptions (every assumption "has to be used", at least once). Moreover, the information whether some assumptions are used more than once is also recorded, in the productivity flag.

Composers. For $m \in M$ and $\Psi \subseteq M \times \mathcal{T}^k$ we use the notation $m \circ \Psi$ for $\{(m \cdot m', \sigma) \colon (m', \sigma) \in \Psi\}$. Given a run descriptor $\sigma = (p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{l+1}, f) \in \mathcal{T}^l$, for $k \in [l, n]$ by $\mathsf{red}^k(\sigma)$ we denote the "reduced" run descriptor $(p, \Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, g) \in \mathcal{T}^k$ in which

$$g = \mathsf{np} \Leftrightarrow (f = \mathsf{np}, \text{ and } \pi_2(\Psi^i) \subseteq \mathcal{T}_{\mathsf{np}} \text{ for each } i \in [l+1, k]).$$

The following proposition is a direct consequence of the definition.

Proposition 7.7. *For $0 \leq l \leq j \leq k \leq n$ and $\sigma \in \mathcal{T}^l$ it holds that $\mathsf{red}^k(\mathsf{red}^j(\sigma)) = \mathsf{red}^k(\sigma)$.* □

We now define composers, which are used to compose run descriptors corresponding to smaller stacks into run descriptors corresponding to greater stacks.

Definition 7.8. Consider a tuple $(\Phi^k, \Phi^{k-1} \dots, \Phi^l; \Psi^k; f)$, where $0 \leq l \leq k \leq n$, $\Phi^i \subseteq M \times \mathcal{T}^i$ 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

(C1) $\Phi^i = \bigcup\{m \circ \mathsf{ass}^i(\sigma) \colon (m, \sigma) \in \Phi^l\}$ for each $i \in [l+1, k]$,

(C2) $\Psi^k = \{(m, \mathsf{red}^k(\sigma)) \colon (m, \sigma) \in \Phi^l\}$,

(C3) $|\pi_2(\Psi^k)| = |\pi_2(\Phi^l)|$ (which means that each $\sigma \in \pi_2(\Phi^l)$ gives a different $\mathsf{red}^k(\sigma)$), and

---

[2] One can imagine two possible definitions of "the same assumption": we may compare either only run descriptors (in our case, $\tau_1 = \tau_3$), or pairs consisting of a monoid element and a run descriptor (in our case, $(\varphi(R{\upharpoonright}_{0,i}), \tau_1) = (\varphi(R{\upharpoonright}_{0,k}), \tau_3)$). At first glance both definitions look equally good, or the latter definition seems to be more natural than the former. It turns out, however, that the latter definition is problematic. The difficulty is that for pairs that are originally different, $(m_1, \sigma_1) \neq (m_2, \sigma)$, it may happen that after multiplying them by a monoid element they become equal, $(m \cdot m_1, \sigma_1) = (m \cdot m_2, \sigma)$. We prefer to avoid this, and hence we stick to the former definition.

(C4) $f = \mathsf{np}$ 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
$\sigma, \tau \in \pi_2(\Phi^l)$ such that $\sigma \neq \tau$.

Suppose that we have a k-stack $t^k = s^k : s^{k-1} : \cdots : s^l$, where, for $i \in [l, k]$, elements of
Φ^i "are assigned" to s^i. Intuitively, a tuple $(\Phi^k, \Phi^{k-1} \ldots, \Phi^l; \Psi^k; f)$ is a composer, if in such
a situation the set Ψ^k can be assigned to the whole stack t^k. As we see, the definition does
not depend on the stacks that are actually composed, only on the sets Φ^i assigned to these
stacks. One can think about $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$ as about inputs to the composer, and about
Ψ^k and f as outputs. Nevertheless, already Φ^l determines all remaining coordinates of the
composer (when l and k are fixed). We remark that not every set $\Phi^l \subseteq M \times \mathcal{T}^l$ can be used
in a composer, due to Condition (C3) of the definition.

The definition says that run descriptors assigned to t^k are of the form $\mathsf{red}^k(\sigma)$ for σ
assigned to s^l (the reason is that if the topmost k-stack of a configuration is t^k, then its
topmost l-stack is s^l). Moreover, the assumptions of σ that are contained in $\mathsf{ass}^i(\sigma)$ for
$i \in [l+1, k]$ have to be realized by the stacks s^i. We notice that the run descriptor $\mathsf{red}^k(\sigma)$
is productive when σ is productive or some of the assumptions in $\mathsf{ass}^i(\sigma)$ for $i \in [l+1, k]$ is
productive (cf. the definition of $\mathsf{red}^k(\sigma)$); in other words, in a run corresponding to the run
descriptor, the part corresponding to the stack t^k is productive when some part corresponding
to s^i for some $i \in [l, k]$ is productive.

The composer itself also has a productivity flag f. The intuition is that we set this flag
to pr if the runs described by elements of Ψ^k read more $\sharp$ symbols than those described
by elements of $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$, in total. This is the case when a productive run descriptor
(coming from some Φ^i for $i \in [l+1, k]$) is used as an assumption for more than one element
of Φ^l. While summing over all run descriptors from all Φ^i, such a run descriptor is added
only once, but it contributes to more than one run descriptor in Ψ^k.

Another important issue is that in $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$ we only have elements that really
contribute while constructing elements of Ψ^k. Simultaneously, we require that every run
descriptor in Ψ^k has exactly one realization by run descriptors from $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$ (i.e.,
it is obtained by reducing exactly one run descriptor from Φ^l). The justification for both
these properties is the same: we want to "approximate" the number of $\sharp$ symbols read by
runs described by elements of Ψ^k while looking at the number of $\sharp$ symbols read by runs
described by elements of $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$. Any redundant run descriptors in $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$
would bias our calculations; multiple decompositions of a single element of Ψ^k would also
bias our calculations.

In the sets $\Phi^k, \Phi^{k-1} \ldots, \Phi^l$ and Ψ^k we also have monoid elements, not only run descriptors.
Intuitively, a pair $(m, \sigma) \in \Psi^k$ (or $(m, \sigma) \in \Phi^i$) corresponds to a run consisting of two parts:
the first part reads a word evaluating to m and uncovers the stack t^k (s^i, respectively);
the second part starts when the topmost k-stack is t^k (the topmost i-stack is s^i), and it
is described by σ. The monoid elements in Ψ^k are the same as in Φ^l, since uncovering t^k
means uncovering s^l. On the other hand, elements in Φ^i for $i \in [l+1, k]$ are obtained as the
composition of m coming from $(m, \sigma) \in \Phi^l$ (describing the word read before uncovering s^l)
and of m' coming from a particular assumption $(m', \tau) \in \mathsf{ass}^i(\sigma)$ (describing the word read
after uncovering s^l, but before uncovering s^i).

Example 7.9. Suppose that $n = 2$. Let $\tau_{\mathsf{np}} \in \mathcal{T}^1 \cap \mathcal{T}_{\mathsf{np}}$ and $\tau_{\mathsf{pr}} \in \mathcal{T}^1 \cap \mathcal{T}_{\mathsf{pr}}$. Consider the
following elements of $\mathcal{T}^0$:

$$\sigma_1 = (p, \Psi^2, \{(m_1, \tau_{\mathsf{np}}), (m_2, \tau_{\mathsf{pr}})\}, \mathsf{np}), \qquad \sigma_2 = (p, \Psi^2, \{(m_3, \tau_{\mathsf{np}})\}, \mathsf{np}),$$

$$\sigma_1' = (p, \Psi^2, \{(m_3, \tau_{\mathsf{np}})\}, \mathsf{pr})\,, \qquad\qquad \sigma_3 = (q, \Phi^2, \{(m_4, \tau_{\mathsf{np}}), (m_5, \tau_{\mathsf{pr}})\}, \mathsf{pr})\,,$$

and the following elements of $\mathcal{T}^1$:

$$\xi_1 = (p, \Psi^2, \mathsf{pr})\,, \qquad \xi_2 = (p, \Psi^2, \mathsf{np})\,, \qquad \xi_3 = (q, \Phi^2, \mathsf{pr})\,.$$

It holds that $\mathsf{red}^1(\sigma_i) = \xi_i$ for $i \in \{1, 2, 3\}$, and $\mathsf{red}^1(\sigma_1') = \xi_1$. We have composers resulting in a single pair from $M \times \mathcal{T}^1$, like

$$(\{(m_6 \cdot m_1, \tau_{\mathsf{np}}), (m_6 \cdot m_2, \tau_{\mathsf{pr}})\}, \{(m_6, \sigma_1)\}; \{(m_6, \xi_1)\}; \mathsf{np})\,, \qquad\qquad \text{or}$$

$$(\{(m_6 \cdot m_3, \tau_{\mathsf{np}})\}, \{(m_6, \sigma_1')\}; \{(m_6, \xi_1)\}; \mathsf{np})\,.$$

Notice that these two composers have the same "output set". We may also repeat the same run descriptor with multiple monoid elements:

$$(\{(m_6 \cdot m_3, \tau_{\mathsf{np}}), (m_7 \cdot m_3, \tau_{\mathsf{np}})\}, \{(m_6, \sigma_1'), (m_7, \sigma_1')\}; \{(m_6, \xi_1), (m_7, \xi_1)\}; \mathsf{np})\,.$$

Here, the situation that $m_6 \cdot m_3 = m_7 \cdot m_3$ is allowed as well (and then the first set has only a single element). Observe that that all the above composers are nonproductive, even though they involve productive run descriptors. Next, we can also have composers involving multiple run descriptors, like

$$(\{(m_6 \cdot m_3, \tau_{\mathsf{np}})\}, \{(m_6, \sigma_1'), (m_6, \sigma_2)\}; \{(m_6, \xi_1), (m_6, \xi_2)\}; \mathsf{np})\,.$$

This composer is again nonproductive: although there is a run descriptor τ_{np} that appears in assumption sets of both σ_1' and σ_2, this run descriptor τ_{np} is nonproductive. But if a productive run descriptor τ_{pr} appears in assumption sets of two run descriptors (like for σ_1 and σ_3), we have a productive composer:

$$(\{(m_6 \cdot m_1, \tau_{\mathsf{np}}), (m_7 \cdot m_4, \tau_{\mathsf{np}}), (m_6 \cdot m_2, \tau_{\mathsf{pr}}), (m_7 \cdot m_5, \tau_{\mathsf{pr}})\}, \{(m_6, \sigma_1), (m_7, \sigma_3)\};$$
$$\{(m_6, \xi_1), (m_7, \xi_3)\}; \mathsf{pr})\,.$$

As negative examples, we have the following tuples, which are not composers:

$$(\{(m_6 \cdot m_3, \tau_{\mathsf{np}}), (m_6 \cdot m_2, \tau_{\mathsf{pr}})\}, \{(m_6, \sigma_1')\}; \{(m_6, \xi_1)\}; \mathsf{np})\,, \qquad\qquad \text{and}$$

$$(\{(m_6 \cdot m_1, \tau_{\mathsf{np}}), (m_6 \cdot m_2, \tau_{\mathsf{pr}}), (m_7 \cdot m_3, \tau_{\mathsf{np}})\}, \{(m_6, \sigma_1), (m_7, \sigma_1')\};$$
$$\{(m_6, \xi_1), (m_7, \xi_1)\}; \mathsf{np})\,.$$

In the first tuple, the first set contains a redundant pair $(m_6 \cdot m_2, \tau_{\mathsf{pr}})$, which is forbidden. In the second tuple, we have simultaneously two realizations of ξ_1: one using σ_1, and one using σ_1'; this is forbidden as well.

In the next proposition we notice that composers are associative.

Proposition 7.10. *Let* $0 \leq l \leq j \leq k \leq n$*. For all fixed* $\Phi^k, \Phi^{k-1}, \ldots, \Phi^l, \Psi^k$ *and* $f \in \{\mathsf{np}, \mathsf{pr}\}$*, the following two conditions are equivalent:*
- $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^l; \Psi^k; f)$ *is a composer, and*
- $(\Phi^j, \Phi^{j-1}, \ldots, \Phi^l; \Psi^j; f_1)$ *and* $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^{j+1}, \Psi^j; \Psi^k; f_2)$ *are composers for some* Ψ^j
  and f_1, f_2 *such that* $(f = \mathsf{np}) \Leftrightarrow (f_1 = f_2 = \mathsf{np})$*.*

Proof. We use symbols X and X_1, X_2 for the tuples from the first and second item, respectively. First, notice that the definition of a composer instantiated either for X, or for both X_1 and X_2 simultaneously, implies that $\Phi^i \subseteq M \times \mathcal{T}^i$ for $i \in [l, k]$, and $\Psi^k \subseteq M \times \mathcal{T}^k$. While proving the right-to-left implication, Condition (C2) of the definition of a composer

instantiated for X_1 implies that $\Psi^j = \{(m, \mathsf{red}^j(\sigma))\colon (m, \sigma) \in \Phi^l\}$. While proving the opposite implication, we choose Ψ^j in this way; the effect is that X_1 satisfies Condition (C2).

We see that $\mathsf{ass}^i(\mathsf{red}^j(\sigma)) = \mathsf{ass}^i(\sigma)$ for $i \in [j+1, k]$ and $\sigma \in \mathcal{T}^l$, so Condition (C1) for X is equivalent to the conjunction of Condition (C1) for X_1 and X_2.

By Proposition 7.7 we have that $\mathsf{red}^k(\mathsf{red}^j(\sigma)) = \mathsf{red}^k(\sigma)$ for $\sigma \in \mathcal{T}^l$, which implies that Condition (C2) instantiated for X is equivalent to Condition (C2) instantiated for X_2.

Notice that $|\pi_2(\Psi^k)| \leq |\pi_2(\Psi^j)| \leq |\pi_2(\Phi^l)|$, because $\pi_2(\Psi^k) = \{\mathsf{red}^k(\sigma)\colon \sigma \in \pi_2(\Psi^j)\}$ and $\pi_2(\Psi^j) = \{\mathsf{red}^j(\sigma)\colon \sigma \in \pi_2(\Phi^l)\}$. Thus $|\pi_2(\Psi^k)| = |\pi_2(\Phi^l)|$ if and only if $|\pi_2(\Psi^k)| = |\pi_2(\Psi^j)| = |\pi_2(\Phi^l)|$; we obtain equivalence for Condition (C3).

Finally, Condition (C4) for X_1 says that $f_1 = \mathsf{np}$ if and only if $\pi_2(\mathsf{ass}^i(\sigma)) \cap \pi_2(\mathsf{ass}^i(\tau)) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [l+1, j]$ and each $\sigma, \tau \in \pi_2(\Phi^l)$ such that $\sigma \neq \tau$; otherwise $f_1 = \mathsf{pr}$. For X_2 and f_2 we have the same with $i \in [j+1, k]$ (here we use again the fact that $\mathsf{ass}^i(\mathsf{red}^j(\sigma)) = \mathsf{ass}^i(\sigma)$). Thus, while proving the right-to-left implication we have these equivalences for f_1 and f_2; they imply that $(f = \mathsf{np}) \Leftrightarrow (f_1 = f_2 = \mathsf{np})$. While proving the left-to-right implication we define f_1 and f_2 so that this is satisfied; then we have Condition (C4) for X_1 and X_2, and the equivalence $(f = \mathsf{np}) \Leftrightarrow (f_1 = f_2 = \mathsf{np})$. □

Derivation Trees. Next, we say when a run descriptor σ from $\mathcal{T}^0$ can be assigned to a stack symbol γ. To this end, we define how statements of the form $\gamma \vdash \sigma$ can be derived. Such a statement means that from a configuration with topmost stack symbol γ one can start a run described by the run descriptor σ (assuming that the stacks below γ satisfy the assumptions of σ). Actually, it is not enough to define when $\gamma \vdash \sigma$ is true; we explicitly need to handle derivation trees justifying such statements. Thus, in Definition 7.11 we define the notion of a *derivation tree for* $\gamma \vdash \sigma$. Having such a derivation tree D, $\gamma \vdash \sigma$ is called the *conclusion* of D, and σ is called the run descriptor of D and is denoted $\mathsf{rd}(D)$.

Definition 7.11. We define the set of *derivation trees* as the smallest set satisfying the following conditions. Let p be a state, and γ a stack symbol.
(1) A triple $(\mathsf{empty}, \gamma, p)$ is a derivation tree for $\gamma \vdash (p, \emptyset, \emptyset, \ldots, \emptyset, \mathsf{np})$.
(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 (read, p, D') is a derivation tree for $\gamma \vdash (p, \Phi^n, \Phi^{n-1}, \ldots, \Phi^1, f)$, 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 $(\mathsf{pop}, \gamma, p, \tau^k)$ is a derivation tree for

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

(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 $i \in [1, n]$. Moreover, suppose that $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$ 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 $(\mathsf{push}, \gamma, p, D', \mathfrak{D})$ 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 a derivation tree D determines its conclusion. This can be seen by induction on the structure of D. In Cases (1) and (3) this is immediate. In Case (2), γ and τ are determined by D', and the letter a is determined by p and by the state of τ (recall that $\vec{q}$ is required to be injective, by the definition of the read operation); this already fixes the whole conclusion of D. Case (4) is the most complicated one. First, we can see that $\pi_2(\Phi^0)$ and all Ψ^i are fixed by D' and $\mathfrak{D}$. Then, by the definition of a composer, we have that the set Φ^0 has to contain those pairs $(m, \tau) \in M \times \pi_2(\Phi^0)$ for which $(m, \mathsf{red}^k(\tau)) \in \Psi^k$. The set Φ^0 fixes the rest of the composer, and thus the whole conclusion of D.

We now comment on the intuitions staying behind Definition 7.11. Let D be a derivation tree for $\gamma \vdash \sigma$, where the state of σ is p. We should have in mind a run R with $R(0) = (p, s^n : s^{n-1} : \cdots : s^1 : u^0_\gamma)$, where $\mathsf{pos}{\downarrow}(u^0_\gamma) = \gamma$. Basically, the run descriptor σ describes such a run, and the derivation tree D specifies parts of this run for which the 0-stack u^0_γ is responsible.

Case (1) corresponds to an empty run ($|R| = 0$). Thus, the assumption sets of σ are empty (stacks s^i are never uncovered), and the run descriptor is nonproductive.

In the remaining cases, nonempty runs are considered. Case (2) talks about a run R starting with a read operation. Say that this operation reads a symbol a, and ends in a state $\vec{q}(a)$. The derivation tree D' (and, in particular, its run descriptor τ) describes the suffix $R{\restriction}_{1,|R|}$. Thus, we require that the state in τ is $\vec{q}(a)$. Because $R{\restriction}_{0,1}$ does not modify the stack, assumptions from $\mathsf{ass}^i(\tau)$, referring to $\mathsf{top}^i(\mathsf{pop}^i(R(1)))$, refer simultaneously to $\mathsf{top}^i(\mathsf{pop}^i(R(0)))$ (for $i \in [1, n]$). This is expressed by the fact that as $\mathsf{ass}^i(\sigma)$ we almost take assumptions from $\mathsf{ass}^i(\tau)$. The only difference is that we multiply the monoid element in every pair by $\varphi(a)$ on the left. This is because monoid elements in $\mathsf{ass}^i(\tau)$ correspond to some prefixes of $R{\restriction}_{1,|R|}$, while monoid elements in $\mathsf{ass}^i(\sigma)$ should talk about prefixes of the whole R; the latter prefixes additionally read the symbol a at the very beginning. Our new run descriptor σ is productive either when the first operation reads the $\sharp$ symbol, or when the run descriptor τ, describing the rest of the run, is productive. Recall that $\vec{q}$ is required to be injective (cf. the definition of the read operation), so seeing the state $\vec{q}(a)$ we can determine the symbol a that was read.

In Case (3) the first operation of R is pop^k, which leads to a configuration of the form $(q, s^n : s^{n-1} : \cdots : s^k)$. The suffix $R{\restriction}_{1,|R|}$ is described by a run descriptor τ^k. In particular, the state in τ^k should be q. For $i \in [1, k-1]$ the i-stack s^i is never uncovered (it is destroyed by the pop^k operation), so the assumption set $\mathsf{ass}^i(\sigma)$ is set to $\emptyset$. The k-stack s^k is uncovered after the first operation, so we put τ^k to the assumption set $\mathsf{ass}^k(\sigma)$, together with the neutral element of the monoid (because the word read by $R{\restriction}_{0,1}$ is empty). This is the only pair in $\mathsf{ass}^k(\sigma)$, because before $R(1)$ no copies of s^k are created, so s^k cannot be uncovered again. For $i \in [k+1, n]$, the assumption set $\mathsf{ass}^i(\tau^k)$ talks about uncovering s^i, and hence it is taken as $\mathsf{ass}^i(\sigma)$ (the monoid elements remain unchanged, because $R{\restriction}_{0,1}$ does not read any symbols). Notice that, unlike for the read operation, we do not include in D any derivation tree talking about τ^k. This is because in D we only describe the part of R for which the

0-stack u_γ^0 is responsible: on the one hand, after the read operation, the 0-stack u_γ^0 is still on the top of the stack and is responsible for continuing the run; on the other hand, after the pop^k operation, u_γ^0 is no longer present on the stack. The run descriptor σ is nonproductive: u_γ^0 is not responsible for reading any $\sharp$ symbol, and every assumption of σ is used only once (for $i \in [k+1, n]$, we simply pass every assumption from $\mathsf{ass}^i(\sigma)$ to τ^k; independently, τ^k may use these assumptions more than once, but this is a responsibility of τ^k, not of σ).

Finally, we have Case (4), where the first operation of R is push_α^k, leading to a configuration of the form $(q, s^n : s^{n-1} : \cdots : s^{k+1} : t^k : \mathsf{p}_{+1}(s^{k-1} : s^{k-2} : \cdots : s^1 : u_\alpha^0))$ with $t^k = s^k : s^{k-1} : \cdots : s^1 : u_\gamma^0$ and with $\mathsf{pos}{\downarrow}(u_\alpha^0) = \alpha$. A run descriptor τ (having state q) describes the suffix $R{\restriction}_{1,|R|}$. In D we include a derivation tree D' for $\alpha \vdash \tau$, talking about the parts of $R{\restriction}_{1,|R|}$ for which the new topmost 0-stack u_α^0 is responsible, because u_γ^0 is responsible for creating u_α^0. For $i \in [1, k-1] \cup [k+1, n]$, the assumption set $\mathsf{ass}^i(\tau)$ refers to the i-stack s^i, and hence we take assumptions from $\mathsf{ass}^i(\tau)$ to $\mathsf{ass}^i(\sigma)$. The assumption set $\mathsf{ass}^k(\tau)$, in turn, refers to the k-stack t^k, not directly to s^k. Because of that, we need a composer to decompose the assumption set $\mathsf{ass}^k(\tau)$ to sets $\Phi^k, \Phi^{k-1}, \ldots, \Phi^0$, referring to stacks $s^k, s^{k-1}, \ldots, s^1, u_\gamma^0$. Then, for $i \in [1, k]$, assumptions from Φ^i are taken to $\mathsf{ass}^i(\sigma)$. Elements of Φ^0 refer to our 0-stack u_γ^0, and hence should be realized by our derivation tree. Thus, for every run descriptor in Φ^0 we provide a derivation tree (in the set $\mathfrak{D}$). We should set σ to be productive if u_γ^0 is responsible for reading some $\sharp$ symbols, or when some productive run descriptor from an assumption set is used more than once. Maybe this happens inside τ or inside some run descriptor from Φ^0; when any of them is productive, we set σ to be productive. But it may also happen that a productive run descriptor is used as an assumption by τ and by some run descriptor from Φ^0, or that a productive run descriptor is used as an assumption by multiple run descriptors from Φ^0 (in the latter case, the composer is productive); in both these situation we also set σ to be productive.

As already said, a derivation tree D is not a representation of the whole run R, it only talks about parts of R for which the topmost 0-stack u_γ^0 is responsible. It is not, however, a precise representation even of these parts. This can be seen in Case (4) of the definition. It is possible that τ uses some run descriptor ξ from the assumption set $\mathsf{ass}^k(\tau)$ more than once. This means that we have multiple suffixes of the run R described by ξ. But in $\mathfrak{D}$ we include only one derivation tree talking about ξ, corresponding to one of the suffixes of R. Thus, in a sense, a derivation tree for $\gamma \vdash \sigma$ is a proof that a run described by σ exists, rather than a way of representing any such run.

Example 7.12. Consider the 2-DPDA $\mathcal{A}_1$ depicted below; its stack alphabet is $\{\gamma\}$, and input alphabet $A = \{a, b, \sharp\}$.
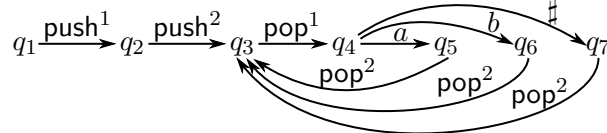
The arrow from q_1 to q_2 denotes that $\delta(q_1, \gamma) = (q_2, \mathsf{push}_\gamma^1)$, the arrow from q_4 to q_5 denotes that $\delta(q_4) = \mathsf{read}(\vec{q})$ with $\vec{q}(a) = q_5$, and so on. Take also a monoid $M = \{1, \mathsf{ne}\}$ with $1 \cdot 1 = 1$ and $1 \cdot \mathsf{ne} = \mathsf{ne} \cdot 1 = \mathsf{ne} \cdot \mathsf{ne} = \mathsf{ne}$, and a morphism $\varphi : A^* \to M$ that maps the empty word to 1 and all nonempty words to ne.

Consider run descriptors

$$\sigma_i = (q_i, \{(1, (q_3, \mathsf{pr}))\}, \emptyset, \mathsf{np}) \qquad \text{for } i \in \{5, 6, 7\}, \text{ and}$$
$$\sigma_{4,f} = (q_4, \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, \emptyset, f) \qquad \text{for } f \in \{\mathsf{np}, \mathsf{pr}\}.$$

They describe runs that are compositions of a 2-return, and of a run that starts in the state q_3 and reads some $\sharp$ symbols (is productive). For $i \in \{5, 6, 7\}$, the 2-return should not read anything. Thus, in this case $D_i = (\mathsf{pop}, \gamma, q_i, (q_3, \mathsf{pr}))$ is a derivation tree for $\gamma \vdash \sigma_i$. On the other hand, for $i = 4$ the 2-return should read a nonempty word, which should contain some $\sharp$ symbol when $f = \mathsf{pr}$. We can derive $\gamma \vdash \sigma_{4,\mathsf{pr}}$ by $D_{4,\sharp} = (\mathsf{read}, q_4, D_7)$, and $\gamma \vdash \sigma_{4,\mathsf{np}}$ by $D_{4,a} = (\mathsf{read}, q_4, D_5)$, as well as by $D_{4,b} = (\mathsf{read}, q_4, D_6)$.

For $f \in \{\mathsf{np}, \mathsf{pr}\}$, denote $\sigma_{4,f}^1 = \mathsf{red}^1(\sigma_{4,f})$, that is, $\sigma_{4,f}^1 = (q_4, \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, f)$. The next run descriptors that we consider are

$$\sigma_{3,f} = (q_3, \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, \{(1, \sigma_{4,f}^1)\}, \mathsf{np}) \qquad \text{for } f \in \{\mathsf{np}, \mathsf{pr}\}.$$

They describe runs whose some suffix (that starts before reading anything, and after a 1-return) is described by $\sigma_{4,f}^1$, and some other suffix (that starts after reading a nonempty word, and after a 2-return) is described by (q_3, pr). We can derive $\gamma \vdash \sigma_{3,f}$ by $D_{3,f} = (\mathsf{pop}, \gamma, q_3, \sigma_{4,f}^1)$, for $f \in \{\mathsf{np}, \mathsf{pr}\}$. Notice that the derivation trees $D_{3,f}$ (and the former ones as well) specify precisely only a prefix of a run; more precisely, the fragment of the run that corresponds to the topmost 0-stack, before using an assumption. It is not checked that the assumption can be fulfilled by any stack that would be placed below the topmost 0-stack. We can equally well have a derivation tree like $(\mathsf{pop}, \gamma, q_3, \xi_4)$ with $\xi_4 = (q_4, \{(1, (q_1, \mathsf{pr}))\}, \mathsf{pr})$, which derives $\gamma \vdash (q_3, \{(1, (q_1, \mathsf{pr}))\}, \{(1, \xi_4)\}, \mathsf{np})$, although it is impossible to deliver ξ_4 by any 1-stack (because, e.g., there is no 2-return that ends in the state q_1).

Next, for $f_1, f_2 \in \{\mathsf{np}, \mathsf{pr}\}$ consider run descriptors

$$\sigma_{2,f_1,f_2} = (q_2, \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, \{(1, \sigma_{4,f_1}^1), (\mathsf{ne}, \sigma_{4,f_2}^1)\}, f_3), \quad \text{where } f_3 = \mathsf{pr} \Leftrightarrow f_1 = f_2 = \mathsf{pr}.$$

We can derive $\gamma \vdash \sigma_{2,f_1,f_2}$ by $D_{2,f_1,f_2} = (\mathsf{push}, \gamma, q_2, D_{3,f_1}, \{D_{3,f_2}\})$. The derivation tree has to specify fragments of the run for which the topmost 0-stack is responsible. When applied to a stack $s^2 : s^1 : s^0$, the push_γ^2 operation (resulting in $(s^2 : s^1 : s^0) : \mathsf{p}_{+1}(s^1 : s^0)$) splits the topmost 0-stack s^0 into two copies; D_{2,f_1,f_2} says that, with the new copy of s^0, we should continue according to the derivation tree D_{3,f_1}. The run descriptor $\mathsf{rd}(D_{3,f_1})$ (i.e., σ_{3,f_1}) says that the 1-stack s^1 will be used with run descriptor σ_{4,f_1}^1 (before reading anything), and that the 2-stack $s^2 : s^1 : s^0$ will be used with run descriptor (q_3, pr) (after reading a nonempty word). The definition of our derivation tree involves a composer

$$(\{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, \{(\mathsf{ne}, \sigma_{4,f_2}^1)\}, \{(\mathsf{ne}, \sigma_{3,f_2})\}; \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}; \mathsf{np}).$$

It says that in order to provide (q_3, pr) by the 2-stack $s^2 : s^1 : s^0$, we have to provide σ_{3,f_2} by s^0, and σ_{4,f_2}^1 by s^1, and (q_3, pr) by s^2. Because (q_3, pr) for $s^2 : s^1 : s^0$ will be used after reading a nonempty word (the pair in the output set of the composer is $(\mathsf{ne}, (q_3, \mathsf{pr}))$), also the run descriptors for s^0, s^1, s^2 will be used after reading a nonempty word. The derivation tree D_{2,f_1,f_2} says also how the lower copy of s^0 provides σ_{3,f_2}; this is described by the derivation tree D_{3,f_2}. In the order-1 assumption set of σ_{2,f_1,f_2} we put both assumptions, $(1, \sigma_{4,f_1}^1)$ and $(\mathsf{ne}, \sigma_{4,f_2}^1)$; the former needs to be provided by the new copy of s^1, and the latter by the lower copy of s^1. Notice that $\sigma_{2,\mathsf{pr},\mathsf{pr}}$ is productive, because for $f_1 = f_2 = \mathsf{pr}$ the same (productive) run descriptor $\sigma_{4,\mathsf{pr}}^1$ is used for s^1 twice.

Consider also run descriptors

$$\sigma_{1,f} = (q_1, \{(\mathsf{ne}, (q_3, \mathsf{pr}))\}, \emptyset, f) \qquad\qquad \text{for } f \in \{\mathsf{np}, \mathsf{pr}\}.$$

We can derive $\gamma \vdash \sigma_{1,\mathsf{np}}$ by $D_{1,a,a} = (\mathsf{push}, \gamma, q_1, D_{2,\mathsf{np},\mathsf{np}}, \{D_{4,a}\})$. After performing push^1_γ from $s^2 : s^1 : s^0$, we obtain $s^2 : (s^1 : s^0) : \mathsf{p}_{+1}(s^0)$. The derivation tree $D_{2,\mathsf{np},\mathsf{np}}$ specifies the behavior of the new topmost 0-stack. The composer

$$(\emptyset, \{(1, \sigma_{4,\mathsf{np}}), (\mathsf{ne}, \sigma_{4,\mathsf{np}})\}; \{(1, \sigma^1_{4,\mathsf{np}}), (\mathsf{ne}, \sigma^1_{4,\mathsf{np}})\}; \mathsf{np})$$

specifies how the assumptions of $\mathsf{rd}(D_{2,\mathsf{np},\mathsf{np}})$ (i.e., $\sigma_{2,\mathsf{np},\mathsf{np}}$) can be realized by the 1-stack $s^1 : s^0$. It says that (the lower copy of) s^0 should provide $\sigma_{4,\mathsf{np}}$, and the derivation tree $D_{4,a}$ specifies how it is provided. The whole $D_{1,a,a}$ corresponds to a run that reads the letter a twice. Similarly, $\gamma \vdash \sigma_{1,\mathsf{np}}$ can be derived by $D_{1,b,b} = (\mathsf{push}, \gamma, q_1, D_{2,\mathsf{np},\mathsf{np}}, \{D_{4,b}\})$, which corresponds to a run that reads the letter b twice. On the other hand, there is no derivation tree corresponding to a run that first reads the letter a, and then the letter b (or vice versa). The reason is that we have to provide exactly one realization of $\sigma_{4,\mathsf{np}}$ for s^0; this can be either $D_{4,a}$, or $D_{4,b}$, but not both. The statement $\gamma \vdash \sigma_{1,\mathsf{pr}}$ can be derived by $D_{1,\sharp,\sharp} = (\mathsf{push}, \gamma, q_1, D_{2,\mathsf{pr},\mathsf{pr}}, \{D_{4,\sharp}\})$, by $D_{1,\sharp,a} = (\mathsf{push}, \gamma, q_1, D_{2,\mathsf{pr},\mathsf{np}}, \{D_{4,\sharp}, D_{4,a}\})$, by $D_{1,a,\sharp} = (\mathsf{push}, \gamma, q_1, D_{2,\mathsf{np},\mathsf{pr}}, \{D_{4,\sharp}, D_{4,a}\})$, and by similar derivation trees using the letter b instead of a. Notice that here we have sets with two derivation trees, $D_{4,\sharp}$ and $D_{4,a}$; this is because one provides a realization for $\sigma_{4,\mathsf{pr}}$, and the other for $\sigma_{4,\mathsf{np}}$.

Finally, consider run descriptors

$$\tau_7 = (q_7, \emptyset, \emptyset, \mathsf{np}),$$
$$\tau_4 = (q_4, \emptyset, \emptyset, \mathsf{pr}), \qquad\qquad \text{and}$$
$$\tau_3 = (q_3, \emptyset, \{(1, \tau^1_4)\}, \mathsf{np}) \qquad\qquad \text{with } \tau^1_4 = \mathsf{red}^1(\tau_4) = (q_4, \emptyset, \mathsf{pr}).$$

Run descriptors τ_4 and τ_7 have empty assumption sets, hence they describe runs that never uncover stacks that are below the topmost 0-stack (in other words, these are runs whose every prefix is 0-upper). We can derive $\gamma \vdash \tau_7$ by $E_7 = (\mathsf{empty}, \gamma, q_7)$, which corresponds to the empty run; $\gamma \vdash \tau_4$ by $E_4 = (\mathsf{read}, q_4, E_7)$, which corresponds to the run that reads $\sharp$ and stops; and $\gamma \vdash \tau_3$ by $E_3 = (\mathsf{pop}, \gamma, q_3, \tau^1_4)$.

Annotated Stacks. A derivation tree provides an information about a part of a run R for which a particular 0-stack is responsible. In order to describe the whole R, we have to specify derivation trees for all 0-stacks in $R(0)$. To the end, we annotate stacks using sets of derivation trees.

An *annotated k-stack* is a positionless[3] 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 are 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 $\mathbf{s}^k$ to the Γ coordinate is denoted by $\mathsf{st}(\mathbf{s}^k)$.

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

$$\mathsf{type}((\gamma, \mathfrak{D})) = \{\mathsf{rd}(D) \colon D \in \mathfrak{D}\}, \qquad\qquad \mathsf{type}([\,]) = \emptyset,$$

---

[3]It turns out that while considering annotated k-stacks we do not need positions, so for notational convenience we assume that annotated stacks are positionless (i.e., their 0-stacks do not contain positions, conversely to non-annotated stacks).

$$\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^{k-1})\}.$$

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 that is below has to deliver annotations fulfilling these assumptions. Simultaneously, all annotations have to be useful: they cannot provide derivation trees for run descriptors that do not appear as assumptions of annotations assigned higher in the stack. To formalize this, we define below when an annotated k-stack $\mathbf{s}^k$ is *well-formed*.

Definition 7.13. 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 $\mathsf{type}(\mathbf{s}^k) = \bigcup\{\pi_2(\mathsf{ass}^k(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{s}^{k-1})\}$, and $|\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1})| = |\mathsf{type}(\mathbf{s}^{k-1})|$.

In the sequel, generally we only consider well-formed annotated stacks (except for some moments when we first define an annotated stack, and then we prove that it is well-formed). Notice that beside of the condition mentioned earlier (saying that $\mathsf{type}(\mathbf{s}^k)$ provides exactly assumptions for run descriptors in $\mathsf{type}(\mathbf{s}^{k-1})$), we also have the second condition, $|\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1})| = |\mathsf{type}(\mathbf{s}^{k-1})|$, saying that every run descriptor in $\mathsf{type}(\mathbf{s}^k : \mathbf{s}^{k-1})$ has exactly one realization by a composition of run descriptors from $\mathsf{type}(\mathbf{s}^k)$ and $\mathsf{type}(\mathbf{s}^{k-1})$. Both conditions have the same goal, which is also the same as for analogous conditions in the definition of a composer (Definition 7.8): we want to estimate the number of $\sharp$ symbols read by a run by looking at the number of productive run descriptors associated to 0-stacks in an annotated stacks. Two realizations of the same run descriptor, as well as realizations of a run descriptor not appearing in an assumption set, would bias these calculations.

The definition of types and well-formedness connects only the type of $\mathbf{s}^k : \mathbf{s}^{k-1}$ with the types of $\mathbf{s}^k$ and $\mathbf{s}^{k-1}$, but similar conditions can be written for a stack of the form $\mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^l$.

Proposition 7.14. *Let* $0 \leq l \leq k \leq n$, *and 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. Then,*

(T1) $\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}$.

Moreover, $\mathbf{s}$ *is well-formed if and only if*

(T2) $\mathsf{type}(\mathbf{s}^i) = \bigcup\{\pi_2(\mathsf{ass}^i(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}$ *for every* $i \in [l+1, k]$,

(T3) $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{s}^l)|$.

Proof. Induction on $k - l$. Suppose first that $k - l = 0$. In this case, Item (T1) holds because $\mathbf{s} = \mathbf{s}^l$ and because $\mathsf{red}^k(\sigma) = \sigma$ when $\sigma \in \mathcal{T}^k$. Moreover $\mathbf{s}$ is well-formed by assumption, Item (T2) is true because $[l+1, k] = \emptyset$ is empty, and Item (T3) is true because $\mathbf{s} = \mathbf{s}^l$; thus we have the equivalence.

Suppose now that $k - l \geq 1$, and denote $\mathbf{t} = \mathbf{s}^{k-1} : \mathbf{s}^{k-2} : \cdots : \mathbf{s}^l$. We then have $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$. We apply the induction assumption to $\mathbf{t} = \mathbf{s}^{k-1} : \mathbf{s}^{k-2} : \cdots : \mathbf{s}^l$. By Item (T1) of the induction assumption, $\mathsf{type}(\mathbf{t}) = \{\mathsf{red}^{k-1}(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}$, and by the definition of the type of $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$,

$$\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{t})\} = \{\mathsf{red}^k(\mathsf{red}^{k-1}(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}.$$

Recalling that $\mathsf{red}^k(\mathsf{red}^{k-1}(\sigma)) = \mathsf{red}^k(\sigma)$, we obtain Item (T1).

Suppose that $\mathbf{s}$ is well-formed. Then, in particular, $\mathbf{t}$ is well-formed, so Items (T2) and (T3) hold for the substack $\mathbf{t}$ by the induction assumption. Item (T2) from the induction

assumption is the same as our Item (T2) for $i \in [l+1, k-1]$. By well-formedness of $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$,

$$\mathsf{type}(\mathbf{s}^k) = \bigcup\{\pi_2(\mathsf{ass}^k(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{t})\} = \bigcup\{\pi_2(\mathsf{ass}^k(\mathsf{red}^{k-1}(\sigma))) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}\,.$$

Recalling that $\mathsf{ass}^k(\mathsf{red}^{k-1}(\sigma)) = \mathsf{ass}^k(\sigma)$ we obtain Item (T2) for $i = k$. Moreover, Item (T3) of the induction assumption says that $|\mathsf{type}(\mathbf{t})| = |\mathsf{type}(\mathbf{s}^l)|$, and by well-formedness of $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$ we have that $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{t})|$, thus $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{s}^l)|$ (Item (T3)).

Conversely, suppose that Items (T2) and (T3) hold for $\mathbf{s}$. Because $\mathsf{type}(\mathbf{s})$ is the image of $\mathsf{type}(\mathbf{t})$ under the function red^k, and $\mathsf{type}(\mathbf{t})$ is the image of $\mathsf{type}(\mathbf{s}^l)$ under the function red^{k-1}, we necessarily have $|\mathsf{type}(\mathbf{s})| \leq |\mathsf{type}(\mathbf{t})| \leq |\mathsf{type}(\mathbf{s}^l)|$, and thus $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{s}^l)|$ (Item (T3)) implies that $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{t})| = |\mathsf{type}(\mathbf{s}^l)|$; we thus have Item (T3) for $\mathbf{t}$. Moreover, Item (T2) for $\mathbf{t}$ is a direct consequence of this item for $\mathbf{s}$ (we only restrict the considered orders i to $[l+1, k-1]$). Thus, by the induction assumption, $\mathbf{t}$ is well-formed. Using Item (T2) for $i = k$, the equality $\mathsf{ass}^k(\sigma) = \mathsf{ass}^k(\mathsf{red}^{k-1}(\sigma))$, and Item (T1) for $\mathbf{t}$ we obtain that

$$\mathsf{type}(\mathbf{s}^k) = \bigcup\{\pi_2(\mathsf{ass}^k(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}$$
$$= \bigcup\{\pi_2(\mathsf{ass}^k(\mathsf{red}^{k-1}(\sigma))) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\} = \bigcup\{\pi_2(\mathsf{ass}^k(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{t})\}\,.$$

Together with the equality $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{t})|$ this implies that $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$ is well-formed. □

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 $(q, \mathsf{pos}_+(\mathsf{st}(\mathbf{s}^n)))$, where q is the state of the only run descriptor in $\mathsf{type}(\mathbf{s}^n)$.

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 remark that in the union we could also allow well-formed annotated stacks $\mathbf{s}^n$ that are not necessarily singular, but are such that $\mathsf{pos}_+(\mathsf{st}(\mathbf{s}^n))$ is the stack of c and the state of all run descriptors in $\mathsf{type}(\mathbf{s}^n)$ is the state of c. On the other hand, in general there does not exist a single well-formed annotated stack $\mathbf{s}^n$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s}^n)) = \mathsf{type}_{\mathcal{A},\varphi}(c)$. Namely, we can have a situation like in Example 7.9, where we cannot assign both σ_1 and σ_1' to the topmost 0-stack, as both of them result in the same run descriptor ξ_1 for the topmost 1-stack.

Actually, we see a direct connection between the well-formedness property and composers.

Proposition 7.15. *Let $0 \leq l \leq k \leq n$, 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 $\pi_2(\Psi^k) = \mathsf{type}(\mathbf{s})$.*

Proof. Suppose first that we have a composer $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^l; \Psi^k; f)$ such that

$$\pi_2(\Phi^i) = \mathsf{type}(\mathbf{s}^i) \qquad\qquad \text{for } i \in [k, l]\,. \qquad\qquad (7.1)$$

Notice that

$$\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\} = \{\mathsf{red}^k(\sigma) \colon \sigma \in \pi_2(\Phi^l)\} = \pi_2(\Psi^k)\,, \qquad (7.2)$$

where the consecutive equalities follow from Item (T1) of Proposition 7.14, from Equality (7.1), and from Condition (C2) of Definition 7.8, respectively. Moreover, for $i \in [l + 1, k]$,

$$\mathsf{type}(\mathbf{s}^i) = \pi_2(\Phi^i) = \bigcup \{\pi_2(\mathsf{ass}^i(\sigma)) \colon \sigma \in \pi_2(\Phi^l)\} = \bigcup \{\pi_2(\mathsf{ass}^i(\sigma)) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}, \quad (7.3)$$

where the equalities are consequences of Equality (7.1), of Condition (C1) of Definition 7.8, and of Equality (7.1) again. Simultaneously,

$$|\mathsf{type}(\mathbf{s})| = |\pi_2(\Psi^k)| = |\pi_2(\Phi^l)| = |\mathsf{type}(\mathbf{s}^l)|, \quad (7.4)$$

where the consecutive equalities follow from Equality (7.2), from Condition (C3) of Definition 7.8, and from Equality (7.1), respectively. Equalities (7.3) and (7.4) give Items (T2) and (T3) of Proposition 7.14, which implies that $\mathbf{s}$ is well-formed; together with Equality (7.2) this gives the thesis.

Conversely, suppose that $\mathbf{s}$ is well-formed and

$$\pi_2(\Psi^k) = \mathsf{type}(\mathbf{s}). \quad (7.5)$$

We define

$$\Phi^l = \{(m, \sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l) \wedge (m, \mathsf{red}^k(\sigma)) \in \Psi^k\}, \qquad \text{and} \qquad (7.6)$$

$$\Phi^i = \bigcup \{m \circ \mathsf{ass}^i(\sigma) \colon (m, \sigma) \in \Phi^l\} \qquad \text{for } i \in [l + 1, k]. \quad (7.7)$$

By Equality (7.5) and by Item (T1) of Proposition 7.14 we have that

$$\pi_2(\Psi^k) = \mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}. \quad (7.8)$$

This means that for every $\sigma \in \mathsf{type}(\mathbf{s}^l)$ there is some m such that $(m, \mathsf{red}^k(\sigma)) \in \Psi^k$. In the light of Equality (7.6) this implies that

$$\pi_2(\Phi^l) = \mathsf{type}(\mathbf{s}^l). \quad (7.9)$$

On the other hand, by Equality (7.7) and by Item (T2) of Proposition 7.14, for $i \in [l + 1, k]$, we have that

$$\pi_2(\Phi^i) = \bigcup \{\pi_2(\mathsf{ass}^i(\sigma)) \colon \sigma \in \pi_2(\Phi^l)\} = \mathsf{type}(\mathbf{s}^i). \quad (7.10)$$

Let us now check particular conditions of Definition 7.8. Condition (C1) is immediate from Equality (7.7). By Equality (7.6), for every pair $(m, \sigma) \in \Phi^l$, the pair $(m, \mathsf{red}^k(\sigma))$ is in Ψ^k. Conversely, by Equality (7.8), every pair in Ψ^k is of the form $(m, \mathsf{red}^k(\sigma))$ with $\sigma \in \mathsf{type}(\mathbf{s}^l)$, hence $(m, \sigma) \in \Phi^l$ by Equality (7.6). This implies Condition (C2). Using consecutively Equality (7.5), Item (T3) of Proposition 7.14, and Equality (7.9), we obtain Condition (C3):

$$|\pi_2(\Psi^k)| = |\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{s}^l)| = |\pi_2(\Phi^l)|.$$

Condition (C4) always holds for some $f \in \{\mathsf{np}, \mathsf{pr}\}$. Thus $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^l; \Psi^k; f)$ is a composer, which together with Equalities (7.9) and (7.10) gives the thesis. $\qquad \square$

Example 7.16. This is a continuation of Example 7.12. With the derivation trees considered there,

$$\mathbf{s}_1 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{1,\sharp,a}\})]] \qquad \text{and}$$

$$\mathbf{s}_2 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}, D_{4,a}\}), (\gamma, \{D_{2,\mathsf{pr},\mathsf{np}}\})]]$$

are well-formed (singular) annotated 2-stacks. On the other hand, the following annotated 2-stacks are not well-formed:

$$\mathbf{s}_3 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \{E_1\}), (\gamma, \{D_{1,\sharp,a}\})]] \,,$$

$$\mathbf{s}_4 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}, D_{4,a}, D_{4,b}\}), (\gamma, \{D_{2,\mathsf{pr},\mathsf{np}}\})]] \,, \qquad \text{and}$$

$$\mathbf{s}_5 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}\}), (\gamma, \{D_{2,\mathsf{pr},\mathsf{np}}\})]] \,.$$

In $\mathbf{s}_3$, we provide a spare derivation tree E_1, not needed by the derivation trees assigned above. In $\mathbf{s}_4$, two derivation trees, $D_{4,a}$ and $D_{4,b}$, provide the same run descriptor. Finally, in $\mathbf{s}_5$, we are missing a derivation tree that would provide $\sigma_{4,\mathsf{np}}$.

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 $\bigcup_{\sigma \in \widetilde{\Phi}} \pi_2(\mathsf{ass}^k(\sigma))$.

Plan for the Remaining Part of the Section. We have already defined types of configurations, as needed for Theorem 7.3. Type of a configuration c is defined via existence of annotated stacks $\mathbf{s}$ such that $\mathsf{conf}(\mathbf{s}) = c$. Theorem 7.3 says that if two configurations c, d have the same type, and we have a run starting in c, then a similar run starts d. Roughly, the strategy of the proof is as follows. First, basing on the run starting in c, we construct an annotated stack $\mathbf{s}$ with $\mathsf{conf}(\mathbf{s}) = c$, corresponding to this run. More precisely, we do not process the whole run in this way, but rather its particular fragments. Then, because the types of c and d equal, there exists an annotated stack $\mathbf{t}$ with $\mathsf{conf}(\mathbf{t}) = d$, and with $\mathsf{rd}(\mathbf{t}) = \mathsf{rd}(\mathbf{s})$. Having $\mathbf{t}$ we proceed in the opposite direction: basing on the annotated stack $\mathbf{t}$ we construct a run starting in d, satisfying the thesis of the theorem.

Recall that an annotated stack is, roughly, a description of a run. The run described by an annotated stack is called *annotated run*, and is defined in Subsection 7.2. In Lemma 7.33 (located in Subsection 7.4) we prove that annotated runs have the expected form, that is, that using an assumption of a run descriptor corresponds to performing a return. In Subsection 7.5 we present the opposite direction: how to construct an annotated stack basing on a run. The proof of Theorem 7.3 is finalized in Subsection 7.6.

Simultaneously, we prepare ourselves for a proof of Theorem 7.4, which additionally talks about the number of $\sharp$ symbols read by a run. We thus need to estimate the number of $\sharp$ symbols read by an annotated run. To this end, in Subsection 7.2 we define two numbers corresponding to an annotated stack $\mathbf{s}$, namely $\mathsf{low}(\mathbf{s})$ and $\mathsf{high}(\mathbf{s})$. They provide a lower bound and, respectively, an upper bound for the number of $\sharp$ symbols read by an annotated run starting in $\mathbf{s}$, as showed in Lemma 7.22. (We also define there a third number, $\mathsf{len}(\mathbf{s})$, and we prove that it gives an upper bound for the length of an annotated run starting in $\mathbf{s}$. Its role is auxiliary: it is only needed for showing that the constructed annotated run is finite. Actually, this is needed already while proving Theorem 7.3.) The essential property is that $\mathsf{high}(\mathbf{s})$ can be bounded by a function of $\mathsf{low}(\mathbf{s})$, as shown in Proposition 7.32, contained in Subsection 7.3. Thus $\mathsf{low}(\mathbf{s})$ itself estimates the number of $\sharp$ symbols read by an annotated

run starting in $\mathbf{s}$. With such a function low we can define sequence-equivalence, as needed in Theorem 7.4. Namely, having a sequence of configurations (all of the same type), for every run descriptor σ it is enough to know one think: whether there is a sequence of annotated stacks corresponding to these configurations (and to the run descriptor σ), and such that the values of low for these annotated stacks are bounded. If this is the case, using these annotated stacks we can reproduce runs that read a bounded number of $\sharp$ symbol. If not, then a sequence of annotated stacks corresponding to the considered configurations also exists, but the values of low for these annotated stacks are unbounded, and in effect the reproduced runs read an unbounded number of $\sharp$ symbols. A proof of Theorem 7.4, following these ideas, is given in Subsection 7.7.

7.2. **Annotated Runs.** In this subsection we describe how, having an annotated stack, we can reproduce a run. This is formalized in the notion of annotated runs. We also relate the number of $\sharp$ symbols read by a run with the number of productive run descriptors in annotations of the starting configuration. Later, in Subsection 7.5, we do the converse: we show how to construct an annotated stack basing on a run. Recall that when a well-formed annotated stack is singular, then its topmost 0-stack is singular as well (cf. Item (T3) of Proposition 7.14).

Definition 7.17. 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 $D = (\mathsf{empty}, \gamma, p)$, then $\mathbf{s}$ has no successor.
(2) If $D = (\mathsf{read}, p, D')$, then the successor is $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\gamma, \{D'\})$.
(3) If $D = (\mathsf{pop}, \gamma, p, \tau^k)$, then the successor is $\mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$.
(4) Suppose that $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$. Let α, k, Ψ^i, Φ^i be as in Definition 7.11(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 $\mathbf{t}^k = \mathbf{s}^k\!\restriction_{\pi_2(\Phi^k)} : \mathbf{s}^{k-1}\!\restriction_{\pi_2(\Phi^{k-1})} : \cdots : \mathbf{s}^1\!\restriction_{\pi_2(\Phi^1)} : (\gamma, \mathfrak{D})$.

We notice that in Case (4) for $i \in [1, k]$ we have that $\pi_2(\Phi^i) \subseteq \mathsf{type}(\mathbf{s}^i)$, and for $i \in [1, k-1]$ we have that $\pi_2(\Psi^i) \subseteq \mathsf{type}(\mathbf{s}^i)$, and thus the restrictions are legal. Indeed, from Proposition 7.14 applied to $\mathbf{s}$ we know that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D)))$ for all $i \in [1, n]$; moreover, by Definition 7.11(4), $\mathsf{ass}^i(\sigma) = \Psi^i \cup \Phi^i$ for $i \in [1, k-1]$, and $\mathsf{ass}^k(\sigma) = \Phi^k$.

Proposition 7.18. *Let $\mathbf{t}$ be the successor of a well-formed singular annotated n-stack $\mathbf{s}$. Then $\mathbf{t}$ is singular, well-formed, and $\mathsf{conf}(\mathbf{t})$ is a successor of $\mathsf{conf}(\mathbf{s})$ (in the considered automaton).*

Proof. Let $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^0$, and $\mathbf{s}^0 = (\gamma, \{D\})$, and $\sigma = \mathsf{rd}(D)$. All $\mathbf{s}^i$ are well-formed, because $\mathbf{s}$ is well-formed. Moreover, by Proposition 7.14, $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\sigma))$ for all $i \in [1, n]$.

We have several cases depending on the shape of $\mathbf{s}$. We cannot have $D = (\mathsf{empty}, \gamma, p)$, as then $\mathbf{s}$ has no successor.

Suppose that $D = (\mathsf{read}, p, D')$. Recall from Definition 7.11 that $\pi_2(\mathsf{ass}^i(\mathsf{rd}(D'))) = \pi_2(\mathsf{ass}^i(\sigma))$ for all $i \in [1, n]$. By Proposition 7.14, $\mathbf{t}$ is singular (its type is a singleton $\{\mathsf{red}^n(\mathsf{rd}(D'))\}$) and well-formed. Moreover, $\delta(\gamma, p) = \mathsf{read}(\vec{q})$, where $\vec{q}(a)$ equals the state of $\mathsf{rd}(D')$ for some $a \in A$. The transition from $\mathsf{conf}(\mathbf{s})$ reading this a leads to $\mathsf{conf}(\mathbf{t})$.

Next, suppose that $D = (\mathsf{pop}, \gamma, p, \tau^k)$. By Definition 7.11 we have that $\pi_2(\mathsf{ass}^k(\sigma)) = \{\tau^k\}$ (hence $\mathsf{type}(\mathbf{s}^k) = \{\tau^k\}$) and $\mathsf{ass}^i(\sigma) = \mathsf{ass}^i(\tau^k)$ (hence $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\tau^k))$) for $i \in [k+1, n]$. Proposition 7.14 applied to $\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$ implies that it is singular and well-formed. Moreover, $\delta(\gamma, p) = (q, \mathsf{pop}^k)$, where q is the state of τ^k, so the transition from $\mathsf{conf}(\mathbf{s})$ leads to $\mathsf{conf}(\mathbf{t})$.

Finally, suppose that $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$. Let α, k, Ψ^i, Φ^i, f be as in Definition 7.11(4), and $\mathbf{t}^k$ as in Definition 7.17(4). Obviously, $\mathsf{type}(\mathbf{s}^i|_{\pi_2(\Phi^i)}) = \pi_2(\Phi^i)$, for all $i \in [1, k]$; moreover, $\mathsf{type}((\gamma, \mathfrak{D})) = \{\mathsf{rd}(E): E \in \mathfrak{D}\} = \pi_2(\Phi^0)$, by Definition 7.11(4). Furthermore, again by this definition, $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$ is a composer. It follows from Proposition 7.15 that $\mathbf{t}^k$ is well-formed, and $\mathsf{type}(\mathbf{t}^k) = \pi_2(\Psi^k)$ (recall that $\mathbf{t}^k = \mathbf{s}^k|_{\pi_2(\Phi^k)} : \mathbf{s}^{k-1}|_{\pi_2(\Phi^{k-1})} : \cdots : \mathbf{s}^1|_{\pi_2(\Phi^1)} : (\gamma, \mathfrak{D}))$. By Definition 7.11(4), for all $i \in [1, n]$ we have that $\Psi^i = \mathsf{ass}^i(\mathsf{rd}(D'))$. Moreover,

- for $i \in [k+1, n]$, we have $\mathsf{ass}^i(\sigma) = \Psi^i$ by Definition 7.11(4), so $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\sigma)) = \pi_2(\Psi^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D')))$;
- $\mathsf{type}(\mathbf{t}^k) = \pi_2(\Psi^k) = \pi_2(\mathsf{ass}^k(\mathsf{rd}(D')))$;
- for $i \in [1, k-1]$, we have that $\mathsf{type}(\mathbf{s}^i|_{\pi_2(\Psi^i)}) = \pi_2(\Psi^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D')))$.

In effect, $\mathbf{t}$, which is a composition of these stacks, and of $(\alpha, \{D'\})$ is singular and well-formed by Proposition 7.14. Additionally, $\delta(\gamma, p) = (q, \mathsf{push}_\alpha^k)$, where q is the state of $\mathsf{rd}(D')$, so the transition from $\mathsf{conf}(\mathbf{s})$ leads to $\mathsf{conf}(\mathbf{t})$. $\qquad\square$

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 denoted $\mathsf{st}(\mathbf{R})$.

Notice that an annotated stack $\mathbf{s}$ may have less successors than $\mathsf{conf}(\mathbf{s})$. Indeed, in the case of $D = (\mathsf{empty}, \gamma, p)$ there are no successors of $\mathbf{s}$, but $\mathsf{conf}(\mathbf{s})$ may have successors. Similarly, in the case of $D = (\mathsf{read}, p, D')$ there is exactly one successor of $\mathbf{s}$ (the state in $\mathsf{rd}(D')$ determines which letter should be read), while in a run from $\mathsf{conf}(\mathbf{s})$ we can read any letter.

Example 7.19. Recall the 2-DPDA $\mathcal{A}_1$ from Example 7.12, and the annotated stack $\mathbf{s}_1 = [[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{1,\sharp,a}\})]]$ from Example 7.16. The successors of $\mathbf{s}_1$ are, consecutively,

$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}, D_{4,a}\}), (\gamma, \{D_{2,\mathsf{pr},\mathsf{np}}\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,a}\}), (\gamma, \{D_{3,\mathsf{np}}\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}\}), (\gamma, \{D_{3,\mathsf{pr}}\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,a}\}), (\gamma, \{D_{3,\mathsf{np}}\})], [(\gamma, \emptyset), (\gamma, \{D_{4,\sharp}\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,a}\}), (\gamma, \{D_{3,\mathsf{np}}\})], [(\gamma, \emptyset), (\gamma, \{D_7\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,a}\}), (\gamma, \{D_{3,\mathsf{np}}\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_{4,a}\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})], [(\gamma, \emptyset), (\gamma, \{D_5\})]],$$
$$[[(\gamma, \{E_4\}), (\gamma, \{E_3\})]],$$
$$[[(\gamma, \{E_4\})]],$$
$$[[(\gamma, \{E_7\})]];$$

the last of them has no more successors. In the transition between the first and the second line, $D_{2,\mathsf{pr},\mathsf{np}}$ says that the new topmost 0-stack should be annotated by $D_{3,\mathsf{pr}}$, and that the

previously topmost 0-stack should be annotated by $D_{3,\mathsf{np}}$. Because $\mathsf{ass}^1(\mathsf{rd}(D_{3,\mathsf{pr}})) = \sigma^1_{4,\mathsf{pr}}$ and $\mathsf{ass}^1(\mathsf{rd}(D_{3,\mathsf{np}})) = \sigma^1_{4,\mathsf{np}}$, we know that $D_{4,\sharp}$ should be taken to the topmost 1-stack (we have that $\mathsf{red}^1(\mathsf{rd}(D_{4,\sharp})) = \sigma^1_{4,\mathsf{pr}}$), and that $D_{4,a}$ should be left in the second topmost 1-stack (we have that $\mathsf{red}^1(\mathsf{rd}(D_{4,a})) = \sigma^1_{4,\mathsf{np}}$).

We can see that not every run is of the form $\mathsf{st}(\mathbf{R})$ for some annotated run $\mathbf{R}$. For example, this is the case for the run that starts in $(q_1, \mathsf{pos}_+([[\gamma, \gamma], [\gamma, \gamma]]))$ and reads a, then b, and then $\sharp$. Indeed, in order to obtain such a run as an annotated run, to the topmost 0-stack we have to assign a hypothetical derivation tree $D_{1,a,b}$, saying that we should first read a and then b, but there is no such derivation tree (as already explained in Example 7.12). Another run that is not of the form $\mathsf{st}(\mathbf{R})$ for any annotated run $\mathbf{R}$ is the run that starts in $(q_2, \mathsf{pos}_+([[\gamma, \gamma], [\gamma, \gamma, \gamma]]))$ and reads a, then b, and then $\sharp$. This time the problem is that to the second topmost 0-stack we cannot assign simultaneously $D_{4,a}$ and $D_{4,b}$, as they both have the same run descriptor. More generally, the push in Case (4) of Definition 7.17 leaves the same annotations in the original substack as in the copied substack, up to a restriction, which causes that fragments of the annotated run corresponding to these substacks have to be the same.

A priori there might exist an infinite annotated run. But, as we see below, this is impossible: always after some number of steps we reach an annotated stack with no successors (Case (1) of Definition 7.17). Moreover, we show that the number of $\sharp$ symbols read by the run starting in an annotated stack $\mathbf{s}$ can be estimated by the number of productive run descriptors in the annotations of $\mathbf{s}$. To this end, 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 give a lower and an upper bound on the number of $\sharp$ symbols read by our run, and the last one gives an upper bound on the length of the run.

Definition 7.20. For positive integers $m_1, \ldots, m_k$ we define $\mathsf{pow}(m_1, \ldots, m_k)$ by induction on k:

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

Notice that, in particular, $\mathsf{pow}(m_1) = m_1$ and $\mathsf{pow}(m_1, m_2) = (1 + m_1)^{m_2} - 1$.

Definition 7.21. 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

$$\mathsf{low}(\mathbf{s}) = |\mathsf{type}(\mathbf{s}) \cap \mathcal{T}_{\mathsf{pr}}|,$$

$$\mathsf{high}(\mathbf{s}) = \prod_{D \in \mathfrak{D} : \ \mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}} C_{\mathsf{depth}(D)}, \qquad\qquad \text{and}$$

$$\mathsf{len}(\mathbf{s}) = \prod_{D \in \mathfrak{D}} C_{\mathsf{depth}(D)},$$

where C_z is defined inductively:

$$C_0 = 2, \qquad\qquad \text{and} \qquad\qquad C_{z+1} = (2|\mathcal{T}^0|)^n \cdot (C_z)^{|\mathcal{T}^0|+1}.$$

- We take $\mathsf{low}([\,]) = 0$ and $\mathsf{high}([\,]) = \mathsf{len}([\,]) = 1$.

- If $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$, we take

$$\mathsf{low}(\mathbf{s}) = \sum_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \left(\mathsf{low}(\mathbf{s}^k \restriction_{\pi_2(\mathsf{ass}^k(\sigma))}) + \mathsf{low}(\mathbf{s}^{k-1} \restriction_{\{\sigma\}}) \right),$$

$$\mathsf{high}(\mathbf{s}) = \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}\left(\mathsf{high}(\mathbf{s}^k \restriction_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{high}(\mathbf{s}^{k-1} \restriction_{\{\sigma\}}) \right), \qquad \text{and}$$

$$\mathsf{len}(\mathbf{s}) = \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}\left(\mathsf{len}(\mathbf{s}^k \restriction_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{len}(\mathbf{s}^{k-1} \restriction_{\{\sigma\}}) \right).$$

The three numbers are interesting for us, because of the following lemma. Recall that, for a run R, by $\sharp(R)$ we denote the number of $\sharp$ symbols read by R.

Lemma 7.22. *If* $\mathbf{R}$ *is an annotated run,*

$$\mathsf{low}(\mathbf{R}(0)) \leq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}(\mathbf{R}(|\mathbf{R}|)),$$
$$\mathsf{high}(\mathbf{R}(0)) \geq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(|\mathbf{R}|)), \qquad \text{and}$$
$$\mathsf{len}(\mathbf{R}(0)) \geq |\mathbf{R}| + \mathsf{len}(\mathbf{R}(|\mathbf{R}|)).$$

This is one of key lemmas of this section. We now give some examples and intuitions staying behind this lemma, and behind the definitions of low, high, and len; after that, we prove this lemma.

We see that the last inequality of Lemma 7.22 bounds the length of an annotated run $\mathbf{R}$ by $\mathsf{len}(\mathbf{R}(0))$, that is, by a function of the annotated stack $\mathbf{R}(0)$, in which the annotated run starts. Similarly, the second inequality bounds the number of $\sharp$ symbols read by $\mathbf{R}$, by another function of $\mathbf{R}(0)$, namely by $\mathsf{high}(\mathbf{R}(0))$. The additional components added on the right of these inequalities only strengthen them. Conversely, the role of the first inequality is to give a lower bound for the number of $\sharp$ symbols read by $\mathbf{R}$. If $\mathbf{R}$ is maximal (i.e., cannot be prolonged), then the topmost 0-stack of $\mathbf{R}(\mathbf{R})$ is annotated by a derivation tree of the form $(\mathsf{empty}, \gamma, q)$, and all other 0-stacks are annotated by empty sets. In effect $\mathsf{low}(\mathbf{R}(|\mathbf{R}|)) = 0$, and we simply obtain that $\mathsf{low}(\mathbf{R}(0)) \leq \sharp(\mathsf{st}(\mathbf{R}))$. But for an arbitrary run $\mathbf{R}$, which may end prematurely, even before reading any $\sharp$ symbol, we have to add $\mathsf{low}(\mathbf{R}(|\mathbf{R}|))$ on the right side of the inequality.

Roughly speaking, $\mathsf{low}(\mathbf{R}(0))$ counts the number of productive run descriptors in the annotations of $\mathbf{R}(0)$. The intuition is that every productive run descriptor is responsible for increasing the number of $\sharp$ symbols read, so the first inequality of Lemma 7.22 should hold with such a definition of low.

We see that the function high takes into account the same productive run descriptors as low, but instead of sums we use products and the pow function. Indeed, it is shown in Proposition 7.27 that if all run descriptors in the annotations of a substack $\mathbf{s}^{k-1}$ are nonproductive, then $\mathsf{high}(\mathbf{s}^{k-1}) = 1$ (and $\mathsf{low}(\mathbf{s}^{k-1}) = 0$). Suppose that $\mathbf{s}^{k-1}$ is singular, that is, its type is a singleton $\{\sigma\}$. If $\mathbf{s}^k : \mathbf{s}^{k-1}$ is well-formed, we have $\mathsf{type}(\mathbf{s}^k) = \pi_2(\mathsf{ass}^k(\sigma))$. Observe that $\mathsf{pow}(x, 1) = (1 + x)^1 - 1 = x$ for every x, and thus

$$\mathsf{high}(\mathbf{s}^k : \mathbf{s}^{k-1}) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^k \restriction_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{high}(\mathbf{s}^{k-1} \restriction_{\{\sigma\}}))$$
$$= \mathsf{pow}(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1})) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^k), 1) = \mathsf{high}(\mathbf{s}^k).$$

This is similar to the behavior of the low function, as in such a case we also have

$$\mathsf{low}(\mathbf{s}^k : \mathbf{s}^{k-1}) = \mathsf{low}(\mathbf{s}^k{\restriction}_{\pi_2(\mathsf{ass}^k(\sigma))}) + \mathsf{low}(\mathbf{s}^{k-1}{\restriction}_{\{\sigma\}})$$
$$= \mathsf{low}(\mathbf{s}^k) + \mathsf{low}(\mathbf{s}^{k-1}) = \mathsf{low}(\mathbf{s}^k) + 0 = \mathsf{low}(\mathbf{s}^k)\,.$$

Technical details of the definition of high were chosen so that it is possible to perform a proof, but two facts are important here. First, nonproductive run descriptors cannot be responsible for increasing the number of $\sharp$ symbols read, and thus (in order to obtain the second inequality of Lemma 7.22) it is enough to have a function high that takes into account only productive run descriptors (i.e., ignores nonproductive run descriptors). Second, because $\mathsf{high}(\mathbf{R}(0))$ and $\mathsf{low}(\mathbf{R}(0))$ are counting the same productive run descriptors, only in a different way, the two values are related. Namely, $\mathsf{high}(\mathbf{R}(0))$ can be bounded by a function of $\mathsf{low}(\mathbf{R}(0))$. This essential property is shown in Proposition 7.32, in the next subsection.

The function len is defined very similarly to high, but it takes into account all run descriptors, not only productive ones. Thus, roughly, it depends on the size of the annotated stack.

Example 7.23. We continue the previous examples, concerning the 2-DPDA $\mathcal{A}_1$ from Example 7.12. Consider the annotated stack $\mathbf{s}_1 = [[\mathbf{t}_4, \mathbf{t}_3], [\mathbf{t}_2, \mathbf{t}_1]]$, where

$$\mathbf{t}_1 = (\gamma, \{D_{1,\sharp,a}\})\,, \qquad\qquad \mathbf{t}_3 = (\gamma, \{E_3\})\,,$$
$$\mathbf{t}_2 = (\gamma, \emptyset)\,, \qquad\qquad \mathbf{t}_4 = (\gamma, \{E_4\})\,.$$

We have that

$$\mathsf{type}(\mathbf{t}_1) = \{\mathsf{rd}(D_{1,\sharp,a})\} = \{\sigma_{1,\mathsf{pr}}\}\,, \qquad \pi_2(\mathsf{ass}^1(\sigma_{1,\mathsf{pr}})) = \emptyset\,,$$
$$\mathsf{type}(\mathbf{t}_2) = \emptyset\,,$$
$$\mathsf{type}(\mathbf{t}_3) = \{\mathsf{rd}(E_3)\} = \{\tau_3\}\,, \qquad \pi_2(\mathsf{ass}^1(\tau_3)) = \{\tau_4^1\}\,,$$
$$\mathsf{type}(\mathbf{t}_4) = \{\mathsf{rd}(E_4)\} = \{\tau_4\}\,. \qquad \pi_2(\mathsf{ass}^1(\tau_4)) = \emptyset\,.$$

In effect

$$\mathsf{type}([\mathbf{t}_2, \mathbf{t}_1]) = \{\mathsf{red}^1(\sigma)\colon \sigma \in \mathsf{type}(\mathbf{t}_1)\} = \{\mathsf{red}^1(\sigma_{1,\mathsf{pr}})\}\,, \quad \pi_2(\mathsf{ass}^2(\mathsf{red}^1(\sigma_{1,\mathsf{pr}}))) = \{(q_3, \mathsf{pr})\}\,,$$
$$\mathsf{type}([\mathbf{t}_2]) = \{\mathsf{red}^1(\sigma)\colon \sigma \in \mathsf{type}(\mathbf{t}_2)\} = \emptyset\,,$$
$$\mathsf{type}([\mathbf{t}_4, \mathbf{t}_3]) = \{\mathsf{red}^1(\sigma)\colon \sigma \in \mathsf{type}(\mathbf{t}_3)\} = \{\mathsf{red}^1(\tau_3)\}\,, \quad \pi_2(\mathsf{ass}^2(\mathsf{red}^1(\tau_3))) = \emptyset\,,$$
$$\mathsf{type}([\mathbf{t}_4]) = \{\mathsf{red}^1(\sigma)\colon \sigma \in \mathsf{type}(\mathbf{t}_4)\} = \{\mathsf{red}^1(\tau_4)\} = \{\tau_4^1\}\,, \qquad \text{and finally}$$
$$\mathsf{type}([[\mathbf{t}_4, \mathbf{t}_3]]) = \{\mathsf{red}^2(\sigma)\colon \sigma \in \mathsf{type}([\mathbf{t}_4, \mathbf{t}_3])\} = \{\mathsf{red}^2(\mathsf{red}^1(\tau_3))\} = \{(q_3, \mathsf{pr})\}\,.$$

Recall that $\tau_3 \in \mathcal{T}_{\mathsf{np}}$ and $\sigma_{1,\mathsf{pr}}, \tau_4 \in \mathcal{T}_{\mathsf{pr}}$. We can compute $\mathsf{low}(\mathbf{s}_1)$ as follows:

$$\mathsf{low}([\mathbf{t}_4]) = \sum_{\sigma \in \mathsf{type}(\mathbf{t}_4)} \left(\mathsf{low}([]{\restriction}_{\pi_2(\mathsf{ass}^1(\sigma))}) + \mathsf{low}(\mathbf{t}_4{\restriction}_{\{\sigma\}})\right)$$
$$= \mathsf{low}([]{\restriction}_\emptyset) + \mathsf{low}(\mathbf{t}_4{\restriction}_{\{\tau_4\}}) = \mathsf{low}([]) + \mathsf{low}(\mathbf{t}_4) = 0 + 1 = 1\,,$$

$$\text{low}([[\mathbf{t}_4, \mathbf{t}_3]]) = \sum_{\sigma \in \text{type}([\mathbf{t}_4, \mathbf{t}_3])} \left(\text{low}([\,]\restriction_{\pi_2(\text{ass}^2(\sigma))}) + \text{low}([\mathbf{t}_4, \mathbf{t}_3]\restriction_{\{\sigma\}})\right)$$

$$= \text{low}([\,]\restriction_{\emptyset}) + \text{low}([\mathbf{t}_4, \mathbf{t}_3]\restriction_{\{\text{red}^1(\tau_3)\}}) = 0 + \text{low}([\mathbf{t}_4, \mathbf{t}_3])$$

$$= \sum_{\sigma \in \text{type}(\mathbf{t}_3)} \left(\text{low}([\mathbf{t}_4]\restriction_{\pi_2(\text{ass}^1(\sigma))}) + \text{low}(\mathbf{t}_3\restriction_{\{\sigma\}})\right)$$

$$= \text{low}([\mathbf{t}_4]\restriction_{\{\tau_4^1\}}) + \text{low}(\mathbf{t}_3\restriction_{\{\tau_3\}}) = \text{low}([\mathbf{t}_4]) + \text{low}(\mathbf{t}_3) = 1 + 0 = 1\,,$$

$$\text{low}([\mathbf{t}_2]) = \sum_{\sigma \in \text{type}(\mathbf{t}_2)} \left(\text{low}([\,]\restriction_{\pi_2(\text{ass}^1(\sigma))}) + \text{low}(\mathbf{t}_2\restriction_{\{\sigma\}})\right) = 0\,,$$

$$\text{low}([\mathbf{t}_2, \mathbf{t}_1]) = \sum_{\sigma \in \text{type}(\mathbf{t}_1)} \left(\text{low}([\mathbf{t}_2]\restriction_{\pi_2(\text{ass}^1(\sigma))}) + \text{low}(\mathbf{t}_1\restriction_{\{\sigma\}})\right)$$

$$= \text{low}([\mathbf{t}_2]\restriction_{\emptyset}) + \text{low}(\mathbf{t}_1\restriction_{\{\sigma_{1,\text{pr}}\}}) = \text{low}([\mathbf{t}_2]) + \text{low}(\mathbf{t}_1) = 0 + 1 = 1\,,$$

$$\text{low}(\mathbf{s}_1) = \sum_{\sigma \in \text{type}([\mathbf{t}_2, \mathbf{t}_1])} \left(\text{low}([[\mathbf{t}_4, \mathbf{t}_3]]\restriction_{\pi_2(\text{ass}^2(\sigma))}) + \text{low}([\mathbf{t}_2, \mathbf{t}_1]\restriction_{\{\sigma\}})\right)$$

$$= \text{low}([[\mathbf{t}_4, \mathbf{t}_3]]\restriction_{\{(q_3,\text{pr})\}}) + \text{low}([\mathbf{t}_2, \mathbf{t}_1]\restriction_{\{\text{red}^1(\sigma_{1,\text{pr}})\}})$$

$$= \text{low}([[\mathbf{t}_4, \mathbf{t}_3]]) + \text{low}([\mathbf{t}_2, \mathbf{t}_1]) = 1 + 1 = 2\,.$$

We see that the restrictions of annotated stacks appearing in the above formulas do not modify these annotated stacks (this is the case because all annotations are either singletons or empty sets).

Next, we compute $\text{high}(\mathbf{s}_1)$. To this end, recall that $\text{depth}(D_{1,\sharp,a}) = 2$ and $\text{depth}(E_4) = 1$. This time we give the formulas ignoring the restrictions, as again they do not change anything:

$$\text{high}([\mathbf{t}_4, \mathbf{t}_3]) = \text{pow}(\text{high}([\mathbf{t}_4]), \text{high}(\mathbf{t}_3)) = \text{pow}(\text{pow}(\text{high}([\,]), \text{high}(\mathbf{t}_4)), \text{high}(\mathbf{t}_3))$$

$$= \text{pow}(\text{pow}(1, C_{\text{depth}(E_4)}), 1) = \text{pow}(\text{pow}(1, C_1), 1) = 2^{C_1} - 1\,,$$

$$\text{high}([\mathbf{t}_2, \mathbf{t}_1]) = \text{pow}(\text{high}([\mathbf{t}_2]), \text{high}(\mathbf{t}_1)) = \text{pow}(1, C_{\text{depth}(D_{1,\sharp,a})}) = 2^{C_2} - 1\,,$$

$$\text{high}(\mathbf{s}_1) = \text{pow}(\text{high}([[\mathbf{t}_4, \mathbf{t}_3]]), \text{high}([\mathbf{t}_2, \mathbf{t}_1]))$$

$$= \text{pow}(\text{pow}(\text{high}([\,]), \text{high}([\mathbf{t}_4, \mathbf{t}_3])), \text{high}([\mathbf{t}_2, \mathbf{t}_1]))$$

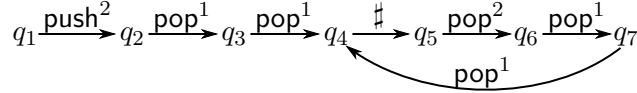$$= \text{pow}(\text{pow}(1, 2^{C_1} - 1), 2^{C_2} - 1) = 2^{(2^{C_1} - 1)(2^{C_2} - 1)} - 1\,.$$

Notice that $\text{high}(\mathbf{t}_3) = 1$, because $\text{type}(\mathbf{t}_3)$ contains only a nonproductive run descriptor.

Similarly, we can compute $\text{len}(\mathbf{s}_1)$, but this should also take into account E_3, whose depth is 0:

$$\text{len}([\mathbf{t}_4, \mathbf{t}_3]) = \text{pow}(\text{len}([\mathbf{t}_4]), \text{len}(\mathbf{t}_3)) = \text{pow}(\text{pow}(\text{len}([\,]), \text{len}(\mathbf{t}_4)), \text{len}(\mathbf{t}_3))$$

$$= \text{pow}(\text{pow}(1, C_{\text{depth}(E_4)}), C_{\text{depth}(E_3)}) = \text{pow}(\text{pow}(1, C_1), C_0) = 2^{C_1 \cdot C_0} - 1\,,$$

$$\text{len}([\mathbf{t}_2, \mathbf{t}_1]) = \text{pow}(\text{len}([\mathbf{t}_2]), \text{len}(\mathbf{t}_1)) = \text{pow}(1, C_{\text{depth}(D_{1,\sharp,a})}) = 2^{C_2} - 1\,,$$

$$\text{len}(\mathbf{s}_1) = \text{pow}(\text{len}([[\mathbf{t}_4, \mathbf{t}_3]]), \text{len}([\mathbf{t}_2, \mathbf{t}_1]))$$

$$= \text{pow}(\text{pow}(\text{len}([\,]), \text{len}([\mathbf{t}_4, \mathbf{t}_3])), \text{len}([\mathbf{t}_2, \mathbf{t}_1]))$$

$$= \text{pow}(\text{pow}(1, 2^{C_1 \cdot C_0} - 1), 2^{C_2} - 1) = 2^{(2^{C_1 \cdot C_0} - 1)(2^{C_2} - 1)} - 1\,.$$

We have said that low counts the number of productive run descriptors in all annotations. This is a good high-level intuition, but strictly speaking this is not true. Indeed, say that we have two run descriptors $\sigma, \sigma' \in \text{type}(\mathbf{s}^{k-1})$ such that $\text{ass}^k(\sigma) = \text{ass}^k(\sigma')$. Then, in the formula for $\text{low}(\mathbf{s}^k : \mathbf{s}^{k-1})$ we add $\text{low}(\mathbf{s}^k|_{\pi_2(\text{ass}^k(\sigma))})$ twice (once for σ, and once for σ'). This is illustrated by the next example.

Example 7.24. Consider the 2-DPDA $\mathcal{A}_2$ depicted below; its stack alphabet is $\{\gamma\}$, and input alphabet $\{\sharp\}$.

$$q_1 \xrightarrow{\text{push}^2} q_2 \xrightarrow{\text{pop}^1} q_3 \xrightarrow{\text{pop}^1} q_4 \xrightarrow{\sharp} q_5 \xrightarrow{\text{pop}^2} q_6 \xrightarrow{\text{pop}^1} q_7$$
$$q_4 \underset{\text{pop}^1}{\longleftarrow} q_5$$

This time we consider the trivial monoid $M = \{1\}$. Denote

$$\sigma_i^1 = (q_i, \{(1, (q_6, \text{pr}))\}, \text{pr}) \qquad \text{for } i \in \{3, 4, 7\}.$$

We are interested in derivation trees

$$D_1 = (\text{push}, \gamma, q_1, (\text{pop}, \gamma, q_2, \sigma_3^1), \{(\text{pop}, \gamma, q_6, \sigma_7^1)\}),$$
$$D_i = (\text{pop}, \gamma, q_i, \sigma_4^1) \qquad \text{for } i \in \{3, 7\}, \text{ and}$$
$$D_4 = (\text{read}, q_4, (\text{pop}, \gamma, q_5, (q_6, \text{pr}))).$$

The annotated 1-stack $\mathbf{s} = [(\gamma, \{D_4\}), (\gamma, \{D_3, D_7\}), (\gamma, \{D_1\})]$ is well-formed. Notice that the run descriptor of D_4 is productive, while run descriptors of D_1, D_3, and D_7 are nonproductive. We can see, though, that D_1 uses both D_3 and D_7 as assumptions, and both D_3 and D_7 use D_4 as an assumption. In effect $\text{low}(\mathbf{s})$ counts the nonproductive run descriptor $\text{rd}(D_4)$ twice:

$$\text{low}(\mathbf{s}) = \text{low}([(\gamma, \{D_4\}), (\gamma, \{D_3, D_7\})]) + \text{low}((\gamma, \{D_1\}))$$
$$= \text{low}([(\gamma, \{D_4\})]) + \text{low}((\gamma, \{D_3\})) + \text{low}([(\gamma, \{D_4\})]) + \text{low}((\gamma, \{D_7\})) + 0$$
$$= 1 + 0 + 1 + 0 + 0 = 2.$$

Remark 7.25. Consider the function $\beth_n(k)$ defined by

$$\beth_0(k) = k \qquad \text{and} \qquad \beth_{n+1}(k) = 2^{\beth_n(k)}.$$

One can construct an n-DPDA $\mathcal{A}$ that recognizes the language $\{\sharp^k a \sharp^{\beth_{n-1}(k)} \colon k \in \mathbb{N}\}$ (see Blumensath [Blu08, Example 9] for a very similar construction). After reading a prefix $\sharp^k a$, the number of 0-stacks in the n-stack s of $\mathcal{A}$ is linear in k. It is possible to annotate s, resulting in an annotated stack $\mathbf{s}$, such that the maximal annotated run starting from $\mathbf{s}$ reads $\beth_{n-1}(k)$ $\sharp$ symbols. It follows that the high function (and thus len as well) has to be at least $(n-1)$-fold exponential in the size of an annotated n-stack. According to our definition, high and len are (in the worst case) $(n+1)$-fold exponential, which is slightly larger than necessary. We believe that it is possible to save these two exponentiations, at the cost of complicating proofs.

We now prove Lemma 7.22, which fills the rest of this subsection. We start by proving some (in)equalities regarding the pow function.

Proposition 7.26. *The following is true for all positive integers:*

$$\text{pow}(a_1, \ldots, a_k, \text{pow}(b_1, \ldots, b_l)) = \text{pow}(a_1, \ldots, a_k, b_1, \ldots, b_l), \tag{7.11}$$

$$\mathsf{pow}(a_1, \ldots, a_k, \mathsf{pow}(c_0, c_1, \ldots, c_l), b_1, \ldots, b_l) \leq$$

$$\leq \mathsf{pow}(a_1, \ldots, a_k, c_0, b_1 c_1, \ldots, b_l c_l), \tag{7.12}$$

$$\mathsf{pow}(a_1, \ldots, a_{i-1}, a_i^x, a_{i+1}, \ldots, a_{k-1}, a_k) \leq \mathsf{pow}(a_1, \ldots, a_{k-1}, x a_k) \qquad \textit{for } i < k, \tag{7.13}$$

$$\mathsf{pow}(a_1, \ldots, a_{k-1}, a_k) + 1 \leq \mathsf{pow}(a_1, \ldots, a_{k-1}, a_k + 1), \tag{7.14}$$

$$\mathsf{pow}(a_1, \ldots, a_k) \cdot \mathsf{pow}(b_1, \ldots, b_k) \leq \mathsf{pow}(a_1 b_1, \ldots, a_k b_k). \tag{7.15}$$

Proof. Equality (7.11) can be shown by induction on k. For $k = 0$ we have that

$$\mathsf{pow}(\mathsf{pow}(b_1, \ldots, b_l)) = (1 + \mathsf{pow}(b_1, \ldots, b_l))^{\mathsf{pow}()} - 1$$

$$= (1 + \mathsf{pow}(b_1, \ldots, b_l))^1 - 1 = \mathsf{pow}(b_1, \ldots, b_l),$$

and for $k > 0$ we directly use the induction assumption:

$$\mathsf{pow}(a_1, \ldots, a_k, \mathsf{pow}(b_1, \ldots, b_l)) = (1 + a_1)^{\mathsf{pow}(a_2, \ldots, a_k, \mathsf{pow}(b_1, \ldots, b_l))} - 1$$

$$= (1 + a_1)^{\mathsf{pow}(a_2, \ldots, a_k, b_1, \ldots, b_l)} - 1$$

$$= \mathsf{pow}(a_1, \ldots, a_k, b_1, \ldots, b_l).$$

For Inequality (7.12) suppose first that $k = 0$. By Inequality (7.15), which we prove below, it follows that

$$\mathsf{pow}(\mathsf{pow}(c_0, c_1, \ldots, c_l), b_1, \ldots, b_l) = (1 + \mathsf{pow}(c_0, c_1, \ldots, c_l))^{\mathsf{pow}(b_1, \ldots, b_l)} - 1$$

$$= (1 + (1 + c_0)^{\mathsf{pow}(c_1, \ldots, c_l)} - 1)^{\mathsf{pow}(b_1, \ldots, b_l)} - 1$$

$$= (1 + c_0)^{\mathsf{pow}(b_1, \ldots, b_l) \cdot \mathsf{pow}(c_1, \ldots, c_l)} - 1$$

$$\leq (1 + c_0)^{\mathsf{pow}(b_1 c_1, \ldots, b_l c_l)} - 1 = \mathsf{pow}(c_0, b_1 c_1, \ldots, b_l c_l).$$

It is easy to see that pow is monotone, thus the general form of Inequality (7.12) follows from the above special form thanks to Equality (7.11):

$$\mathsf{pow}(a_1, \ldots, a_k, \mathsf{pow}(c_0, c_1, \ldots, c_l), b_1, \ldots, b_l)$$

$$= \mathsf{pow}(a_1, \ldots, a_k, \mathsf{pow}(\mathsf{pow}(c_0, c_1, \ldots, c_l), b_1, \ldots, b_l))$$

$$\leq \mathsf{pow}(a_1, \ldots, a_k, \mathsf{pow}(c_0, b_1 c_1, \ldots, b_l c_l)) = \mathsf{pow}(a_1, \ldots, a_k, c_0, b_1 c_1, \ldots, b_l c_l).$$

Heading toward proving Inequality (7.13), we first show that

$$x \cdot \mathsf{pow}(a_{i+1}, \ldots, a_k) \leq \mathsf{pow}(a_{i+1}, \ldots, a_{k-1}, x a_k), \tag{7.16}$$

where $i < k$, and the numbers $x, a_{i+1}, \ldots, a_k$ are positive integers. This is shown by induction on $k - i$. When $k - i = 1$, we simply have that

$$x \cdot \mathsf{pow}(a_k) = x \cdot ((1 + a_k)^1 - 1) = (1 + x a_k)^1 - 1 = \mathsf{pow}(x a_k).$$

Suppose that $k - i > 1$. Notice that $xb \leq b^x$ for all $x \in \mathbb{N}$ and $b \geq 2$. Thus,

$$x \cdot \mathsf{pow}(a_{i+1}, \ldots, a_k) = x \cdot ((1 + a_{i+1})^{\mathsf{pow}(a_{i+2}, \ldots, a_k)} - 1)$$

$$\leq x \cdot (1 + a_{i+1})^{\mathsf{pow}(a_{i+2}, \ldots, a_k)} - 1 \leq (1 + a_{i+1})^{x \cdot \mathsf{pow}(a_{i+2}, \ldots, a_k)} - 1$$

$$\leq (1 + a_{i+1})^{\mathsf{pow}(a_{i+2}, \ldots, a_{k-1}, x a_k)} - 1 = \mathsf{pow}(a_{i+1}, \ldots, a_{k-1}, x a_k).$$

Above, the first inequality holds because $x \geq 1$; the second inequality follows from the inequality $xb \leq b^x$, where we notice that $(1 + a_{i+1})^{\mathsf{pow}(a_{i+2}, \ldots, a_k)} \geq 2$ (because $a_{i+1} \geq 1$ and

$\mathsf{pow}(\dots) \geq 1$); the third inequality is an application of the induction assumption. From Inequality (7.16) it follows that

$$\mathsf{pow}(a_i^x, a_{i+1}, \dots, a_k) = (1 + a_i^x)^{\mathsf{pow}(a_{i+1}, \dots, a_k)} - 1 \leq (1 + a_i)^{x \cdot \mathsf{pow}(a_{i+1}, \dots, a_k)} - 1$$

$$\leq (1 + a_i)^{\mathsf{pow}(a_{i+1}, \dots, a_{k-1}, xa_k)} - 1 = \mathsf{pow}(a_{i+1}, \dots, a_{k-1}, xa_k).$$

This gives the thesis, by Equality (7.11) and by monotonicity of pow:

$$\mathsf{pow}(a_1, \dots, a_{i-1}, a_i^x, a_{i+1}, \dots, a_{k-1}, a_k) = \mathsf{pow}(a_1, \dots, a_{i-1}, \mathsf{pow}(a_i^x, a_{i+1}, \dots, a_{k-1}, a_k))$$

$$\leq \mathsf{pow}(a_1, \dots, a_{i-1}, \mathsf{pow}(a_i, \dots, a_{k-1}, xa_k)) = \mathsf{pow}(a_1, \dots, a_{k-1}, xa_k).$$

Inequality (7.14) is shown by induction on k. For $k = 1$ we simply have

$$\mathsf{pow}(a_1) + 1 = (1 + a_1)^1 - 1 + 1 = (1 + a_1 + 1)^1 - 1 = \mathsf{pow}(a_1 + 1).$$

For $k > 1$ we use the induction assumption as follows (the first inequality below holds because $a_1 \geq 1$):

$$\mathsf{pow}(a_1, \dots, a_{k-1}, a_k) + 1 = (1 + a_1)^{\mathsf{pow}(a_2, \dots, a_k)} - 1 + 1 \leq (1 + a_1)^{\mathsf{pow}(a_2, \dots, a_k)+1} - 1$$

$$\leq (1 + a_1)^{\mathsf{pow}(a_2, \dots, a_{k-1}, a_k+1)} - 1 = \mathsf{pow}(a_1, \dots, a_{k-1}, a_k + 1).$$

Inequality (7.15) is also shown by induction on k. For $k = 0$ the thesis is trivial:

$$\mathsf{pow}() \cdot \mathsf{pow}() = 1 \cdot 1 = 1 = \mathsf{pow}().$$

Suppose now that $k \geq 1$, and denote $x = \mathsf{pow}(a_2, \dots, a_k)$ and $y = \mathsf{pow}(b_2, \dots, b_k)$. We claim that

$$((1 + a_1)^x - 1)((1 + b_1)^y - 1) \leq (1 + a_1 b_1)^{xy} - 1. \tag{7.17}$$

Let us prove this inequality. By symmetry, we can assume that $x \geq y$. We have three cases. If $x = y = 1$, Inequality (7.17) simply says that

$$((1 + a_1)^1 - 1)((1 + b_1)^1 - 1) = a_1 b_1 \leq (1 + a_1 b_1)^{1 \cdot 1} - 1.$$

Next, suppose that $x \geq 2$ and $y = 1$. We see that

$$0 \leq (b_1 - 1)^2,$$

$$0 \leq b_1^2 - 2b_1 + 1,$$

$$4b_1 \leq b_1^2 + 2b_1 + 1,$$

$$4b_1 \leq (b_1 + 1)^2,$$

$$b_1 \leq \left(\frac{b_1 + 1}{2}\right)^2.$$

Because $x \geq 2$ and $b_1 \geq 1$, it follows that

$$b_1 \leq \left(\frac{b_1 + 1}{2}\right)^x. \tag{7.18}$$

Next, observe that

$$0 \leq (a_1 - 1)(b_1 - 1),$$

$$0 \leq a_1 b_1 - a_1 - b_1 + 1,$$

$$a_1 b_1 + a_1 + b_1 + 1 \leq 2a_1 b_1 + 2,$$

$$(1 + a_1)(b_1 + 1) \leq 2 + 2a_1 b_1,$$

$$(1 + a_1)\left(\frac{b_1 + 1}{2}\right) \leq 1 + a_1 b_1. \tag{7.19}$$

Using Inequalities (7.18) and (7.19) we obtain Inequality (7.17):

$$((1 + a_1)^x - 1)((1 + b_1)^1 - 1) = (1 + a_1)^x \cdot b_1 - b_1 \leq (1 + a_1)^x \cdot b_1 - 1$$

$$\leq (1 + a_1)^x \left(\frac{b_1 + 1}{2}\right)^x - 1 \leq (1 + a_1 b_1)^{x \cdot 1} - 1.$$

The remaining case is when $x \geq y \geq 2$. In this case we have that

$$((1 + a_1)^x - 1)((1 + b_1)^y - 1) \leq (1 + a_1)^x (1 + b_1)^y - 1 \leq (1 + a_1)^x (1 + b_1)^x - 1$$

$$\leq (1 + a_1 b_1)^x (1 + a_1 b_1)^x - 1 = (1 + a_1 b_1)^{x \cdot 2} - 1$$

$$\leq (1 + a_1 b_1)^{x \cdot y} - 1.$$

Thus, we have shown Inequality (7.17) in all cases. Using this inequality and the induction assumption, we can conclude that

$$\mathsf{pow}(a_1, \ldots, a_k) \cdot \mathsf{pow}(b_1, \ldots, b_k) = ((1 + a_1)^x - 1)((1 + b_1)^y - 1) \leq (1 + a_1 b_1)^{x \cdot y} - 1$$

$$\leq (1 + a_1 b_1)^{\mathsf{pow}(a_2 b_2, \ldots, a_k b_k)} - 1 = \mathsf{pow}(a_1 b_1, \ldots, a_k b_k).$$

$\square$

Heading toward the proof of Lemma 7.22, we now observe some auxiliary properties.

Proposition 7.27. *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* $\mathsf{low}(\mathbf{s}) \geq 1$ *and* $\mathsf{high}(\mathbf{s}) \geq 2$.

Proof. By induction on the structure of $\mathbf{s}$. In the base cases of a 0-stack and of an empty k-stack, the thesis follows directly from Definition 7.21. In the induction step denote $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$. Recall that $\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^{k-1})\}$ (by the definition of types), so

$$\mathsf{type}(\mathbf{s}) \subseteq \mathcal{T}_{\mathsf{np}} \Leftrightarrow \forall_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{red}^k(\sigma) \in \mathcal{T}_{\mathsf{np}}.$$

Moreover,

$$\mathsf{red}^k(\sigma) \in \mathcal{T}_{\mathsf{np}} \Leftrightarrow (\sigma \in \mathcal{T}_{\mathsf{np}} \wedge \pi_2(\mathsf{ass}^k(\sigma)) \subseteq \mathcal{T}_{\mathsf{np}})$$

for $\sigma \in \mathcal{T}^{k-1}$ (by the definition of red^k). It follows that

$$\mathsf{type}(\mathbf{s}) \subseteq \mathcal{T}_{\mathsf{np}} \Leftrightarrow \forall_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} (\sigma \in \mathcal{T}_{\mathsf{np}} \wedge \pi_2(\mathsf{ass}^k(\sigma)) \subseteq \mathcal{T}_{\mathsf{np}}).$$

If $\mathsf{type}(\mathbf{s}) \subseteq \mathcal{T}_{\mathsf{np}}$ then, by the induction assumption, $\mathsf{low}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}) = \mathsf{low}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}}) = 0$ and $\mathsf{high}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}) = \mathsf{high}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}}) = 1$ for all $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$; in effect

$$\mathsf{low}(\mathbf{s}) = \sum_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \left(\mathsf{low}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}) + \mathsf{low}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}})\right) = \sum_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} (0 + 0) = 0, \quad \text{and}$$

$$\mathsf{high}(\mathbf{s}) = \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}\left(\mathsf{high}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{high}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}})\right).$$

$$= \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}(1, 1) = \prod_{\sigma \in \mathsf{type}(\mathbf{s}^{k-1})} 1 = 1.$$

Conversely, if $\mathsf{type}(\mathbf{s}) \not\subseteq \mathcal{T}_{\mathsf{np}}$ then, by the induction assumption, at least one among $\mathsf{low}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))})$ and $\mathsf{low}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}})$ for $\sigma \in \mathsf{type}(\mathbf{s}^{k-1})$ is positive (and all other are non-negative); in effect $\mathsf{low}(\mathbf{s})$, being their sum, is positive. Similarly, at least one among $\mathsf{high}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))})$ and $\mathsf{high}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}})$ is greater than 1 (and all other are positive); in effect some $\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}), \mathsf{high}(\mathbf{s}^{k-1}\!\restriction_{\{\sigma\}})\big)$ is greater than 1 (notice that $\mathsf{pow}(2,1) = (1+2)^1 - 1 = 2$, and $\mathsf{pow}(1,2) = (1+1)^2 - 1 = 3$, and that pow is monotone), and thus their product $\mathsf{high}(\mathbf{s})$ is greater than 1. $\qquad\square$

Proposition 7.28. *For every well-formed annotated stack* $\mathbf{s}$,

$$\mathsf{low}(\mathbf{s}) = \sum_{\sigma \in \mathsf{type}(\mathbf{s})} \mathsf{low}(\mathbf{s}\!\restriction_{\{\sigma\}}), \qquad \mathsf{high}(\mathbf{s}) = \prod_{\sigma \in \mathsf{type}(\mathbf{s})} \mathsf{high}(\mathbf{s}\!\restriction_{\{\sigma\}}), \qquad and$$

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

Proof. We analyze Definition 7.21. Suppose first that $\mathbf{s} = (\gamma, \mathfrak{D})$ (i.e., that $\mathbf{s}$ is of order 0). Because $\mathsf{type}(\mathbf{s}\!\restriction_{\{\sigma\}}) = \{\sigma\}$,

$$\mathsf{low}(\mathbf{s}) = |\mathsf{type}(\mathbf{s}) \cap \mathcal{T}_{\mathsf{pr}}| = \sum_{\sigma \in \mathsf{type}(\mathbf{s})} |\{\sigma\} \cap \mathcal{T}_{\mathsf{pr}}|$$

$$= \sum_{\sigma \in \mathsf{type}(\mathbf{s})} |\mathsf{type}(\mathbf{s}\!\restriction_{\{\sigma\}}) \cap \mathcal{T}_{\mathsf{pr}}| = \sum_{\sigma \in \mathsf{type}(\mathbf{s})} \mathsf{low}(\mathbf{s}\!\restriction_{\{\sigma\}}).$$

Recall that $\mathsf{type}(\mathbf{s}) = \{\mathsf{rd}(D) \colon D \in \mathfrak{D}\}$, and that $\mathbf{s}\!\restriction_{\{\sigma\}} = (\gamma, \{D \in \mathfrak{D} \colon \mathsf{rd}(D) = \sigma\})$; thus,

$$\mathsf{high}(\mathbf{s}) = \prod_{D \in \mathfrak{D} \colon \, \mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}} C_{\mathsf{depth}(D)}$$

$$= \prod_{\sigma \in \mathsf{type}(\mathbf{s})} \left(\prod_{D \in \mathfrak{D} \colon \, \mathsf{rd}(D) = \sigma \in \mathcal{T}_{\mathsf{pr}}} C_{\mathsf{depth}(D)} \right) = \prod_{\sigma \in \mathsf{type}(\mathbf{s})} \mathsf{high}(\mathbf{s}\!\restriction_{\{\sigma\}}), \qquad and$$

$$\mathsf{len}(\mathbf{s}) = \prod_{D \in \mathfrak{D}} C_{\mathsf{depth}(D)} = \prod_{\sigma \in \mathsf{type}(\mathbf{s})} \left(\prod_{D \in \mathfrak{D} \colon \, \mathsf{rd}(D) = \sigma} C_{\mathsf{depth}(D)} \right) = \prod_{\sigma \in \mathsf{type}(\mathbf{s})} \mathsf{len}(\mathbf{s}\!\restriction_{\{\sigma\}}).$$

If $\mathbf{s} = [\,]$, then $\mathsf{type}(\mathbf{s}) = \emptyset$, and thus

$$\mathsf{low}(\mathbf{s}) = 0 = \sum_{\sigma \in \emptyset} \mathsf{low}(\mathbf{s}\!\restriction_{\{\sigma\}}), \qquad \mathsf{high}(\mathbf{s}) = 1 = \prod_{\sigma \in \emptyset} \mathsf{high}(\mathbf{s}\!\restriction_{\{\sigma\}}), \qquad and$$

$$\mathsf{len}(\mathbf{s}) = 1 = \prod_{\sigma \in \emptyset} \mathsf{len}(\mathbf{s}\!\restriction_{\{\sigma\}}).$$

Finally, suppose that $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1}$. Recall that $\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\tau) \colon \tau \in \mathsf{type}(\mathbf{s}^{k-1})\}$. Moreover, by well-formedness of $\mathbf{s}$, for every $\sigma \in \mathsf{type}(\mathbf{s})$ there is exactly one $\tau \in \mathsf{type}(\mathbf{s}^{k-1})$ such that $\mathsf{red}^k(\tau) = \sigma$; denote it τ_σ. By the definition of a restriction, we have that

$\mathbf{s}\restriction_{\{\sigma\}} = \mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau_\sigma))} : \mathbf{s}^{k-1}\restriction_{\{\tau_\sigma\}}$. Recalling that $\mathsf{type}(\mathbf{s}\restriction_{\{\sigma\}}) = \{\sigma\}$, we obtain

$$\mathsf{low}(\mathbf{s}) = \sum_{\tau\in\mathsf{type}(\mathbf{s}^{k-1})} \left(\mathsf{low}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau))}) + \mathsf{low}(\mathbf{s}^{k-1}\restriction_{\{\tau\}})\right)$$

$$= \sum_{\sigma\in\mathsf{type}(\mathbf{s})} \left(\mathsf{low}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau_\sigma))}) + \mathsf{low}(\mathbf{s}^{k-1}\restriction_{\{\tau_\sigma\}})\right) = \sum_{\sigma\in\mathsf{type}(\mathbf{s})} \mathsf{low}(\mathbf{s}\restriction_{\{\sigma\}}) ,$$

$$\mathsf{high}(\mathbf{s}) = \prod_{\tau\in\mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}\left(\mathsf{high}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau))}), \mathsf{high}(\mathbf{s}^{k-1}\restriction_{\{\tau\}})\right)$$

$$= \prod_{\sigma\in\mathsf{type}(\mathbf{s})} \mathsf{pow}\left(\mathsf{high}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau_\sigma))}), \mathsf{high}(\mathbf{s}^{k-1}\restriction_{\{\tau_\sigma\}})\right) = \prod_{\sigma\in\mathsf{type}(\mathbf{s})} \mathsf{high}(\mathbf{s}\restriction_{\{\sigma\}}), \quad \text{and}$$

$$\mathsf{len}(\mathbf{s}) = \prod_{\tau\in\mathsf{type}(\mathbf{s}^{k-1})} \mathsf{pow}\left(\mathsf{len}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau))}), \mathsf{len}(\mathbf{s}^{k-1}\restriction_{\{\tau\}})\right)$$

$$= \prod_{\sigma\in\mathsf{type}(\mathbf{s})} \mathsf{pow}\left(\mathsf{len}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\tau_\sigma))}), \mathsf{len}(\mathbf{s}^{k-1}\restriction_{\{\tau_\sigma\}})\right) = \prod_{\sigma\in\mathsf{type}(\mathbf{s})} \mathsf{len}(\mathbf{s}\restriction_{\{\sigma\}}) . \qquad \square$$

Proposition 7.29. *Let $0 \le l \le k \le n$, and let $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^l$ be a well-formed annotated k-stack that is singular. In this situation*

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

$$\mathsf{high}(\mathbf{s}) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \ldots, \mathsf{high}(\mathbf{s}^l)) , \qquad and$$

$$\mathsf{len}(\mathbf{s}) = \mathsf{pow}(\mathsf{len}(\mathbf{s}^k), \mathsf{len}(\mathbf{s}^{k-1}), \ldots, \mathsf{len}(\mathbf{s}^l)) .$$

Proof. Induction on $k - l$. For $k - l = 0$, we simply have $\mathbf{s} = \mathbf{s}^k$; both sides of each equality are the same (recall that $\mathsf{pow}(x) = x$ for every x).

Suppose that $k - l \ge 1$, and denote $\mathbf{t} = \mathbf{s}^{k-1} : \mathbf{s}^{k-2} : \cdots : \mathbf{s}^l$; we have that $\mathbf{s} = \mathbf{s}^k : \mathbf{t}$. Because $\mathbf{s}$ is well-formed and $\mathsf{type}(\mathbf{s})$ is a singleton, also $\mathsf{type}(\mathbf{t})$ is a singleton $\{\sigma\}$, where $\mathsf{type}(\mathbf{s}^k) = \pi_2(\mathsf{ass}^k(\sigma))$. In effect, restricting $\mathbf{t}$ to $\{\sigma\}$ or $\mathbf{s}^k$ to $\pi_2(\mathsf{ass}^k(\sigma))$ does not change the annotated stacks, so, by definition,

$$\mathsf{low}(\mathbf{s}) = \sum_{\sigma\in\mathsf{type}(\mathbf{t})} \left(\mathsf{low}(\mathbf{s}^k\restriction_{\pi_2(\mathsf{ass}^k(\sigma))}) + \mathsf{low}(\mathbf{t}\restriction_{\{\sigma\}})\right) = \mathsf{low}(\mathbf{s}^k) + \mathsf{low}(\mathbf{t}) .$$

Similarly,

$$\mathsf{high}(\mathbf{s}) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{t})) \qquad \text{and} \qquad \mathsf{len}(\mathbf{s}) = \mathsf{pow}(\mathsf{len}(\mathbf{s}^k), \mathsf{len}(\mathbf{t})) .$$

From the induction assumption we know that

$$\mathsf{low}(\mathbf{t}) = \sum_{i=l}^{k-1} \mathsf{low}(\mathbf{s}^i) ,$$

$$\mathsf{high}(\mathbf{t}) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^{k-1}), \mathsf{high}(\mathbf{s}^{k-2}), \ldots, \mathsf{high}(\mathbf{s}^l)) , \qquad \text{and}$$

$$\mathsf{len}(\mathbf{t}) = \mathsf{pow}(\mathsf{len}(\mathbf{s}^{k-1}), \mathsf{len}(\mathbf{s}^{k-2}), \ldots, \mathsf{len}(\mathbf{s}^l)) .$$

By substituting this to equalities for $\mathsf{low}(\mathbf{s})$, $\mathsf{high}(\mathbf{s})$, and $\mathsf{len}(\mathbf{s})$ we obtain the thesis, where in the case of high and len we additionally use Equality (7.11). $\qquad \square$

Proposition 7.30. *Let $0 \leq l \leq k \leq n$, let $\mathbf{s} = \mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^l$ be a well-formed annotated k-stack, and let $\sigma \in \mathsf{type}(\mathbf{s}^l)$. In this situation*

$$\mathbf{s}\!\restriction_{\{\mathsf{red}^k(\sigma)\}} = \mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))} : \mathbf{s}^{k-1}\!\restriction_{\pi_2(\mathsf{ass}^{k-1}(\sigma))} : \cdots : \mathbf{s}^{l+1}\!\restriction_{\pi_2(\mathsf{ass}^{l+1}(\sigma))} : \mathbf{s}^l\!\restriction_{\{\sigma\}} .$$

Proof. Proposition 7.14 used for $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \mathsf{type}(\mathbf{s}^l)\}$ and $|\mathsf{type}(\mathbf{s})| = |\mathsf{type}(\mathbf{s}^l)|$; the latter means that $\mathsf{red}^k(\sigma) = \mathsf{red}^k(\sigma')$ implies $\sigma = \sigma'$ for $\sigma, \sigma' \in \mathsf{type}(\mathbf{s}^l)$.

Observe that $\mathsf{top}^l(\mathbf{s}\!\restriction_{\{\mathsf{red}^k(\sigma)\}})$ equals $\mathbf{s}^l$ restricted to a subset of $\mathsf{type}(\mathbf{s}^l)$. Proposition 7.14 used for $\mathbf{s}\!\restriction_{\{\mathsf{red}^k(\sigma)\}}$ implies that this subset is a singleton $\{\sigma'\}$, and that $\mathsf{red}^k(\sigma') = \mathsf{red}^k(\sigma)$. This implies that $\sigma' = \sigma$, by the previous paragraph. Using also Item (T2) of Proposition 7.14, we see that

$$\mathbf{s}\!\restriction_{\{\mathsf{red}^k(\sigma)\}} = \mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))} : \mathbf{s}^{k-1}\!\restriction_{\pi_2(\mathsf{ass}^{k-1}(\sigma))} : \cdots : \mathbf{s}^{l+1}\!\restriction_{\pi_2(\mathsf{ass}^{l+1}(\sigma))} : \mathbf{s}^l\!\restriction_{\{\sigma\}} ,$$

as required. $\qquad\square$

Next, we observe how the functions low, high, and len interplay with composing annotated stacks.

Lemma 7.31. *Let $0 \leq l \leq k \leq n$, let $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^l; \Psi^k; f)$ 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*

$$\sum_{i=l}^{k} \mathsf{low}(\mathbf{s}^i) \leq \mathsf{low}(\mathbf{s}) , \tag{7.20}$$

$$\sum_{i=l}^{k} \mathsf{low}(\mathbf{s}^i) < \mathsf{low}(\mathbf{s}) \qquad\qquad\qquad\qquad\qquad \textit{if } f = \mathsf{pr} , \tag{7.21}$$

$$\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \ldots, \mathsf{high}(\mathbf{s}^{l+1}), \big|\mathcal{T}^0\big|^n \cdot \mathsf{high}(\mathbf{s}^l)\big) \geq \mathsf{high}(\mathbf{s}) , \tag{7.22}$$

$$\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^k), \mathsf{high}(\mathbf{s}^{k-1}), \ldots, \mathsf{high}(\mathbf{s}^{l+1}), \mathsf{high}(\mathbf{s}^l)\big) \geq \mathsf{high}(\mathbf{s}) \qquad \textit{if } f = \mathsf{np} , \tag{7.23}$$

$$\mathsf{pow}\big(\mathsf{len}(\mathbf{s}^k), \mathsf{len}(\mathbf{s}^{k-1}), \ldots, \mathsf{len}(\mathbf{s}^{l+1}), \big|\mathcal{T}^0\big|^n \cdot \mathsf{len}(\mathbf{s}^l)\big) \geq \mathsf{len}(\mathbf{s}) . \tag{7.24}$$

Proof. Because $\mathsf{type}(\mathbf{s}^l) = \pi_2(\Phi^l)$, Proposition 7.14 used for $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}) = \{\mathsf{red}^k(\sigma) \colon \sigma \in \pi_2(\Phi^l)\}$ and $|\mathsf{type}(\mathbf{s})| = |\pi_2(\Phi^l)|$, which means that the mapping defined by $\sigma \mapsto \mathsf{red}^k(\sigma)$ is a bijection between $\pi_2(\Phi^l)$ and $\mathsf{type}(\mathbf{s})$. Moreover, for $\sigma \in \pi_2(\Phi^l)$,

$$\mathbf{s}\!\restriction_{\{\mathsf{red}^k(\sigma)\}} = \mathbf{s}^k\!\restriction_{\pi_2(\mathsf{ass}^k(\sigma))} : \mathbf{s}^{k-1}\!\restriction_{\pi_2(\mathsf{ass}^{k-1}(\sigma))} : \cdots : \mathbf{s}^{l+1}\!\restriction_{\pi_2(\mathsf{ass}^{l+1}(\sigma))} : \mathbf{s}^l\!\restriction_{\{\sigma\}} ,$$

by Proposition 7.30.

For $i \in [l+1, k]$ and $\sigma \in \pi_2(\Phi^l)$ denote $H_\sigma^i = \mathsf{high}(\mathbf{s}^i{\restriction}_{\pi_2(\mathsf{ass}^i(\sigma))})$. Using Proposition 7.28, the above property, and Proposition 7.29, we obtain that

$$\mathsf{low}(\mathbf{s}) = \sum_{\tau \in \mathsf{type}(\mathbf{s})} \mathsf{low}(\mathbf{s}{\restriction}_{\{\tau\}}) = \sum_{\sigma \in \pi_2(\Phi^l)} \mathsf{low}(\mathbf{s}{\restriction}_{\{\mathsf{red}^l(\sigma)\}})$$

$$= \sum_{\sigma \in \pi_2(\Phi^l)} \left(\mathsf{low}(\mathbf{s}^l{\restriction}_{\{\sigma\}})) + \sum_{i=l+1}^{k} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\mathsf{ass}^i(\sigma))}) \right)$$

$$= \mathsf{low}(\mathbf{s}^l) + \sum_{i=l+1}^{k} \sum_{\sigma \in \pi_2(\Phi^l)} \mathsf{low}(\mathbf{s}^i{\restriction}_{\pi_2(\mathsf{ass}^i(\sigma))}), \qquad \text{and} \qquad (7.25)$$

$$\mathsf{high}(\mathbf{s}) = \prod_{\tau \in \mathsf{type}(\mathbf{s})} \mathsf{high}(\mathbf{s}{\restriction}_{\{\tau\}}) = \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{high}(\mathbf{s}{\restriction}_{\{\mathsf{red}^k(\sigma)\}})$$

$$= \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{pow}\big(H_\sigma^k, H_\sigma^{k-1}, \ldots, H_\sigma^{l+1}, \mathsf{high}(\mathbf{s}^l{\restriction}_{\{\sigma\}})\big). \qquad (7.26)$$

For each $i \in [l+1, k]$ it holds that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Phi^i) = \bigcup\{\pi_2(\mathsf{ass}^i(\sigma)) : \sigma \in \pi_2(\Phi^l)\}$ (by the definition of a composer, Condition (C1)), so, by Proposition 7.28,

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

$$(7.27)$$

Altogether, Equality (7.25) and Inequality (7.27) used for all $i \in [l+1, k]$ imply Inequality (7.20).

For Inequality (7.21), recall from the definition of a composer (Condition (C4)) that if $f = \mathsf{pr}$, then for some $i \in [l+1, k]$, some $\tau \in \mathcal{T}_{\mathsf{pr}}$ appears in $\pi_2(\mathsf{ass}^i(\sigma))$ simultaneously for two different $\sigma \in \pi_2(\Phi^l)$. By Proposition 7.27, $\mathsf{low}(\mathbf{s}^i{\restriction}_{\{\tau\}}) \ge 1$ for this τ (because it is productive). Thus, some positive component $\mathsf{low}(\mathbf{s}^i{\restriction}_{\{\tau\}})$ appears in two sums $\sum_{\tau \in \pi_2(\mathsf{ass}^i(\sigma))} \mathsf{low}(\mathbf{s}^i{\restriction}_{\{\tau\}})$ in Inequality (7.27) used for this i, so this inequality (and, in effect, Inequality (7.20)) becomes strict.

Using Equality (7.26), Inequality (7.15), and Proposition 7.28 we obtain

$$\mathsf{high}(\mathbf{s}) = \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{pow}\big(H_\sigma^k, H_\sigma^{k-1}, \ldots, H_\sigma^{l+1}, \mathsf{high}(\mathbf{s}^l{\restriction}_{\{\sigma\}})\big)$$

$$\le \mathsf{pow}\Big(\prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^k, \prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^{k-1}, \ldots, \prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^{l+1}, \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{high}(\mathbf{s}^l{\restriction}_{\{\sigma\}}) \Big)$$

$$= \mathsf{pow}\Big(\prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^k, \prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^{k-1}, \ldots, \prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^{l+1}, \mathsf{high}(\mathbf{s}^l) \Big). \qquad (7.28)$$

Observe that by restricting an annotated stack we can only decrease the value of high. Thus, for each $i \in [l+1, k]$,

$$\prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^i = \prod_{\sigma \in \pi_2(\Phi^l)} \mathsf{high}(\mathbf{s}^i \restriction_{\pi_2(\mathsf{ass}^i(\sigma))})$$

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

The last inequality is true because $|\pi_2(\Phi^l)| \leq |\mathcal{T}^l| \leq |\mathcal{T}^0|$. Using Inequality (7.13) we move the $|\mathcal{T}^0|$ exponents (there is at most n of them) into the last argument of pow and we obtain Inequality (7.22):

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

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

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 (cf. Proposition 7.27):

$$\prod_{\sigma \in \pi_2(\Phi^l)} H_\sigma^i = \prod_{\sigma \in \pi_2(\Phi^l)} \prod_{\tau \in \pi_2(\mathsf{ass}^i(\sigma))} \mathsf{high}(\mathbf{s}^i \restriction_{\{\tau\}}) = \prod_{\tau \in \pi_2(\Phi^i)} \mathsf{high}(\mathbf{s}^i \restriction_{\{\tau\}}) = \mathsf{high}(\mathbf{s}^i).$$

By substituting this to Inequality (7.28) we obtain Inequality (7.23).

Inequality (7.24) is obtained in the same way as Inequality (7.22), because the definitions of len and high differ only in the base case. $\square$

We are now ready to prove Lemma 7.22.

Proof of Lemma 7.22. 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 $\mathbf{s}^0 = (\gamma, \{D\})$. Recall that our goal is to prove the following inequalities:

$$\mathsf{low}(\mathbf{R}(0)) \leq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}(\mathbf{R}(1)),$$
$$\mathsf{high}(\mathbf{R}(0)) \geq \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)), \qquad \text{and}$$
$$\mathsf{len}(\mathbf{R}(0)) \geq 1 + \mathsf{len}(\mathbf{R}(1)).$$

We have four cases, depending on the shape of D.

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

Case 2. Suppose that $D = (\mathsf{read}, p, D')$. Then $\mathbf{R}(1) = \mathbf{s}^n : \mathbf{s}^{n-1} : \dots : \mathbf{s}^1 : (\gamma, \{D'\})$. Because both $\mathbf{R}(0)$ and $\mathbf{R}(1)$ are singular, by Proposition 7.29 we have that

$$\mathsf{low}(\mathbf{R}(0)) = \mathsf{low}(\mathbf{s}^0) + \sum_{i=1}^n \mathsf{low}(\mathbf{s}^i) \quad \text{and} \quad \mathsf{low}(\mathbf{R}(1)) = \mathsf{low}((\gamma, \{D'\})) + \sum_{i=1}^n \mathsf{low}(\mathbf{s}^i).$$

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 $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$ (as then $\mathsf{low}(\mathbf{s}^0) = 0$). 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.

Again by Proposition 7.29 we have that

$$\mathsf{high}(\mathbf{R}(0)) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), \mathsf{high}(\mathbf{s}^0)), \qquad \text{and}$$

$$\mathsf{high}(\mathbf{R}(1)) = \mathsf{pow}(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^1), \mathsf{high}((\gamma, \{D'\}))).$$

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 $\mathsf{high}(\mathbf{s}^0) = \mathsf{high}((\gamma, \{D'\})) = 1$, so the two sides of the inequality are equal:

$$\mathsf{high}(\mathbf{R}(0)) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)).$$

If $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$, then using Inequality (7.14) we obtain

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

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

Case 3. Suppose that $D = (\mathsf{pop}, \gamma, p, \tau^k)$. Then $\mathbf{R}(1) = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$. The operation between $\mathsf{conf}(\mathbf{R}(0))$ and $\mathsf{conf}(\mathbf{R}(1))$ is pop^k, so $\sharp(\mathsf{st}(\mathbf{R})) = 0$. For $i \in [1, k-1]$, by Definition 7.11(3) we have that $\mathsf{ass}^i(\mathsf{rd}(D)) = \emptyset$, and by well-formedness of $\mathbf{R}(0)$ we have that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D)))$ (cf. Proposition 7.14); in effect $\mathsf{type}(\mathbf{s}^i) = \emptyset$, hence $\mathsf{low}(\mathbf{s}^i) = 0$ and $\mathsf{high}(\mathbf{s}^i) = 1$. By Definition 7.11(3), $\mathsf{rd}(R) \in \mathcal{T}_{\mathsf{np}}$, so $\mathsf{low}(\mathbf{s}^0) = 0$ and $\mathsf{high}(\mathbf{s}^0) = 1$; moreover $\mathsf{len}(\mathbf{s}^0) = C_0 = 2$. Because $\mathbf{R}(0)$ and $\mathbf{R}(1)$ are singular, by Proposition 7.29 we can write

$$\mathsf{low}(\mathbf{R}(0)) = \sum_{i=0}^{n} \mathsf{low}(\mathbf{s}^i) = \sum_{i=k}^{n} \mathsf{low}(\mathbf{s}^i) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{low}(\mathbf{R}(1)),$$

$$\begin{aligned}
\mathsf{high}(\mathbf{R}(0)) &= \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^k), 1, \dots, 1\big) \\
&= \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \dots, \mathsf{high}(\mathbf{s}^k)\big) = \sharp(\mathsf{st}(\mathbf{R})) + \mathsf{high}(\mathbf{R}(1)), \qquad \text{and}
\end{aligned}$$

$$\begin{aligned}
\mathsf{len}(\mathbf{R}(0)) &= \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^n), \mathsf{len}(\mathbf{s}^{n-1}), \dots, \mathsf{len}(\mathbf{s}^k), 1, \dots, 1, 2\big) \\
&\geq \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^n), \mathsf{len}(\mathbf{s}^{n-1}), \dots, \mathsf{len}(\mathbf{s}^k), 1, \dots, 1, 1\big) + 1 \\
&= \mathsf{pow}\big(\mathsf{len}(\mathbf{s}^n), \mathsf{len}(\mathbf{s}^{n-1}), \dots, \mathsf{len}(\mathbf{s}^k)\big) + 1 = 1 + \mathsf{len}(\mathbf{R}(1)),
\end{aligned}$$

as required.

Case 4. Suppose that $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$. Let α and k be such that $\delta(\gamma, p)$ performs push_α^k; this transition does not read anything, so $\sharp(\mathsf{st}(\mathbf{R})) = 0$. By Definition 7.11(4) we have a composer $(\Phi^k, \Phi^{k-1}, \dots, \Phi^0; \Psi^k; f)$ such that $\pi_2(\Phi^0) = \{\mathsf{rd}(E) \colon E \in \mathfrak{D}\}$. Denote also $\Psi^i = \mathsf{ass}^i(\mathsf{rd}(D'))$ for $i \in [1, k-1]$. By Definition 7.17(4),

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

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 7.31 we obtain the following inequality, which is strict if $f = \mathsf{pr}$:

$$\sum_{i=1}^{k} \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Phi^i)}) + \mathsf{low}((\gamma, \mathfrak{D})) \leq \mathsf{low}(\mathbf{t}^k). \tag{7.29}$$

Because $\mathbf{R}(0)$ is well-formed, $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\mathsf{rd}(D)))$ for all $i \in [1, n]$ (cf. Proposition 7.14). By Definition 7.11(4), $\mathsf{ass}^k(\sigma) = \Phi^k$, and $\mathsf{ass}^i(\sigma) = \Psi^i \cup \Phi^i$ for $i \in [1, k-1]$. In effect, $\mathsf{type}(\mathbf{s}^k) = \pi_2(\Phi^k)$ and $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i) \cup \pi_2(\Phi^i)$ for $i \in [1, k-1]$, so

$$\mathsf{low}(\mathbf{s}^k) = \mathsf{low}(\mathbf{s}^k\!\restriction_{\pi_2(\Phi^k)}), \qquad\qquad\qquad \text{and} \qquad\qquad (7.30)$$

$$\mathsf{low}(\mathbf{s}^i) = \sum_{\sigma \in \mathsf{type}(\mathbf{s}^i)} \mathsf{low}(\mathbf{s}^i\!\restriction_{\{\sigma\}})$$

$$\leq \sum_{\sigma \in \pi_2(\Psi^i)} \mathsf{low}(\mathbf{s}^i\!\restriction_{\{\sigma\}}) + \sum_{\sigma \in \pi_2(\Phi^i)} \mathsf{low}(\mathbf{s}^i\!\restriction_{\{\sigma\}})$$

$$= \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Psi^i)}) + \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Phi^i)}) \qquad\qquad \text{for } i \in [1, k-1], \qquad (7.31)$$

where the equalities in the second formula are by Proposition 7.28. Moreover, if $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \not\subseteq \mathcal{T}_{\mathsf{np}}$ for some $i \in [1, k-1]$, then the appropriate inequality is strict (since the component corresponding to $\tau \in \pi_2(\Psi^i) \cap \pi_2(\Phi^i) \cap \mathcal{T}_{\mathsf{pr}}$, which is positive by Proposition 7.27, appears in both sums on the right side, and only once on the left side). Because $\mathbf{R}(0)$ and $\mathbf{R}(1)$ are singular,

$$\mathsf{low}(\mathbf{R}(0)) = \sum_{i=0}^{n} \mathsf{low}(\mathbf{s}^i), \qquad\qquad\qquad\qquad \text{and}$$

$$\mathsf{low}(\mathbf{R}(1)) = \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) + \mathsf{low}(\mathbf{t}^k) + \sum_{i=1}^{k-1} \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Psi^i)}) + \mathsf{low}((\alpha, \{D'\}))$$

by Proposition 7.29. We apply (In)equalities (7.30) and (7.31) to the formula for $\mathsf{low}(\mathbf{R}(0))$; next, we substitute Inequality (7.29); we obtain

$$\mathsf{low}(\mathbf{R}(0)) \leq \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) + \sum_{i=1}^{k-1} \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Psi^i)}) + \sum_{i=1}^{k} \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Phi^i)}) + \mathsf{low}(\mathbf{s}^0) \leq$$

$$\leq \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) + \sum_{i=1}^{k-1} \mathsf{low}(\mathbf{s}^i\!\restriction_{\pi_2(\Psi^i)}) + \mathsf{low}(\mathbf{t}^k) - \mathsf{low}((\gamma, \mathfrak{D})) + \mathsf{low}(\mathbf{s}^0) =$$

$$= \mathsf{low}(\mathbf{R}(1)) - \mathsf{low}((\alpha, \{D'\})) - \mathsf{low}((\gamma, \mathfrak{D})) + \mathsf{low}(\mathbf{s}^0) \leq \mathsf{low}(\mathbf{R}(1)) + \mathsf{low}(\mathbf{s}^0).$$

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 of 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, then $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ 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 7.11(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

$$a_i = \mathsf{high}(\mathbf{s}^i) \qquad\qquad\qquad\qquad \text{for } i \in [k+1, n],$$

$$a_i = \mathsf{high}(\mathbf{s}^i\!\restriction_{\pi_2(\Psi^i)}) \qquad\qquad\qquad \text{for } i \in [1, k-1],$$

$$b_i = \mathsf{high}(\mathbf{s}^i\!\restriction_{\pi_2(\Phi^i)}) \qquad\qquad\qquad \text{for } i \in [1, k].$$

Suppose first that $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{np}}$. Then, by Definition 7.11(4), $\{\mathsf{rd}(D')\} \cup \pi_2(\Phi^0) \subseteq \mathcal{T}_{\mathsf{np}}$, and $f = \mathsf{np}$, and $\pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ for each $i \in [1, k-1]$. In consequence, $\mathsf{high}(\mathbf{s}^0) =$

$\mathsf{high}((\gamma, \mathfrak{D})) = \mathsf{high}((\alpha, \{D'\})) = 1$. Recall that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i) \cup \pi_2(\Phi^i)$ for $i \in [1, k-1]$. Thus, by Proposition 7.28,

$$\mathsf{high}(\mathbf{s}^i) = \prod_{\sigma \in \mathsf{type}(\mathbf{s}^i)} \mathsf{high}(\mathbf{s}^i\!\upharpoonright_{\{\sigma\}}) = \prod_{\sigma \in \pi_2(\Psi^i)} \mathsf{high}(\mathbf{s}^i\!\upharpoonright_{\{\sigma\}}) \cdot \prod_{\sigma \in \pi_2(\Phi^i)} \mathsf{high}(\mathbf{s}^i\!\upharpoonright_{\{\sigma\}})$$
$$= a_i \cdot b_i \qquad\qquad\qquad\qquad \text{for } i \in [1, k-1]\,; \qquad (7.32)$$

the second equality above holds because for $\sigma \in \pi_2(\Psi^i) \cap \pi_2(\Phi^i) \subseteq \mathcal{T}_{\mathsf{np}}$ we have $\mathsf{high}(\mathbf{s}^i\!\upharpoonright_{\{\sigma\}}) = 1$ (cf. Proposition 7.27). Because $f = \mathsf{np}$, from Lemma 7.31 we know that

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

Using Proposition 7.29 and Equalities (7.32), then Inequality (7.12), then the above inequality, and then again Proposition 7.29, we obtain

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

Next, suppose that $\mathsf{rd}(D) \in \mathcal{T}_{\mathsf{pr}}$. Then Lemma 7.31 gives us the inequality

$$\mathsf{pow}(b_k, b_{k-1}, \ldots, b_1, |\mathcal{T}^0|^n \cdot \mathsf{high}((\gamma, \mathfrak{D}))) \geq \mathsf{high}(\mathbf{t}^k)\,. \qquad (7.33)$$

By definition it holds

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

Using Inequality (7.13) we replace 2^{k-1} in the last argument of pow by 2 in the $k-1$ previous arguments; then we observe that $(\mathsf{high}(\mathbf{s}^i))^2 \geq a_i b_i$ for each $i \in [1, k-1]$; then we use Inequality (7.12), and finally Inequality (7.33):

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

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

7.3. **Relating Upper and Lower Bounds.** As already mentioned, it is meaningful to consider the functions low and high because they are closely related: one is bounded if the other is bounded. This is shown in the following proposition.

Proposition 7.32. *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 7.21.

By definition of a type, for each configuration c and each run descriptor $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(c)$ there exists a well-formed annotated n-stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{conf}(\mathbf{s}) = c$. We can assume without loss of generality that all derivation trees in $\mathbf{s}$ 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 7.27. 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 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

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

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 7.33. *Let* $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^0$ *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* $\mathsf{top}^r(\mathbf{R}(|\mathbf{R}|)) = \mathbf{s}^r\!\upharpoonright_{\{\xi\}}$.

Proof. We use induction on $\mathsf{len}(\mathbf{s})$. Thanks to Lemma 7.22 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 6.9.

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

Case 2. Suppose that $D = (\mathsf{read}, p, D')$. 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 $\mathsf{conf}(\mathbf{t})$. Because $(m, \xi) \in \mathsf{ass}^r(\sigma)$, there exists 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 $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r\!\upharpoonright_{\{\xi\}}$. Together with the step between $\mathbf{s}$ and $\mathbf{t}$, it gives us an annotated run as required.

Case 3. Suppose that $D = (\mathsf{pop}, \gamma, p, \tau)$, where $\tau \in \mathcal{T}^k$. The successor of $\mathbf{s}$ is $\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$. Recall that $\mathsf{ass}^i(\sigma) = \emptyset$ for $i < k$, so $r \geq k$. Moreover, $\mathsf{ass}^k(\sigma) = \{(\mathbf{1}_M, \tau)\}$, and $\mathsf{type}(\mathbf{s}^k) = \pi_2(\mathsf{ass}^k(\sigma)) = \{\tau\}$ by well-formedness of $\mathbf{s}$ (cf. Proposition 7.14). If $r = k$, then $(m, \xi) = (\mathbf{1}_M, \tau)$. In this case the annotated run of length 1 satisfies the thesis. Otherwise $r > k$, and $(m, \xi) \in \mathsf{ass}^r(\sigma) = \mathsf{ass}^r(\tau)$. Then as well $(m, \xi) \in \mathsf{ass}^r(\tau')$, where τ' is the run descriptor in the type of $\mathsf{top}^0(\mathbf{s}^k)$ (since $\tau = \mathsf{red}^k(\tau')$). 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 $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r\!\upharpoonright_{\{\xi\}}$. Together with the step between $\mathbf{s}$ and $\mathbf{t}$, it gives us an annotated run as required.

Case 4. Suppose that $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$. Let α, k, Ψ^i, Φ^i be as in Definition 7.11(4). The successor of $\mathbf{s}$ is
$$\mathbf{t} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{t}^k : \mathbf{s}^{k-1}\!\upharpoonright_{\pi_2(\Psi^{k-1})} : \mathbf{s}^{k-2}\!\upharpoonright_{\pi_2(\Psi^{k-2})} : \cdots : \mathbf{s}^1\!\upharpoonright_{\pi_2(\Psi^1)} : (\alpha, \{D'\}),$$
where $\mathbf{t}^k = \mathbf{s}^k\!\upharpoonright_{\pi_2(\Phi^k)} : \mathbf{s}^{k-1}\!\upharpoonright_{\pi_2(\Phi^{k-1})} : \cdots : \mathbf{s}^1\!\upharpoonright_{\pi_2(\Phi^1)} : (\gamma, \mathfrak{D})$. Recall from our previous proofs that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i)$ for $i \in [k+1, n]$, and $\mathsf{type}(\mathbf{t}^k) = \pi_2(\Psi^k)$. 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 $\mathsf{top}^r(\mathbf{S}(|\mathbf{S}|)) = \mathbf{s}^r\!\upharpoonright_{\{\xi\}}$; 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 $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$. The definition of a composer (Condition (C1)) gives us some $(m_1, \tau) \in \Phi^0$ and $m_2 \in M$ such that $m = m_1 \cdot m_2$ and $(m_2, \xi) \in \mathsf{ass}^r(\tau)$. We see that $(m_1, \mathsf{red}^k(\tau)) \in \Psi^k$ (by Condition (C2) of the definition). 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 $\mathsf{top}^k(\mathbf{S}(|\mathbf{S}|)) = \mathbf{t}^k\!\upharpoonright_{\{\mathsf{red}^k(\tau)\}}$. By Proposition 7.30,
$$\mathbf{t}^k\!\upharpoonright_{\{\mathsf{red}^k(\tau)\}} = \mathbf{s}^k\!\upharpoonright_{\pi_2(\mathsf{ass}^k(\tau))} : \mathbf{s}^{k-1}\!\upharpoonright_{\pi_2(\mathsf{ass}^{k-1}(\tau))} : \cdots : \mathbf{s}^1\!\upharpoonright_{\pi_2(\mathsf{ass}^1(\tau))} : (\gamma, \mathfrak{D})\!\upharpoonright_{\{\tau\}}.$$

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. $\qquad\square$

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. This is shown in Lemma 7.34, which is a converse of Lemma 7.33. While reversing Lemma 7.33 we have to remember that not every run can be extended to an annotated run, so in Lemma 7.34 we need to use runs, not annotated runs.

Lemma 7.34. *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 $(\varphi(R), \mathsf{red}^r(\xi)) \in \mathsf{ass}^r(\sigma)$.*

Before proving Lemma 7.34, we first state four auxiliary lemmas. These lemmas are used not only in the proof of Lemma 7.34, but also in the next subsection. Actually, every of these lemmas has a part denoted by $(\star)$; these parts are needed only in the next subsection.

Lemma 7.35. *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 $i \in [1, n]$. Moreover, there exists a well-formed singular annotated 0-stack $\mathbf{v}^0$ such that $\mathsf{type}(\mathbf{v}^0) = \{\sigma\}$, and the following is satisfied.*

$(\star)$ *Let $\mathbf{s}'$ be a well-formed annotated n-stack with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Then there exists an annotated run $\mathbf{S}$ of length 1 such that $\mathbf{S}(0) = \mathbf{s}'$, the transition of $\mathsf{st}(\mathbf{S})$ is read, it reads the same letter as the transition of R, and $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\tau\}$.*

Proof. By definition of $\mathsf{type}_{\mathcal{A},\varphi}$, there exists a well-formed annotated stack $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\gamma, \{D'\})$ such that $\mathsf{rd}(D') = \tau$ and $\mathsf{conf}(\mathbf{s}) = R(1)$. Well-formedness of $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\tau))$ for each $i \in [1, n]$ (cf. Proposition 7.14). When p is the state of $R(0)$, Definition 7.11(2) implies that $D = (\mathsf{read}, p, D')$ 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 (again, cf. Proposition 7.14), so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$.

In order to prove Property $(\star)$, as $\mathbf{v}^0$ we take $(\gamma, \{D\})$. Clearly $\mathsf{type}(\mathbf{v}^0) = \{\mathsf{rd}(D)\} = \{\sigma\}$. Consider now any well-formed annotated n-stack $\mathbf{s}'$ with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Let $\mathbf{S}$ be the annotated run from $\mathbf{s}'$ to its successor. Because the topmost 0-stack of $\mathbf{s}'$ is annotated by $D = (\mathsf{read}, p, D')$, the successor of $\mathbf{s}'$ indeed exists, and the transition of $\mathsf{st}(\mathbf{S})$ is read. The state of $R(1)$ and the state of $\mathsf{conf}(\mathbf{S}(1))$ are the same (namely $\pi_1(\mathsf{rd}(D'))$), so $\mathsf{st}(\mathbf{S})$ reads the same letter as R. Moreover, $\mathsf{top}^0(\mathbf{S}(1)) = (\gamma, \{D'\})$, so $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\mathsf{rd}(D')\} = \{\tau\}$, as required. $\qquad\square$

Lemma 7.36. *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 $\mathsf{ass}^k(\sigma) = \{(\mathbf{1}_M, \mathsf{red}^k(\tau))\}$. Moreover, there exists a well-formed singular annotated k-stack $\mathbf{v}^k$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{v}^k)) = \{\sigma\}$, and the following is satisfied.*

($\star$) *Let* $\mathbf{s}'$ *be a well-formed annotated n-stack with* $\mathsf{top}^k(\mathbf{s}') = \mathbf{v}^k$. *Then there exists an annotated run* $\mathbf{S}$ *of length 1 such that* $\mathbf{S}(0) = \mathbf{s}'$, *and* $\mathsf{st}(\mathbf{S})$ *performs* pop^k, *and* $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\tau\}$.

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 $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^k$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s}^k)) = \{\tau\}$, and $\mathsf{st}(\mathbf{s}^i) = \mathsf{pos}{\downarrow}(s^i)$ for each $i \in [k, n]$. Then, by Proposition 7.14, $\mathsf{type}(\mathbf{s}^k) = \{\mathsf{red}^k(\tau)\}$. Well-formedness of $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\mathsf{ass}^i(\tau))$ for each $i \in [k+1, n]$ (cf. Proposition 7.14). 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 $\mathsf{st}(\mathbf{s}^i) = \mathsf{pos}{\downarrow}(s^i)$ (we annotate s^i by empty sets). Finally, by Definition 7.11(3), $D = (\mathsf{pop}, s^0, p, \mathsf{red}^k(\tau))$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where $\gamma = \mathsf{pos}{\downarrow}(s^0)$ and

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

so $\mathbf{s}^0 = (\gamma, \{D\})$ has type $\{\sigma\}$. Using Proposition 7.14 we observe that $s^n : s^{n-1} : \cdots : s^0$ is well-formed, so $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$.

In order to prove Property ($\star$), as $\mathbf{v}^k$ we take $\mathbf{s}^k : \mathbf{s}^{k-1} : \cdots : \mathbf{s}^0$. Clearly $\mathsf{type}(\mathsf{top}^0(\mathbf{v}^k)) = \mathsf{type}(\mathbf{s}^0) = \{\sigma\}$. Consider now any well-formed annotated n-stack $\mathbf{s}'$ with $\mathsf{top}^k(\mathbf{s}') = \mathbf{v}^k$. Let $\mathbf{S}$ be the annotated run from $\mathbf{s}'$ to its successor. Because the topmost 0-stack of $\mathbf{s}'$ is annotated by D, the successor of $\mathbf{s}'$ indeed exists, and $\mathsf{st}(\mathbf{S})$ performs pop^k. Moreover, $\mathsf{top}^k(\mathbf{S}(1)) = \mathbf{s}^k$, so $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \mathsf{type}(\mathsf{top}^0(\mathbf{s}^k)) = \{\tau\}$. $\square$

Lemma 7.37. *Let R be a run of length 1 performing push_α^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 $\mathsf{ass}^i(\tau) \subseteq \mathsf{ass}^i(\sigma)$ for each $i \in [1, n] \setminus \{k\}$. Moreover, there exists a well-formed singular annotated 0-stack $\mathbf{v}^0$ such that $\mathsf{type}(\mathbf{v}^0) = \{\sigma\}$, and the following is satisfied.*

($\star$) *Let $\mathbf{s}'$ be a well-formed annotated n-stack with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Then there exists an annotated run $\mathbf{S}$ of length 1 such that $\mathbf{S}(0) = \mathbf{s}'$, and $\mathsf{st}(\mathbf{S})$ performs push_α^k, and $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\tau\}$.*

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 $R(0) = (p, s^n : s^{n-1} : \cdots : s^1 : (\gamma, x))$; then $\pi_2(R(1))$ equals

$$s^n : s^{n-1} : \cdots : s^{k+1} : (s^k : s^{k-1} : \cdots : s^1 : (\gamma, x)) : \mathsf{p}_{+1}(s^{k-1} : s^{k-2} : \cdots : s^1 : (\alpha, x)).$$

By definition of $\mathsf{type}_{\mathcal{A},\varphi}$, there exists a well-formed annotated stack $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\alpha, \{D'\})$ in which $\mathbf{s}^k = \mathbf{t}^k : \mathbf{t}^{k-1} : \cdots : \mathbf{t}^1 : (\gamma, \mathfrak{D})$, such that $\mathsf{rd}(D') = \tau$, and $\mathsf{st}(\mathbf{s}^i) = \mathsf{pos}{\downarrow}(s^i)$ for each $i \in [1, n] \setminus \{k\}$, and $\mathsf{st}(\mathbf{t}^i) = \mathsf{pos}{\downarrow}(s^i)$ for each $i \in [1, k]$. Denote $\Psi^i = \mathsf{ass}^i(\tau)$ for each $i \in [1, n]$. Well-formedness of $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i)$ for each $i \in [1, n]$ (cf. Proposition 7.14), and, thanks to Proposition 7.15, there exists a composer $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$ 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 7.11(4), $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where

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

Using Proposition 7.14 we observe that the annotated stack

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

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

In order to prove Property $(\star)$, as $\mathbf{v}^0$ we take $(\gamma, \{D\})$. Clearly $\mathsf{type}(\mathbf{v}^0) = \{\mathsf{rd}(D)\} = \{\sigma\}$. Consider now any well-formed annotated n-stack $\mathbf{s}'$ with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Let $\mathbf{S}$ be the annotated run from $\mathbf{s}'$ to its successor. Because the topmost 0-stack of $\mathbf{s}'$ is annotated by D, the successor of $\mathbf{s}'$ indeed exists, and $\mathsf{st}(\mathbf{S})$ performs push_α^k. Moreover, $\mathsf{top}^0(\mathbf{S}(1)) = (\alpha, \{D'\})$, so $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\mathsf{rd}(D')\} = \{\tau\}$. $\qquad\square$

Lemma 7.38. *Let R be a run in which $R{\restriction}_{0,1}$ performs push_α^k 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 $i \in [1, k]$. Moreover, there exists a well-formed singular annotated 0-stack $\mathbf{v}^0$ such that $\mathsf{type}(\mathbf{v}^0) = \{\sigma\}$, and the following is satisfied.*

$(\star)$ *Let $\mathbf{s}'$ be a well-formed annotated n-stack with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Then there exists an annotated run $\mathbf{S}$ of length 1 such that $\mathbf{S}(0) = \mathbf{s}'$, and $\mathsf{st}(\mathbf{S})$ performs push_α^k, and $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\rho\}$, and $\tau \in \mathsf{type}(\mathsf{top}^0(\mathsf{pop}^k(\mathbf{S}(1))))$.*

Proof. Denote $R(0) = (p, s^n : s^{n-1} : \cdots : s^1 : (\gamma, x))$; then $\pi_2(R(1))$ is as in the previous lemma, and $\mathsf{top}^k(R(0)) \cong \mathsf{top}^k(R(|R|))$ (due to Proposition 6.7). The definition of $\mathsf{type}_{\mathcal{A}, \varphi}$ gives us a well-formed annotated n-stack $\mathbf{u}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{u})) = \{\tau\}$ and $\mathsf{conf}(\mathbf{u}) = R(|R|)$, and a well-formed annotated stack $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^1 : (\alpha, \{D'\})$ such that $\mathsf{rd}(D') = \rho$ and $\mathsf{conf}(\mathbf{s}) = R(1)$. Denote $\mathbf{u}^k = \mathsf{top}^k(\mathbf{u})$. Then $\mathsf{type}(\mathsf{top}^0(\mathbf{u}^k)) = \{\tau\}$, and $\mathsf{st}(\mathbf{u}^k) = \mathsf{pos}{\downarrow}(s^k : s^{k-1} : \cdots : s^1 : (\gamma, x))$, and $\mathsf{st}(\mathbf{s}^i) = \mathsf{pos}{\downarrow}(s^i)$ for each $i \in [1, n] \setminus \{k\}$, and $\mathsf{st}(\mathbf{s}^k) = \mathsf{pos}{\downarrow}(s^k : s^{k-1} : \cdots : s^1 : (\gamma, x))$. Denote $\Psi^i = \mathsf{ass}^i(\rho)$ for each $i \in [1, n]$. Well-formedness of $\mathbf{s}$ implies that $\mathsf{type}(\mathbf{s}^i) = \pi_2(\Psi^i)$ for each $i \in [1, n]$ (cf. Proposition 7.14). By Proposition 7.14, $\mathsf{type}(\mathbf{u}^k) = \{\mathsf{red}^k(\tau)\}$. Thanks to the assumption $\mathsf{red}^k(\tau) \in \pi_2(\mathsf{ass}^k(\rho))$ we have that $\mathsf{type}(\mathbf{u}^k) \subseteq \pi_2(\Psi^k)$. In effect, the annotated stack $\mathbf{u}^k \oplus \mathbf{s}^k$ has type $\pi_2(\Psi^k)$, equal to the type of $\mathbf{s}^k$, but additionally $\tau \in \mathsf{type}(\mathsf{top}^0(\mathbf{u}^k \oplus \mathbf{s}^k))$ (recalling the construction from the previous proof, 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 $\mathbf{u}^k \oplus \mathbf{s}^k = \mathbf{t}^k : \mathbf{t}^{k-1} : \cdots : \mathbf{t}^1 : (\gamma, \mathfrak{D})$. By Proposition 7.15 we have a composer $(\Phi^k, \Phi^{k-1}, \ldots, \Phi^0; \Psi^k; f)$ 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 $\tau \in \pi_2(\Phi^0)$ and $(\varphi(R), \mathsf{red}^k(\tau)) \in \Psi^k$, it holds $(\varphi(R), \tau) \in \Phi^0$ (thanks to Conditions (C2) and (C3) of the definition of a composer), which implies $\varphi(R) \circ \mathsf{ass}^i(\tau) \subseteq \Phi^i$ for each $i \in [1, k]$ (thanks to Condition (C1) of the definition). By Definition 7.11(4), $D = (\mathsf{push}, \gamma, p, D', \mathfrak{D})$ is a derivation tree with conclusion $\gamma \vdash \sigma$, where

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

We observe that the annotated stack

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

is well-formed (by Proposition 7.14), so $\sigma \in \mathsf{type}_{\mathcal{A}, \varphi}(R(0))$.

In order to prove Property $(\star)$, as $\mathbf{v}^0$ we take $(\gamma, \{D\})$. Clearly $\mathsf{type}(\mathbf{v}^0) = \{\mathsf{rd}(D)\} = \{\sigma\}$. Consider now any well-formed annotated n-stack $\mathbf{s}'$ with $\mathsf{top}^0(\mathbf{s}') = \mathbf{v}^0$. Let $\mathbf{S}$ be the annotated run from $\mathbf{s}'$ to its successor. Because the topmost 0-stack of $\mathbf{s}'$ is annotated by D, the successor of $\mathbf{s}'$ indeed exists, and $\mathsf{st}(\mathbf{S})$ performs push_α^k. Moreover, $\mathsf{top}^0(\mathbf{S}(1)) = (\alpha, \{D'\})$, so $\mathsf{type}(\mathsf{top}^0(\mathbf{S}(1))) = \{\mathsf{rd}(D')\} = \{\rho\}$. On the other hand, $\mathsf{top}^0(\mathsf{pop}^k(\mathbf{S}(1)))$ is $(\gamma, \mathfrak{D})$, so its type is $\{\mathsf{rd}(E)\colon E \in \mathfrak{D}\} = \pi_2(\Phi^0)$, and we know that $\tau \in \pi_2(\Phi^0)$. $\qquad\square$

Proof of Lemma 7.34. Recall that we are given an r-return R, and a run descriptor $\xi \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$, and we have to show existence of 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 use induction on the length of the r-return R. Proposition 6.9 gives us possible forms of R; we analyze these cases.

Suppose first that $|R| = 1$ and the only transition of R performs pop^r. We take σ from Lemma 7.36, 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_α^k 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 7.35, or 7.36, or 7.37, 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_α^k 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 7.38 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 $(\varphi(R), \mathsf{red}^r(\xi)) = (\varphi(S) \circ \varphi(T), \mathsf{red}^r(\xi)) \in \varphi(S) \circ \mathsf{ass}^r(\tau) \subseteq \mathsf{ass}^r(\sigma)$. $\qquad\square$

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 6.8 we can also describe r-upper runs. This is stated in the following lemma.

Lemma 7.39. *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.*

$(\star)$ *Let $\mathbf{s}$ be a well-formed annotated n-stack such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{top}^r(\mathsf{conf}(\mathbf{s})) \cong \mathsf{top}^r(R(0))$. 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 $\mathsf{conf}(\mathbf{t})$ that is (r, φ)-parallel to R, and such that*

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

The idea staying behind a proof of this lemma is that the run R can be split into parts of two kinds. First, we have parts for which the topmost r-stack of $R(0)$ is responsible. Since in $\mathsf{conf}(\mathbf{s})$ the topmost r-stack is the same, we can execute them also from $\mathsf{conf}(\mathbf{s})$. Second, we have parts not controlled by the topmost r-stack of $R(0)$, but (according to Proposition 6.8) these are returns. Analogous returns can be executed from $\mathsf{conf}(\mathbf{s})$, because of the run descriptor σ in $\mathsf{type}(\mathsf{top}^0(\mathbf{s}))$.

Two aspects of the statement of the lemma can be understood basing on the above idea. First, the correction function f_R is really needed (i.e., the lemma would be false if the identity function was always taken as f_R). Second, there does not need to exist an annotated run from $\mathbf{s}$ to $\mathbf{t}$; we only prove existence of a (non-annotated) run from $\mathsf{conf}(\mathbf{s})$ to $\mathsf{conf}(\mathbf{t})$. The justification of both these phenomena is the same: while creating the run from $\mathsf{conf}(\mathbf{s})$, we completely ignore the annotations contained in the topmost r-stack of $\mathbf{s}$ (while an annotated run from $\mathbf{s}$ necessarily follows them); instead, as long as the topmost r-stack of $\mathsf{st}(\mathbf{s})$ controls the run, we copy steps of the run R (only after leaving the topmost r-stack, we start using the annotations from $\mathbf{s}$). In a sense, f_R describes how much can be lost while ignoring annotations in the topmost r-stack of $\mathbf{s}$ (recall that, while ignoring annotation, this r-stack is the same as in $R(0)$, thus fixed; only the annotations are not fixed).

Before proving Lemma 7.39 we show how Theorem 7.3 follows from it. For this purpose the inequalities regarding low and high are redundant; they are 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 positionless topmost k-stack as $R(0)$. Consider the run descriptor $\tau = (\pi_1(R(|R|)), \emptyset, \ldots, \emptyset, \mathsf{np})$ and observe that $\tau \in \mathsf{type}_{\mathcal{A},\varphi}(R(|R|))$ (we annotate the topmost 0-stack of $R(|R|)$ by the derivation tree from Definition 7.11(1)). Applying Lemma 7.39 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 $\mathsf{top}^k(\mathsf{conf}(\mathbf{s})) \cong \mathsf{top}^k(R(0))$, from Property $(\star)$ of Lemma 7.39 we obtain a run S that starts in c and is (k, φ)-parallel to R, as required. $\qquad \square$

Below, we give an auxiliary lemma, showing how to construct a function f_R needed for Lemma 7.39.

Lemma 7.40. *Let $k \in [0, n]$, and let $\mathbf{v}^k$ be a well-formed singular annotated k-stack. Then there exists a monotone function $f_{\mathbf{v}^k} \colon \mathbb{N} \to \mathbb{N}$ such that for all well-formed singular annotated n-stacks $\mathbf{s}, \mathbf{s}'$ with $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{s}^k$ and $\mathbf{s}' = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{v}^k$ and $\mathsf{st}(\mathbf{s}^k) = \mathsf{st}(\mathbf{v}^k)$ it holds that $\mathsf{low}(\mathbf{s}) \leq f_{\mathbf{v}^k}(\mathsf{low}(\mathbf{s}'))$ and $f_{\mathbf{v}^k}(\mathsf{high}(\mathbf{s})) \geq \mathsf{high}(\mathbf{s}')$.*

Proof. We first define the function $f_{\mathbf{v}^k}$, and then we show that it satisfies the thesis. Suppose that k and $\mathbf{v}^k$ are fixed. We construct $f_{\mathbf{v}^k}(N)$ by induction on N. Consider some $N \in \mathbb{N}$. First, we ensure that $f_{\mathbf{v}^k}(N) \geq N + \mathsf{low}(\mathbf{s}^k)$ for all annotated k-stacks $\mathbf{s}^k$ such that $\mathsf{st}(\mathbf{s}^k) = \mathsf{st}(\mathbf{v}^k)$. This is possible, because $\mathsf{low}(\mathbf{s}^k)$ equals the number of productive run descriptors altogether in the types of all 0-stacks in $\mathbf{s}^k$. So, although there are infinitely many annotated k-stacks $\mathbf{s}^k$ such that $\mathsf{st}(\mathbf{s}^k) = \mathsf{st}(\mathbf{v}^k)$, the value of $\mathsf{low}(\mathbf{s}^k)$ is bounded by the number of 0-stacks in $\mathsf{st}(\mathbf{s}^k)$ times $|\mathcal{T}^0|$. Next, we ensure that $f_{\mathbf{v}^k}(N) \geq \mathsf{pow}(a_n, a_{n-1}, \ldots, a_{k+1}, \mathsf{high}(\mathbf{v}^k))$ for all tuples $(a_n, a_{n-1}, \ldots, a_{k+1}, a_k)$ of positive integers such that $\mathsf{pow}(a_n, a_{n-1}, \ldots, a_{k+1}, a_k) = N$. Notice that there are only finitely many such tuples (in particular, none of a_i may be greater than N). Finally, we ensure that $f_{\mathbf{v}^k}(N) \geq f_{\mathbf{v}^k}(N - 1)$ (unless $N = 0$), in order to ensure monotonicity of $f_{\mathbf{v}^k}$ (since we are defining $f_{\mathbf{v}^k}$ by induction, $f_{\mathbf{v}^k}(N - 1)$ is already defined).

Consider now well-formed singular annotated n-stacks $\mathbf{s}, \mathbf{s}'$ such that $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{s}^k$, and $\mathbf{s}' = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{v}^k$, and $\mathsf{st}(\mathbf{s}^k) = \mathsf{st}(\mathbf{v}^k)$. Using Proposition 7.29

and properties of $f_{\mathbf{v}^k}$ ensured in its definition, we obtain the required inequalities:

$$\mathsf{low}(\mathbf{s}) = \mathsf{low}(\mathbf{s}^k) + \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i) \leq f_{\mathbf{v}^k}\Big(\sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i)\Big)$$

$$\leq f_{\mathbf{v}^k}\Big(\mathsf{low}(\mathbf{v}^k) + \sum_{i=k+1}^{n} \mathsf{low}(\mathbf{s}^i)\Big) = f_{\mathbf{v}^k}(\mathsf{low}(\mathbf{s}')) ,$$

$$f_{\mathbf{v}^k}(\mathsf{high}(\mathbf{s})) = f_{\mathbf{v}^k}\big(\mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \ldots, \mathsf{high}(\mathbf{s}^{k+1}), \mathsf{high}(\mathbf{s}^k)\big)\big)$$

$$\geq \mathsf{pow}\big(\mathsf{high}(\mathbf{s}^n), \mathsf{high}(\mathbf{s}^{n-1}), \ldots, \mathsf{high}(\mathbf{s}^{k+1}), \mathsf{high}(\mathbf{v}^k)\big) = \mathsf{high}(\mathbf{s}') . \qquad \square$$

Proof of Lemma 7.39. The proof is by induction on the length of the r-upper run R. Proposition 6.8 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 as S we take the run of length 0 from $\mathsf{conf}(\mathbf{s})$.

Suppose that R has length 1, and its transition either is read or performs push_α^k. Then we construct σ and $\mathbf{v}^0$ out of R and τ as in Lemma 7.35 or Lemma 7.37, respectively. Recall that $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$, as needed. As f_R we take the function $f_{\mathbf{v}^0}$ constructed in Lemma 7.40 for the annotated 0-stack $\mathbf{v}^0$. Next, we are given a well-formed annotated n-stack $\mathbf{s}$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{top}^r(\mathsf{conf}(\mathbf{s})) \cong \mathsf{top}^r(R(0))$. Let $\mathbf{s}'$ be the annotated n-stack obtained from $\mathbf{s}$ by replacing its topmost 0-stack with $\mathbf{v}^0$. Because $\mathsf{type}(\mathbf{v}^0) = \{\sigma\} = \mathsf{type}(\mathsf{top}^0(\mathbf{s}))$, we have that $\mathbf{s}'$ is also well-formed. As $\mathbf{t}$ we take the successor of $\mathbf{s}'$, and as S the one-step run from $\mathsf{conf}(\mathbf{s}')$ (i.e., from $\mathsf{conf}(\mathbf{s})$) to $\mathsf{conf}(\mathbf{t})$. By Property $(\star)$ of Lemma 7.35 or Lemma 7.37, respectively, the successor of $\mathbf{s}'$ indeed exists, and $\mathsf{type}(\mathsf{top}^0(\mathbf{t})) = \{\tau\}$; moreover, the run 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. Recalling that f_R satisfies the thesis of Lemma 7.40, and using Lemma 7.22, we obtain the required inequalities:

$$\mathsf{low}(\mathbf{s}) \leq f_R(\mathsf{low}(\mathbf{s}')) \leq f_R(\sharp(S) + \mathsf{low}(\mathbf{t})) , \qquad\qquad \text{and}$$
$$f_R(\mathsf{high}(\mathbf{s})) \geq \mathsf{high}(\mathbf{s}') \geq \sharp(S) + \mathsf{high}(\mathbf{t}) .$$

Next, suppose that R has length 1 and performs pop^k for $k \leq r$. We construct σ and $\mathbf{v}^k$ out of R and τ as in Lemma 7.36. Recall that $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$, as needed. As f_R we take the function $f_{\mathbf{v}^k}$ constructed in Lemma 7.40 for the annotated k-stack $\mathbf{v}^k$. Then, we are given an annotated stack $\mathbf{s} = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{s}^k$ such that $\mathsf{type}(\mathsf{top}^0(\mathbf{s})) = \{\sigma\}$ and $\mathsf{top}^r(\mathsf{conf}(\mathbf{s})) \cong \mathsf{top}^r(R(0))$. Consider $\mathbf{s}' = \mathbf{s}^n : \mathbf{s}^{n-1} : \cdots : \mathbf{s}^{k+1} : \mathbf{v}^k$. Because $\mathsf{type}(\mathsf{top}^0(\mathbf{v}^k)) = \{\sigma\} = \mathsf{type}(\mathsf{top}^0(\mathbf{s}))$, by Proposition 7.14 we have that $\mathsf{type}(\mathbf{v}^k) = \{\mathsf{red}^k(\sigma)\} = \mathsf{type}(\mathsf{top}^k(\mathbf{s}))$; in effect $\mathbf{s}'$ is also well-formed. As $\mathbf{t}$ we take the successor of $\mathbf{s}'$, and as S the one-step run from $\mathsf{conf}(\mathbf{s}')$ to $\mathsf{conf}(\mathbf{t})$. By Property $(\star)$ of Lemma 7.36, the successor of $\mathbf{s}$ indeed exists, and $\mathsf{type}(\mathsf{top}^0(\mathbf{t})) = \{\tau\}$, and the transition of S performs pop^k. Clearly S is (r,φ)-parallel to R. The required inequalities are obtained in the same way as in the previous case, due to Lemmas 7.40 and 7.22.

Next, suppose that $R{\restriction}_{0,1}$ performs push_α^k 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 7.34, we construct a run descriptor $\rho \in \mathsf{type}_{\mathcal{A},\varphi}(R(1))$ such that $(\varphi(R), \mathsf{red}^k(\tau)) \in \mathsf{ass}^k(\rho)$. Then, we construct σ and $\mathbf{v}^0$ out of R, τ, and ρ as in Lemma 7.38. Recall that $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0))$, as needed. As f_R we take the function $f_{\mathbf{v}^0}$ constructed in Lemma 7.40 for the annotated 0-stack $\mathbf{v}^0$. When we are given $\mathbf{s}$, we proceed as follows. First, as $\mathbf{s}'$ we take the annotated n-stack

obtained from $\mathbf{s}$ by replacing its topmost 0-stack with $\mathbf{v}^0$. As in the previous cases, $\mathbf{s}'$ is well-formed. Let $\mathbf{S}$ be the one-step annotated run from $\mathbf{s}'$, and let $\mathbf{S}(1) = \mathbf{u}^n : \mathbf{u}^{n-1} : \cdots : \mathbf{u}^0$. By Property $(\star)$ of Lemma 7.38, $\mathbf{S}$ indeed exists, and $\mathsf{st}(\mathbf{S})$ performs push_α^k, and $\mathsf{type}(\mathbf{u}^0) = \{\rho\}$, and $\tau \in \mathsf{type}(\mathsf{top}^0(\mathbf{u}^k))$. Because $(\varphi(R), \mathsf{red}^k(\tau)) \in \mathsf{ass}^k(\rho)$, Lemma 7.33 gives us an annotated run $\mathbf{T}$ starting in $\mathbf{S}(1)$ such that $\mathsf{st}(\mathbf{T})$ is a k-return, $\varphi(\mathsf{st}(\mathbf{T})) = \varphi(R)$, and $\mathsf{top}^k(\mathbf{T}(|\mathbf{T}|)) = \mathbf{u}^k{\restriction}_{\mathsf{red}^k(\tau)}$. As S we take $\mathsf{st}(\mathbf{S} \circ \mathbf{T})$, and as $\mathbf{t}$ we take $\mathbf{T}(|\mathbf{T}|)$. Proposition 7.30 implies that $\{\tau\} = \mathsf{type}(\mathsf{top}^0(\mathbf{u}^k{\restriction}_{\mathsf{red}^k(\tau)})) = \mathsf{type}(\mathsf{top}^0(\mathbf{t}))$. By Proposition 6.7 we obtain that $\mathsf{top}^k(R(0)) \cong \mathsf{top}^k(R(|R|))$ and $\mathsf{top}^k(S(0)) \cong \mathsf{top}^k(S(|S|))$. Since $k \geq r + 1$, $\mathsf{top}^r(R(|R|)) \cong \mathsf{top}^r(S(|S|))$ as well. Together with $\varphi(S) = \varphi(S{\restriction}_{1,|S|}) = \varphi(R{\restriction}_{1,|R|}) = \varphi(R)$ this means that R and S are (r, φ)-parallel, because by definition no suffix of a k-return can be $(k-1)$-upper (r-upper). The inequalities are obtained as in the previous cases.

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 $\mathsf{top}^r(\mathsf{conf}(\mathbf{s})) \cong \mathsf{top}^r(R(0))$. 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

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

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

Definition 7.41. 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 (notice that we require the same type $\{\sigma\}$ for all i). 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 that $\mathsf{stype}((c_i)_{i=1}^\infty) = \mathsf{stype}((d_i)_{i=1}^\infty)$.

Proof of Theorem 7.4. Recall that we are given a run $R \circ R'$ in which R is k-upper and R' is an n-return; we are also given two infinite sequences of configurations $c_1, c_2, \ldots$ and $d_1, d_2, \ldots$ that are $(\mathcal{A}, \varphi)$-sequence-equivalent, and in which all configurations have the same $(\mathcal{A}, \varphi)$-type and the same positionless topmost k-stack as $R(0)$. Our goal is to construct, for each i, 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 the sequences $\sharp(S_1 \circ S_1'), \sharp(S_2 \circ S_2'), \ldots$ and $\sharp(T_1 \circ T_1'), \sharp(T_2 \circ T_2'), \ldots$ are either both bounded

or both unbounded. Let $\xi = (\pi_1(R'(|R'|)), \emptyset, \ldots, \emptyset, \mathsf{np})$. We see that $\xi \in \mathsf{type}_{\mathcal{A},\varphi}(R'(|R'|))$, because we can annotate the topmost 0-stack (γ, x) by $\{(\mathsf{empty}, \gamma, \pi_1(R'(|R'|)))\}$ and all other 0-stacks by $\emptyset$. Lemma 7.34 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 7.39 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 $\sigma \in \mathsf{stype}((c_i)_{i=1}^\infty)$ (hence also $\sigma \in \mathsf{stype}((d_i)_{i=1}^\infty)$). 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 positionless-equal, for each i. We use Property $(\star)$ of Lemma 7.39 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 and 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 7.33 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 $\mathsf{type}(\mathbf{S}_i'(|\mathbf{S}_i'|)) = \{\mathsf{red}^n(\xi)\}$. Let $S_i' = \mathsf{st}(\mathbf{S}_i')$, and $\mathbf{u}_i = \mathbf{S}_i'(|\mathbf{S}_i'|)$. Thanks to Lemma 7.22, $\mathsf{high}(\mathbf{t}_i) \geq \sharp(S_i') + \mathsf{high}(\mathbf{u}_i)$. Because $(\mathsf{high}(\mathbf{s}_i))_{i=1}^\infty$ is bounded, we see that the sequence $(\sharp(S_i \circ S_i'))_{i=1}^\infty$ is bounded as well.

We perform the same construction for $(d_i)_{i=1}^\infty$, obtaining runs $T_i \circ T_i'$ from d_i, such that $(\sharp(T_i \circ T_i'))_{i=1}^\infty$ is bounded.

Case 2. This is the opposite case: we suppose that $\sigma \notin \mathsf{stype}((c_i)_{i=1}^\infty)$. Recall that $\sigma \in \mathsf{type}_{\mathcal{A},\varphi}(R(0)) = \mathsf{type}_{\mathcal{A},\varphi}(c_i)$ for each i. Using Proposition 7.32 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 that $\mathsf{low}(\mathbf{s}_i) \leq f_R(\sharp(S_i) + \mathsf{low}(\mathbf{t}_i)) \leq f_R(\sharp(S_i \circ S_i') + \mathsf{low}(\mathbf{u}_i))$. Additionally $\mathsf{low}(\mathbf{u}_i) = 0$, because $\mathsf{type}(\mathbf{u}_i) = \mathsf{type}(\mathbf{S}_i'(|\mathbf{S}_i'|)) = \{\mathsf{red}^n(\xi)\} \subseteq \mathcal{T}_{\mathsf{np}}$ (cf. Proposition 7.27). It follows that $(\sharp(S_i \circ S_i'))_{i=1}^\infty$ is unbounded, and similarly $(\sharp(T_i \circ T_i'))_{i=1}^\infty$. $\qquad\square$

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). Locally, however, some parts of the stack might be constructed, and a few steps later removed. In order to handle this behavior, we concentrate on those configurations of the run in which the stack is minimal (in appropriate sense) and will not be destroyed later; they are called milestone configurations.

The idea of considering milestone configurations comes from Kartzow [Kar10], 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 all $i, j \in I$, $i \leq j$.

Example 8.2. Consider a DPDA of order 3. Suppose that there is a run that begins in a stack $\mathsf{pos}_+([[[a,a]]])$, and performs forever the following sequence of operations, in a loop:

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

Then the positionless topmost 2-stack is, alternately, $[[a,a]]$, or $[[a,a],[a,a]]$, or $[[a,a],[a]]$. This run does not read any symbols, so it is a degenerate case of an infinite run that reads only stars. Configurations with positionless topmost 2-stack $[[a,a]]$ 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 $[[a,a]]$ 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 indices i for which $R(i)$ is a milestone.

Lemma 8.3. *Let R be a run between two milestone configurations. If R reads only stars, then it 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 $k > 0$, take a run R between two milestone configurations, and suppose that it reads only stars. Let S be the infinite run that starts in $R(0)$ and 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 the run $S\restriction_{0,|R|} \circ S\restriction_{|R|,i}$. $\qquad\square$

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 β, assigning a natural number to every positionless l-stack, having the following property. Let R be a run that reads only stars, let $s_{|R|}^l$ be an l-stack of $R(|R|)$, and let $s_i^l = \mathsf{hist}(R\restriction_{i,|R|}, s_{|R|}^l)$ for all $i \in [0, |R|]$. If there exist at least $\beta(\mathsf{pos}\downarrow(s_0^l))$ indices i such that $s_i^l = \mathsf{top}^l(R(i))$, then for some index i the configuration $R(i)$ is a milestone and $s_i^l = \mathsf{top}^l(R(i))$.*

Corollary 8.5. *If R is an infinite run reading only stars, then for infinitely many indices 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 $\beta(\mathsf{pos}\downarrow(\pi_2(R(0))))$. We repeat this procedure for the remaining suffix of R.[4] $\qquad\square$

In order to get some intuitions on Lemma 8.4, let us first see why it works for $l = n$. In this case s_0^l is just the whole n-stack of $R(0)$. Moreover, the assumption that there exist at least $\beta(\mathsf{pos}\downarrow(s_0^l))$ indices i such that $s_i^l = \mathsf{top}^l(R(i))$ simply expresses that $|R| + 1 \geq \beta(\mathsf{pos}\downarrow(s_0^l))$.

---

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

Thus, the lemma says that if we have a long enough run that starts in a configuration with stack s_0^l and reads only stars, then the run reaches a milestone configuration. This, in turn, means that we cannot decrease the stack s_0^l forever. Indeed, recall the intuition that a milestone configuration is a minimal configuration, that is, such that the run reading only stars never visits a "smaller" configuration. It is just enough to consider the infinite run reading only stars, and take the „smallest" configuration visited by this run; this should be a milestone configuration.

When $l < n$, the lemma concentrates on the history of a single l-stack $s_{|R|}^l$ (another point of view is that it concentrates on the future of a single l-stack s_0^l). We look at the fragments of R where this l-stack is the topmost l-stack; the length of these fragments is required to be at least $\beta(\mathsf{pos}{\downarrow}(s_0^l))$ in total. The lemma says that regardless of what happens in other fragments of R, controlled by other parts of the n-stack of $R(0)$ (outside of s_0^l), the stack s_0^l can itself ensure that a milestone configuration is reached.

In the remaining part of the section we prove Lemma 8.4. Our proof strategy is as follows. The indices i for which s_i^l is 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 k-upper runs is 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 $\min I^k \leq i \leq j \in I^k$, then i belongs to 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 these k-upper runs cannot be further 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 that reads only stars, and let I^k be a k-advancing set. If $|I^k| \geq f_N^k(|\mathsf{top}^k(R(\min I^k))|)$, then there exists a $(k-1)$-advancing subset $I^{k-1} \subseteq I^k$ of size at least N, such that $\mathsf{pos}{\downarrow}(\mathsf{top}^{k-1}(R(\min I^{k-1})))$ is one of the $(k-1)$-stacks in $\mathsf{pos}{\downarrow}(\mathsf{top}^k(R(\min I^k)))$.*

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 $a := \min I^k$ and $r := |\mathsf{top}^k(R(\min I^k))|$. For each $j \in I^k$ denote

$$r_j := |\mathsf{top}^k(R(j))| \qquad \text{and} \qquad m_j := \min\{r_i \colon i \in I^k \wedge i \leq j\} \,.$$

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 $m_j = m$, by the pigeonhole principle (if for every m there were at most $f_{N-1}^k(m+1)$ such

indices, in total we would have at most $\sum_{m=1}^{r} f_{N-1}^k(m+1) = f_N^k(r) - 1$ indices in I^k, but we have at least $f_N^k(r)$ of them). Choose some such m; let b be the first index such that $m_b = m$, and e the last such index. We see that $m = r_b$.

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

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

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

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

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 $r_i \geq m_i = m = r_b$ for each $i \in I^k \cap [b, e]$ (hence, in particular, for each $i \in I^k \cap [b, \max J^{k-1}]$), 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 $\mathsf{pos}\!\downarrow(\mathsf{top}^{k-1}(R(b)))$ is one of the $(k-1)$-stacks in $\mathsf{pos}\!\downarrow(\mathsf{top}^k(R(a)))$. We know that $r_i \geq m_i > m = r_b$ for each $i \in I^k \cap [a, b-1]$, 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 ignoring positions annotating the stacks). $\qquad\square$

While using Theorem 7.3 we need to ensure that the replicated run reads only stars. For this reason, fix a finite monoid M, and a morphism $\varphi \colon A^* \to M$, such that its value $\varphi(w)$ determines whether a word w consists only of stars. Our second auxiliary lemma is 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 a composition for some r. Then Theorem 7.3 applied to this composition and to $S(|S|)$ (where we use the fact that $S(|S|)$ has the same $(\mathcal{A}, \varphi)$-type and the same positionless topmost 0-stack as $(S_1 \circ \cdots \circ S_r)(0)$) gives us a run that starts in $S(|S|)$ and is $(\mathcal{A}, \varphi)$-parallel to $S_1 \circ \cdots \circ S_r$. Recalling the definition of being $(\mathcal{A}, \varphi)$-parallel, we see that this run is a composition $S_1' \circ \cdots \circ S_r'$ of r nonempty 0-upper runs reading only stars. 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 in our alphabet Γ, times the number of $(\mathcal{A}, \varphi)$-types, plus two. Then, by the pigeonhole principle, 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
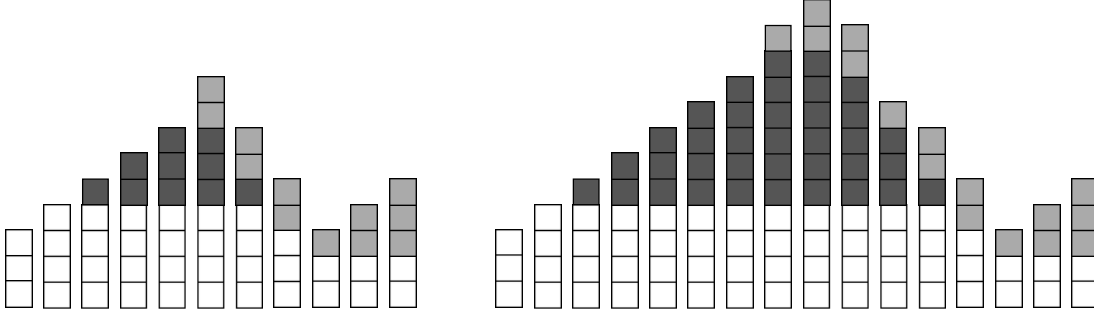
Figure 5: An example configuration at the end of a run of a 2-DPDA, and an analogous configuration after pumping. The 2-stack grows from left to right. White symbols were already present in $R(0)$; dark gray symbols were created while reading stars at the beginning of R; light gray symbols were created later.

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. $\qquad\square$

Proof of Lemma 8.4. Fix some l-stack s_0^l. Let N_0 be equal to the number of stack symbols in the alphabet, 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 $N_k = f_{N_{k-1}}^k(r_k)$, where r_k is the maximal size of a k-stack that appears in s_0^l, and $f_{N_{k-1}}^k$ is the function from Lemma 8.6. We define $\beta(\mathsf{pos}{\downarrow}(s_0^l)) := N_l$.

Now take a run R and l-stacks s_i^l for $i \in [1, |R|]$, 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 $\mathsf{pos}{\downarrow}(\mathsf{top}^k(R(\min I^k)))$ is one of the k-stacks in $\mathsf{pos}{\downarrow}(s_0^l)$.

As I^l we take the set of those indices i for which $s_i^l = \mathsf{top}^l(R(i))$. It is immediate from the definitions that I^l is l-advancing (recall that $s_i^l = \mathsf{hist}(R{\upharpoonright}_{i,j}, s_j^l)$ for $i \leq j$). By assumption $|I^l| \geq \beta(\mathsf{pos}{\downarrow}(s_0^l)) = N_l$. Moreover, $s_{\min I^l}^l$ was not modified from the beginning of the run (as it was not the topmost l-stack), so this l-stack is positionless-equal to s_0^l.

Then by induction on $l - k$, we construct I^{k-1} out of I^k using 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 have a 0-advancing set I^0 such that $|I^0| \geq N_0$. 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 $R(i)$, by the pigeonhole principle (recall the definition of N_0 from the first paragraph of the proof). Lemma 8.7 applied to $R{\upharpoonright}_{i,j}$ proves that $R(i)$ is a milestone. By construction $i \in I^0 \subseteq \cdots \subseteq I^l$, so $s_i^l = \mathsf{top}^l(R(i))$. $\qquad\square$

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 that we consider, we have a milestone configuration from which we start runs that first read 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 are 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 that 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 5. 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 that 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 that 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 a 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 was presented in a former paper by the author [Par12c]. There are several differences between these two lemmas. An advantage of the former lemma 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 former pumping lemma 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 former pumping lemma was later generalized to collapsible pushdown automata [KP12].

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 ceases to be $(k-1)$-upper. One possibility is as depicted in Figure 5: this happens during a pop^k operation; our topmost k-stack becomes positionless-equal to $\mathsf{pop}^k(\mathsf{top}^k(R(0)))$. This can also happen during a pop^r operation for some $r > k$. 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, by the pigeonhole principle; 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 positionless topmost k-stack as in the original run, when the run ceases 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 ceases 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 is based on Lemma 9.2, in which we analyze the situation just after reading the stars.

In order to state Lemma 9.2, we need two definitions. For a run R starting in a configuration c, and for a k-stack s^k in some configuration $R(i)$, where $k \in [1, n]$, we say that s^k is *c-clear* in $R(i)$ (with respect to R) when $\mathsf{hist}(R{\restriction}_{0,i}, \mathsf{top}^{k-1}(s^k)) \neq \mathsf{top}^{k-1}(c)$. Moreover, for a configuration c, and for $k \in [1, n]$, let $\mathcal{S}^k(c)$ be the smallest set of positionless k-stacks such that if R is a run that starts in c and reads only stars, and s^k is a k-stack of $R(|R|)$ that is c-clear with respect to R, then $\mathsf{pos}{\downarrow}(s^k) \in \mathcal{S}^k(c)$.

Lemma 9.2. *For each milestone configuration c, and for $k \in [1, n]$, the set $\mathcal{S}^k(c)$ is finite.*

Proof. Let $\mathcal{X}(c)$ be the set containing all positionless k-stacks of c, and additionally $\mathsf{pos}{\downarrow}(\mathsf{pop}^k(\mathsf{top}^k(c)))$; clearly $\mathcal{X}(c)$ is finite. We claim that every positionless k-stack in $\mathcal{S}^k(c)$ can be obtained from a positionless k-stack $s^k_- \in \mathcal{X}(c)$ by applying at most $\beta(s^k_-)$ push and pop operations, where β is the function from Lemma 8.4; this immediately implies that $\mathcal{S}^k(c)$ is finite.

Fix a run R that starts in c and reads only stars, and fix a c-clear k-stack s^k of $R(|R|)$. Consider the smallest index i for which the k-stack $\mathsf{hist}(R{\restriction}_{i,|R|}, s^k)$ is c-clear in $R(i)$; denote $t^k = \mathsf{hist}(R{\restriction}_{i,|R|}, s^k)$. We claim that $\mathsf{pos}{\downarrow}(t^k) \in \mathcal{X}(c)$. Indeed, either $i = 0$ and t^k is one of the k-stacks of c, or $\mathsf{hist}(R{\restriction}_{i-1,|R|}, s^k)$ is not c-clear in $R(i-1)$. 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 $t^k = \mathsf{top}^k(R(i)) \cong \mathsf{pop}^k(\mathsf{top}^k(c))$; thus, $\mathsf{pos}{\downarrow}(t^k) \in \mathcal{X}(c)$.

Observe that t^k can be changed in $R{\restriction}_{i,|R|}$ only when it is the topmost k-stack. If there exist at most $\beta(\mathsf{pos}{\downarrow}(t^k))$ indices $j \in [i, |R|]$ such that $\mathsf{hist}(R{\restriction}_{j,|R|}, s^k) = \mathsf{top}^k(R(j))$, then s^k can be obtained from t^k by applying at most $\beta(\mathsf{pos}{\downarrow}(t^k))$ push and pop operations, as we wanted to prove.

It remains to prove that indeed there are at most $\beta(\mathsf{pos}{\downarrow}(t^k))$ indices $j \in [i, |R|]$ such that $\mathsf{hist}(R{\restriction}_{j,|R|}, s^k) = \mathsf{top}^k(R(j))$. Suppose to the contrary that there are more than $\beta(\mathsf{pos}{\downarrow}(t^k))$ such indices j. Then we can use Lemma 8.4 for $R{\restriction}_{i,|R|}$; it gives us an index j such that the configuration $R(j)$ is a milestone and $\mathsf{hist}(R{\restriction}_{j,|R|}, s^k) = \mathsf{top}^k(R(j))$. 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 $\mathsf{hist}(R, s^k) \neq \mathsf{top}^k(c)$ (because, by the definition of i, $\mathsf{hist}(R, s^k)$ is c-clear in c), and we know that $\mathsf{hist}(R{\restriction}_{j,|R|}, s^k) = \mathsf{top}^k(R(j))$, so $R{\restriction}_{0,j}$ is not k-upper; in particular it cannot be 0-upper. Otherwise, as already observed, $R{\restriction}_{0,i}$ is not $(k-1)$-upper and $\mathsf{hist}(R{\restriction}_{i,|R|}, s^k) = \mathsf{top}^k(R(i))$, which implies that $R{\restriction}_{i,j}$ is k-upper. But $R{\restriction}_{0,j}$ is $(k-1)$-upper, so this contradicts Proposition 6.4 applied for $R{\restriction}_{0,i} \circ R{\restriction}_{i,j}$. $\qquad\square$

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 positionless topmost k-stack as $R(|R|)$.

Proof. There are only finitely many possible values of an $(\mathcal{A}, \varphi)$-type of a configuration. Thus, it is enough to show, for each k, that there are only finitely many possible positionless topmost k-stacks over all configurations $R(|R|)$ satisfying the assumptions. For $k = 0$ this is trivial as a positionless 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, $\mathsf{top}^k(R(|R|)) \cong \mathsf{pop}^k(\mathsf{top}^k(R(0)))$ (cf. Proposition 6.6); 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; it thus belongs to the set $\mathcal{S}^k(c)$, which is finite by Lemma 9.2. □

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 that starts in c and 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 positionless 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 that $T_i = T_{\mathsf{pb}(c)}$ for all $i \geq \mathsf{pb}(c)$.

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'. Without loss of generality, 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. Without loss of generality, 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 "without loss of generality" assumption. We obtain some $d \in \mathcal{S}(c)$ that has the same $(\mathcal{A}, \varphi)$-type and the same positionless 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 the language U cannot be recognized by a deterministic higher-order pushdown automaton. 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, for instance, 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 is by contradiction: suppose that for some n we have an $(n-1)$-DPDA recognizing U. We construct an n-DPDA $\mathcal{A}$ that 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 (here we use the fact that no word in U is a prefix of another word in U). Such a normalization allows us to use Theorem 7.4, as in $\mathcal{A}$ every accepting run is an n-return.

Fix a finite monoid M and a morphism $\lambda\colon A^* \to M$ that 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 $\lambda(u) \neq \lambda(v)$ for all words u, v being of different forms. 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:

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

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 $|w_n|$ (recall that n is the order of $\mathcal{A}$), 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. The 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 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 a 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 is 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 $\sharp$ symbol). This contradicts the following key lemma (taken for $k = n-1$ and $u = \varepsilon$).
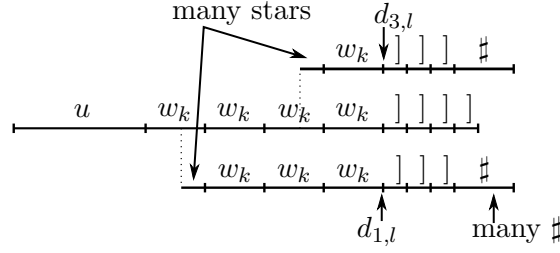
Figure 6: Illustration of runs appearing in the proof (where $N = 4$, $x = 1$, $y = 3$). Recall that stars can appear between letters of the words.

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 $v \neq w_{k+1}$ and $\langle uv, uw_{k+1}\rangle$ is not k-upper.*

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

Let now $k \geq 0$. Figure 6 may 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 $\heartsuit$.

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

In the proof, we construct two sequences of accepting runs, with many extra stars inserted in two different places. Namely, in one sequence we insert extra stars before the last opening bracket that is not closed, and in the other sequence—after this bracket. In effect, the number of sharp symbols read by runs in one of these sequences should be unbounded, and in the other—bounded. The sequences are constructed in such a way that this violates Theorem 7.4, which says that the number of sharp symbols read by runs in these sequences is either bounded in both sequences or unbounded in both sequences.

Now we come to details. By the induction assumption (where uw_k^{i-1} is taken as u), for each $i \in [1, N]$ there exists 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 $\heartsuit$), 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 $\mathsf{type}_{\mathcal{A},\lambda}(d_{i,l})$ that appears infinitely often, and then we take the subsequence of only these $d_{i,l}$ that 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

$d_{x,1}, d_{x,2}, \ldots$ and $d_{y,1}, d_{y,2}, \ldots$ 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 positionless-equal.

Let R' be a prefix of $S'_{x,1}$ that 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 that starts in $R'(|R'|)$ and reads only $\sharp$ symbols (because $\mathcal{A}$ recognizes U, there is an accepting run R'' that starts in $R'(|R'|)$ and reads only $\sharp$ symbols; by construction of $\mathcal{A}$, it is an n-return).

Finally, we use Theorem 7.4 for λ (as φ), sequences $d_{x,1}, d_{x,2}, \ldots$ (as $c_1, c_2, \ldots$) and $d_{y,1}, d_{y,2}, \ldots$ (as $d_1, d_2, \ldots$), and for run the $R' \circ R''$.[5] 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 positionless topmost k-stacks. Thus, the assumptions of the theorem are satisfied. For each l, we obtain runs $S_l = S'_l \circ S''_l$ (from $d_{x,l}$) and $T_l = T'_l \circ T''_l$ (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 that 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 $\sharp(S_1), \sharp(S_2), \ldots$ 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 that 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 $\sharp(T_1), \sharp(T_2), \ldots$ has to be constant, hence bounded. This contradicts the thesis of Theorem 7.4, which says that these sequences are either both bounded or both unbounded. □

Acknowledgment

References

[AdMO05] Klaus Aehlig, Jolie G. de Miranda, and C.-H. Luke Ong. Safety is not a restriction at level 2 for string languages. In Vladimiro Sassone, editor, *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 490–504. Springer, 2005.

[BCHS12] Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, and Olivier Serre. A saturation method for collapsible pushdown systems. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2012.

[Blu08] Achim Blumensath. On the structure of graphs in the Caucal hierarchy. *Theor. Comput. Sci.*, 400(1-3):19–45, 2008.

[Cau02] Didier Caucal. On infinite terms having a decidable monadic theory. In Krzysztof Diks and Wojciech Rytter, editors, *MFCS*, volume 2420 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2002.

---

[5] 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 $R(z(uw_k^N))$.

[Eng91]   Joost Engelfriet. Iterated stack automata and complexity classes. *Inf. Comput.*, 95(1):21–75, 1991.

[Gil96]   Robert H. Gilman. A shrinking lemma for indexed languages. *Theor. Comput. Sci.*, 163(1&2):277–281, 1996.

[Hay73]   Takeshi Hayashi. On derivation trees of indexed grammars. *Publ. RIMS, Kyoto Univ.*, 9:61–92, 1973.

[HK13]   Alexander Heußner and Alexander Kartzow. Reachability in higher-order-counters. In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*, pages 528–539. Springer, 2013.

[HMOS08]   Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *LICS*, pages 452–461. IEEE Computer Society, 2008.

[Kar10]   Alexander Kartzow. Collapsible pushdown graphs of level 2 are tree-automatic. In Jean-Yves Marion and Thomas Schwentick, editors, *STACS*, volume 5 of *LIPIcs*, pages 501–512. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[Kar11]   Alexander Kartzow. A pumping lemma for collapsible pushdown graphs of level 2. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 322–336. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[KNU02]   Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *FoSSaCS*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002.

[KNUW05]   Teodor Knapik, Damian Niwiński, Paweł Urzyczyn, and Igor Walukiewicz. Unsafe grammars and panic automata. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1450–1461. Springer, 2005.

[KO09]   Naoki Kobayashi and C.-H. Luke Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2009.

[Kob09]   Naoki Kobayashi. Model-checking higher-order functions. In António Porto and Francisco Javier López-Fraguas, editors, *PPDP*, pages 25–36. ACM, 2009.

[KP12]   Alexander Kartzow and Paweł Parys. Strictness of the collapsible pushdown hierarchy. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *MFCS*, volume 7464 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2012.

[Mas74]   A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet Math. Dokl.*, 15:1170–1174, 1974.

[Mas76]   A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12:38–43, 1976.

[Ong06]   C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS*, pages 81–90. IEEE Computer Society, 2006.

[Par11]   Paweł Parys. Collapse operation increases expressive power of deterministic higher order pushdown automata. In Thomas Schwentick and Christoph Dürr, editors, *STACS*, volume 9 of *LIPIcs*, pages 603–614. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[Par12a]   Paweł Parys. Higher-order pushdown systems with data. In Marco Faella and Aniello Murano, editors, *GandALF*, volume 96 of *EPTCS*, pages 210–223, 2012.

[Par12b]   Paweł Parys. On the significance of the collapse operation. In *LICS*, pages 521–530. IEEE, 2012.

[Par12c]   Paweł Parys. A pumping lemma for pushdown graphs of any level. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 54–65. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.