Reachability in Higher-Order-Counters*

Alexander Heußner[1] and Alexander Kartzow[2]

[1] Otto-Friedrich-Universität Bamberg, Germany
[2] Universität Leipzig, Germany

Abstract. Higher-order counter automata (HOCA) can be either seen as a restriction of higher-order pushdown automata (HOPA) to a unary stack alphabet, or as an extension of counter automata to higher levels. We distinguish two principal kinds of HOCA: those that can test whether the topmost counter value is zero and those which cannot.

We show that control-state reachability for level k HOCA with 0-test is complete for $(k-2)$-fold exponential space; leaving out the 0-test leads to completeness for $(k-2)$-fold exponential time. Restricting HOCA (without 0-test) to level 2, we prove that global (forward or backward) reachability analysis is **P**-complete. This enhances the known result for pushdown systems which are subsumed by level 2 HOCA without 0-test.

We transfer our results to the formal language setting. Assuming that $\mathbf{P} \subsetneq \mathbf{PSPACE} \subsetneq \mathbf{EXPTIME}$, we apply proof ideas of Engelfriet and conclude that the hierarchies of languages of HOPA and of HOCA form strictly interleaving hierarchies. Interestingly, Engelfriet's constructions also allow to conclude immediately that the hierarchy of collapsible pushdown languages is strict level-by-level due to the existing complexity results for reachability on collapsible pushdown graphs. This answers an open question independently asked by Parys and by Kobayashi.

1 From Higher-Order Pushdowns to Counters and Back

Higher-order pushdown automata (HOPA) — also known as iterated pushdown automata — were first introduced by Maslov in [15] and [16] as an extension of classical pushdown automata where the pushdown storage is replaced by a nested pushdown of pushdowns of ... of pushdowns. After being originally studied as acceptors of languages, these automata have nowadays obtained renewed interest as computational model due to their connection to safe higher-order recursion schemes. Recent results focus on algorithmic questions concerning the underlying configuration graphs, e.g., Carayol and Wöhrle [5] showed decidability of the monadic second-order theories of higher-order pushdown graphs due to the pushdown graph's connection to the Caucal-hierarchy [6], and Hague and Ong determined the precise complexity of the global backwards reachability problem for HOPA: for level k it is complete for $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k-1}(n^d))$ [9].[1]

* The second author is supported by the DFG research project GELO. We both thank M. Bojańczyk, Ch. Broadbent, and M. Lohrey for helpful discussions and comments.

[1] We define $\exp_0(n) := n$ and $\exp_{k+1}(n) := \exp(\exp_k(n))$ for any natural number k.

K. Chatterjee and J. Sgall (Eds.): MFCS 2013, LNCS 8087, pp. 528–539, 2013.

In the setting of classical pushdown automata it is well known that restricting the stack alphabet to one single symbol, i.e., reducing the pushdown storage to a counter, often makes solving algorithmic problems easier. For instance, control state reachability for pushdown automata is **P**-complete whereas it is **NSPACE**$(\log(n))$-complete for counter automata. Then again, results from counter automata raise new insights to the pushdown case by providing algorithmic lower bounds and important subclasses of accepted languages separating different classes of complexity. In this paper we lift this idea to the higher-order setting by investigating reachability problems for higher-order counter automata (HOCA), i.e., HOPA over a one-element stack alphabet. Analogously to counter automata, we introduce level k HOCA in two variants: with or without 0-tests. Throughout this paper, we write k-HOCA$^-$ for the variant without 0-tests and k-HOCA$^+$ for the variant with 0-tests. Transferring our results' constructions back to HOPA will then allow to answer a recent open question [17,14].

To our knowledge, the only existing publication on HOCA is by Slaats [18]. She proved that $(k+1)$-HOCA$^+$ can simulate level k pushdown automata (abbreviated k-HOPA). In fact, even $(k+1)$-HOCA$^-$ simulate k-HOPA. Slaats conjectured that $L(k\text{-HOCA}^+) \subsetneq L(k\text{-HOPA})$ where $L(X)$ denotes the languages accepted by automata of type X. We can confirm this conjecture by combining the proof ideas of Engelfriet [7] with our main result on control-state reachability for HOCA in Theorems 13 and 14: control state reachability on k-HOCA$^+$ is complete for $\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$ and control state reachability on k-HOCA$^-$ is complete for $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$. These results are obtained by adapting a proof strategy relying on reductions to bounded space storage automata originally stated for HOPA by Engelfriet [7]. His main tool are auxiliary $\mathbf{SPACE}(b(n))$ P^k automata where P^k denotes the storage type of a k-fold nested pushdown (see Section 2 for a precise definition). Such a (two-way) automaton has an additional storage of type P^k, and a Turing machine worktape with space $b(n)$. His main technical result shows a trade off between the space bound b and the number of iterated pushdowns k. Roughly speaking, exponentially more space allows to reduce the number of nestings of pushdowns by one. Similarly, at the cost of another level of pushdown, one can trade alternation against nondeterminism. Here, we also restate reachability on k-HOCA$^+$ as a membership problem on alternating auxiliary $\mathsf{SPACE}(\exp_{k-3}(n))$ $\mathcal{Z}+$ automata (where $\mathcal{Z}+$ is the new storage type of a counter with 0-test). For our $\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$-hardness proof we provide a reduction of $\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp(\exp_{k-3}(n^d)))$ to alternating auxiliary $\mathbf{SPACE}(\exp_{k-3}(n))$ $\mathcal{Z}+$ automata that is inspired by Jancar and Sawa's **PSPACE**-completeness proof for the non-emptiness of alternating automata [11]. For containment we adapt the proof of Engelfriet [7] and show that membership for alternating auxiliary $\mathbf{SPACE}(\exp_{k-3}(n))$ $\mathcal{Z}+$ automata can be reduced to alternating reachability on counter automata of size $\exp_{k-2}(n)$, where n is the size of the original input, which is known to be in $\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$ (cf. [8]).

For the case of k-HOCA$^-$ the hardness follows directly from the hardness of reachability for level $(k-1)$ pushdown automata and the fact that the latter can

be simulated by k-HOCA$^-$. For containment in $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$ the mentioned machinery of Engelfriet reduces the problem to the case $k = 2$.

The proof that control-state reachability on 2-HOCA$^-$ is in $\mathbf{P}$ is implied by Theorem 5 which proves a stronger result: both the global regular forward and backward reachability problems for 2-HOCA$^-$ are $\mathbf{P}$-complete. The backward reachability problem asks, given a regular set C of configurations, for a (regular) description of all configurations that allow to reach one in C. This set is typically denoted as $\mathsf{pre}^*(C)$. Note that there is no canonical way of defining a regular set of configurations of 2-HOCA$^-$. We are aware of at least three possible notions: regularity via 2-store automata [2], via sequences of pushdown-operations [4], and via encoding in regular sets of trees. We stick to the latter, and use the encoding of configurations as binary trees introduced in [12]: We call a set C of configurations regular if the set of encodings of configurations $\{\mathsf{E}(c) \mid c \in C\}$ is a regular set of trees (where E denotes the encoding function from [12]). Note that the other two notions of regularity are both strictly weaker (with respect to expressive power) than the notion of regularity we use here. Nevertheless, our result does not carry over to these other notions of regularity as they admit more succinct representations of certain sets of configurations. See [10] for details.

Besides computing $\mathsf{pre}^*(C)$ in polynomial time our algorithm also allows to compute the reachable configurations $\mathsf{post}^*(C)$ in polynomial time. Thus, 2-HOCA$^-$ subsumes the well-known class of pushdown systems [1] while still possessing the same good complexity with respect to reachability problems.

Due to the lack of space, detailed formal proofs are deferred to a long version of this article [10].

2 Formal Model of Higher-Order Counters

2.1 Storage Types and Automata

An elegant way for defining HOCA and HOPA is the use of storage types and operators on these (following [7]). For simplicity, we restrict ourselves to what Engelfriet calls *finitely encoded* storage types.

Definition 1. *For X some set, we call a function $t : X \to \{true, false\}$ an X-test and a partial function $f : X \to X$ an X-operation.*
A storage type *is a tuple $\mathcal{S} = (X, T, F, x_0)$ where X is the set of $\mathcal{S}$-configurations, $x_0 \in X$ the initial $\mathcal{S}$-configuration, T a finite set of X-tests and F a finite set of X-operations containing the identity on X, i.e., $\mathsf{id}_X \in F$.*

Let us fix some finite alphabet Σ with a distinguished symbol $\bot \in \Sigma$. Let $\mathcal{P}_\Sigma = (X, T, F, x_0)$ be the *pushdown storage type* where $X = \Sigma^+$, $x_0 = \bot$, $T = \{\mathsf{top}_\sigma \mid \sigma \in \Sigma\}$ with $\mathsf{top}_\sigma(w) = true$ if $w \in \Sigma^*\sigma$, and $F = \{\mathsf{push}_\sigma \mid \sigma \in \Sigma\} \cup \{\mathsf{pop}, \mathsf{id}\}$ with $\mathsf{id} = \mathsf{id}_X$, $\mathsf{push}_\sigma(w) = w\sigma$ for all $w \in X$, and $\mathsf{pop}(w\sigma) = w$ for all $w \in \Sigma^+$ and $\sigma \in \Sigma$ and $\mathsf{pop}(\sigma)$ undefined for all $\sigma \in \Sigma$. Hence, $\mathcal{P}_\Sigma$ represents a classical pushdown stack over the alphabet Σ. We write $\mathcal{P}$ for $\mathcal{P}_{\{\bot,0,1\}}$.

We define the storage type *counter without* 0*-test* $\mathcal{Z} = \mathcal{P}_{\{\bot\}}$, which is the pushdown storage over a unary pushdown alphabet. We define the storage type *counter with* 0*-test* $\mathcal{Z}+$ exactly like $\mathcal{Z}$ but we add the test *empty*? to the set of tests where $\mathit{empty}?(x) = \mathit{true}$ if $x = \bot$ (the plus in $\mathcal{Z}+$ stands for "with 0-test"). In other words, *empty*? returns false iff the operation pop is applicable.

Definition 2. *For a storage type* $\mathcal{S} = (X, T, F, x_0)$ *we define an* $\mathcal{S}$ automaton *as a tuple* $\mathcal{A} = (Q, q_0, q_f, \Delta)$ *where as usual* Q *is a finite set of states with initial state* q_0 *and final state* q_f *and* Δ *is the transition relation. The difference to a usual automaton is the definition of* Δ *by* $\Delta = Q \times \{\mathit{true}, \mathit{false}\}^T \times Q \times F$.

For $q \in Q$ and $x \in X$, a transition $\delta = (q, R, p, f)$ is applicable to the *configuration* (q, x) if $f(x)$ is defined and if for each test $t \in T$ we have $R(t) = t(x)$, i.e., the result of the storage-tests on the storage configuration x agree with the test results required by the transition δ. If δ is applicable, application of δ leads to the configuration $(p, f(x))$. The notions of a run, the accepted language, etc. are now all defined as expected.

The Pushdown Operator We also consider $\mathcal{P}_\Sigma$ as an operator on other storage types as follows. Given a storage type $\mathcal{S} = (X, T, F, x_0)$ let the storage type *pushdown of* $\mathcal{S}$ be $\mathcal{P}_\Sigma(\mathcal{S}) = (X', T', F', x_0')$ where $X' = (\Sigma \times X)^+$, $x_0' = (\bot, x_0)$, $T' = \{\mathsf{top}_\sigma \mid \sigma \in \Sigma\} \cup \{\mathit{test}(t) \mid t \in T\}$, $F' = \{\mathsf{push}_{\gamma,f} \mid \gamma \in \Sigma, f \in F\} \cup \{\mathsf{stay}_f \mid f \in F\} \cup \{\mathit{pop}\}$, and where for all $x' = \beta(\sigma, x)$, $\beta \in (\Sigma \times X)^*$, $\sigma \in \Sigma$, $x \in X$ it holds that

- $\mathsf{top}_\tau(x') = (\tau = \sigma)$,
- $\mathit{test}(t)(x') = t(x)$,
- $\mathsf{push}_{\tau,f}(x') = \beta(\sigma, x)(\tau, f(x))$ if f is defined on x (and undefined otherwise),
- $\mathsf{stay}_f(x') = \beta(\sigma, f(x))$ if f is defined on x (and undefined otherwise), and
- $\mathsf{pop}(x') = \beta$ if β is nonempty (and undefined otherwise).

Note that $\mathsf{stay}_{\mathsf{id}_X} = \mathsf{id}_{X'}$ whence F' contains the identity. As for storages, we define the operator $\mathcal{P}$ to be the operator $\mathcal{P}_{\{\bot,0,1\}}$.

2.2 HOPA, HOCA, and Their Reachability Problems

We can define the iterative application of the operator $\mathcal{P}$ on some storage $\mathcal{S}$ as follows: let $\mathcal{P}^0(\mathcal{S}) = \mathcal{S}$ and $\mathcal{P}^{k+1}(\mathcal{S}) = \mathcal{P}(\mathcal{P}^k(\mathcal{S}))$. A *level* k *higher-order pushdown automaton* is a $\mathcal{P}^{k-1}(\mathcal{P})$ automaton. We abbreviate the class of all these automata with k-HOPA. A *level* k *higher-order counter automaton with zero-test* is a $\mathcal{P}^{k-1}(\mathcal{Z}+)$ automaton and k-HOCA$^+$ denotes the corresponding class.[2]. Similarly, k-HOCA$^-$ denotes the class of *level* k *higher-order counter automata without zero-test* which is the class of $\mathcal{P}^{k-1}(\mathcal{Z})$ automata. Obviously,

[2] A priori our definition of k-HOCA$^+$ results in a stronger automaton model than that used by Slaats. In fact, both models are equivalent (cf. [10]).

for any level k it holds that $L(\text{k-HOCA}^-) \subseteq L(\text{k-HOCA}^+) \subseteq L(\text{k-HOPA})$ where $L(X)$ denotes the languages accepted by automata of type X.

We next define the reachability problems which we study in this paper.

Definition 3. *Given an $\mathcal{S}$ automaton and one of its control states $q \in Q$, then the* control state reachability problem *asks whether there is a configuration (q, x) that is reachable from (q_0, x_0) where $x \in X$ is an arbitrary $\mathcal{S}$-configuration.*

Assuming a notion of regularity for sets of $\mathcal{S}$ configurations (and hence for sets of configurations of $\mathcal{S}$ automata), we can also define a global variant of the control state reachability problem.

Definition 4. *Given an $\mathcal{S}$ automaton $\mathcal{A}$ and a regular set of configurations C, the* regular backwards reachability problem *demands a description of the set of configurations from which there is a path to some configuration $c \in C$.*

Analogously, the regular forward reachability problem asks for a description of the set of configurations reachable from a given regular set C. In the following section, we consider the regular backwards (and forwards) reachability problem for the class of 2-HOCA$^-$ only.

3 Regular Reachability for 2-HOCA$^-$

The goal of this section is to prove the following theorem extending a known result on regular reachability on pushdown systems to 2-HOCA$^-$:

Theorem 5. *Reg. backwards/forwards reachability on* 2-HOCA$^-$ *is* **P***-complete.*

3.1 Returns, Loops, and Control State Reachability

Proving Theorem 5 is based on the "returns-&-loops" construction for 2-HOPA of [12]. As a first step, we consider the simpler case of control-state reachability:

Proposition 6. *Control state reachability for* 2-HOCA$^-$ *is* **P***-complete.*

In [12] it has been shown that certain runs, so-called *loops* and *returns*, are the building blocks of any run of a 2-HOPA in the sense that solving a reachability problem amounts to deciding whether certain loops and returns exist. Here, we analyse these notions more precisely in the context of 2-HOCA$^-$ in order to derive a polynomial control state reachability algorithm. Using this algorithm we can then also solve the regular backwards reachability problem efficiently.

For this section, we fix a $\mathcal{P}(\mathcal{Z})$-automaton $\mathcal{A} = (Q, q_0, F, \Delta)$. Recall that the $\mathcal{P}(\mathcal{Z})$-configurations of $\mathcal{A}$ are elements of $(\Sigma \times \{\bot\}^+)^+$. We identify $\bot^{m+1}$ with the natural number m and the set of storage configurations with $(\Sigma \times \mathbb{N})^+$.

Definition 7. *Let $s \in (\Sigma \times \mathbb{N})^+$, $t \in \Sigma \times \mathbb{N}$ and $q, q' \in Q$ be states of $\mathcal{A}$. A* return *of $\mathcal{A}$ from (q, st) to (q', s) is a run r from (q, st) to (q', s) such that except for the final configuration no configuration of r is in $Q \times \{s\}$.*

Let $s \in (\Sigma \times \mathbb{N})^*$, $t \in \Sigma \times \mathbb{N}$. A loop *of $\mathcal{A}$ from (q, st) to (q', st) is a run r from (q, st) to (q', st) such that no configuration of r is in $Q \times \{s\}$.*

One of the underlying reasons why control state reachability for pushdown systems can be efficiently solved is the fact that it is always possible to reach a certain state without increasing the pushdown by more than polynomially many elements. In the following, we prove an analogue of this fact for $\mathcal{P}(\mathcal{Z})$. For a given configuration, if there is a return or loop starting in this configuration, then this return or loop can be realised without increasing the (level 2) pushdown more than polynomially. This is due to the monotonic behaviour of $\mathcal{Z}$: given a $\mathcal{Z}$ configuration x, if we can apply a sequence φ of transitions to x then we can apply φ to all bigger configurations, i.e., to any configuration of the form $\mathsf{push}_\perp^n(x)$. Note that this depends on the fact that $\mathcal{Z}$ contains only trivial tests (the test $\mathsf{top}_\perp$ always returns true). In contrast, for $\mathcal{Z}+$, if φ applies a couple of pop operations and then tests for zero and performs a transition, then this is not applicable to a bigger counter because the 0-test would now fail.

For a $\mathcal{P}(\mathcal{Z})$ configuration $x = (\sigma_1, n_1)(\sigma_2, n_2)\dots(\sigma_m, n_m)$, let $|x| = m$ be its *height*. Let r be some run starting in (q, x) for some $q \in Q$. The run r *increases the height by at most* k if $|x'| \leq |x| + k$ for all configurations (q', x') of r.

Definition 8. *Let $s \in (\{\perp\} \times \mathbb{N})^+$. We write $\mathsf{ret}_k(s)$ and $\mathsf{lp}_k(s)$, resp., for the set of pairs of initial and final control states of returns or loops starting in s and increasing the height by at most k. We write $\mathsf{ret}_\infty(s)$ and $\mathsf{lp}_\infty(s)$,resp., for the union of all $\mathsf{ret}_k(sw)$ or $\mathsf{lp}_k(s)$.*

The existence of a return (or loop) starting in sw (or $s'w$) (with $s \in (\{\perp\} \times \mathbb{N})^+, s' \in (\{\perp\} \times \mathbb{N})^*$ and $w \in \{\perp\} \times \mathbb{N}$) does not depend on the concrete choice of s or s'. Thus, we also write $\mathsf{ret}_k(w)$ for $\mathsf{ret}_k(sw)$ and $\mathsf{lp}_k(w)$ for $\mathsf{lp}_k(s'w)$.

By induction on the length of a run, we first prove that $\mathcal{P}(\mathcal{Z})$ is *monotone* in the following sense: let $s \in (\Sigma \times \mathbb{N})^*, t = (\sigma, n) \in \Sigma \times \mathbb{N}$, $q, q' \in Q$ and r a run starting in (q, st) and ending in state q'. If the topmost counter of each configuration of r is at least m, then for each $n' \geq n - m$ there is a run r' starting in $(q, s(\sigma, n'))$ and performing exactly the same transitions as r. In particular, for all $k \in \mathbb{N} \cup \{\infty\}$, $\sigma \in \Sigma$ and $m_1 \leq m_2 \in \mathbb{N}$, $\mathsf{ret}_k((\sigma, m_1)) \subseteq \mathsf{ret}_k((\sigma, m_2))$ and $\mathsf{lp}_k((\sigma, m_1)) \subseteq \mathsf{lp}_k((\sigma, m_2))$.

We next show that the sequence $(\mathsf{ret}_k((\sigma, m)))_{m \in \mathbb{N}}$ stabilises at $m = |\Sigma||Q|^2$. From this we conclude that $\mathsf{ret}_\infty = \mathsf{ret}_{|\Sigma|^2|Q|^4}$, i.e., in order to realise a return with arbitrary fixed initial and final configuration, we do not have to increase the height by more than $|\Sigma|^2|Q|^4$ (if there is such a return at all).

Lemma 9. *For $k \in \mathbb{N} \cup \{\infty\}$, $\sigma \in \Sigma$, $m \geq |\Sigma||Q|^2$, and $m' \geq 2 \cdot |\Sigma||Q|^2$, we have $\mathsf{ret}_k((\sigma, m)) = \mathsf{ret}_k((\sigma, |\Sigma||Q|^2))$ and $\mathsf{lp}_\infty((\sigma, m')) = \mathsf{lp}_\infty((\sigma, 2 \cdot |\Sigma||Q|^2))$.*

The proof uses the fact that we can find an $m' \leq |\Sigma||Q|^2$ with $\mathsf{ret}_k((\sigma, m')) = \mathsf{ret}_k(\sigma, m' + 1)$ for all σ by the pigeonhole-principle. Using monotonicity of $\mathcal{P}(\mathcal{Z})$ we conclude that $\mathsf{ret}_k(\sigma, m') = \mathsf{ret}_k(\sigma, m)$ for all $m \geq m'$. A similar application of the pigeonhole-principle shows that there is a $k \leq |\Sigma|^2 \cdot |Q|^4$ such that $\mathsf{ret}_k((\sigma, i)) = \mathsf{ret}_{k+1}((\sigma, i))$ for all σ and all $i \leq |\Sigma||Q|^2$ (or equivalently for all $i \in \mathbb{N}$). By induction on $k' \geq k$ we show that $\mathsf{ret}_{k'} = \mathsf{ret}_k$ because any subreturn that increases the height by $k + 1$ can be replaced by a subreturn that only increases the height by k. Thus, we obtain the following lemma.

GeneratePDA$(\mathcal{A}, A)$:
Input: 2-HOCA$^-$ $\mathcal{A} = (Q, q_0, \Delta)$ over Σ, matrix $A = (a_{\sigma,p,q})_{(\sigma,p,q)\in\Sigma\times Q^2}$ over $\mathbb{N}\cup\{\infty\}$
Output: 1-HOPA $\mathcal{A}'$ simulating $\mathcal{A}$

$k_0 := |\Sigma|^2 \cdot |Q|^4$; $h_0 := |\Sigma| \cdot |Q^2|$; $\Delta' := \emptyset$
foreach $\delta \in \Delta$:
  if $\delta == (q, (\sigma, \bot), \mathsf{stay}_{\mathsf{pop}}, p)$:
    foreach i **in** $\{0, \dots, h_0\}$: $\Delta' := \Delta' \cup \{((q,\sigma), \bot_i, \mathsf{pop}, (p,\sigma)), ((q,\sigma), , \bot_\infty, \mathsf{pop}, (p,\sigma))\}$
  elseif $\delta == (q, (\sigma, \bot), \mathsf{stay}_{\mathsf{push}_\bot}, p)$
    $\Delta' := \Delta' \cup \{((q,\sigma), \bot_\infty, \mathsf{push}_{\bot_\infty}, (p,\sigma))\} \cup \{((q,\sigma), \bot_{h_0}, \mathsf{push}_{\bot_\infty}, (p,\sigma))\}$
    foreach i **in** $\{0, \dots, h_0 - 1\}$: $\Delta' := \Delta' \cup \{((q,\sigma), \bot_i, \mathsf{push}_{\bot_{i+1}}, (p,\sigma))\}$
  elseif $\delta == (q, (\sigma, \bot), \mathsf{push}_{\tau,\mathsf{id}}, p)$
    foreach r **in** Q **such that** $a_{\tau,p,r} \neq \infty$:
      foreach i **in** $\{a_{\tau,p,r}, a_{\tau,p,r}+1, \dots, h_0\} \cup \{\infty\}$: $\Delta' := \Delta' \cup \{((q,\sigma), \bot_i, \mathsf{id}, (r,\sigma))$
$\mathcal{A}' := (Q \times \Sigma, (q_0, \bot), \Delta')$
return $\mathcal{A}'$

Fig. 1. 2-HOCA$^-$ to 1-HOPA Reduction Algorithm

Lemma 10. *For all $i \in \mathbb{N}$ and $\sigma \in \Sigma$, we have $\mathsf{ret}_\infty((\sigma, i)) = \mathsf{ret}_{|\Sigma|^2\cdot|Q|^4}((\sigma, i))$ and $\mathsf{lp}_\infty = \mathsf{lp}_{|\Sigma|^2|Q|^4+1}$.*

We now can prove that control-state reachability on 2-HOCA$^-$ is **P**-complete.

Proof (of Proposition 6). Since 2-HOCA$^-$ can trivially simulate pushdown automata, hardness follows from the analogous hardness result for pushdown automata. Containment in **P** uses the following ideas:

1. We assume that the input $(\mathcal{A}, q)$ satisfies that q is reachable in $\mathcal{A}$ iff $(q, (\bot, 0))$ is reachable and that $\mathcal{A}$ only uses instructions of the forms pop, $\mathsf{push}_{\sigma,\mathsf{id}}$, and stay_f. Given any 2-HOCA$^-$ $\mathcal{A}'$ and a state q, it is straightforward to construct (in polynomial time) a 2-HOCA$^-$ $\mathcal{A}$ that satisfies this condition such that q is reachable in $\mathcal{A}'$ iff it is reachable in $\mathcal{A}$.
2. Recall that $\mathsf{ret}_\infty(w) = \mathsf{ret}_{k_0}(w)$ for $k_0 = |\Sigma|^2 \cdot |Q|^4$ and for all $w \in \Sigma\times\mathbb{N}$. Set $h_0 = |\Sigma| \cdot |Q^2|$. We want to compute a table $(a_{\sigma,p,q})_{\sigma,p,q\in\Sigma\times Q^2}$ with values in $\{\infty, 0, 1, 2, \dots, h_0\}$ such that $a_{\sigma,p,q} = \min\{i \mid (p,q) \in \mathsf{ret}_{k_0}((\sigma,i))\}$ (where we set $\min\{\emptyset\} = \infty$). Due to Lemmas 9 and 10 such a table represents ret_∞ in the sense that $(p,q) \in \mathsf{ret}_\infty((\sigma,i))$ iff $i \geq a_{\sigma,p,q}$.
3. With the help of the table $(a_{\sigma,p,q})_{(\sigma,p,q)\in\Sigma\times Q^2}$ we compute in polynomial time a $\mathcal{P}$ automaton $\mathcal{A}_\infty$ which executes the same level 1 transitions as $\mathcal{A}$ and simulates loops of $\mathcal{A}$ in the following sense: if there is a loop of $\mathcal{A}$ starting in $(q,(\sigma,i))$ performing first a $\mathsf{push}_{\tau,\mathsf{id}}$ operation and then performing a return with final state p, we allow $\mathcal{A}'$ to perform an id-transition from $(q,(\sigma,i))$ to $(p,(\sigma,i))$. This new system basically keeps track of the height of the pushdown up to h_0 by using a pushdown alphabet $\{\bot_0, \dots, \bot_{h_0}, \bot_\infty\}$ where the topmost symbol of the pushdown is $\bot_i$ iff the height of the pushdown is i (where ∞ stands for values above h_0). After this change of pushdown alphabet, the additional id-transitions are easily computable from the table $(a_{\sigma,p,q})_{(\sigma,p,q)\in\Sigma\times Q^2}$. The resulting system has size $O(h_0^2 \cdot (|\mathcal{S}| + 1))$, i.e., is polynomial in the original system $\mathcal{A}$.
4. Using [1], check for reachability of q in the pushdown automaton $\mathcal{A}_\infty$.

```
ReachHOCA-(A, q_f):
Input: 2-HOCA⁻ A = (Q, q_0, Δ) over Σ, q_f ∈ Q
Output: whether q_f is reachable in A

1  k_0 := |Σ|^2 · |Q|^4;  h_0 :=|Σ| · |Q^2|;
2  foreach (σ, p, q) in Σ × Q^2: a_{σ,p,q} := ∞
3  for k = 1, 2, ..., k_0:
4     A_k := GeneratePDA(A, (a_{σ,p,q})_{(σ,p,q)∈Σ×Q^2})
5     foreach (r, (τ, ⊥), pop, q) in Δ and (σ, p) in Σ × Q:
6        for i = h_0, h_0 − 1, ..., 1, 0:
7        if ReachPDA(A_k, ((p, σ), i), (r, τ)): a'_{σ,p,q} := i
8     foreach (σ, p, q) in Σ × Q^2: a_{σ,p,q} :=a'_{σ,p,q}
9  A_∞ :=GeneratePDA(A, (a_{σ,p,q})_{(σ,p,q)∈Σ×Q^2})
10 if Reach(A_∞, ((q_0, ⊥), 0), (q_f, ⊥)): return true else return false
```

Fig. 2. Reachability on 2-HOCA$^-$ Algorithm 2

In fact, for step 2 we already use a variant of steps 3 and 4: we compute $\mathsf{ret}_\infty = \mathsf{ret}_{|\Sigma|^2|Q|^4}$ by induction starting with ret_0. If we remove all level 2 operations from $\mathcal{A}$ and store the topmost level 2 stack-symbol in the control state we obtain a pushdown automaton $\mathcal{B}$ such that $(q, q') \in \mathsf{ret}_0(\sigma, k)$ (w.r.t. $\mathcal{A}$) iff there is a transition $(p, (\sigma, \bot), \mathsf{pop}, q')$ of $\mathcal{A}$ and the control state (p, σ) is reachable from $((p, \sigma), k)$ in $\mathcal{B}$. Thus, the results of polynomially many reachability queries for $\mathcal{B}$ determine the table for ret_0. Similarly, we can use the table of ret_i to compute the table of ret_{i+1} as follows. A return extending the height of the pushdown by $i+1$ decomposes into parts that do not increase the height at all and parts that perform a $\mathsf{push}_{\tau,\mathsf{id}}$ followed by a return increasing the height by at most i. Using the table for ret_i we can easily enrich $\mathcal{B}$ by id-transitions that simulate such push operations followed by returns increasing the height by at most i. Again, determining whether $(q, q') \in \mathsf{ret}_{i+1}(\sigma, k)$ reduces to one reachability query on this enriched $\mathcal{B}$ for each pop-transition of $\mathcal{A}$.

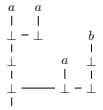
With these ideas in mind, it is straightforward to check that algorithm `ReachHOCA-` in Figure 2 (using algorithm `GeneratePDA` of Figure 1 as subroutine for step 3) solves the reachability problem for 2-HOCA$^-$ (of the form described in step 1) in polynomial time. In this algorithm, `ReachPDA` $(\mathcal{A}', c, q)$ refers to the classical polynomial time algorithm that determines whether in the (level 1) pushdown automaton $\mathcal{A}'$ state q is reachable when starting in configuration c; a transition $(q, (\sigma, \tau), f, p)$ refers to a transition from state q to state p applying operation f that is executable if the (level 2) test top_σ and the (level 1) test $test(\mathsf{top}_\tau)$ both succeed. □

3.2 Regular Reachability

In order to define regular sets of configurations, we recall the encoding of 2-HOPA configurations as trees from [12]. Let $p = (\sigma_1, v_1)(\sigma_2, v_2)\dots(\sigma_m, v_m) \in \mathcal{P}(\mathcal{Z})$. If $v_1 = 0$, we set $p_l = \emptyset$ and $p_r = (\sigma_2, v_2)\dots(\sigma_m, v_m)$. Otherwise, there is

a maximal $1 \leq j \leq m$ such that $v_1, \ldots, v_j \geq 1$ and we set $p_l = (\sigma_1, v_1 - 1) \ldots (\sigma_j, v_j - 1)$ and $p_r = (\sigma_{j+1}, v_{j+1}) \ldots (\sigma_m, v_m)$ if $j < m$ and $p_r = \emptyset$ if $j = m$. The *tree-encoding* E of p is given as follows:

$$\mathsf{E}(p) = \begin{cases} \emptyset & \text{if } p = \emptyset \\ \bot(\sigma_1, \mathsf{E}(p_r)) & \text{if } p = (\sigma_1, 0)p_r \\ \bot(\mathsf{E}(p_l), \mathsf{E}(p_r)) & \text{otherwise,} \end{cases}$$



where $\bot(t_1, t_2)$ is the tree with root $\bot$ whose left subtree is t_1 and whose right subtree is t_2. For a configuration $c = (q, p)$ we define $\mathsf{E}(c)$ to be the tree $q(\mathsf{E}(p), \emptyset)$. The picture beside the definition of E shows the encoding of the configuration $(q, (a, 2)(a, 2)(a, 0)(b, 1))$. Note that for each element (σ, i) of p, there is a path to a leaf l which is labelled by σ such that the path to l contains $i + 2$ left successors. Moreover, the inorder traversal of the tree induces an order of the leaves which corresponds to the left-to-right order of the elements of p. We call a set C of configurations *regular* if the set $\{\mathsf{E}(c) \mid c \in C\}$ is a regular set of trees.

E turns the reachability predicate on 2-HOCA$^-$ into a tree-automatic relation [12], i.e., for a given 2-HOCA$^-$ $\mathcal{A}$, there is a tree-automaton $\mathcal{T}_\mathcal{A}$ accepting the convolution of $\mathsf{E}(c_1)$ and $\mathsf{E}(c_2)$ for 2-HOCA$^-$ configurations c_1 and c_2 iff there is a run of $\mathcal{A}$ from c_1 to c_2. This allows to solve the regular backwards reachability problem as follows. On input a 2-HOCA$^-$ and a tree automaton $\mathcal{T}$ recognising a regular set C of configurations, we first compute the tree-automaton $\mathcal{T}_\mathcal{A}$. Then using a simple product construction of $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}$ and projection, we obtain an automaton $\mathcal{T}_{\mathsf{pre}}$ which accepts $\mathsf{pre}^*(C) = \{\mathsf{E}(c) \mid \exists c' \in C$ and a run from c to $c'\}$. The key issue for the complexity of this construction is the computation of $\mathcal{T}_\mathcal{A}$ from $\mathcal{A}$. The explicit construction of $\mathcal{T}_\mathcal{A}$ in [12] involves an exponential blow-up. In this construction the blow-up is only caused by a part of $\mathcal{T}_\mathcal{A}$ that computes $\mathsf{ret}_\infty(\sigma, m)$ for each $\sigma \in \Sigma$ on input a path whose labels form the word $\bot^m$. Thus, we can exhibit the following consequence.

Corollary 11 ([12]). *Given a* 2-HOCA$^-$ $\mathcal{A}$ *with state set* Q*, we can compute the tree automaton* $\mathcal{T}_\mathcal{A}$ *in* $\mathbf{P}$*, if we can compute from* $\mathcal{A}$ *in* $\mathbf{P}$ *a deterministic word automaton* $\mathcal{T}'$ *with state set* $Q' \subseteq \prod_{\sigma \in \Sigma}(2^{Q \times Q})^2$ *such that for all* $m \in \mathbb{N}$ *the state of* $\mathcal{T}'$ *on input* $\bot^m$ *is* $(\mathsf{ret}_\infty(\sigma, m), \mathsf{lp}_\infty(\sigma, m))_{\sigma \in \Sigma}$*.*

Thus, the following lemma completes the proof of Theorem 5.

Lemma 12. *Let* $\mathcal{A}$ *be a* 2-HOCA$^-$ *with state set* Q*. We can compute in polynomial time a deterministic finite word automaton* $\mathcal{A}'$ *with state set* Q' *of size at most* $2 \cdot (|\Sigma| \cdot |Q|^2 + 1)$ *such that* $\mathcal{A}'$ *is in state* $(\mathsf{ret}_\infty((\sigma, n)), \mathsf{lp}_\infty((\sigma, n)))_{\sigma \in \Sigma}$ *after reading* $\bot^n$ *for every* $n \in \mathbb{N}$*.*

Proof. Let $n_0 = 2 \cdot |\Sigma| \cdot |Q|^2$. Recall algorithm `ReachHOCA-` of Figure 2. In this polynomial time algorithm we computed a matrix $A = (a_{\sigma,p,q})_{(\sigma,p,q) \in \Sigma \times Q^2}$ representing ret_∞ and a pushdown automaton $\mathcal{A}_\infty$ (of level 1) simulating $\mathcal{A}$ in the sense that $\mathcal{A}_\infty$ reaches a configuration $((q, \sigma)p)$ for a pushdown p of height n if and only if $\mathcal{A}$ reaches $(q, (\sigma, n))$. It is sufficient to describe a polynomial time algorithm that computes $M_i = (\mathsf{ret}_\infty((\sigma, n)), \mathsf{lp}_\infty((\sigma, n)))_{\sigma \in \Sigma}$ for all $n \leq n_0$. $\mathcal{A}'$

is then the automaton with state set $\{M_i \mid i \leq n_0\}$, transitions from M_i to M_{i+1} for each $i < n_0$ and a transition from M_{n_0} to M_{n_0}. The correctness of this construction follows from Lemma 9.

Let us now describe how to compute M_i in polynomial time. Since $\mathcal{A}_\infty$ simulates $\mathcal{A}$ correctly, there is a loop from $(q,(\sigma,i))$ to $(q',(\sigma,i))$ of $\mathcal{A}$ if and only if there is a run of $\mathcal{A}_\infty$ from $((q,\sigma),p_i)$ to $((q',\sigma),p_i)$ for $p_i = \bot_0\bot_1\ldots\bot_i$ (where we identify $\bot_j$ with $\bot_\infty$ for all $j > h_0$). Thus, we can compute the loop part of M_i by n_0 many calls to an algorithm for reachability on pushdown systems. Note that $(p,q) \in \mathsf{ret}_\infty((\sigma,i))$ with respect to $\mathcal{A}$ if there is a state r and some $\tau \in \Sigma$ such that (r,τ,pop,q) is a transition of $\mathcal{A}$ and (r,τ) is reachable in $\mathcal{A}_\infty$ from $((q,\sigma),i)$. Thus, with a loop over all transitions of $\mathcal{A}$ we reduce the computation of the returns component of M_i to polynomially many control state reachability problems on a pushdown system. □

4 Reachability for k-HOCA$^-$ and k-HOCA$^+$

Using slight adaptations of Engelfriet's seminal paper [7], we can lift the result on reachability for 2-HOCA$^-$ to reachability for k-HOCA$^-$ (cf. [10]).

Theorem 13. *For $k \geq 2$, the control state reachability problem for* k-HOCA$^-$ *is complete for* $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$. *For $k \geq 1$, the alternating control state reachability problem for* k-HOCA$^-$ *is complete for* $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k-1}(n^d))$.

Hardness follows from the hardness of control state reachability for $(\mathsf{k}-1)$-HOPA [7] and the trivial fact that the storage type $\mathcal{P}^{k-1}$ of $(\mathsf{k}-1)$-HOPA can be trivially simulated by the storage type $\mathcal{P}^{k-1}(\mathcal{Z})$ of k-HOCA$^-$. Containment for the first claim is proved by induction on k (the base case $k = 2$ has been proved in the previous section). For $k \geq 3$, we use Lemma 7.11, Theorems 2.2 and 2.4 from [7] and reduce reachability of k-HOCA$^-$ to reachability on (exponentially bigger) $(\mathsf{k}-1)$-HOCA$^-$. For the second claim, we adapt Engelfriet's Lemma 7.11 to a version for the setting of alternating automata (instead of nondeterministic automata) and use his Theorems 2.2. and 2.4 in order to show equivalence (up to logspace reductions) of alternating reachability for $(\mathsf{k}-1)$-HOCA$^-$ and reachability for k-HOCA$^-$.

We can also reduce reachability for k-HOCA$^+$ to reachability for $(k-1)$-fold exponentially bigger 1-HOCA$^+$. Completeness for $\mathbf{NSPACE}(\log(n))$ of reachability for 1-HOCA$^+$ (cf. [8]) yields the upper bounds for reachability for k-HOCA$^+$. The corresponding lower bounds follow by applications of Engelfriet's theorems and an adaptation of the **PSPACE**-hardness proof for emptiness of alternating finite automata by Jancar and Sawa [11].

Theorem 14. *For $k \geq 2$, (alternating) control state reachability for* k-HOCA$^+$ *is complete for (*$\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_{k-1}(n^d))$ $\mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_{k-2}(n^d))$.

5 Back to HOPS: Applications to Languages

Engelfriet [7] also discovered a close connection between the complexity of the control state reachability problem for a class of automata and the class of languages recognised by this class. We restate a slight extension (cf. [10]) of these results and use them to confirm Slaat's conjecture from [18].

Proposition 15. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be storage types and C_1, C_2 complexity classes such that $C_1 \subsetneq C_2$. If control state reachability for nondeterministic $\mathcal{S}_i$ automata is complete for C_i, then there is a deterministic $\mathcal{S}_2$ automaton accepting some language L such that no nondeterministic $\mathcal{S}_1$-automaton accepts L.*

In fact, Engelfriet's proof can be used to derive a separating language. For a storage type $\mathcal{S} = (X, T, F, x_0)$, we define the *language of valid storage sequences* $\mathsf{VAL}(\mathcal{S})$ as follows. For each test $t \in T$ and $r \in \{true, false\}$ we set $t_r := \mathsf{id}\restriction_{\{x \in X \mid t(x) = r\}}$ and set $\Sigma = F \cup \{t_r \mid t \in T, r \in \{true, false\}\}$. For $s \in \Sigma^*$ such that $s = a_1 \dots a_n$, and $x \in X$ we write $s(x)$ for $a_n(a_{n-1}(\dots a_1(x)\dots))$. We define $\mathsf{VAL}(\mathcal{S}) = \{s \in \Sigma^* \mid s(x_0) \text{ is defined}\}$.

If the previous proposition separates the languages of $\mathcal{S}_2$ automata from those of $\mathcal{S}_1$ automata, then it follows from the proof that $\mathsf{VAL}(\mathcal{S}_2)$ is not accepted by any $\mathcal{S}_1$ automato (cf. [10]).

Corollary 16. *If* $\mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_k(n^d)) \subsetneq \mathbf{DSPACE}(\bigcup_{d\in\mathbb{N}} \exp_k(n^d)) \subsetneq \mathbf{DTIME}(\bigcup_{d\in\mathbb{N}} \exp_{k+1}(n^d))$, *then* $L((\mathsf{k}-1)\text{-}\mathsf{HOPA}) \subsetneq L(\mathsf{k}\text{-}\mathsf{HOCA}^-) \subsetneq L(\mathsf{k}\text{-}\mathsf{HOCA}^+) \subsetneq L(\mathsf{k}\text{-}\mathsf{HOPA})$.

The crucial underlying construction detail of the proof of Proposition 15 is quite hidden within the details of Engelfriet's technical and long paper. Its usefulness in other contexts — e.g., for higher-order pushdowns or counters — has been overseen so far. Here we give another application to collapsible pushdown automata: reachability for collapsible pushdown automata of level k is $\mathbf{DSPACE}(\exp_{k-1}(n))$-complete (cf. [3]). Thus, Proposition 15 trivially shows that the language of valid level $(k+1)$ collapsible pushdown storage sequences separates the collapsible pushdown languages of level $k+1$ from those of level k. This answers a question asked by several experts in this field (cf. [17,14]). In fact, [17] uses a long and technical construction to prove the weaker result that there are more level $2k$ collapsible pushdown languages than level k collapsible pushdown languages. From Proposition 15 one also easily derives the level-by-level strictness of the collapsible pushdown tree hierarchy and the collapsible pushdown graph hierarchy (cf. [13,14]).

6 Future Work

Our result on regular reachability gives hope that also complexity results on model checking for logics like the μ-calculus extend from pushdown automata to 2-HOCA. 2-HOCA^- probably is a generalisation of pushdown automata that retains the good complexity results for basic algorithmic questions. It is also

interesting whether the result on regular reachability extends to the different notions of regularity for k-HOCA mentioned in the introduction. HOCA also can be seen as a new formalism in the context of register machines as currently used in the verification of concurrent systems. HOCA allow to store pushdown-like structures of register values and positive results on model checking HOCA could be transferred to verification questions in this concurrent setting.

References

1. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
2. Bouajjani, A., Meyer, A.: Symbolic reachability analysis of higher-order context-free processes. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 135–147. Springer, Heidelberg (2004)
3. Broadbent, C., Carayol, A., Hague, M., Serre, O.: A saturation method for collapsible pushdown systems. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 165–176. Springer, Heidelberg (2012)
4. Carayol, A.: Regular sets of higher-order pushdown stacks. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 168–179. Springer, Heidelberg (2005)
5. Carayol, A., Wöhrle, S.: The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 112–123. Springer, Heidelberg (2003)
6. Caucal, D.: On infinite terms having a decidable monadic theory. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 165–176. Springer, Heidelberg (2002)
7. Engelfriet, J.: Iterated stack automata and complexity classes. Inf. Comput. 95(1), 21–75 (1991)
8. Göller, S.: Reachability on prefix-recognizable graphs. Inf. Process. Lett. 108(2), 71–74 (2008)
9. Hague, M., Ong, C.-H.L.: Symbolic backwards-reachability analysis for higher-order pushdown systems. LMCS 4(4) (2008)
10. Heußner, A., Kartzow, A.: Reachability in higher-order-counters. CoRR, arxiv:1306.1069 (2013), http://arxiv.org/abs/1306.1069
11. Jancar, P., Sawa, Z.: A note on emptiness for alternating finite automata with a one-letter alphabet. Inf. Process. Lett. 104(5), 164–167 (2007)
12. Kartzow, A.: Collapsible pushdown graphs of level 2 are tree-automatic. Logical Methods in Computer Science 9(1) (2013)
13. Kartzow, A., Parys, P.: Strictness of the collapsible pushdown hierarchy. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 566–577. Springer, Heidelberg (2012)
14. Kobayashi, N.: Pumping by typing. To appear in Proc. LICS (2013)
15. Maslov, A.N.: The hierarchy of indexed languages of an arbitrary level. Sov. Math., Dokl. 15, 1170–1174 (1974)
16. Maslov, A.N.: Multilevel stack automata. Problems of Information Transmission 12, 38–43 (1976)
17. Parys, P.: Variants of collapsible pushdown systems. In: Proc. of CSL 2012. LIPIcs, vol. 16, pp. 500–515 (2012)
18. Slaats, M.: Infinite regular games in the higher-order pushdown and the parametrized setting. PhD thesis, RWTH Aachen (2012)