2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science

On the Significance of the Collapse Operation

Paweł Parys
Institute of Informatics
University of Warsaw
Warsaw, Poland

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

***Index Terms*—Higher-order pushdown systems, collapse, higher-order recursion schemes.**

I. Introduction

Already in the 70's, Maslov ([13], [14]) generalized the concept of pushdown automata to higher-order pushdown automata and studied such devices as acceptors of string languages. In the last decade, renewed interest in these automata has arisen. They are now studied also as generators of graphs and trees. It was an interesting problem whether the class of trees generated by deterministic level-n pushdown systems coincides with the class of trees generated by level-n recursion schemes. Knapik et al. [10] showed instead that this class coincides with the class of trees generated by *safe* level-n recursion schemes,[1] and Caucal [3] gave another characterization: trees on level $n+1$ are obtained from trees on level n by an MSO-interpretation of a graph, followed by application of unfolding. Carayol and Wöhrle [2] studied the ε-closures of configuration graphs of level-n pushdown systems and proved that these graphs are exactly the graphs in the n-th level of the Caucal hierarchy.

Driven by the question whether safety implies a semantical restriction to recursion schemes, Hague et al. [5] extended the model of level-n pushdown systems to level-n collapsible pushdown systems by introducing a new stack operation called collapse. Intuitively, this operation allows the removal of all stacks on which a copy of the currently topmost stack symbol is present. They showed that the trees generated by such systems coincide exactly with the class of trees generated by all higher-order recursion schemes and this correspondence is level-by-level. Let us mention that these trees have decidable MSO theory [15], and that higher-order recursion schemes have close connections with verification of some real life higher order programs [12].

[1]Safety is a syntactic restriction on the recursion scheme.

Nevertheless, it was still an open question whether these two hierarchies are possibly the same hierarchy? This problem was stated in [10] and repeated in other papers concerning higher order PDA [11], [1], [15], [5]. A partial answer to this question was given in [16]: there exists a language recognized by a deterministic collapsible pushdown automaton of the second level, which is not recognized by any deterministic higher order pushdown automaton (without collapse) of the second level. We prove a stronger property, that the sum of both hierarchies is different, which is our main theorem.

Theorem 1.1. *There exists a language recognized by a deterministic collapsible pushdown automaton of the second level, which is not recognized by any deterministic higher order pushdown automaton (without collapse) of any level.*

The result can be also stated as follows (the parts about recursion schemes follow from the equivalences mentioned above).

Corollary 1.2. *There exists a tree generated by a collapsible pushdown system of the second level (equivalently: by a recursion scheme of the second level), which is not generated by any higher order pushdown system (without collapse) of any level (equivalently: by a safe recursion scheme of any level).*

This confirms that the correspondence between higher order recursion schemes and higher order pushdown systems is not perfect. The language used in Theorem 1.1 (after some adaptations) comes from [10] and from that time was conjectured to be a good example.

As a side effect, in Section VII we present a pumping lemma for deterministic higher order pushdown automata. Although its formulation is not very natural, we believe it may be useful for some other applications. Our lemma is similar to the pumping lemma from [17] (see Section VII for some comments). Earlier, several pumping lemmas related to the second level of the pushdown hierarchy were proposed [6], [4], [8].

Related work: One may ask a similar question for nondeterministic automata rather than for deterministic ones: is there a language recognized by a nondeterministic collapsible pushdown automaton, which is not recognized by any nondeterministic higher order pushdown automaton (without collapse). This is an independent problem. The answer is known only for level 2 and is opposite: one can see that for level 2 the collapse operation can be simulated by nondeterminism,

1043-6871/12 $26.00 © 2012 IEEE
DOI 10.1109/LICS.2012.62

hence normal and collapsible nondeterministic PDA of level 2 recognize the same languages [1].

We know [5] that there is a collapsible pushdown graph of level 2, which has undecidable MSO theory, hence which is not a pushdown graph of any level (as they all have decidable MSO theory).

In [9] we simultaneously prove that the hierarchy of collapsible pushdown trees (and also graphs) is infinite, i.e. that for each level there exists a tree generated by a collapsible pushdown system of that level which is not generated by any collapsible pushdown system of a lower level.

II. Preliminaries

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

- $\mathsf{read}(f)$, where $f : A \to Q$ is an injective function,
- $\mathsf{pop}^k(q)$, where $1 \leq k \leq n$ and $q \in Q$, or
- $\mathsf{push}^k(t^0, q)$, where $1 \leq k \leq n$, and $t^0 \in \Gamma$, and $q \in Q$.

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

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

$$\Gamma_*^0 = \Gamma,$$
$$\Gamma_*^k = (\Gamma_*^{k-1})^* \qquad \text{for } 1 \leq k \leq n.$$

In other words, a 0-stack is just a single symbol, and a k-stack for $1 \leq k \leq n$ is a (possibly empty) sequence of $(k-1)$-stacks. Top of a stack is on the right. The *size* of a k-stack is just the number of $(k-1)$-stacks it contains. For any $s^k \in \Gamma_*^k$ and $s^{k-1} \in \Gamma_*^{k-1}$ we write $s^k : s^{k-1}$ for the k-stack obtained from s^k by placing s^{k-1} at its end. The operator ":" is assumed to be right associative, i.e. $s^2 : s^1 : s^0 = s^2 : (s^1 : s^0)$. We say that an n-stack s is a *pushdown store* (*pds* for short) if every k-stack in s is nonempty, for $1 \leq k \leq n$ (including the whole s).

A *configuration* of $\mathcal{A}$ consists of a state and of a pds, i.e. is an element of $Q \times \Gamma_*^n$ in which the n-stack is a pds. The *initial* configuration consists of the initial state q_I and of the n-stack containing only one 0-stack, which is the initial stack symbol γ_I. For a configuration c, its state is denoted $state(c)$, and its n-stack is denoted $\pi(c)$.

Next, we define when a configuration d is a *successor* of a configuration c. Let $p = state(c)$, and let s^0 be the topmost 0-stack of $\pi(c)$. We have three cases depending on $\delta(p, s^0)$:

- if $\delta(p, s^0) = \mathsf{read}(f)$ then $state(d) = f(a)$ for some $a \in A$, and $\pi(d) = \pi(c)$,
- if $\delta(p, s^0) = \mathsf{pop}^k(q)$ then $state(d) = q$, and $\pi(d)$ is obtained from $\pi(c)$ by replacing its topmost k-stack $s^k : s^{k-1}$ by s^k (i.e. we remove the topmost $(k-1)$-stack; in particular the topmost k-stack of $\pi(c)$ has to contain at least two $(k-1)$-stacks),
- if $\delta(p, s^0) = \mathsf{push}^k(t^0, q)$ then $state(d) = q$, and $\pi(d)$ is obtained from $\pi(c)$ by replacing its topmost k-stack $s^k : s^{k-1}$ by $(s^k : s^{k-1}) : s^{k-1}$, and then by replacing its topmost 0-stack by t^0 (i.e. we copy the topmost $(k-1)$-stack, and then we change the topmost symbol in the copy).[2]

Notice that most configurations have exactly one successor. However when the operation is read, there are several successors. It is also possible that there are no successors: when the operation is pop^k but there is only one $(k-1)$-stack on the topmost k-stack.

Next, we define a *run* of $\mathcal{A}$. For $0 \leq i \leq m$, 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 $1 \leq i \leq m$, 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 which is defined as expected.

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 operation between them is pop or push. If the operation is $\mathsf{read}(f)$, this is the one-letter word consisting of the letter a for which $state(d) = f(a)$ (this letter is determined uniquely, as f is injective). For a longer run R this is defined as the concatenation of the words read by the subruns $R\restriction_{i-1,i}$ for $1 \leq i \leq |R|$. A word w is accepted by $\mathcal{A}$ if it is read by some run from the initial configuration to a configuration having an accepting state. The *language* recognized by $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$.

Collapsible 2-HOPDA: In Section III we also use deterministic collapsible pushdown automata of the second level (2-CPDA for short). Such automata are defined like 2-HOPDA, with the following differences. A 0-stack contains now two parts: a symbol from Γ, and a natural number, but still only the symbol (together with a state) is used to determine which transition is performed from a configuration. The push^1 operation sets the number in the topmost 0-stack to the current size of the 2-stack. We have a new operation $\mathsf{collapse}(q)$. When it is performed between configurations c and d, then $state(d) = q$, and $\pi(d)$ is obtained from $\pi(c)$ by removing its topmost 1-stacks, so that only $k-1$ of them is left, where k is the number stored in the topmost 0-stack of c (intuitively, we remove all 1-stacks on which the topmost 0-stack is present).

III. The Separating Language

In this section we define a language U which can be recognized by a 2-CPDA, but not by any n-HOPDA, 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

[2] In the classical definition the topmost symbol can be changed only when $k = 1$ (for $k \geq 2$ it has to be $s^0 = t^0$). We make this (not important) extension to have an unified definition of push^k for every k.

TABLE I

STACK CONTENTS OF THE EXAMPLE RUN, AND SUBRUNS BEING k-UPPER RUNS AND k-RETURNS

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

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

Finally the $\sharp$ symbol is read. If the topmost symbol is X, we have read as many opening brackets as closing brackets, hence we should accept one $\sharp$ symbol. Otherwise, the topmost Y symbol corresponds to the last opening bracket which is not closed. We execute the 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 make pop^2), and then accept.

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

IV. THE HISTORY FUNCTION, AND SPECIAL RUNS

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

A *position* is a vector $x = (x_n, x_{n-1}, \dots, x_1)$ of n positive integers. The symbol at position x in configuration c (which is an element of the stack alphabet) is defined in the natural way (we take the x_n-th $(n-1)$-stack of $\pi(c)$, then its x_{n-1}-th $(n-2)$-stack, and so on; elements of stacks are numbered from bottom to top). We say that x is a position of c, if at position x there is a symbol in c. For $0 \le k \le n$, by $top^k(c)$ we denote the position of the bottommost symbol of the topmost k-stack of c. In particular $top^0(c)$ is the topmost position of c.

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

For $0 \le k \le n$, we say that a run R is *k-upper* if $hist(R, top^k(R(|R|))) = top^k(R(0))$; 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, $up^k \subseteq up^l$ for $k \le l$, and $R \in up^k \iff R \circ S \in up^k$ for $S \in up^k$.

For $1 \le k \le n$, a run R is a *k-return* if

- $hist(R, top^{k-1}(R(|R|)))$ is the bottommost position of the second topmost $(k-1)$-stack[3] of $R(0)$, and
- $R\restriction_{i,|R|} \notin up^{k-1}$ for all $0 \le i < |R|$.

Let ret^k be the set of k-returns. Observe that $ret^k \subseteq up^k$.

Example 4.1. Consider a HOPDA of level 2. Below, brackets are used to group symbols in one 1-stack. Consider a run R of length 6 in which $\pi(R(0)) = [ab][cd]$, and the operations between consecutive configurations are:

$$\mathsf{push}^2(e),\ \mathsf{pop}^1,\ \mathsf{pop}^2,\ \mathsf{pop}^1,\ \mathsf{push}^1(d),\ \mathsf{pop}^1.$$

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

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

at position $(2,2)$ in $R(5)$ we have d and at position $(2,1)$ in $R(0)$ we have c.

Next we state several easy propositions, which are useful later, and also give more intuition about the above definitions.

Proposition 4.2. *Let R be a k-upper run (where $0 \le k \le n$) such that $R\restriction_{i,|R|} \notin up^k$ for $0 < i < |R|$. Then*

- *the topmost k-stack of $R(0)$ and $R(|R|)$ is the same; additionally for every position x in the topmost k-stack of $R(|R|)$, $hist(R,x)$ is the corresponding position in the topmost k-stack of $R(0)$, or*
- *$|R| = 1$ and the only operation of R is pop^r for $r \le k$, or push^r for $r \le k$.*

Proposition 4.3. *Let R be a k-upper run, where $1 \le k \le n$. Then R is $(k-1)$-upper if and only if the size of the topmost k-stack of $R(0)$ is not greater than the size of the topmost k-stack of $R(i)$ for every $0 \le i \le |R|$ such that $R\restriction_{i,|R|} \in up^k$.*

Proposition 4.4. *Let $R \circ S$ be a run such that $R \notin up^{k-1}$ and $S \in up^k$, where $1 \le k \le n$. Then $R \circ S \notin up^{k-1}$.*

Proposition 4.5. *Let R be a run (where $1 \le k \le n$) such that $R\restriction_{0,|R|-1} \in up^{k-1}$ and $R\restriction_{|R|-1,|R|} \in up^k$, but $R \notin up^{k-1}$. Then R is a k-return.*

Proposition 4.6. *Let R be a k-return, where $1 \le k \le n$. Then the topmost k-stack of $R(0)$ after removing its topmost $(k-1)$-stack is equal to the topmost k-stack of $R(|R|)$. Additionally for every position x in the topmost k-stack of $R(|R|)$, $hist(R,x)$ is the corresponding position in the topmost k-stack of $R(0)$.*

V. Types and Sequence Equivalence

In this section we assign to each configuration a type from a finite set which, in some sense, describes possible returns and upper runs from this configuration. Additionally, we also assign to each sequence of configurations some information from a finite set, which says whether runs from these configurations can read an unbounded number of $\sharp$ symbols. For this section we fix an be an n-HOPDA $\mathcal{A}$ with stack alphabet Γ, state set Q, and input alphabet A which contains a distinguished symbol $\sharp$. Moreover we fix a morphism $\varphi\colon A^* \to M$ into a finite monoid.

We begin by types describing returns. To every k-stack s^k (where $0 \le k \le n$) we assign a set $type(s^k) \subseteq \mathcal{T}^k$; it contains some *run descriptors*. The sets $\mathcal{T}^k$ are defined inductively as follows (where $\mathcal{P}(X)$ denotes the power set of X):

$$\mathcal{T}^k = \{(\mathsf{ne},\mathsf{tr})\} \cup \big(\mathcal{P}(\mathcal{T}^n) \times \mathcal{P}(\mathcal{T}^{n-1}) \times \cdots \times \mathcal{P}(\mathcal{T}^{k+1}) \times \times Q \times \mathcal{D}^k \times \{\mathsf{tr},\mathsf{nt}\}\big), \quad \text{where}$$

$$\mathcal{D}^k = \bigcup_{r=k+1}^{n} M \times \{r\} \times \mathcal{P}(\mathcal{T}^n) \times \mathcal{P}(\mathcal{T}^{n-1}) \times \cdots \times \times \mathcal{P}(\mathcal{T}^{r+1}) \times Q,$$

We say that a run R *agrees* with $(m, r, \Sigma^n, \Sigma^{n-1}, \dots, \Sigma^{r+1}, q) \in \mathcal{D}^0$ if

- the word read by R evaluates to m under φ, and
- R is an r-return, and
- $\Sigma^i \subseteq type(t^i)$ for $r+1 \le i \le n$, where $\pi(R(|R|)) = t^n : t^{n-1} : \cdots : t^r$, and
- $q = state(R(|R|))$.

The acronym ne stands for nonempty; we have $(\mathsf{ne},\mathsf{tr}) \in type(s^k)$ when s^k is nonempty. A typical run descriptor in $\mathcal{T}^k$ is of the form $\sigma = (\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, p, \widehat{\sigma}, fl)$. By adding σ to the type of some s^k, we claim the following. If for each $k+1 \le i \le n$ we take an i-stack t^i that satisfies the claims of Ψ^i, then there is a run which starts in state p and stack $t^n : t^{n-1} : \cdots : t^{k+1} : s^k$, and agrees with $\widehat{\sigma}$ (for a moment let us ignore the last coordinate of σ; it will be useful later, while defining sequence equivalence). In other words we have the following lemma.

Lemma 5.1. *Let $0 \le k \le n$, let $\widehat{\sigma} \in \mathcal{D}^k$, and let $c = (p, s^n : s^{n-1} : \cdots : s^k)$. The following two conditions are equivalent:*

1) *there exists a run from c which agrees with $\widehat{\sigma}$,*
2) *$type(s^k)$ contains a run descriptor $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, p, \widehat{\sigma}, fl)$ such that $\Psi^i \subseteq type(s^i)$ for $k+1 \le i \le n$.*

Types of stacks are defined as the smallest set closed under some natural rules (saying when we should add a run descriptor to a type). One rule is for the composition of stacks: if $type(s^k)$ contains a run descriptor $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, p, \widehat{\sigma}, fl)$, and $\Psi^{k+1} \subseteq type(s^{k+1})$, and $\widehat{\sigma} \in \mathcal{D}^{k+1}$, then $type(s^{k+1} : s^k)$ contains the run descriptor $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+2}, p, \widehat{\sigma}, fl)$. The other rules correspond to some possible forms of returns; a typical rule is as follows. Let s^0 be a symbol and p a state such that $\delta(s^0, p) = \mathsf{pop}^k(q)$. Then for each $\sigma = (\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, q, \widehat{\sigma}, fl) \in \mathcal{T}^k$ we have $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, \{\sigma\}, \emptyset, \emptyset, \dots, \emptyset, p, \widehat{\sigma}, \mathsf{tr}) \in type(s^0)$. This rule corresponds to a situation when we can perform pop^k to a configuration from which we have a run which agrees with $\widehat{\sigma}$; then the whole run also agrees with $\widehat{\sigma}$ (because $\widehat{\sigma} \in \mathcal{D}^k$, it describes an r-return for $r > k$). It is possible to write rules describing all returns, because we have the following characterization (for case 3 it is also important that we have Proposition 4.6).

Proposition 5.2. *A run R is an r-return (where $1 \le r \le n$) if and only if*

1) *$|R| = 1$, and the operation performed by R is pop^r, or*
2) *the first operation performed by R is read, or pop^k for $k < r$, or push^k for $k \ne r$, and $R\restriction_{1,|R|}$ is an r-return, or*
3) *the first operation performed by R is push^k for $k \ge r$, and $R\restriction_{1,|R|}$ is a composition of a k-return and an r-return.*

It is not difficult (but technically complicated) to prove Lemma 5.1 directly from the definition of $type$. For the 2⇒1 implication, we decompose the run according to Proposition 5.2; this gives us a list of rules which imply that an appropriate run descriptor is in the type. For the opposite implication, a run descriptor is in the type, because it has some derivation using the rules; by composing all these rules we obtain appropriate run.

Let us mention that a similar notion of types was also present in [17]. Those types were defined in a different, semantical way. Namely, our Lemma 5.1 is used as a definition; then it is necessary to prove that the type of s^k does not depend on the choice of $s^n, s^{n-1}, \dots, s^{k+1}$ present in the assumptions of the lemma. Our approach is better for the following reason: when we add some run descriptor $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, p, \widehat{\sigma}, fl)$ to the type of some k-stack, then in the sets Ψ^i we only have the assumptions which are really useful (we never put there redundant run descriptors). This will be important while defining sequence equivalence. Comparing to [17] we have also added the morphism φ, but this is a very easy extension.

Next, let us define the type of a whole configuration $c = (p, s^n : s^{n-1} : \dots : s^0)$:

$$type_{\mathcal{A},\varphi}(c) = \left(type(s^n), type(s^{n-1}), \dots, type(s^0), p\right).$$

Let $\mathcal{T}_{\mathcal{A},\varphi}$ denote the (finite) set of all such tuples. Basing on $type_{\mathcal{A},\varphi}$, for each $0 \le k \le n$ we define a function $type^k_{\mathcal{A},\varphi}$ which assigns to every configuration c of $\mathcal{A}$ a pair from $\mathcal{T}_{\mathcal{A},\varphi} \times \Gamma^k_*$, which is $type_{\mathcal{A},\varphi}(c)$, and the topmost k-stack of c. Notice that the range of $type^k_{\mathcal{A},\varphi}$ for $k \ge 1$ is not finite. We also define a partial order $\sqsubseteq$ on $\mathcal{T}_{\mathcal{A},\varphi} \times \Gamma^k_*$: we say that $((\Psi^n, \Psi^{n-1}, \dots, \Psi^0, p), s^k) \sqsubseteq ((\Phi^n, \Phi^{n-1}, \dots, \Phi^0, q), t^k)$ if and only if $p = q$ and $s^k = t^k$ and $\Psi^i \subseteq \Phi^i$ for each $0 \le i \le n$.

The main purpose for introducing types is to describe upper runs. Notice that upper runs are closely connected to returns thanks to the following characterization.

Proposition 5.3. *A run R is k-upper (where $0 \le k \le n$) if and only if*

1) $|R| = 0$, *or*
2) $|R| = 1$, *and the operation performed by R is* read, *or* push^r *for any r, or* pop^r *for $r \le k$, or*
3) *the first operation performed by R is* push^r *for $r \ge k+1$, and $R\lceil_{1,|R|}$ is an r-return, or*
4) *R is a composition of two nonempty k-upper runs.*

In case 2 we see that $type^k_{\mathcal{A},\varphi}$ after the operation is determined by $type^k_{\mathcal{A},\varphi}$ before the operation, as only the topmost k-stack is changed. Similarly if the operation is push^r (first step of case 3), because the type of a composition of stacks $s^r : s^{r-1}$ is (by definition) determined by the types of s^r and s^{r-1}. Thanks to that we obtain the following theorem, which will be important in the next sections.

Theorem 5.4. *Let R be a k-upper run (where $0 \le k \le n$), and let c be a configuration such that $type^k_{\mathcal{A},\varphi}(R(0)) \sqsubseteq type^k_{\mathcal{A},\varphi}(c)$. Then there exists a k-upper run S from c such that*

1) *if $|R| > 0$ then $|S| > 0$, and*
2) *the words read by R and by S evaluate to the same under φ, and*
3) $type^k_{\mathcal{A},\varphi}(R(|R|)) \sqsubseteq type^k_{\mathcal{A},\varphi}(S(|S|))$.

This theorem allows us to transfer k-upper runs to configurations having greater $type^k_{\mathcal{A},\varphi}$. When we start using the theorem, we even have $type^k_{\mathcal{A},\varphi}(R(0)) = type^k_{\mathcal{A},\varphi}(c)$. After that we can again use it for some run starting in $R(|R|)$ and for $S(|S|)$; then we only know that $type^k_{\mathcal{A},\varphi}(R(|R|)) \sqsubseteq type^k_{\mathcal{A},\varphi}(S(|S|))$, we do not have the equality (that is the reason of introducing the $\sqsubseteq$ order).

The statement of our second theorem, about sequence equivalence, is technically more complicated (we were unable to prove its simpler variant, which we believe is also true). Thus before stating the theorem, we give the desired properties of the sequence equivalence. We will define an equivalence relation over infinite sequences of configurations of $\mathcal{A}$, called $(\mathcal{A},\varphi)$-sequence equivalence, which has finitely many equivalence classes. Assume we have two infinite sequences of configurations $c_1, c_2, c_3, \dots$ and $d_1, d_2, d_3, \dots$, and an n-return R such that $type_{\mathcal{A},\varphi}(R(0)) \sqsubseteq type_{\mathcal{A},\varphi}(c_i)$ and $type_{\mathcal{A},\varphi}(R(0)) \sqsubseteq type_{\mathcal{A},\varphi}(d_i)$ for each i. Then, by Lemma 5.1, there exists an n-return from each of c_i and d_i (there might be multiple n-returns, let us fix one of them for every c_i and d_i). Let x_i be the number of the $\sharp$ symbols read by the return from c_i; similarly y_i for d_i. If the sequences $c_1, c_2, c_3, \dots$ and $d_1, d_2, d_3, \dots$ are $(\mathcal{A},\varphi)$-sequence equivalent, it should hold that either the sequences $x_1, x_2, x_3, \dots$ and $y_1, y_2, y_3, \dots$ are both bounded, or both unbounded.

Till now it is trivial to obtain an equivalence relation satisfying the above property: we just need one class for "bounded" sequences, and one for "unbounded". What we would like to have as well is the following transfer property. Assume we have two infinite sequences of configurations $c_1, c_2, c_3, \dots$ and $d_1, d_2, d_3, \dots$, and a k-upper run R such that $type^k_{\mathcal{A},\varphi}(R(0)) \sqsubseteq type^k_{\mathcal{A},\varphi}(c_i)$ and $type^k_{\mathcal{A},\varphi}(R(0)) \sqsubseteq type^k_{\mathcal{A},\varphi}(d_i)$ for each i. Then Theorem 5.4 gives us a k-upper run R from each of c_i and d_i; let c'_i (d'_i) be the configuration at the end of this run. It would be useful to say that if the original sequences are $(\mathcal{A},\varphi)$-sequence equivalent, then $c'_1, c'_2, c'_3, \dots$ and $d'_1, d'_2, d'_3, \dots$ also are. Unfortunately, instead of this property we have the following theorem, which combines the above two properties together (it will be used only in a situation when the k-upper runs does not read the $\sharp$ symbols, only the n-return does).

Theorem 5.5. *Let $0 \le k \le n$, and let $m_1, m_2, \dots, m_r$ be elements of M. Denote by $\mathcal{R}$ the set of runs $R_1 \circ R_2 \circ \dots \circ R_r$ such that $R_1, R_2, \dots, R_{r-1} \in up^k$, and $R_r \in ret^n$, and each R_i reads a word evaluating under φ to m_i. Let $R \in \mathcal{R}$, and let $c_1, c_2, c_3, \dots$ and $d_1, d_2, d_3, \dots$ be infinite sequences of configurations which are $(\mathcal{A},\varphi)$-sequence equivalent. Assume that $type^k_{\mathcal{A},\varphi}(R(0)) = type^k_{\mathcal{A},\varphi}(c_i) = type^k_{\mathcal{A},\varphi}(d_i)$ for each i. Then for each i there exist runs $S_i \in \mathcal{R}$ from c_i, and $T_i \in \mathcal{R}$ from d_i such that either the sequences $x_1, x_2, x_3, \dots$ and $y_1, y_2, y_3, \dots$ are both bounded, or both unbounded, where for each i, x_i and y_i is the number of the $\sharp$ symbols read by the run S_i and T_i.*

How to obtain this theorem? Now we use the last coordinate of run descriptors. We divide run descriptors into two kinds: trivial (tr) and nontrivial (nt). The intended meaning is that a run descriptor $(\Psi^n, \Psi^{n-1}, \dots, \Psi^{k+1}, p, \widehat{\sigma}, fl)$ is nontrivial (i.e. has $fl = \mathsf{nt}$) if it (more precisely: the run described by it) reads more $\sharp$ symbols than the run descriptors in the assumptions Ψ^i. It is the case when either it reads some $\sharp$ symbol itself, or uses some nontrivial run descriptor at least twice. Otherwise the run descriptor will be trivial. Here it is important that all run descriptors in the assumptions are used at least once. But there is also some fixed maximal number of times each of the assumptions can be used (which depends on the size of the automaton).

Consider a sequence of configurations $c_1, c_2, c_3, \dots$; decompose the stack of c_i as $s_i^n : s_i^{n-1} : \dots : s_i^0$. We can assume that $type_{\mathcal{A},\varphi}(c_i)$ is the same for every i (only such sequences are described by Theorem 5.5). For each i, each $1 \leq k \leq n$, and each $\sigma \in type(s_i^k)$, we choose a subset of the type of each 0-stack of s_i^k, such that all run descriptors in these subsets are used to obtain that σ is in $type(s_i^k)$ (they can be possibly chosen in multiple ways, we fix one choice). Let $nt(\sigma, i)$ be the number of chosen run descriptors which are nontrivial (it can be quite big, as s_i^k can be big, but it can be also much smaller than the size of s_i^k). Finally, to the sequence $c_1, c_2, c_3, \dots$ we assign the set X of those σ for which $\{nt(\sigma, i) : i \in \mathbb{N}\}$ is bounded (finite). If for two sequences these sets are equal, we say that these sequences are $(\mathcal{A}, \varphi)$-sequence equivalent.

Let us now argue that Theorem 5.5 holds for such definition of $(\mathcal{A}, \varphi)$-sequence equivalence. The main idea is that when we construct the runs $S_i \in \mathcal{R}$ from $c_i = (p_i, s_i^n : s_i^{n-1} : \dots : s_i^0)$, we use (independent on i) the same set of run descriptors (which is a subset of $\bigcup_k type(s_i^k)$). It can be shown that if all this run descriptors are in the set X, the runs which we obtain read only a bounded number of the $\sharp$ symbols. On the other hand, if some of the run descriptors is not in the set X, the runs read an unbounded number of the $\sharp$ symbols.

VI. Milestone Configurations

In this section we define so-called milestone configurations and we show their basic properties. The idea of considering milestone configurations comes from [7], but our definition is slightly different (namely, their definition is relative to a run, while our definition is absolute, we always consider the run reading only stars). For this section we fix an n-HOPDA $\mathcal{A}$ with stack alphabet Γ and with input alphabet A containing a distinguished symbol denoted $\star$ (star).

Definition 6.1. 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 $R\restriction_{i,j} \in up^0$ for each $i, j \in I$, $i \leq j$.

Example 6.2. Consider a HOPDA of level 3. Assume there is a run which begins in a stack $[[aa]]$, and performs forever the following sequence of operations, in a loop:

$$\mathsf{push}^2(a),\ \mathsf{push}^3(a),\ \mathsf{pop}^1,\ \mathsf{push}^3(a),\ \mathsf{pop}^2,\ \mathsf{push}^3(a).$$

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

Lemma 6.3. *Let R be an infinite run reading only stars. Then, for infinitely many i the configuration $R(i)$ is a milestone.*

Proof: Let $I^n = \mathbb{N}$. For $k = n-1, n-2, \dots, 0$ we define

$$I^k = \{i \in I^{k+1} | \forall_{j \geq i} (j \in I^{k+1} \Rightarrow R\restriction_{i,j} \in up^k)\}.$$

It is enough to show that set I^0 is infinite. Then, by definition, I^0 contains only milestone configurations.

We prove that I^k is infinite by induction on k, from $k = n$ down to $k = 0$. The induction basis for $k = n$ is true, because $I^n = \mathbb{N}$. Let now $k \leq n-1$; assume that I^{k+1} is infinite. For each index l, we want to find an index $i \geq l$ which is in I^k. By s_j denote the size of the topmost $(k+1)$-stack of $R(j)$. We can choose an index $i \in I^{k+1}$ such that s_i is minimal among all s_j for $j \in I^{k+1} \cap \{l, l+1, l+2, \dots\}$. By Proposition 4.3 (used for $k+1$ as k) we see that $i \in I^k$. ∎

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 also $R(i)$ is a milestone for each $i \in I$. The following lemma shows that in fact the set I can contain all i for which $R(i)$ is a milestone.

Lemma 6.4. *Let R be a run reading only stars, which begins and ends in a milestone. Then R is 0-upper.*

Proof: Consider the infinite run S from $R(0)$ reading only stars (since $R(0)$ is a milestone, the run is really infinite); R is its prefix. We use the sets I^k from the proof of Lemma 6.3 (for run S). We will show that if $S(i)$ is a milestone, then $i \in I^0$. It will mean that both 0 and $|R|$ are in I^0, which will imply that $R = S\restriction_{0,|R|} \in up^0$.

We prove by induction on k, from $k = n$ down to $k = 0$, that if $S(i)$ is a milestone, then $i \in I^k$. We have $i \in I^n$ for each i. Let $k \leq n-1$. Assume that $S(i)$ is a milestone, and that for each milestone $S(j)$ we have $j \in I^{k+1}$. Choose any $j \in I^{k+1}$, $j \geq i$. We need to prove that $S\restriction_{i,j}$ is k-upper. By definition of a milestone, we have arbitrarily large l (in particular $l \geq j$) such that $S\restriction_{i,l}$ is 0-upper (thus as well k-upper) and that $S(l)$ is a milestone. From the induction assumption we have $l \in I^{k+1}$, so $S\restriction_{j,l}$ is $(k+1)$-upper. We conclude that $S\restriction_{i,j} \in up^k$ using Proposition 4.4 for runs $S\restriction_{i,j}$, $S\restriction_{j,l}$, and for $k+1$ as k. ∎

We also prove a finitary version of Lemma 6.3, which is used in the proof of the pumping lemma in the next section.

Lemma 6.5. *Let $1 \leq k \leq n$. There exists a function $b\colon \Gamma_*^k \to \mathbb{N}$, assigning a number to a k-stack, having the*

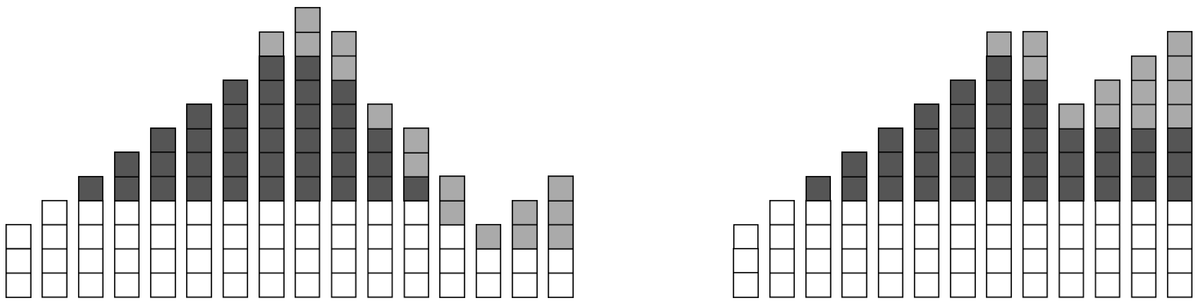

Fig. 1. Two example configurations at the end of a run of a 2-HOPDA. The 2-stack grows from left to right. White symbols were already present in $R(0)$; dark gray symbols were created while reading stars at the beginning of R. light gray symbols were created later. The run on the right is 0-upper; the one on the left is not 0-upper

following property. Let R be a run which reads only stars, and let y be a position of $R(|R|)$. Let s^k be the k-stack of $R(0)$ containing $hist(R, y)$. Assume that there exist at least $b(s^k)$ indices i such that position $hist(R\restriction_{i,|R|}, y)$ is in the topmost k-stack of $R(i)$. Then for some i, configuration $R(i)$ is a milestone and position $hist(R\restriction_{i,|R|}, y)$ is in the topmost k-stack of $R(i)$.

Proof (sketch): In the first step we define sets I^m for $0 \leq m \leq k$, similar to those in the proof of Lemma 6.3. More precisely, as I^k we take the set of indices i such that position $hist(R\restriction_{i,|R|}, y)$ is in the topmost k-stack of $R(i)$. Notice that $R\restriction_{i,j} \in up^k$ for any $i, j \in I^k$, $i \leq j$. Then by induction, if the set I^m is big enough for some m, it can be shown that then we can choose its subset I^{m-1} such that $R\restriction_{i,j} \in up^{m-1}$ for any $i, j \in I^k$, $i \leq j$. In the induction hypothesis it is also necessary to say that the topmost m-stack of $R(\min(I^m))$ is one of the m-stacks of s^k.

In the second step we prove that configuration $R(i)$ is a milestone for some $i \in I^0$. Let $\varphi\colon A^* \to M$ be a morphism into a finite monoid which checks if a word consists of only $\star$ symbols. Since the set I^0 is big enough, we can find two indices $i, j \in I^0$, $i < j$ such that $type^0_{\mathcal{A},\varphi}(R(i)) = type^0_{\mathcal{A},\varphi}(R(j))$. Applying Theorem 5.4 to run $R\restriction_{i,j}$ and to configuration $R(j)$, we obtain a 0-upper run from $R(j)$ of positive length, reading only stars. We can apply Theorem 5.4 again to the configuration at the end of this new run, and obtain as its continuation another 0-upper run. Continuing like this, we obtain an infinite run from $R(i)$ reading only stars. Additionally, because it is composed from 0-upper runs, $R(i)$ is a milestone. By construction $hist(R\restriction_{i,|R|}, y)$ is in the topmost k-stack of $R(i)$, so we obtain the thesis. ∎

VII. Pumping Lemma

In this section we present a pumping lemma which can be used to prove that a language cannot be recognized by an n-HOPDA. We begin by the statement of this lemma. For this section we fix an n-HOPDA $\mathcal{A}$ with input alphabet A containing a $\star$ symbol. Fix also a morphism $\varphi\colon A^* \to M$ into a finite monoid. For uniformity of presentation, we define $up^{-1} = \emptyset$ (no run is -1-upper).

Definition 7.1. We say that a run R' is a *pumping witness* for a run R with respect to φ, if $R(0) = R'(0)$, and the words read by R and by R' evaluate to the same under φ, and for each $0 \leq k \leq n$,

- $type^k_{\mathcal{A},\varphi}(R(|R|)) \sqsubseteq type^k_{\mathcal{A},\varphi}(R'(|R'|))$, or
- R is $(k-1)$-upper.

We say that a run R *can be pumped* with respect to φ if for each r' there exists a pumping witness R' such that the word read by R' begins with at least r' stars.

The above definition will only be used for runs R starting in a milestone. Intuitively, a run can be pumped if we can change the number of stars at its beginning in such a way that the final configuration does not change too much. In the definition we describe the behavior of the topmost k-stack (for each k). The second option, that R is $(k-1)$-upper, corresponds to a situation when a part of the topmost k-stack of $R'(|R'|)$ was created while reading the initial stars; then we do not say anything about this topmost k-stack. Otherwise, the topmost k-stack of $R(|R|)$, in some sense, was not touched while reading the initial stars; then we guarantee that $type^k_{\mathcal{A},\varphi}(R(|R|)) \sqsubseteq type^k_{\mathcal{A},\varphi}(R'(|R'|))$ (in particular the topmost k-stack of $R(|R|)$ and of $R'(|R'|)$ are the same).

Theorem 7.2 (Pumping lemma)**.** *Let c be a milestone configuration. There exists $r > 0$ such that each run from c, which reads a word beginning with (at least) r stars, can be pumped with respect to φ.*

Figure 1 presents the pumping lemma and the definition of a pumping witness for $n = 2$ and $k = 1$. The run on the left is not 0-upper (the topmost 1-stack does not contain symbols created while reading the stars). Then the theorem says that we can increase the number of stars read (i.e. increase the dark gray part of the figure) so that the topmost 1-stack is not changed. Possibly while adding just one star we obtain some completely different run. But when the number of stars read is big, we will observe some regularity in the possible ways how the dark gray part "behaves"; by choosing appropriate number of stars we will remove the last dark gray symbol from the topmost 1-stack in exactly the same way as in the original run. Notice also that it is not enough to change only the number of stars read at the beginning of the word; possibly some other parts of the word have to be changed as well (in particular the part responsible for removing the dark gray symbols has to be longer). On the right side of the figure we have a run which is 0-upper. Then we still can increase the number of stars read and obtain a similar but bigger configuration. But now probably the topmost 1-stack will change (its dark gray part will become bigger); in such situation we only say that the topmost symbol (and the types) will be preserved.

Let us mention that another pumping lemma for higher-order pushdown automata appears in [17]. There are several differences between these two lemmas. An advantage of the lemma in [17] is that it gives a precise value of r for which the lemma holds, in terms of the size of c. Moreover it works not only for deterministic HOPDA, but also for nondeterministic HOPDA in which the ε-closure of the configuration graph is finitely-branching. On the other hand in [17] we obtain the pumping witness property only for $k = 0$. Additionally it is just said that the number of symbols read by the run increases (not necessarily the number of stars at the beginning).

The rest of this section is devoted to a proof of this theorem. Notice that while checking that R' is a pumping witness for R it is enough to concentrate on one value of k, namely on the greatest k such that R is not $(k-1)$-upper (then the conditions for all other values of k easily follow). The proof is divided into three parts, as also in the run R which should be pumped we can distinguish three fragments. A first fragment of R reads only stars; its behavior is described by Lemma 7.3. The second fragment ends in the first moment j (after the initial stars are read) such that $R\restriction_{j,|R|}$ is k-upper and $R\restriction_{0,j}$ is not $(k-1)$-upper. On Figure 1 this is the moment when the last gray symbol is removed from the topmost 1-stack. The behavior of the first two fragments is described by Corollary 7.4.

Lemma 7.3. *Let c be a milestone configuration, and let $1 \leq k \leq n$. Then there exists a finite set $\mathcal{S}^k$ of k-stacks having the following property. Let R be a run from c reading only stars. Let x be the bottommost position of the topmost $(k-1)$-stack in some k-stack of $R(|R|)$. Assume that $hist(R, x) \neq top^{k-1}(c)$. Then the k-stack of $R(|R|)$ containing x is in $\mathcal{S}^k$.*

Proof: Let $\mathcal{X}$ be the set containing all k-stacks of c, and additionally the topmost k-stack of c with its topmost $(k-1)$-stack removed. Let $\mathcal{S}^k$ contain all k-stack which can be obtained from a k-stack $s^k \in \mathcal{X}$ by applying at most $b(s^k)$ of push and pop operations, where b is the function from Lemma 6.5.

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

Let us fix some c-clear k-stack of $R(|R|)$, let y be its bottommost position. Consider the smallest index i for which this k-stack (more precisely the k-stack of $R(i)$ containing $hist(R\restriction_{i,|R|}, y)$) is c-clear in $R(i)$. Then this k-stack of $R(i)$ is in $\mathcal{X}$. Indeed, either $i = 0$ and it is one of the k-stacks of c, or it is derived from the topmost k-stack of c, and we have just removed from it the remainings of the topmost $(k-1)$-stack of c (thus it is the topmost k-stack of c with its topmost $(k-1)$-stack removed).

Next we observe that this k-stack s^k of $R(i)$ could not be modified more than $b(s^k)$ times before $R(|R|)$. Otherwise, by Lemma 6.5, some configuration $R(j)$ in which this is the topmost k-stack would be a milestone. As also c is a milestone, Lemma 6.4 would imply that $R\restriction_{0,j}$ is 0-upper, but this is impossible when the topmost k-stack of $R(j)$ is c-clear. ■

Corollary 7.4. *Let c be a milestone configuration. Then there exists a finite set $\mathcal{S}$ of configurations having the following property. Let $0 \leq k \leq n$, let R be a run from c, and let $0 \leq r \leq |R|$ be such that $R\restriction_{0,r}$ reads only stars. Assume that R is not $(k-1)$-upper, but for $r \leq i < |R|$ either $R\restriction_{0,i}$ is $(k-1)$-upper or $R\restriction_{i,|R|}$ is not k-upper. Then for some configuration $d \in \mathcal{S}$, we have $type^k_{\mathcal{A},\varphi}(R(|R|)) = type^k_{\mathcal{A},\varphi}(d)$.*

Proof: Recall that $type^k_{\mathcal{A},\varphi}(d)$ returns an element of $\mathcal{T}_{\mathcal{A},\varphi}$, and the topmost k-stack of d. We have only finitely elements of $\mathcal{T}_{\mathcal{A},\varphi}$. So it is enough to show, for each k, that there are only finitely many possible topmost k-stacks over all configurations $R(|R|)$ satisfying the assumptions. For $k = 0$ this is trivial as 0-stack contains just one symbol. So let $1 \leq k \leq n$. We have two cases.

First assume that $R\restriction_{i,|R|}$ is k-upper for some $r \leq i < |R|$; let i be the greatest index in this set. Then $R\restriction_{0,i}$ is $(k-1)$-upper, but R is not. This means that the topmost k-stack of $R(|R|)$ can be obtained from the topmost k-stack of c by removing the topmost $(k-1)$-stack; thus the content of this k-stack is fixed.

The other case is that $R\restriction_{i,|R|}$ is not k-upper for every $r \leq i < |R|$. This means that the topmost k-stack of $R(|R|)$ is an unchanged copy of some k-stack of $R(r)$. As R is not $(k-1)$-upper, this k-stack of $R(r)$ is c-clear, and by Lemma 7.3 it is in the finite set $\mathcal{S}^k$. ■

Proof of Theorem 7.2: Consider the infinite run S starting at the milestone configuration c and reading only stars. Consider first the degenerate case when in S only finitely many stars are read. As r we take their number, plus one. Then the thesis is satisfied trivially, as there is no run from c which reads a word beginning with r stars. So for the rest of the proof assume that S reads infinitely many stars.

Let $\mathcal{S}$ be the set from Corollary 7.4 (used for c). For each $i \geq 1$ we define the set $T_i \subseteq \{0, 1, \ldots, n\} \times \mathcal{S} \times M$ as follows. A triple (k, 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, evaluates to m under φ, and $type^k_{\mathcal{A},\varphi}(R(|R|)) = type^k_{\mathcal{A},\varphi}(d)$. By definition $T_i \subseteq T_{i+1}$ (for each i), and there are only finitely many possible sets, so from some moment every T_i is the same. As the required number of stars (in the statement of the pumping lemma) we take such $r > 0$ that $T_i = T_r$ for each $i \geq r$.

Consider now any number r' (we may assume that $r' \geq r$) and any run R from c which reads a word beginning with at least r stars. Our goal is to show a pumping witness R' for

R such that the word read by R' begins with at least r' stars. Let k be the greatest number ($0 \leq k \leq n$) such that R is not $(k-1)$-upper. Such k exists, as $k=0$ is always good (recall that by definition no run is -1-upper). Let i be an index such that $R{\restriction}_{0,i}$ reads exactly r stars. Let $j \geq i$ be the smallest index such that $R{\restriction}_{j,|R|}$ is k-upper and $R{\restriction}_{0,j}$ is not $(k-1)$-upper. Such j exists, as $j = |R|$ is always good.

We use Corollary 7.4 for k, for $R{\restriction}_{0,j}$ (as R), and for i (as r). Its assumptions are satisfied by minimality of j. So we get that $type^k_{\mathcal{A},\varphi}(R(j)) = type^k_{\mathcal{A},\varphi}(d)$, for some $d \in \mathcal{S}$. It means that $(k, d, m) \in T_r$, where m is the image under φ of the word read by $R{\restriction}_{0,j}$. Because $T_r = T_{r'}$, there exists a run U from c such that the word read by U begins with (at least) r' stars, evaluates to m under φ, and $type^k_{\mathcal{A},\varphi}(U(|U|)) = type^k_{\mathcal{A},\varphi}(R(j))$.

Finally, we use Theorem 5.4 for $R{\restriction}_{j,|R|}$ is order to obtain a k-upper run U' from $U(|U|)$, and we observe that $U \circ U'$ is a pumping witness for R as required. ∎

VIII. Why U Cannot Be Recognized?

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

Of course our proof goes by contradiction: assume that for some n we have an $(n-1)$-HOPDA recognizing U. We construct an n-HOPDA $\mathcal{A}$ which works as follows. First it makes a push^n operation. Then it simulates the $(n-1)$-HOPDA (not using the push^n and pop^n operations). When the $(n-1)$-HOPDA is going to accept, $\mathcal{A}$ makes the pop^n operation and afterwards accepts. Clearly, $\mathcal{A}$ recognizes U as well. Such normalization allows us to use Theorem 5.5, as in $\mathcal{A}$ every accepting run is an n-return.

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

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

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

- if word w contains the $\sharp$ symbol, and
- if the pattern of w is longer than $|w_n|$, and
- the exact value of the pattern of w, assuming that the pattern is not longer than $|w_n|$.

We fix a run R, and an index $z(w)$ for each prefix w of w_n, such that the following holds. Run R begins in the initial configuration. Between $R(0)$ and $R(z(\varepsilon))$ only stars are read. For each prefix w of w_n, configuration $R(z(w))$ is a milestone. Just after $z(w)$ run R reads r stars, where r is the constant from Theorem 7.2 used for morphism φ and for $R(z(w))$ (as c). If $w = va$ (where a is a single letter), the word read by R between $R(z(w))$ and $R(z(wa))$ consists of a surrounded by some number of stars. Of course such run R exists: we read stars until we reach a milestone (succeeds thanks to Lemma 6.3), then we read as many stars as required by the pumping lemma, then we read the next letter of w_n, and so on (because $\mathcal{A}$ accepts U, it will never block).

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

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

Lemma 8.1. *Let $-1 \leq k \leq n-1$, and let u be a word such that uw_{k+1} is a prefix of w_n. Then there exist a prefix v of w_{k+1} such that $\langle uv, uw_{k+1}\rangle$ is not k-upper.*

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

Let now $k \geq 0$. Figure 2 might be helpful in finding different runs present in the proof below. Assume 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.

From the induction assumption (where uw_k^{i-1} is taken as u), for each $1 \leq i \leq N$ there exist a prefix v_i of w_k such that $\langle uw_k^{i-1}v_i, uw_k^i\rangle$ is not $(k-1)$-upper. As $\langle uw_k^i, uw_k^N\rangle$ is k-upper (property $\heartsuit$), from Proposition 4.4 we know that $\langle uw_k^{i-1}v_i, uw_k^N\rangle$ is not $(k-1)$-upper.

Now we are ready to use the pumping lemma (Theorem 7.2). For each $1 \leq i \leq N$ we use it for $\langle uw_k^{i-1}v_i, uw_k^N\rangle$. Recall from the definition of R that the word read by this subrun begins with such a number of stars that the pumping lemma can be used. So this subrun can be pumped. For each number l we obtain a pumping witness $S_{i,l}$ which reads a word beginning with at least l stars; let $d_{i,l} = S_{i,l}(|S_{i,l}|)$. From the definition of a pumping witness (Definition 7.1), we have a run $S'_{i,l} := R{\restriction}_{0,z(uw_k^{i-1}v_i)} \circ S_{i,l}$ from the initial configuration to $d_{i,l}$ which reads a word having pattern uw_k^N. Moreover, because $\langle uw_k^{i-1}v_i, uw_k^N\rangle$ is not $(k-1)$-upper, we obtain that $type^k_{\mathcal{A},\varphi}(R(z(uw_k^N))) \sqsubseteq type^k_{\mathcal{A},\varphi}(d_{i,l})$.

Because there are only finitely many possible values of $type_{\mathcal{A},\lambda}$, we can assume that $type_{\mathcal{A},\lambda}(d_{i,l}) = type_{\mathcal{A},\lambda}(d_{i,j})$ for $1 \leq i \leq N$ and each l and j. Indeed, we can choose (for each i separately) some value of $type_{\mathcal{A},\lambda}(d_{i,l})$ which appears

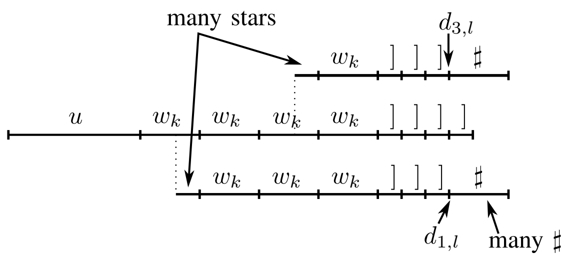

Fig. 2. Illustration of runs appearing in the proof (where $N = 4$, $x = 1$, $y = 3$). Recall that stars can appear between letters of the words

infinitely often, and then we take the subsequence of only these $d_{i,l}$ which give this value.

Since there are more possible indices $i \in \{1, 2, \dots, N\}$ than the number of classes of the $(\mathcal{A}, \lambda)$-sequence equivalence relation, times $|\mathcal{T}_{\mathcal{A},\lambda}|$, there have to exist two indices $1 \leq x < y \leq N$ such that $type_{\mathcal{A},\lambda}(d_{x,1}) = type_{\mathcal{A},\lambda}(d_{y,1})$, and the sequences $d_{x,1}, d_{x,2}, d_{x,3}, \dots$ and $d_{y,1}, d_{y,2}, d_{y,3}, \dots$ are $(\mathcal{A}, \lambda)$-sequence equivalent. From now we fix these two indices x, y. Furthermore, because $type^k_{\mathcal{A},\varphi}(R(z(uw_k^N))) \sqsubseteq type^k_{\mathcal{A},\varphi}(d_{i,l})$ for each $1 \leq i \leq N$ and each l, we know that the topmost k-stacks of all $d_{x,l}$ and $d_{y,l}$ are the same. Thus $type^k_{\mathcal{A},\lambda}(d_{x,l}) = type^k_{\mathcal{A},\lambda}(d_{y,j})$ for each l and j.

Let $r = N - x + 1$. We will construct a run $R' = R'_1 \circ R'_2 \circ \dots \circ R'_r$ from $d_{x,1}$ such that for $1 \leq i < r$, R'_i is a k-upper run reading word of the form $\star^*]\star^*$ (a closing bracket surrounded by some number of stars), and R'_r is an n-return reading only $\sharp$ symbols. Notice that $type^k_{\mathcal{A},\varphi}(R(z(uw_k^N))) \sqsubseteq type^k_{\mathcal{A},\varphi}(d_{x,1})$, and that $\langle uw_k^N]^{i-1}, uw_k^N]^i\rangle$ is k-upper for each i (property $\heartsuit$). Consecutively for $1 \leq i < r$, R'_i is obtained by applying Theorem 5.4 to $\langle uw_k^N]^{i-1}, uw_k^N]^i\rangle$ and to configuration $R'_{i-1}(|R'_{i-1}|)$ (or $d_{x,1}$ for $i = 1$). Because $\mathcal{A}$ recognizes U, there is a run from $R'_{r-1}(|R'_{r-1}|)$ to an accepting configuration reading only $\sharp$ symbols; we take it as R'_r. By construction of $\mathcal{A}$, R'_r is an n-return. So we obtain a run R' as declared.

Finally we use Theorem 5.5 for λ (as φ), k, sequences $d_{x,1}, d_{x,2}, d_{x,3}, \dots$ (as $c_1, c_2, c_3, \dots$) and $d_{y,1}, d_{y,2}, d_{y,3}, \dots$ (as $d_1, d_2, d_3, \dots$), and for run R' (as R).[4] As noticed above (in particular because $R'(0) = d_{x,1}$) we have $type^k_{\mathcal{A},\lambda}(R'(0)) = type^k_{\mathcal{A},\lambda}(d_{x,l}) = type^k_{\mathcal{A},\lambda}(d_{y,l})$ for each l. Thus the assumptions of the theorem are satisfied. For each l, we obtain runs S_l (from $d_{x,l}$) and T_l (from $d_{y,l}$). The word read by any of these runs contains $r - 1 = N - x$ closing brackets with some number of stars around them, and after them some number of the $\sharp$ symbols.

For each l, let x_l and y_l be the number of the $\sharp$ symbols read by S_l and T_l, respectively. The pattern of the word read by $S'_{x,l} \circ S_l$, for each l, is $uw_k^N]^{N-x}$; the same for $S'_{x,l} \circ T_l$. In this pattern the last opening bracket which is not closed is the last bracket of the x-th w_k after u. Recall that configurations $d_{x,l}$ were obtained by pumping inside the x-th w_k, so before this bracket; for $l \to \infty$ the number of stars inserted there is unbounded. From the definition of language U it follows that the sequence $x_1, x_2, x_3, \dots$ has to be unbounded. On the other hand, configurations $d_{y,l}$ were obtained by pumping inside the y-th w_k, so after the last opening bracket which was not closed (as $y > x$). For each l the number of stars before this bracket is the same. From the definition of language U it follows that the sequence $y_1, y_2, y_3, \dots$ has to be constant, hence bounded. This contradicts with the thesis of Theorem 5.5, which says that either both these sequences are bounded or both unbounded. ■

[4]We cannot use $R\restriction_{z(uw_k^N),|R|}$ instead of R', because do not know anything about $type_{\mathcal{A},\lambda}(R(z(uw_k^N)))$.

ACKNOWLEDGMENT

We thank A. Kartzow and the anonymous reviewers for their constructive comments. Work partially supported by the Polish Ministry of Science grant nr N N206 567840.

REFERENCES

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

[2] A. Carayol and S. Wöhrle, "The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata," in *FSTTCS*, ser. Lecture Notes in Computer Science, P. K. Pandya and J. Radhakrishnan, Eds., vol. 2914. Springer, 2003, pp. 112–123.

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

[4] R. H. Gilman, "A shrinking lemma for indexed languages," *Theor. Comput. Sci.*, vol. 163, no. 1&2, pp. 277–281, 1996.

[5] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre, "Collapsible pushdown automata and recursion schemes," in *LICS*. IEEE Computer Society, 2008, pp. 452–461.

[6] T. Hayashi, "On derivation trees of indexed grammars," *Publ. RIMS, Kyoto Univ.*, vol. 9, pp. 61–92, 1973.

[7] A. Kartzow, "Collapsible pushdown graphs of level 2 are tree-automatic," in *STACS*, ser. LIPIcs, J.-Y. Marion and T. Schwentick, Eds., vol. 5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 501–512.

[8] ——, "A pumping lemma for collapsible pushdown graphs of level 2," in *CSL*, ser. LIPIcs, M. Bezem, Ed., vol. 12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 322–336.

[9] A. Kartzow and P. Parys, "Strictness of the collapsible pushdown hierarchy," *CoRR*, vol. abs/1201.3250, 2012.

[10] T. Knapik, D. Niwinski, and P. Urzyczyn, "Higher-order pushdown trees are easy," in *FoSSaCS*, ser. Lecture Notes in Computer Science, M. Nielsen and U. Engberg, Eds., vol. 2303. Springer, 2002, pp. 205–222.

[11] T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz, "Unsafe grammars and panic automata," in *ICALP*, ser. Lecture Notes in Computer Science, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580. Springer, 2005, pp. 1450–1461.

[12] N. Kobayashi, "Model-checking higher-order functions," in *PPDP*, A. Porto and F. J. López-Fraguas, Eds. ACM, 2009, pp. 25–36.

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

[14] A. N. Maslov, "Multilevel stack automata," *Problems of Information Transmission*, vol. 12, pp. 38–43, 1976.

[15] C.-H. L. Ong, "On model-checking trees generated by higher-order recursion schemes," in *LICS*. IEEE Computer Society, 2006, pp. 81–90.

[16] P. Parys, "Collapse operation increases expressive power of deterministic higher order pushdown automata," in *STACS*, ser. LIPIcs, T. Schwentick and C. Dürr, Eds., vol. 9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 603–614.

[17] P. Parys, "A pumping lemma for pushdown graphs of any level," 2011, accepted to STACS 2012.