Ordered Tree-Pushdown Systems

Lorenzo Clemente, Pawel Parys, Sylvain Salvati, Igor Walukiewicz

▶ **To cite this version:**

Lorenzo Clemente, Pawel Parys, Sylvain Salvati, Igor Walukiewicz. Ordered Tree-Pushdown Systems. FSTTCS 2015, Dec 2015, Bangalore, India. hal-01145598

HAL Id: hal-01145598
https://hal.science/hal-01145598

Submitted on 24 Apr 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ordered Tree-Pushdown Systems

Lorenzo Clemente[1], Sylvain Salvati[2], and Igor Walukiewicz[2]

[1] University of Warsaw, Poland
[2] CNRS, Université de Bordeaux, INRIA, France

Abstract. We define a new class of pushdown systems where the pushdown is a tree instead of a word. We allow a limited form of lookahead on the pushdown conforming to a certain ordering restriction, and we show that the resulting class enjoys a decidable reachability problem. This follows from a preservation of recognizability result for the backward reachability relation of such systems. As an application, we show that our simple model can encode several formalisms generalizing pushdown systems, such as ordered multi-pushdown systems, annotated higher-order pushdown systems, the Krivine machine, and ordered annotated multi-pushdown systems. In each case, our procedure yields tight complexity.

1 Introduction

Context. Modeling complex systems requires to strike the right balance between the accuracy of the model, and the complexity of its analysis. A particularly successful example is given by *pushdown systems*, which are a popular class of infinite-state systems arising in diverse contexts, such as language processing, data-flow analysis, security, computational biology, and program verification. Many interesting analyses reduce to checking reachability in pushdown systems, which can be decided in PTIME using, e.g., the popular *saturation technique* [**?**,**?**] (cf. also the recent survey [**?**]). Pushdown systems have been generalized in several directions. In [**?**] *tree-pushdown systems* are introduced, where the pushdown is a tree instead of a linear word. It is observed that, unlike ordinary pushdown systems, non-destructive lookahead on the pushdown cannot be introduced without leading to undecidability. It is proved that, when seen as automata, the pushdown can be linearized without changing the expressiveness of the model w.r.t. the class of recognized tree languages.

A seemingly unrelated generalization is *ordered multi-pushdown systems* [**?**,**?**], where several linear pushdowns are available instead of just one. Since already two unrestricted linear pushdowns can simulate a Turing machine, an ordering restriction is put on popping transitions, requiring that all pushdowns smaller than the popped one are empty. Reachability in this model is 2-EXPTIMEc [**?**].

Higher-order pushdown systems generalize ordinary pushdown systems by allowing pushdowns to be nested inside other pushdowns [**?**,**?**]. *Collapsible pushdown systems* [**?**,**?**] additionally enrich pushdown symbols with *collapse links* to inner sub-pushdowns. This allows the automaton to push a new symbol and to save, at the same time, the current context in which the symbol is pushed,

and to later return to this context via a collapse operation. *Annotated pushdown systems* [**?**] provide a simplification of collapsible pushdown systems by replacing collapse links with arbitrary pushdown annotations[3]. The *Krivine machine* [**?**] is a related model which evaluates terms in simply-typed λY-calculus. Reachability in all these models is $(n-1)$-EXPTIMEc [**?**,**?**], and one exponential higher in the presence of alternation. The more general *ordered annotated multi-pushdown systems* [**?**] have several annotated pushdown systems under an ordering restriction similar to [**?**] in the first-order case. They subsume both ordered multi-pushdown systems and annotated pushdown systems. The saturation method (cf. [**?**]) has been adapted to most of these models, and it is the basis of the prominent MOPED tool [**?**] for the analysis of pushdown systems, as well as the C-SHORe model-checker for annotated pushdown systems [**?**].

Contributions. Motivated by a unification of the results above, we introduce *ordered tree-pushdown systems*. This is a natural model extending tree-pushdown systems by allowing a limited form of partially destructive lookahead on the pushdown. We introduce an order between pushdown symbols, and we require that, whenever a sub-pushdown is read, all sub-pushdowns of smaller order must be discarded. The obtained model is expressive enough to simulate all systems mentioned above, and still not Turing-powerful thanks to the ordering condition. Our contributions are: i) A general preservation of recognizability result for ordered tree-pushdown systems. ii) Direct encoding of several popular extensions of pushdown systems, such as ordered multi-pushdown systems, annotated pushdown systems, the Krivine machine, and ordered annotated multi-pushdown systems. iii) A conceptually simple saturation algorithm working on finite tree automata representing sets of configurations, subsuming and unifying previous constructions. iv) A complete complexity characterization of reachability in ordered tree-pushdown systems and natural subclasses thereof. v) A short and simple correctness proof, simplifying substantially the previous literature.

Related work. Apart from the results mentioned above, we should note a saturation method for recursive program schemes [**?**]. Since schemes and λY-calculus are equi-expressive, our method can also be adapted to schemes. Concerning multi-pushdown systems, apart from the restriction based on the pushdown order there are other restrictions making their analysis decidable. Indeed, in [**?**] decidability is proved for annotated multi-pushdowns with phase-bounded and scope-bounded restrictions. It is not clear how to treat these restrictions in the simple framework presented here. For standard multi-pushdown systems split-width has been proposed as a unifying restriction [**?**]. It is not known how to extend this method to annotated multi-pushdown systems, or to our tree-pushdown systems.

[3] Collapsible and annotated systems generate the same configuration graphs when started from the same initial configuration, since new annotations can only be created to sub-pushdowns of the current pushdown. However, annotated pushdown systems have a richer backward reachability set which includes non-constructible pushdowns.

Outline. In Sec. 2 we introduce common notions. In Sec. 3 we define our model and we present our saturation-based algorithm to decide reachability. In Sec. 4 we show that ordered systems can optimally encode several popular formalisms. In Sec. 5 we conclude with some perspectives on open problems.

2 Preliminaries

We will work with rewriting systems on ranked trees, and with alternating tree automata. The novelty is that every letter of the ranked alphabet will have an order. A tree will have the order determined by the letter in the root. The order will be used later to constrain rewriting rules.

An *alternating transition system* is a tuple $\mathcal{S} = \langle \mathcal{C}, \rightarrow \rangle$, where $\mathcal{C}$ is the set of configurations and $\rightarrow \subseteq \mathcal{C} \times 2^{\mathcal{C}}$ is the alternating transition relation. We lift the relation $\rightarrow$ to sets of configurations $A, B \subseteq \mathcal{C}$ by defining $A \rightarrow B$ iff, for every $c \in A$, there exists $C \subseteq B$ s.t. $c \rightarrow C$, and we denote by $\rightarrow^*$ its reflexive and transitive closure. The set of *predecessors* of a set $C \subseteq \mathcal{C}$ of configurations is $\mathrm{Pre}^*(C) = \{c \mid \{c\} \rightarrow^* C\}$.

Ordered trees. Let $\mathbb{N}$ be the set of non-negative integers, and let $\mathbb{N}_{>0}$ be the set of strictly positive integers. A *node* is an element $u \in \mathbb{N}_{>0}^*$. A node u is a *predecessor* of a node v iff u is a prefix of v, i.e., there exists $w \in \mathbb{N}_{>0}^*$ s.t. $v = uw$, and similarly for the notion of *successor*. A *tree domain* is a non-empty prefix-closed set of nodes $D \subseteq \mathbb{N}_{>0}^*$ s.t., if $u \cdot (i+1) \in D$, then $u \cdot i \in D$ for every $i \in \mathbb{N}_{>0}$. A *leaf* is a node u in D without successors. A *ranked alphabet* is a pair (Σ, rank) of a set of symbols Σ together with a ranking function $\mathsf{rank} : \Sigma \rightarrow \mathbb{N}$. An *ordered alphabet* is a triple $(\Sigma, \mathsf{rank}, \mathsf{ord})$ where (Σ, rank) is a ranked alphabet and $\mathsf{ord} : \Sigma \rightarrow \mathbb{N}_{>0}$. For a ranked/ordered alphabet $(\Sigma, \mathsf{rank}, \mathsf{ord})$ and a finite tree domain D, a *finite Σ-tree* (or just *tree*) is a function $t : D \rightarrow \Sigma$ that assigns to each node u in D with n successors a label $a = t(u)$ in Σ s.t. $\mathsf{rank}(a) = n$. The *size* of t is $|t| := |D|$. For a Σ-tree $t : D \rightarrow \Sigma$ and label $a \in \Sigma$, let $t^{-1}(a) = \{u \in D \mid t(u) = a\}$ be the set of nodes labelled with a. We denote by $\mathcal{T}(\Sigma)$ the set of Σ-trees. The *order* of a tree t is $\mathsf{ord}(t) := \mathsf{ord}(t(\varepsilon))$.

Rewriting. For each order $n \in \mathbb{N}$, let $\mathcal{V}_n$ be a countably infinite set of variables s.t. $\mathcal{V}_0, \mathcal{V}_1, \ldots$ are pairwise disjoint, and let $\mathcal{V} = \bigcup_n \mathcal{V}_n$. We consider the extended ordered alphabet $(\Sigma \cup \mathcal{V}, \mathsf{rank}, \mathsf{ord})$ where a variable $\mathsf{x} \in \mathcal{V}_n$ has $\mathsf{rank}(x) = 0$ and $\mathsf{ord}(\mathsf{x}) = n$. Let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of $(\Sigma \cup \mathcal{V})$-trees. Fix a $(\Sigma \cup \mathcal{V})$-tree t, and let $\mathcal{V}(t)$ be the set of variables appearing in it. t is *linear* if each variable in $\mathcal{V}(t)$ appears exactly once in t. For a $(\Sigma \cup \mathcal{V})$-tree u, t is *u-ground* if $\mathcal{V}(t) \cap \mathcal{V}(u) = \varnothing$. A tree $t = a(t_1, \ldots, t_n)$ is *u-shallow* if each t_i is either u-ground, or just a variable (possibly appearing in u). A *substitution* is a finite mapping $\sigma : \mathcal{V} \rightarrow \mathcal{T}(\Sigma \cup \mathcal{V})$ s.t. $\mathsf{ord}(\sigma(\mathsf{x})) = \mathsf{ord}(\mathsf{x})$. Given a $(\Sigma \cup \mathcal{V})$-tree t and a substitution σ, $t\sigma$ is the $(\Sigma \cup \mathcal{V})$-tree obtained by replacing each variable x in t in the domain of σ with $\sigma(\mathsf{x})$. A *rewrite rule* over Σ is a pair $l \rightarrow r$ of $(\Sigma \cup \mathcal{V})$-trees l and r. A rewrite rule $l \rightarrow r$ is *left-linear* if l is linear, and it is *shallow* if l is r-shallow.

Alternating tree automata. An *alternating tree automaton* (or just *tree automaton*) is a tuple $\mathcal{B} = \langle \Sigma, Q, \Delta \rangle$ where Σ is a finite ranked alphabet, Q is a finite set of states, and $\Delta \subseteq Q \times \Sigma \times (2^Q)^*$ is a set of alternating transitions of the form $p \xrightarrow{a} P_1 \cdots P_n$, with a of rank n. We say that $\mathcal{B}$ is *non-deterministic* if, for every transition as above, all P_j's are just singletons, and we omit the braces in this case. An automaton is *ordered* if, for every state p and symbols a, b s.t. $p \xrightarrow{a} \cdots$ and $p \xrightarrow{b} \cdots$, we have $\mathsf{ord}(a) = \mathsf{ord}(b)$. We assume w.l.o.g. that automata are ordered, and we denote by $\mathsf{ord}(p)$ the order of state p. The transition relation is extended to a set of states $P \subseteq Q$ by defining $P \xrightarrow{a} P_1 \cdots P_n$ iff, for every $p \in P$, there exists a transition $p \xrightarrow{a} P_1^p \cdots P_n^p$, and $P_j = \bigcup_{p \in P} P_j^p$ for every $1 \leqslant j \leqslant n$. It will be useful later in the definition of the saturation procedure to define run trees not just on ground trees, but also on trees possibly containing variables. A variable of order k is treated like a leaf symbol which is accepted by all states of the same order. Let $P \subseteq Q$ be a set of states, and let $t : D \to (\Sigma \cup \mathcal{V})$ be an input tree. A *run tree from P on t* is a 2^Q-tree[4] $s : D \to 2^Q$ over the same tree domain D s.t. $s(\varepsilon) = P$, and: i) if $t(u) = a$ is not a variable and of rank n, then $s(u) \xrightarrow{a} s(u \cdot 1) \cdots s(u \cdot n)$, and ii if $t(u) = \mathsf{x}$ then $\forall p \in s(u), \mathsf{ord}(p) = \mathsf{ord}(\mathsf{x})$. The *language* recognized by a set of states $P \subseteq Q$, denoted by $\mathcal{L}(P)$, is the set of Σ-trees t s.t. there exists a run tree from P on t.

3 Ordered tree-pushdown systems

We introduce a generalization pushdown systems, where the pushdown is a tree instead of a linear word. An *alternating ordered tree-pushdown system* of order $n \in \mathbb{N}_{>0}$ is a tuple $\mathcal{S} = \langle n, \Sigma, P, \mathcal{R} \rangle$ where Σ is an ordered alphabet containing symbols of order at most n, P is a finite set of *control locations*, and $\mathcal{R}$ is a set of rules of the form $p, l \to S, r$ s.t. $p \in P$ and $S \subseteq P$. Moreover, $l \to r$ is a left-linear rewrite rule over Σ with $l := a(u_1, \ldots, u_m)$ such that at most one $u_{\hat{\imath}}$ is neither r-ground nor a variable, and if such an $u_{\hat{\imath}}$ exists then

(1) $u_{\hat{\imath}} = b(v_1, \ldots, v_n)$ with $b \in \Sigma$ and all v_j's either r-ground or a variable, and
(2) for every $j \in \{1, \ldots, m\}$, $j \neq \hat{\imath}$: if $\mathsf{ord}(u_j) \leqslant \mathsf{ord}(b)$ then u_j is r-ground.

We refer to (2) as *the ordering condition*. A rule is called *deep* if $u_{\hat{\imath}}$ exists, otherwise it is *shallow*; let $\mathcal{R}_d$ be the set of deep rules. For example, $a(\mathsf{x}, \mathsf{y}) \to c(a(\mathsf{x}, \mathsf{y}), \mathsf{x})$ is shallow, but $a(b(\mathsf{x}), \mathsf{y}) \to c(\mathsf{x}, \mathsf{y})$ is deep; here necessarily $\mathsf{ord}(\mathsf{y}) > \mathsf{ord}(b)$. In Sec. 4 we provide more examples of such rewrite rules by encoding many popular formalisms. Note that r can be non-linear, thus sub-trees can be duplicated. The *size* of $\mathcal{S}$ is $|\mathcal{S}| := |\Sigma| + |P| + |\mathcal{R}|$, where $|\mathcal{R}| := \sum_{p, l \to S, r \in \mathcal{R}} (1 + |l| + |S| + |r|)$. We say that $\mathcal{S}$ is *non-deterministic* if S is a singleton for every rule $p, l \to S, r \in \mathcal{R}$, and *shallow* if every rule is shallow. Rewrite rules induce an alternating transition system $\langle \mathcal{C}, \to_{\mathcal{S}} \rangle$, where the set of configurations $\mathcal{C}$ consists of pairs (p, t) with

[4] Strictly speaking 2^Q does not have a rank/order. It is easy to duplicate each subset at every rank/order to obtain an ordered alphabet, which we avoid for simplicity.

$p \in P$ and $t \in \mathcal{T}(\Sigma)$, and, for every configuration (p,t), $S \subseteq P$, and tree u, $(p,t) \rightarrow_{\mathcal{S}} S \times \{u\}$ if, and only if, there exists a rule $(p,l) \rightarrow (S,r) \in \mathcal{R}$ and a substitution σ s.t. $t = l\sigma$ and $u = r\sigma$.

Let $\mathcal{A} = \langle \Sigma, Q, \Delta \rangle$ be a tree automaton s.t. $P \subseteq Q$. The *language of configurations* recognized by $\mathcal{A}$ from P is $\mathcal{L}(\mathcal{A}, P) := \{(p,t) \in \mathcal{C} \mid p \in P \text{ and } t \in \mathcal{L}(p)\}$. Given an initial configuration $(p_0, t_0) \in \mathcal{C}$ and a tree automaton $\mathcal{A}$ recognizing a regular set of target configurations $\mathcal{L}(\mathcal{A}, P) \subseteq \mathcal{C}$, the *reachability problem* for $\mathcal{S}$ amounts to determine whether $(p_0, t_0) \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.

3.1 Reachability analysis

We present a saturation-based procedure to decide reachability in ordered tree-pushdown systems. This follows from a general preservation of recognizability result for the backward reachability relation.

Theorem 1 (Preservation of recognizability). *Let $\mathcal{S}$ be an order-n tree-pushdown system and let C be regular set of configurations. Then, $\mathrm{Pre}^*(C)$ is effectively regular, and an automaton recognizing it can be built in n-fold exponential time.*

Let $\mathcal{S} = \langle n, \Sigma, P, \mathcal{R} \rangle$ be a tree-pushdown system. The target set C is given as a tree automaton $\mathcal{A}$ s.t. $\mathcal{L}(\mathcal{A}, P) = C$. We construct a tree automaton $\mathcal{B} = \langle \Sigma, Q', \Delta' \rangle$ recognizing $\mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$. Q' is obtained by adding states to Q, and Δ' by adding transitions to Δ according to a saturation procedure. For every rule $p, l \rightarrow S, r \in \mathcal{R}$ and for every v which is a r-ground subtree of l we create a new state p^v of order $\mathsf{ord}(v)$ recognizing all ground trees that can be obtained by replacing variables in v by arbitrary trees, i.e., $\mathcal{L}(p^v) = \{v\sigma \mid \sigma : \mathcal{V} \rightarrow \mathcal{T}(\Sigma) \cdot v\sigma \text{ is ground}\}$. Let Q_0 be the set of such p^v's and let Δ_0 the required transitions. Notice that $|Q_0| \leqslant |\mathcal{R}|$.

In order to deal with with deep rules we add new states in the following stratified way: Let $Q'_n = Q \cup Q_0$, and, for every order $1 \leqslant i < n$, let $Q'_i = Q'_{i+1} \cup \bigcup_{g \in \mathcal{R}_d} \{g\} \times \left(2^{Q'_{i+1}}\right)^{\mathsf{rank}(g)-1}$, where $\mathsf{rank}(g = p, l \rightarrow S, r)$ is the rank of the root symbol in l. We define the set of states in $\mathcal{B}$ to be $Q' := Q'_1$. We add transitions to $\mathcal{B}$ in an iterative process until no more transitions can be added. During the saturation process, we maintain the following invariant: *For $1 \leqslant k < n$, states in $Q'_k \backslash Q'_{k+1}$ recognize only trees of order k.* Therefore, $\mathcal{B}$ is also an ordered tree automaton. Formally, Δ' is the least set containing $\Delta \cup \Delta_0$ and closed under adding transitions according to the following condition: For every rule $g = p, l \rightarrow S, r \in \mathcal{R}$ with $l = a(u_1, \ldots, u_m)$, and for every run tree t from S on r in $\mathcal{B}$, we add a transition

$$p \xrightarrow{a} P_1 \cdots P_m \qquad (\Delta'\text{-shallow})$$

s.t., for every $1 \leqslant i \leqslant m$, i) if $u_i = \mathsf{x}$ is a variable, then $P_i := \bigcup t(r^{-1}(\mathsf{x}))$, ii) if u_i is not a variable but is r-ground, then $P_i := \{p^{u_i}\}$, and iii) if g is a deep rule, then for the unique $u_{\hat{\imath}} = b(v_1, \ldots, v_n)$ we let $P_{\hat{\imath}} := \{p_{\hat{\imath}}\}$ where $p_{\hat{\imath}} :=$

$(g, P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_m)$. Thanks to the ordering condition, if $\mathsf{ord}(b) = k$, then $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_m \subseteq Q'_{k+1}$, and thus p_i is indeed a state of order k in $Q'_k \backslash Q'_{k+1}$. Then, we add the transition

$$(g, P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_m) \xrightarrow{b} S_1 \cdots S_n \qquad (\Delta'\text{-deep})$$

s.t., for every $1 \leqslant j \leqslant n$, iii.1) if $v_j = \mathsf{x}$ is a variable, then $S_j := \bigcup t(r^{-1}(\mathsf{x}))$, and iii.2) if v_j is not a variable (and therefore necessarily r-ground), then $S_j := \{p^{v_j}\}$.

Lemma 1. *For $\mathcal{A}$ and $\mathcal{B}$ be as above, $\mathcal{L}(\mathcal{B}, P) = \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.*

The correctness proof, even though short, is presented in the appendix. The right-to-left direction is by straightforward induction on the number of rewrite steps to reach $\mathcal{L}(\mathcal{A}, P)$. The left-to-right direction is more subtle, but with an appropriate invariant of the saturation process it also follows by a direct inspection.

3.2 Complexity

For an initial configuration $(p_0, t_0) \in \mathcal{C}$ and an automaton $\mathcal{A}$ recognizing a regular set of target configurations $\mathcal{L}(\mathcal{A}, P)$, we can decide the reachability problem by constructing $\mathcal{B}$ as in the previous section, and then testing $(p_0, t_0) \in \mathcal{L}(\mathcal{B}, P)$. In the following, let $m > 1$ be the maximal rank of any symbol in Σ. Using the notation from the previous subsection we have that $|Q'_n| \leqslant |Q| + |\mathcal{R}|$, and, for every $1 \leqslant i < n$, $|Q'_i| \leqslant |Q'_{i+1}| + |\mathcal{R}| \cdot 2^{(m-1)\cdot|Q'_{i+1}|} \leqslant O\left(|\mathcal{R}| \cdot 2^{(m-1)\cdot|Q'_{i+1}|}\right)$, and thus $|Q'| \leqslant \mathsf{exp}_{n-1}(O((m-1)\cdot(|Q| + |\mathcal{R}|)))$, where $\mathsf{exp}_0(x) = x$ and, for $i \geqslant 0$, $\mathsf{exp}_{i+1}(x) = 2^{\mathsf{exp}_i(x)}$. Similarly, we derive $|\Delta'| \leqslant \mathsf{exp}_n(O((m-1)\cdot(|Q| + |\mathcal{R}|)))$.

Theorem 2. *Reachability in order n tree-pushdown systems is n-EXPTIMEc.*

We identify three subclasses of ordered tree-pushdown systems, for which the reachability problem is of decreasing complexity. First we consider the class of *linear non-deterministic systems*. Assume that $\mathcal{A}$ is non-deterministic. When $\mathcal{S}$ is linear, i.e., variables in the r.h.s. of rules in $\mathcal{R}$ appear exactly once, all P_i's in (Δ'-shallow) are singletons, and thus also $\mathcal{B}$ is also non-deterministic. Consequently, the construction of Q'_i can be simplified to avoid the exponential blow-up at each order: $Q'_i = Q'_{i+1} \cup \bigcup_{g \in \mathcal{R}_d} \{g\} \times \left(Q'_{i+1}\right)^{\mathsf{rank}(g)-1}$, yielding $|Q'| \leqslant O((|Q| + |\mathcal{R}|)^{(m-1)^n})$ and $|\Delta'| \leqslant O(|\mathcal{R}| \cdot |Q'|^m)$. Therefore, $\mathcal{B}$ is doubly exponential in n.

Theorem 3. *Reachability in linear non-deterministic ordered tree-pushdown systems is 2-EXPTIMEc.*

If the system is *shallow*, then we do not need to add states recursively ($Q' := Q \cup Q_0$), and we thus avoid the multiple exponential blow-up. The same happens if the system is *unary*, i.e., $m = 1$.

Theorem 4. *Reachability in shallow/unary ordered tree-pushdown systems is EXPTIMEc, and in PTIME for the non-deterministic variant.*

All lower-bounds follow from the reductions presented in Sec. 4.

4 Applications

In this section, we give examples of systems that can be encoded as ordered tree-pushdown systems. Ordinary alternating pushdown systems (and even prefix-rewrite systems) can be easily encoded as alternating ordered tree-pushdown systems by viewing a word as a linear tree; the ordering condition is trivial since symbols have arity $\leqslant 1$. This yields an EXPTIME reachability procedure, by Theorem 4. *Tree-pushdown systems* [**?**] can be seen as ordered tree-pushdown systems where every rule is shallow. By Theorem 4, reachability in alternating tree-pushdown systems is in EXPTIME. Both complexities above reduce to PTIME for non-alternating systems. In the rest of the section, we show how to encode four more sophisticated classes of systems, namely *ordered multi-pushdown systems* (Sec. 4.1), *annotated high-order pushdown systems* (Sec. 4.2), the *Krivine machine with states* (Sec. 4.3), and *ordered annotated multi-pushdown systems* (Sec. 4.4), and we show that reachability for these models can be decided with tight complexity bounds using our saturation procedure.

4.1 Ordered multi-pushdown systems

In an ordered multi-pushdown systems there are n pushdowns. Symbols can be pushed on any pushdown, but only the first non-empty pushdown can be popped [**?**]. This is equivalent to say that to pop a symbol from the k-th pushdown, the contents of the previous pushdowns $1, \dots, k-1$ should be discarded. Formally, an *alternating ordered multi-pushdown system* is a tuple $\mathcal{O} = \langle n, \Gamma, Q, \Delta \rangle$, where $n \in \mathbb{N}_{>0}$ is the *order* of the system (i.e., the number of pushdowns), Γ is a finite pushdown alphabet, Q is a finite set of control locations, and $\Delta \subseteq Q \times O_n \times 2^Q$ is a set of rules of the form (p, o, P) with $p \in Q$, $P \subseteq Q$, and o a pushdown operation in $O_n := \{\mathsf{push}_k(a), \mathsf{pop}_k(a) \mid 1 \leqslant k \leqslant n, a \in \Gamma\}$. We say that $\mathcal{O}$ is *non-deterministic* when P is a singleton for every rule. A multi-pushdown system induces an alternating transition system $\langle \mathcal{C}, \rightarrow_{\mathcal{O}} \rangle$ where the set of configurations is $\mathcal{C} = Q \times (\Gamma^*)^n$, and transitions are defined as follows: for every $(p, \mathsf{push}_k(a), P) \in \Delta$ there exists a transition $(p, w_1, \dots, w_n) \rightarrow_{\mathcal{O}} P \times \{(w_1, \dots, a \cdot w_k, \dots, w_n)\}$, and for every $(p, \mathsf{pop}_k(a), P) \in \Delta$ there exists a transition $(p, w_1, \dots, a \cdot w_k, \dots, w_n) \rightarrow_{\mathcal{O}} P \times \{(\varepsilon, \dots, \varepsilon, w_k, \cdots, w_n)\}$. For $\{p_0\}, T \subseteq Q$, the *reachability problem* for $\mathcal{O}$ asks whether $(p_0, \varepsilon, \dots, \varepsilon) \in \mathrm{Pre}^*(T \times (\Gamma^*)^n)$.

Encoding. We show that an ordered multi-pushdown system can be simulated by an ordered tree-pushdown system. The idea is to encode the k-th pushdown as a linear tree of order k, and to encode a multi-pushdown as a tree of linear pushdowns. Let $\perp$ and $\bullet$ be two new symbols not in Γ, let $\Gamma_\perp = \Gamma \cup \{\perp\}$, and let $\Sigma = \bigcup_{1 \leqslant i \leqslant n} \Gamma_\perp \times \{i\} \cup \{\bullet\}$ be an ordered alphabet, where a symbol $(a, i) \in \Gamma_\perp \times \{i\}$ has order i, rank 1 if $a \in \Gamma$ and rank 0 if $a = \perp$. Moreover, $\bullet$ has rank n and order 1. For simplicity, we write a^i instead of (a, i). A multi-pushdown $w_1, \dots, w_n$, where each $w_j = a_{j,1} \dots a_{j,n_j}$ is encoded as the tree $\mathsf{enc}(w_1, \dots, w_n) := \bullet(a^1_{1,1}(a^1_{1,2}(\dots \perp^1)), \dots, a^n_{n,1}(a^n_{n,2}(\dots \perp^n)))$. For an ordered multi-pushdown system $\mathcal{O} = \langle n, \Gamma, Q, \Delta \rangle$ we define an equivalent ordered

tree-pushdown system $\mathcal{S} = \langle n, \Sigma, Q, \mathcal{R}\rangle$ with Σ defined as above, and set of rules $\mathcal{R}$ defined as follows (we use the convention that variable x_k has order k): For every push rule $(p, \mathsf{push}_k(a), P) \in \Delta$, we have a rule $p, \bullet(\mathsf{x}_1, \dots, \mathsf{x}_n) \to P, \bullet(\mathsf{x}_1, \dots, a^k(\mathsf{x}_k), \dots, \mathsf{x}_n) \in \mathcal{R}$, and for every pop rule $(p, \mathsf{pop}_k(a), P) \in \Delta$, we have $p, \bullet(\mathsf{x}_1, \dots, a^k(\mathsf{x}_k), \dots, \mathsf{x}_n) \to P, \bullet(\bot^1, \dots, \bot^{k-1}, \mathsf{x}_k, \mathsf{x}_{k+1}, \dots, \mathsf{x}_n) \in \mathcal{R}$. Both kind of rules above are linear, and the latter one satisfies the ordering condition since lower-order variables $\mathsf{x}_1, \dots, \mathsf{x}_{k-1}$ are discarded. It is easy to see that $(p, w_1, \dots, w_n) \to_{\mathcal{O}} P \times \{(w'_1, \dots, w'_n)\}$ if, and only if, $(p, \mathsf{enc}(w_1, \dots, w_n)) \to_{\mathcal{S}} P \times \{\mathsf{enc}(w'_1, \dots, w'_n)\}$. Thus, the encoding preserves reachability properties. By Theorem 2, we obtain a n-EXPTIME upper-bound for reachability in alternating multi-pushdown systems of order n. Moreover, since $\mathcal{S}$ is linear, and since $\mathcal{S}$ is non-deterministic when $\mathcal{O}$ is non-deterministic, by Theorem 3 we recover the optimal 2-EXPTIMEc complexity of [**?**].

Theorem 5 ([?]). *Reachability in alternating ordered multi-pushdown systems is in n-EXPTIME. Reachability in non-deterministic ordered multi-pushdown systems is 2-EXPTIMEc.*

4.2 Annotated higher-order pushdown systems

Let Γ be a finite pushdown alphabet. In the following, we fix an order $n \geqslant 1$, and we let $1 \leqslant k \leqslant n$ range over orders. For our purpose, it is convenient to expose the topmost pushdown at every order recursively[5]. We define Γ_k, the set of *pushdowns of order k*, simultaneously for all $1 \leqslant k \leqslant n$, as the least set containing the empty pushdown $\langle\,\rangle$, and, whenever $u_1 \in \Gamma_1, \dots, u_k \in \Gamma_k$, $v_j \in \Gamma_j$ for some $1 \leqslant j \leqslant n$, then $\langle a^{v_j}, u_1, \dots, u_k\rangle \in \Gamma_k$. Pushdown operations are as follows. The operation push^b_k pushes a symbol $b \in \Gamma$ on the top of the topmost order-1 stack and annotates it with the topmost order-k stack, push_k duplicates the topmost order-$(k-1)$ stack, pop_k removes the topmost order-$(k-1)$ stack, and $\mathsf{collapse}_k$ replaces the topmost order-k stack with the topmost order-k stack annotating the topmost symbol:

$$\mathsf{push}^b_k(\langle a^u, u_1, \dots, u_n\rangle) = \langle b^{\langle a^u, u_1, \dots, u_k\rangle}, \langle a^u, u_1\rangle, u_2, \dots, u_n\rangle$$
$$\mathsf{push}_k(\langle a^u, u_1, \dots, u_n\rangle) = \langle a^u, u_1, \dots, u_{k-1}, \langle a^u, u_1, \dots, u_k\rangle, u_{k+1}, \dots, u_n\rangle$$
$$\mathsf{pop}_k(\langle a^u, v_1, \dots, v_{k-1}, \langle b^v, u_1, \dots, u_k\rangle, u_{k+1}, \dots, u_n\rangle) = \langle b^v, u_1, \dots, u_n\rangle$$
$$\mathsf{collapse}_k(\langle a^{\langle b^v, v_1, \dots, v_k\rangle}, u_1, \dots, u_n\rangle) = \langle b^v, v_1, \dots, v_k, u_{k+1}, \dots, u_n\rangle$$

Let $O_n = \bigcup_{k=1}^n \{\mathsf{push}^b_k, \mathsf{push}_k, \mathsf{pop}_k, \mathsf{collapse}_k \mid b \in \Gamma\}$ be the set of stack operations. An *alternating order-n annotated pushdown automaton* is a tuple $\mathcal{P} = \langle n, \Gamma, Q, \Delta\rangle$, where Γ is a finite stack alphabet, Q is a finite set of control locations, and $\Delta \subseteq Q \times \Gamma \times O_n \times 2^Q$ is a set of rules. An annotated pushdown automaton induces a transition system $\langle \mathcal{C}, \to_{\mathcal{P}}\rangle$, where $\mathcal{C} = Q \times \Gamma_n$, and the transition relation is defined as $(p, w) \to_{\mathcal{P}} P \times \{w'\}$ whenever $(p, a, o, P) \in \Delta$ with $w = \langle a^u, \cdots\rangle$ and $w' = o(w)$. Given $\{p_0\}, T \subseteq Q$, the *reachability problem* for $\mathcal{P}$ asks whether $(p_0, \langle\,\rangle) \in \mathrm{Pre}^*(T \times \Gamma_n)$.

[5] Our definition is equivalent to [**?**].

Encoding. We represent annotated pushdowns as trees. Let Σ be the ordered alphabet containing, for each $1 \leqslant k \leqslant n$, an end-of-stack symbol $\bot^k \in \Sigma$ of rank 0 and order k. Moreover, for each $a \in \Gamma$ and order $1 \leqslant k \leqslant n$, there is a symbol $\langle a, k\rangle \in \Sigma$ of order k and rank $k+1$ representing the root of a tree encoding a stack of order k. An order-k stack is encoded as a tree recursively by $\mathsf{enc}_k(\langle\,\rangle) = \bot^k$ and $\mathsf{enc}_k(\langle a^u, u_1, \dots, u_k\rangle) = \langle a, k\rangle(\mathsf{enc}_1(u_1), \dots, \mathsf{enc}_k(u_k), \mathsf{enc}_i(u))$, where i is the order of u. Let $\mathcal{P} = \langle n, \Gamma, Q, \Delta\rangle$ be an annotated pushdown system. We define an equivalent ordered tree-pushdown system $\mathcal{S} = \langle n, \Sigma, Q, \mathcal{R}\rangle$, where Σ is as defined above, and $\mathcal{R}$ contains a rule $p, l \to P, r$ for each rule in $(p, a, o, P) \in \Delta$, where $l \to r$ is as follows (cf. also Fig. 1 in the appendix for a pictorial representation). We use the convention that a variable subscripted by i has order i, and we write $\mathsf{x}_{i..j}$ for $(\mathsf{x}_i, \dots, \mathsf{x}_j)$, and similarly for $\mathsf{z}_{i..j}$:

$$
\begin{aligned}
&\langle a, n\rangle(\mathsf{x}_{1..n}, \mathsf{y}) \to \langle b, n\rangle(\langle a, 1\rangle(\mathsf{x}_1, \mathsf{y}), \mathsf{x}_{2..n}, \langle a, k\rangle(\mathsf{x}_{1..k}, \mathsf{y}) && \text{if } o = \mathsf{push}^b_k\\
&\langle a, n\rangle(\mathsf{x}_{1..n}, \mathsf{y}) \to \langle a, n\rangle(\mathsf{x}_{1..k-1}, \langle a, k\rangle(\mathsf{x}_{1..k}, \mathsf{y}), \mathsf{x}_{k+1..n}, \mathsf{y}) && \text{if } o = \mathsf{push}_k\\
&\langle a, n\rangle(\mathsf{z}_{1..k-1}, \langle b, k\rangle(\mathsf{x}_{1..k}, \mathsf{y}), \mathsf{x}_{k+1..n}, \mathsf{z}) \to \langle b, n\rangle(\mathsf{x}_{1..n}, \mathsf{y}) && \text{if } o = \mathsf{pop}_k\\
&\langle a, n\rangle(\mathsf{z}_{1..k}, \mathsf{x}_{k+1..n}, \langle b, k\rangle(\mathsf{x}_{1..k}, \mathsf{y})) \to \langle b, n\rangle(\mathsf{x}_{1..n}, \mathsf{y}) && \text{if } o = \mathsf{collapse}_k
\end{aligned}
$$

The last two rules satisfy the ordering condition of tree-pushdown systems since only higher-order variables $\mathsf{x}_{k+1}, \dots, \mathsf{x}_n$ are not discarded. It is easy to see that $(p, w) \to_{\mathcal{P}} P \times \{w'\}$ if, and only if, $(p, \mathsf{enc}_n(w)) \to_{\mathcal{S}} P \times \{\mathsf{enc}_n(w')\}$. Consequently, the encoding preserves reachability properties. Since an annotated pushdown system of order n is simulated by a tree-pushdown system of the same order, the following complexity result is an immediate consequence of Theorem 2.

Theorem 6 ([?]). *Reachability in alternating annotated pushdown systems of order n is n-EXPTIMEc.*

4.3 Krivine machine with states

We show that the Krivine machine evaluating on simply-typed λY-terms can be encoded as an ordered tree-pushdown system. This provides the first saturation algorithm for the Krivine machine, yielding an optimal reachability procedure.

A *type* is either the basic type 0 or $\alpha \to \beta$ for types α, β. The *level* of a type is $\mathsf{level}(0) = 1$ and $\mathsf{level}(\alpha \to \beta) = \max(\mathsf{level}(\alpha) + 1, \mathsf{level}(\beta))$. We abbreviate $\alpha \to \dots \to \alpha \to \beta$ as $\alpha^k \to \beta$. Let $\mathcal{V} = \{x_1^{\alpha_1}, x_2^{\alpha_2}, \dots\}$ be a countably infinite set of typed variables, and let Γ be a ranked alphabet. A *term* is either (i) a constant $f^{0^k \to 0} \in \Gamma$, (ii) a variable $x^\alpha \in \mathcal{V}$, (iii) an abstraction $(\lambda x^\alpha \cdot M^\beta)^{\alpha \to \beta}$, (iv) an application $(M^{\alpha \to \beta} N^\alpha)^\beta$, or (v) a fixpoint $(Y M^{\alpha \to \alpha})^\alpha$. For a given term M, its set of *free variables* is defined as usual. A term M is *closed* if it does not have any free variable. We denote by $\Lambda(M)$ be the set of *sub-terms* of M. An *environment* ρ is a finite type-preserving function assigning closures to variables, and a *closure* C^α is a pair consisting of a term of type α and an environment, as expressed by the following mutually recursive grammar: $\rho ::= \varnothing \mid \rho[x^\alpha \mapsto C^\alpha]$ and $C^\alpha ::= (M^\alpha, \rho)$. In a closure (M^α, ρ), M^α is called the *skeleton*, and it determines the type and level of the closure. Let $Cl^\alpha(M)$ be the set of closures of type α with skeleton

in $\Lambda(M)$. A *alternating Krivine machine*[6] *with states* of level $l \in \mathbb{N}_{>0}$ is a tuple $\mathcal{M} = \langle l, \Gamma, Q, M^0, \Delta \rangle$, where $\langle \Gamma, Q, \Delta \rangle$ is an alternating tree automaton, and M^0 is a closed term of type 0 s.t. the level of any sub-term in $\Lambda(M^0)$ is at most l. In the following, let $\tau = \tau_1 \to \cdots \to \tau_k \to 0$. The Krivine machine $\mathcal{M}$ induces a transition system $\langle \mathcal{C}, \to_{\mathcal{M}} \rangle$, where in a configuration $(p, C^\tau, C_1^{\tau_1}, \ldots, C_k^{\tau_k}) \in \mathcal{C}$, $p \in Q$, $C^\tau \in Cl^\tau(M^0)$ is the *head closure*, and $C_1^{\tau_1} \in Cl^{\tau_1}(M^0), \ldots, C_k^{\tau_k} \in Cl^{\tau_k}(M^0)$ are the *argument closures*. The transition relation $\to_{\mathcal{M}}$ depends on the structure of the skeleton of the head closure. It is deterministic except when the head is a constant in Γ, in which case Δ controls how the state changes:

$$
\begin{aligned}
(p, (x^\tau, \rho), C_1^{\tau_1}, \ldots, C_k^{\tau_k}) &\to_{\mathcal{M}} \{(p, \rho(x)^\tau, C_1^{\tau_1}, \ldots, C_k^{\tau_k})\} \\
(p, (M^\tau N^{\tau_1}, \rho), C_2^{\tau_2}, \ldots, C_k^{\tau_k}) &\to_{\mathcal{M}} \{(p, (M^\tau, \rho), (N^{\tau_1}, \rho), C_2^{\tau_2}, \ldots, C_k^{\tau_k})\} \\
(p, (YM^{\tau \to \tau}, \rho), C_1^{\tau_1}, \ldots, C_k^{\tau_k}) &\to_{\mathcal{M}} \{(p, (M^{\tau \to \tau}, \rho), ((YM)^\tau, \rho), C_1^{\tau_1}, \ldots, C_k^{\tau_k})\} \\
(p, (\lambda x^{\tau_0} \cdot M^\tau, \rho), C_0^{\tau_0}, \ldots, C_k^{\tau_k}) &\to_{\mathcal{M}} \{(p, (M^\tau, \rho[x^{\tau_0} \mapsto C_0^{\tau_0}]), C_1^{\tau_1}, \ldots, C_k^{\tau_k})\} \\
(p, (a^{0^k \to 0}, \rho), C_1^0, \ldots, C_k^0) &\to_{\mathcal{M}} P_1 \times \{C_1^0\} \cup \cdots \cup P_k \times \{C_k^0\} \\
&\qquad \text{for every } p \xrightarrow{a} P_1 \cdots P_k \in \Delta
\end{aligned}
$$

Given $\{p_0\}, T \subseteq Q$, the *reachability problem* for $\mathcal{M}$ asks whether $(p_0, M^0) \in \mathrm{Pre}^*(T \times (\bigcup_{\tau = \tau_1 \to \cdots \to \tau_k \to 0} Cl^\tau(M^0) \times Cl^{\tau_1}(M^0) \times \cdots \times Cl^{\tau_k}(M^0)))$.

Encoding. Following [**?**], we encode closures and configurations of the Krivine machine as trees. Fix a Krivine machine $\mathcal{M} = \langle l, \Gamma, Q, M^0, \Delta \rangle$ of level l. We assume a total order on all variables $\langle x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \rangle$ appearing in M^0. For a type τ, we define $\mathsf{ord}(\tau) = l - \mathsf{level}(\tau)$. We construct an ordered tree-pushdown system $\mathcal{S} = \langle l, \Sigma, Q', \mathcal{R} \rangle$ of order l as follows. The ordered alphabet is $\Sigma = \{N^\tau, [N^\tau] \mid N^\tau \in \Lambda(M^0)\} \cup \{\bot\}$. Here, N^τ is a symbol of $\mathsf{rank}(N^\tau) = n$ and $\mathsf{ord}(N^\tau) = \mathsf{ord}(\tau)$. Moreover, if $\tau = \tau_1 \to \cdots \to \tau_k \to 0$ for some $k \geqslant 0$, then $[N^\tau]$ is a symbol of $\mathsf{rank}([N^\tau]) = n + k$ and $\mathsf{ord}([N^\tau]) = \mathsf{ord}(\tau)$. Finally, $\bot$ is a leaf of order 1. The set of states is $Q' = Q \cup \bigcup_{p \xrightarrow{a} P_1 \cdots P_k \in \Delta} \{(1, P_1), \ldots, (k, P_k)\}$. A closure (N^τ, ρ) is encoded recursively as $\mathsf{enc}(N^\tau, \rho) = N^\tau(t_1, \ldots, t_n)$, where, for every $1 \leqslant i \leqslant n$, i) if $x_i \in \mathsf{FV}(N^\tau)$ then $t_i = \mathsf{enc}(\rho(x_i))$, and ii) $t_i = \bot$ otherwise. A configuration $c = (p, (N^\tau, \rho), C_1^{\tau_1}, \ldots, C_k^{\tau_k})$ is encoded as the tree $\mathsf{enc}(c) = [N^\tau](t_1, \ldots, t_n, \mathsf{enc}(C_1^{\tau_1}), \ldots, \mathsf{enc}(C_k^{\tau_k}))$, where the first n subtrees encode the closure (N^τ, ρ), i.e., $\mathsf{enc}(N^\tau, \rho) = N^\tau(t_1, \ldots, t_n)$. The encoding is extended pointwise to sets of configurations. Below, we assume that $\tau = \tau_1 \to \cdots \to \tau_k \to 0$, that variable y_j has order $\mathsf{ord}(\tau_j)$ for every $1 \leqslant j \leqslant k$, and that variables x_i and x'_i have order $\mathsf{ord}(\alpha_i)$ for every $1 \leqslant i \leqslant n$. Notice that $\mathsf{ord}(\tau) < \mathsf{ord}(\tau_1), \ldots, \mathsf{ord}(\tau_k)$. Moreover, we write $\mathbf{x} = \langle \mathsf{x}_1, \ldots, \mathsf{x}_n \rangle$, $\mathbf{z} = \langle \mathsf{z}_1, \ldots, \mathsf{z}_n \rangle$, and $\mathbf{y} = \langle \mathsf{y}_1, \ldots, \mathsf{y}_k \rangle$. $\mathcal{R}$ contains the following rules:

$$
\begin{aligned}
p, [x_i^\tau](\mathsf{z}_1, \ldots, M^\tau(\mathbf{x}), \ldots, \mathsf{z}_n, \mathbf{y}) &\to \{p\}, [M^\tau](\mathbf{x}, \mathbf{y}) \\
p, [M^\tau N^{\tau_1}](\mathbf{x}, \mathsf{y}_2, \ldots, \mathsf{y}_k) &\to \{p\}, [M^\tau](\mathbf{x}, N^{\tau_1}(\mathbf{x}), \mathsf{y}_2, \ldots, \mathsf{y}_k) \\
p, [YM^{\tau \to \tau}](\mathbf{x}, \mathbf{y}) &\to \{p\}, [M^{\tau \to \tau}](\mathbf{x}, YM(\mathbf{x}), \mathbf{y}) \\
p, [\lambda x_i^{\tau_0} \cdot M^\tau](\mathbf{x}, \mathsf{y}_0, \mathbf{y}) &\to \{p\}, [M^\tau](\mathsf{x}_1, \ldots, \mathsf{x}_{i-1}, y_0, \mathsf{x}_{i+1}, \ldots, \mathsf{x}_n, \mathbf{y})
\end{aligned}
$$

[6] Cf. also [**?**] for a definition of the Krivine machine in a different context.

$$p, [a^{0^k \to 0}](\mathbf{x}, \mathbf{y}) \to \{(1, P_1), \ldots, (k, P_k)\}, [a^{0^k \to 0}](\mathbf{x}, \mathbf{y}) \qquad \forall (p \xrightarrow{a} P_1 \cdots P_k \in \Delta)$$
$$(i, P_i), [a^{0^k \to 0}](\mathbf{z}, \mathsf{y}_1, \ldots, M_i^0(\mathbf{x}), \ldots, \mathsf{y}_k) \to P_i, [M_i^0](\mathbf{x})$$

The first rule satisfies the ordering condition since the shared variables y_i's are of order strictly higher than M^τ. A direct inspection of the rules shows that, for a configuration c and a set of configurations D, where $\{c\} \cup D$ does not contain an intermediate configuration of the form $((i, P_i), \cdots)$, we have $c \to^*_{\mathcal{M}} D$ if, and only if, $\mathsf{enc}(c) \to^*_{\mathcal{S}} \mathsf{enc}(D)$. Therefore, the encoding preserves reachability properties. Since a Krivine machine of level n is simulated by a tree-pushdown system of order n, the following is an immediate consequence of Theorem 2.

Theorem 7 ([?]). *Reachability in alternating Krivine machines with states of level n is n-EXPTIMEc.*

4.4 Ordered annotated multi-pushdown systems

Ordered annotated multi-pushdown systems are the common generalization of ordered multi-pushdown systems and annotated pushdown systems [**?**]. Such a system is comprised of $m > 0$ annotated higher-order pushdowns arranged from left to right, where each pushdown is of order $n > 0$. While push operations are unrestricted, pop and collapse operations implicitly destroy all pushdowns to the left of the pushdown being manipulated, in the spirit of [**?**]. [**?**] has shown that reachability in this model can be decided in mn-fold exponential time, by using a saturation-based construction leveraging on the previous analysis for the first-order case [**?**]. In Sec. B in the appendix we provide a simple encoding of an annotated multi-pushdown system with parameters (m, n) into a tree-pushdown system of order mn. It is essentially obtained by taking together our previous encodings of ordered (cf. Sec.4.1) and annotated systems (cf. Sec. 4.2). The following complexity result is a direct consequence of Theorem 2.

Theorem 8 ([?]). *Reachability in alternating ordered annotated multi-pushdown systems of parameters (m, n) is in (mn)-EXPTIME.*

5 Conclusions

We have introduced a novel extension of pushdown automata which is able to capture several sophisticated models thanks to a simple ordering condition on the tree-pushdown. As future research it would be interesting to study other restrictions, such as *phase-bounding* [**?**] or *scope-bounding* [**?**]. Our general saturation algorithm can be used to verify reachability properties. We plan to extend it to the more general *parity properties*, in the spirit of [**?**,**?**]. We leave as future work implementing our saturation algorithm, leveraging on subsumption techniques to keep the search space small.

Acknowledgments. We kindly acknowledge stimulating discussions with Irène Durand, Géraud Sénizergues, and Jean-Marc Talbot.

A Proof of Lemma 1

Let $\mathcal{A}$ be the automaton recognizing the target set of configurations, and let $\mathcal{B}$ be the automaton obtained at the end of the saturation procedure (cf. page 5).

Lemma 1. *For $\mathcal{A}$ and $\mathcal{B}$ be as above, $\mathcal{L}(\mathcal{B}, P) = \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.*

We prove the two inclusions of the lemma separately.

Lemma 2 (Completeness). *For $\mathcal{A}$ and $\mathcal{B}$ as above, $\mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P)) \subseteq \mathcal{L}(\mathcal{B}, P)$.*

Proof. Let (p, t) be a configuration in $\mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$ with $p \in P$ and $t = a(t_1, \ldots, t_m)$. We show $(p, t) \in \mathcal{L}(\mathcal{B}, P)$ by induction on the length $d \geqslant 0$ of the shortest sequence of rewrite steps from (p, t) to $\mathcal{L}(\mathcal{A}, P)$. If $d = 0$, then $(p, t) \in \mathcal{L}(\mathcal{A}, P)$. Since the saturation procedure only adds states and transitions to $\mathcal{A}$, we directly have $(p, t) \in \mathcal{L}(\mathcal{B}, P)$. Inductively, assume that the property holds for all configurations reaching $\mathcal{L}(\mathcal{A}, P)$ in at most $d \geqslant 0$ steps, and let configuration (p, t) be at distance $d + 1 > 0$ from $\mathcal{L}(\mathcal{A}, P)$. There exists a rule $p, l \to S, r \in \mathcal{R}$ with $l = a(u_1, \ldots, u_m)$ and a substitution σ s.t. $t = l\sigma$ and

$$(p, t) \to_{\mathcal{S}} S \times \{r\sigma\} \subseteq \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$$

By induction hypothesis, $S \times \{r\sigma\} \subseteq \mathcal{L}(\mathcal{B}, P)$. By definition of Δ', automaton $\mathcal{B}$ contains a transition $p \xrightarrow{a} P_1 \cdots P_m$. It thus suffices to show that $t_1 \in \mathcal{L}(P_1), \ldots, t_m \in \mathcal{L}(P_m)$. If $u_i = \mathsf{x}$ is a variable, then by definition $P_i = \bigcup t(r^{-1}(\mathsf{x}))$. Since $S \times \{r\sigma\} \subseteq \mathcal{L}(\mathcal{B}, P)$, $t_i = \sigma(\mathsf{x}) \in \mathcal{L}(P_i)$. If u_i is not a variable and r-ground, then $P_i = \{p^{u_i}\}$ and $t_i \in \mathcal{L}(p^{u_i})$ by construction. Finally, if u_i is not r-ground, then $u_i = b(v_1, \ldots, v_n)$, $t_i = b(s_1, \ldots, s_n)$, and $\mathcal{B}$ contains a transition $(g, P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_m) \xrightarrow{b} S_1 \cdots S_n$. Thus, it suffices to show $s_1 \in \mathcal{L}(S_1), \ldots, s_n \in \mathcal{L}(S_n)$, which is done as above.

Lemma 3 (Soundness). *For $\mathcal{A}$ and $\mathcal{B}$ as above, $\mathcal{L}(\mathcal{B}, P) \subseteq \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.*

The soundness proof requires several steps. We assume w.l.o.g that in the automaton $\mathcal{A}$ initial states in P have no incoming transitions. Notice that this property is preserved during the saturation procedure, therefore also in $\mathcal{B}$ there are no transitions entering initial states in P. We also assume that in deep rules $p, a(u_1, \ldots, u_m) \to S, r$, the unique $u_i = b(\cdots)$ which is not r-ground actually occurs in the first position, i.e, $i = 1$. This is w.l.o.g. since we can always add shallow rules that just reshuffle subtrees. First, we assign a semantics $[\![p]\!] \subseteq \mathcal{T}(\Sigma)$ to all states p in $\mathcal{B}$. For a set of states $S \subseteq Q'$, $[\![S]\!] := \bigcap_{p \in S} [\![p]\!]$, where $[\![p]\!]$ is defined as follows:

$$[\![p]\!] := \begin{cases} \{t \mid (p, t) \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))\} & \text{if } p \in P \\ \mathcal{L}(p) & \text{if } p \in Q \backslash P \text{ or } p = p^v \in Q_0 \\ \left\{ t_1 \mid \begin{array}{c} \forall (t_2 \in [\![P_2]\!], \ldots, t_m \in [\![P_m]\!]) \cdot \\ (q, a(t_1, \ldots, t_m)) \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P)) \end{array} \right\} & \begin{array}{l} \text{if } p \in Q'_i \backslash Q'_{i+1} \text{ for some } 1 \leqslant i < n \\ \quad \text{with } p = (g, P_2, \ldots, P_m), \\ \quad \text{and } g = (q, a(\cdots) \to \cdots) \end{array} \end{cases}$$

In the second case, $\mathcal{L}(p)$ refers to the language of p in the automaton $\mathcal{B}$. In the last case, $[\![p]\!]$ is defined in terms of $[\![P_2]\!], \ldots, [\![P_m]\!]$, which is well-defined by induction on the order since $P_2, \ldots, P_m \subseteq Q'_{i+1}$. Second, we define sound transitions as those respecting the semantics: Formally, a transition $P \xrightarrow{a} P_1 \cdots P_m$ is *sound* iff whenever $\forall(t_1 \in [\![P_1]\!], \ldots, t_m \in [\![P_m]\!])$, $a(t_1, \ldots, t_m) \in [\![P]\!]$.

Proposition 1. *If all transitions are sound, then $\mathcal{L}(p) \subseteq [\![p]\!]$ for every $p \in Q'$.*

Proof. Let $t \in \mathcal{L}(p)$. We proceed by complete induction on the height of t. If $t = a$ is a leaf, then there exists a sound transition $p \xrightarrow{a}$, and thus $a \in [\![p]\!]$ by definition of sound transition. For the inductive step, let $t = a(t_1, \ldots, t_m)$. There exists a sound transition $p \xrightarrow{a} P_1 \cdots P_m$ s.t. $t_1 \in \mathcal{L}(P_1), \ldots, t_m \in \mathcal{L}(P_m)$. By induction hypothesis, $t_1 \in [\![P_1]\!], \ldots, t_m \in [\![P_m]\!]$, and thus by the definition of sound transition, $a(t_1, \ldots, t_m) \in [\![p]\!]$.

Proposition 2. *Transitions in $\Delta \cup \Delta_0$ are sound.*

Proof. Let $p \xrightarrow{a} P_1 \cdots P_m \in \Delta \cup \Delta_0$, and let $t_1 \in [\![P_1]\!], \ldots, t_m \in [\![P_m]\!]$. Since we assume that there are no transitions back to the initial states in P, we have $P_1, \ldots, P_m \subseteq Q \backslash P$, and thus $t_1 \in \mathcal{L}(P_1), \ldots, t_m \in \mathcal{L}(P_m)$ by the definition of the semantics. Consequently, $t := a(t_1, \ldots, t_m) \in \mathcal{L}(p)$. If $p \notin P$ we are done, since $[\![p]\!] = \mathcal{L}(p)$ in this case. Otherwise, if $p \in P$ then $(p, t) \in \mathcal{L}(\mathcal{A}, P)$, which is included in $\mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$, and thus we have $t \in [\![p]\!]$ by definition.

Proposition 3. *The saturation procedure adds only sound transitions.*

Proof. Let $g = p, l \to S, r$ with $l = a(u_1, \ldots, u_m)$, and let t be a sound run tree in $\mathcal{B}$ from S on r. We show that the transition $p \xrightarrow{a} P_1 \cdots P_m$ as added by rule (Δ'-shallow) is sound. To this end, let $t_1 \in [\![P_1]\!], \ldots, t_m \in [\![P_m]\!]$, and we show $t' := a(t_1, \ldots, t_m) \in [\![p]\!]$. Since $p \in P$, this amounts to showing that $(p, t') \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$. We apply the rewrite rule g above to configuration (p, t'):

$$(p, t') \to S \times \{r\sigma\},$$

where σ is the unique substitution s.t. $t' = l\sigma$. (σ is unique since l is linear.) It thus suffices to show $S \times \{r\sigma\} \subseteq \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$. First, assume that g is shallow. Every variable x appearing in r is labelled by t by the set of states $P_i = \bigcup t(r^{-1}(\mathsf{x}))$, for some i. Since t uses only sound transitions and $t_1 \in [\![P_1]\!], \ldots, t_m \in [\![P_m]\!]$, by induction on its height we have $r\sigma \in [\![S]\!]$, which implies $S \times \{r\sigma\} \subseteq \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$ by the definition of the semantics since $S \subseteq P$. If g is not shallow, then $P_1 = \{(g, P_2, \ldots, P_m)\}$. In this case, since $t_1 \in [\![P_1]\!]$, by the definition of the semantics, we deduce directly $(p, t') \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.

When g is not shallow, the transition $(g, P_2, \ldots, P_m) \xrightarrow{b} S_1 \cdots S_n$ is additionally added by rule (Δ'-deep), and we have to show that this transition is sound too. Let $w_1 \in [\![S_1]\!], \ldots, w_n \in [\![S_n]\!]$, and we show $t_1 := b(w_1, \ldots, w_n) \in [\![(g, P_2, \ldots, P_m)]\!]$. To this end, let $t_2 \in [\![P_2]\!], \ldots, t_m \in [\![P_m]\!]$, and we show

$(p, a(t_1, \dots, t_m)) \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$. The proof is as above, noticing that t labels a variable x in r by either P_i for some $2 \leqslant i \leqslant m$, or S_j for some $1 \leqslant j \leqslant n$, and we can again conclude $r\sigma \in [\![S]\!]$ by induction on the height of t.

Proof (of Lemma 3). By Proposition 2, the initial transitions in $\Delta \cup \Delta_0$ are sound, and by Proposition 3, all transitions in Δ' are sound. Let $(p, t) \in \mathcal{L}(\mathcal{B}, P)$. Thus, $t \in \mathcal{L}(p)$. By Proposition 1, $t \in [\![p]\!]$. Since $p \in P$, by the definition of the semantics, $(p, t) \in \mathrm{Pre}^*(\mathcal{L}(\mathcal{A}, P))$.

B Ordered annotated multi-pushdown systems

We encode ordered annotated multi-pushdown systems [**?**] into tree-pushdown systems. Formally, an *alternating ordered annotated multi-pushdown system* is a tuple $\mathcal{R} = \langle m, n, \Gamma, Q, \Delta \rangle$, where $m \in \mathbb{N}_{>0}$ is the the number of higher-order pushdowns, $n \in \mathbb{N}_{>0}$ is the order of each of the m higher-order pushdowns, Γ is a finite pushdown alphabet, Q is a finite set of control locations, and $\Delta \subseteq Q \times \Gamma_\perp^m \times O \times 2^Q$ is a set of rules. Let $O = \bigcup_{l=1}^m \bigcup_{k=1}^n \{l\} \times \{\mathsf{push}_k^b, \mathsf{push}_k, \mathsf{pop}_k, \mathsf{collapse}_k \mid b \in \Gamma\}$. Pop and collapse operations are called *consuming*. An alternating ordered annotated multi-pushdown system $\mathcal{R}$ induces an alternating transition system $\langle \mathcal{C}, \to_{\mathcal{R}} \rangle$, $\mathcal{C} = Q \times \Gamma_n^m$, and $(p, w_1, \dots, w_m) \to_{\mathcal{R}} P \times \{(w_1', \dots, w_m')\}$ if, and only if, there exists a rule $(p, (a_1, \dots, a_m), (l, o), P) \in \Delta$ s.t. 1) $w_1 = \langle a_1^{u_1}, \cdots \rangle, \dots, w_m = \langle a_m^{u_m}, \cdots \rangle$, 2) if o is consuming, then $w_1' = \cdots = w_{l-1}' = \langle\,\rangle$, 3) if o is not consuming, then $w_1' = w_1, \dots, w_{l-1}' = w_{l-1}$, 4) $w_l' = o(w_l)$, and 5) $w_{l+1}' = w_{l+1}, \dots, w_m' = w_m$. For $\{p_0\}, T \subseteq Q$, the *reachability problem* for $\mathcal{R}$ asks whether $(p_0, \langle\,\rangle, \dots, \langle\,\rangle) \in \mathrm{Pre}^*(T \times \Gamma_n^m)$.

Encoding. Let Σ be an ordered alphabet containing, for every $a \in \Gamma_\perp$, pushdown index $1 \leqslant l \leqslant m$, and order $1 \leqslant k \leqslant n$ a symbol (l, k, a) of order $(l-1) \cdot n + k$ and rank $k + 1$. Moreover, Σ also contains, for every tuple $(a_1, \dots, a_m) \in \Gamma_\perp^m$, a symbol $(a_1, \dots, a_m)$ of order 1 and rank $m \cdot (n+1)$. Thus Σ has order mn. Fix a pushdown index l. An order-k pushdown is encoded as the tree

$$\mathsf{enc}_{l,k}(\langle a^{\hat{u}}, u_1, \dots, u_k \rangle) = \quad \begin{array}{ccccccc} & & & (l,k,a) & & & \\ & \swarrow & & \downarrow & & \searrow & \\ \mathsf{enc}_{l,1}(u_1) & & \cdots & & \mathsf{enc}_{l,k}(u_k) & & \mathsf{enc}_{l,i}(\hat{u}) \end{array},$$

where i is the order of $\hat{u}$, and a m-tuple of order-n pushdowns

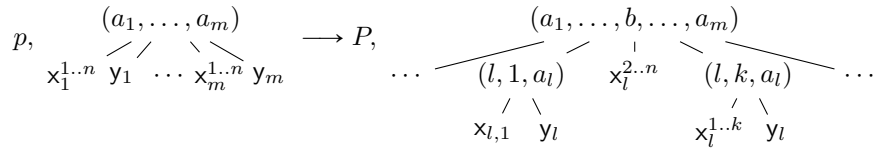
$$w = \langle a_1^{\hat{u}_1}, u_{1,1}, \dots, u_{1,n} \rangle, \dots, w_m = \langle a_m^{\hat{u}_m}, \dots, u_{m,n} \rangle$$

is encoded as the as a the following tree $\mathsf{enc}_n(w)$

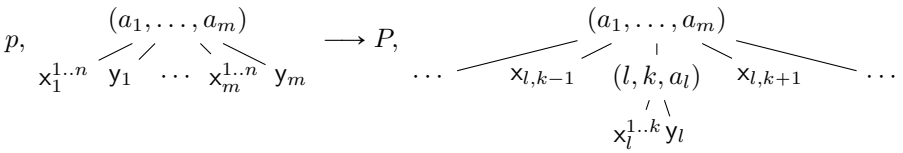
$$\begin{array}{ccccccccc} & & & & (a_1, \dots, a_m) & & & & \\ & \swarrow & & & \downarrow & & & \searrow & \\ \mathsf{enc}_{1,1}(u_{1,1}) & & \cdots & & \mathsf{enc}_{1,n}(u_{1,n}) & \mathsf{enc}_{1,i_1}(\hat{u}_1) & \cdots & & \mathsf{enc}_{m,i_m}(\hat{u}_m) \end{array}$$

where i_1 is the order of $\hat{u}_1$, …, i_m is the order of $\hat{u}_m$.

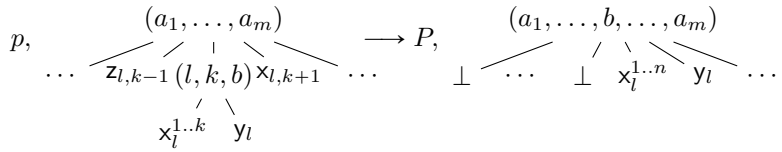
Let $\langle m, n, \Gamma, Q, \Delta\rangle$ be an ordered annotated multi-pushdown system. We define an equivalent ordered tree-pushdown system $\mathcal{S} = \langle mn, \Sigma, Q, \mathcal{R}\rangle$ of order mn, where Σ and Q are as defined above, and $\mathcal{R}$ contains a rule for each rule in Δ, as follows. We use the convention that a variable subscripted by (l, k) has order $(l-1)\cdot n + k$, and we write $\mathsf{x}_l^{i..j}$ (with $i \leqslant j$) for the tuple of variables $(\mathsf{x}_{l,i}, \ldots, \mathsf{x}_{l,j})$. If $(p, (a_1, \ldots, a_m), (l, \mathsf{push}_k^b), P) \in \Delta$, then there is the following shallow rule in $\mathcal{R}$:



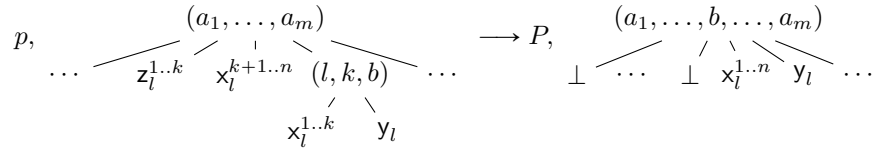
If $(p, (a_1, \ldots, a_m), (l, \mathsf{push}_k), P) \in \Delta$, then there is the following shallow rule in $\mathcal{R}$:



If $(p, (a_1, \ldots, a_m), (l, \mathsf{pop}_k), P) \in \Delta$, then there is the following deep rule in $\mathcal{R}$:



Finally, if $(p, (a_1, \ldots, a_m), (l, \mathsf{collapse}_k), P) \in \Delta$, then there is the following deep rule in $\mathcal{R}$:



The two deep rules above satisfy the ordering condition since (l, k, b) has order $(l-1)\cdot n + k$, and all other variables $x_l^{k+1..n}$, $x_{l+1}^{1..n}$, y_{l+1}, …, $x_m^{1..n}$, y_m have strictly higher order.

Lemma 4 (Simulation). *We have that* $(p, w) \to_{\mathcal{R}} P \times \{w'\}$ *if, and only if,* $(p, \mathsf{enc}_n(w)) \to_{\mathcal{S}} P \times \{\mathsf{enc}_n(w')\}$. *Thus, the reachability problem for* $\mathcal{R}$ *is equivalent to the reachability problem for* $\mathcal{S}$.

C The translation for annotated pushdown systems

We present graphically the rewrite rules of the resulting ordered tree transition system. The rules in Figure 1 are the same as in the main text. We hope that the graphical presentation better conveys the intuition behind them.

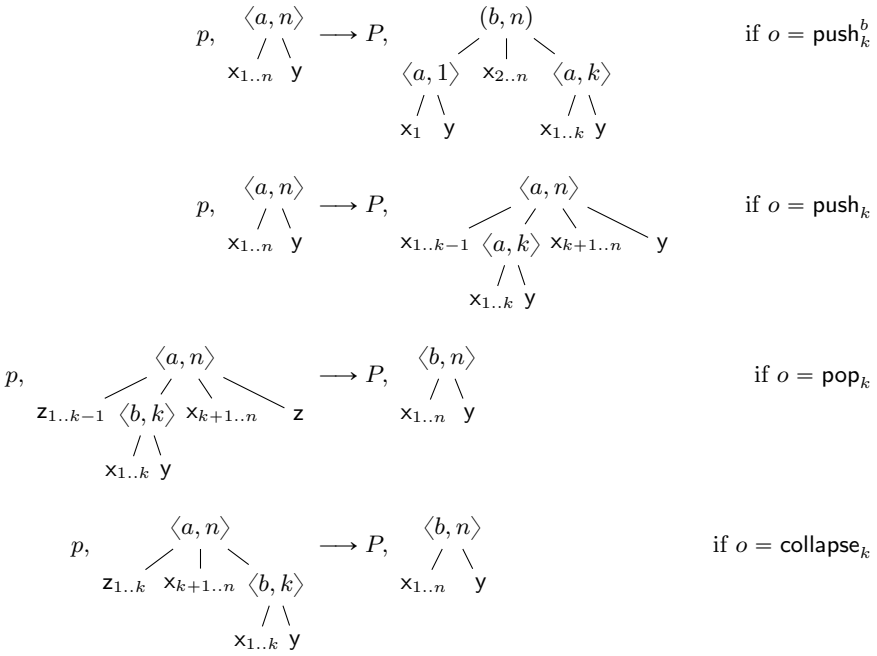


Fig. 1: Translation from annotated pushdowns to ordered tree-pushdown systems.

D The translation for Krivine machines

We present graphically the rewrite rules of the resulting ordered tree transition system. The rules in Figure 2 are the same as in the main text. We hope that the graphical presentation better conveys the intuition behind them.

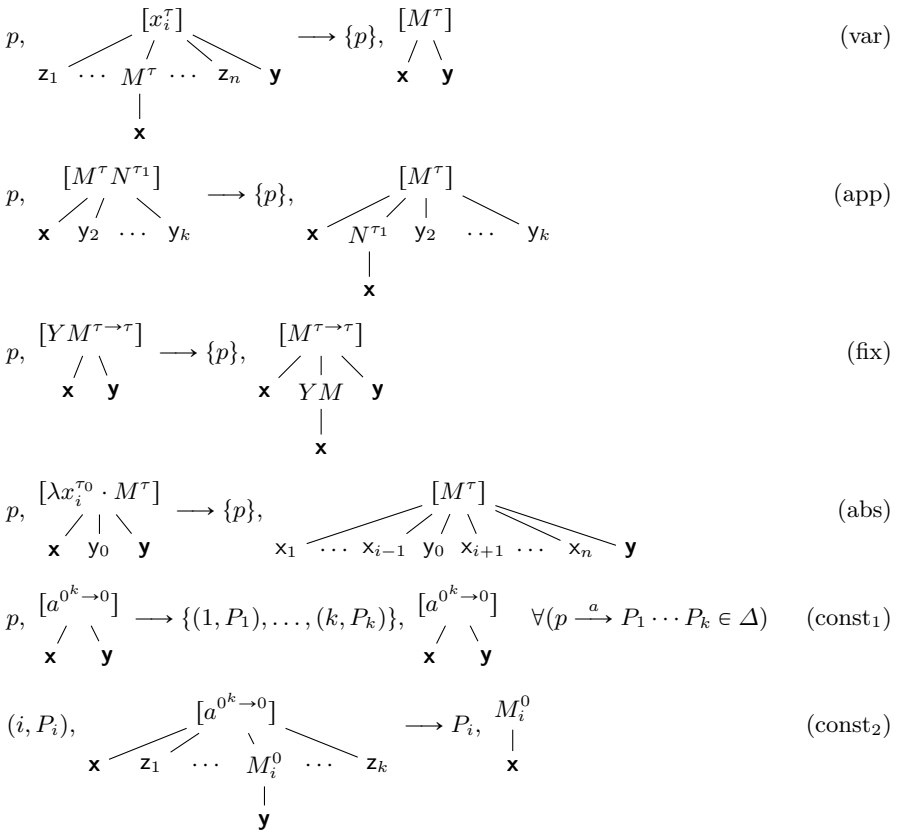


Fig. 2: Translation from the Krivine machine to ordered tree-pushdown systems.