Streamlining Collapsible Pushdown Systems and their Model-Checking

Christopher Broadbent* and Naoki Kobayashi

The University of Tokyo

Abstract. Collapsible pushdown systems (CPDS) provide a pushdown-like model equi-expressive with higher-order recursion schemes (HORS), a type of grammar that can accurately model higher-order control-flow. Despite their extensional equivalence with HORS, CPDS are still a loose fit since they exhibit behaviour with no natural interpretation in terms of HORS evaluation. This has made it difficult to formally understand the precise relationship between two general approaches to higher-order model-checking: *automata-theoretic* algorithms operating on CPDS and *type-theroetic* model-checking algorithms operating on HORS. The purpose of this work is to help formally unify these two approaches. To this end we study *streamlined collapsible pushdown systems (SCPDS)*, which can be seen as the subclass of CPDS that provide precisely what is necessary to evaluate HORS and nothing more. We then consider the SCPDS analogue of a recent *'saturation algorithm'* for CPDS. The important point is that streamlining the automata-theoretic approach in this way allows it to be given a precise type-theoretic interpretation.

1 Introduction

Higher-order recursion schemes (HORS) are grammars providing a natural model for higher-order functional programs [12]. A number of promising approaches have been proposed for model-checking *safety properties* of HORS in practice, despite the problem's non-elementary complexity [15]. Most practical algorithms [11, 14, 13, 16] use the framework proposed by Kobayashi [12], which reduces the problem to typing the HORS in an intersection type system characterising safety.

More recently an approach based on *collapsible pushdown systems* (CPDS) [8] has been considered. CPDS generate the same class of trees as HORS and, as a generalisation of pushdown automata, invite the application of automata theoretic techniques. In particular, the *saturation* algorithm [1, 7] for pushdown systems was generalised by Broadbent *et al.* to CPDS [3] (further generalising [9, 2, 19]). A variant of this algorithm was implemented in the tool CSHORe [4].

In this paper we study a version of this saturation algorithm [3] for *streamlined CPDS (SCPDS)* corresponding to the special case when a CPDS is obtained by compiling a HORS [6]. This pushdown-like model constitutes a closer fit to HORS than arbitrary CPDS in a manner similar to Salvati and Walukiewicz's

* Supported by a JSPS short-term fellowship.

work with Krivine machines [17, 18]. It turns out that SCPDS saturation can also be naturally interpreted as computing intersection types for a HORS, although these types are *dual* to those of Kobayashi's original work [12] in the sense that they describe *unsafe* terms whilst Kobayashi's original work use types describing *safe* terms. Since this automata-theoretic algorithm acting on (a special case of) CPDS has a type-theoretic Doppelgänger, it constitutes a first formal step towards consolidating type-based and automata-based approaches.

More concretely, the type-based incarnation of SCPDS saturation amounts to a naïve version of HorSat, one of two saturation-like algorithms [5] that work directly on HORS. In *op. cit.* we remarked informally on general parallels between HorSat and CPDS saturation but offered no precise comparison.

We intend this paper to be of interest even without knowledge of HorSat or CPDS. In particular we introduce only the streamlined version of the latter, making separate remarks to help the CPDS expert understand how SCPDS and its saturation algorithm can be seen as restrictions of their CPDS counterparts. The reader unfamiliar with CPDS can ignore these remarks.

2 Preliminaries

2.1 Ranked Trees and Automata

A *ranked alphabet* Σ, which we fix, is a set of symbols a each assigned an arity $\mathtt{ar}(a)$. Given a set U, $U^\perp := U \uplus \{\perp\}$ and if ranked we set $\mathtt{ar}(\perp) := 0$.

An (unlabeled) tree T is a subset of $\{1, \ldots, m\}^*$ such that $\epsilon \in T$, and $\pi k \in T$ implies $\pi \cup \{\pi i \mid 1 \leq i \leq k\} \subseteq T$. For a set S of symbols, an S-labeled tree is a map T from a tree to S. The *leaves* of T are the elements of $dom(T)$ that are maximal with respect to the prefix relation. A ***ranked-tree*** T over Σ requires that $T(\pi) = a$ (for $\pi \in dom(t)$) imply that $\pi i \in dom(t)$ iff $1 \leq i \leq \mathtt{ar}(a)$.

Definition 1. *An **alternating co-trivial tree automaton** is a quadruple $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta_\mathcal{A}, q_I)$, where Σ is a ranked alphabet, $\mathcal{Q}$ is a set of states, $q_I \in \mathcal{Q}$ is the initial state, and $\Delta \subseteq \mathcal{Q} \times \Sigma \times 2^{\mathcal{Q} \times [1,r]}$ is a transition function where $r := \max\{\ \mathtt{ar}(a)\ \mid a \in \Sigma\ \}$. If $(q, a, S) \in \Delta_\mathcal{A}$, then $S \subseteq \mathcal{Q} \times \{1, \ldots, \mathtt{ar}(a)\}$.*

For a $\Sigma^\perp$-labeled ranked tree T, a $2^\mathcal{Q}$-labeled tree R is a ***run-tree*** of $\mathcal{A}$ over T if (i) $dom(R) = dom(T)$ with $R(\epsilon) = \{q_I\}$, (ii) for every $\pi \in dom(T)$ and $q \in R(\pi)$, there exists $S_q \subseteq \mathcal{Q} \times [1, r]$ such that $(q, T(\pi), S_q) \in \Delta_\mathcal{A}$ and for each i such that $\pi i \in dom(T)$, we have $R(\pi i) = \left\{\ q' \ \middle|\ (q', i) \in \bigcup_{q \in R(w)} S_q\ \right\}$. A $\Sigma^\perp$-labeled ranked tree T is ***accepted by*** $\mathcal{A}$ if there is a run-tree of $\mathcal{A}$ over T such that $R(\pi) \neq \emptyset$ for only finitely many π. $\mathcal{L}(\mathcal{A})$ is the set of accepted trees.

2.2 Higher-Order Recursion Schemes (HORS)

The set of ***sorts*** is given by: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The ***order*** and ***arity*** of sorts are $\mathtt{ord}(\mathtt{o}) := 0$, $\mathtt{ar}(\mathtt{o}) := 0$ and $\mathtt{ord}(\kappa_1 \to \kappa_2) := \max(\mathtt{ord}(\kappa_1) + 1, \mathtt{ord}(\kappa_2))$ and $\mathtt{ar}(\kappa_1 \to \kappa_2) := 1 + \mathtt{ar}(\kappa_2)$. We fix a *finite* set of ***non-terminals***

$\mathcal{N}$. Each $F \in \mathcal{N}$ is assigned a sort $\mathcal{K}(F)$. We also assume a set of *variables* $\mathcal{V}$ such that each $x \in \mathcal{V}$ is assigned a sort $\mathcal{K}(x)$ and a positive integer $\boldsymbol{pos}(x)$. The latter describes the argument to which it is bound in a HORS rule (defined below). A ***terminal*** $a \in \Sigma$ has sort $\mathcal{K}(a) := \mathtt{o}_1 \to \cdots \to \mathtt{o}_{\mathtt{ar}(a)} \to \mathtt{o}$ The set of ***atoms*** is defined by $\mathbf{At} := \mathcal{N} \uplus \mathcal{V} \uplus \Sigma$. We define a set $\mathbf{AppTerms}$ of ***applicative terms*** by induction and extend $\mathcal{K}$ to each member of this set: $\mathbf{At} \subseteq \mathbf{AppTerms}$ and if $u, v \in \mathbf{AppTerms}$ with $\mathcal{K}(u) = \kappa_1 \to \kappa_2$ and $\mathcal{K}(v) = \kappa_1$, then $(uv) \in \mathbf{AppTerms}$ with $\mathcal{K}((uv)) = \kappa_2$. For $t \in \mathbf{AppTerms}$, $\mathtt{ord}(t) := \mathtt{ord}(\mathcal{K}(t))$ and $\mathtt{ar}(t) := \mathtt{ar}(\mathcal{K}(t))$. $\mathbf{FV}(t)$ is the set of variables in a term t. Define ***the head of*** t by $\boldsymbol{hd}(t) := t$ if $t \in \mathbf{At}$ and $\boldsymbol{hd}(uv) := \boldsymbol{hd}(u)$. Where $t = \boldsymbol{hd}(t)u_1 \cdots u_\ell$, we define $\boldsymbol{w}(t) := \ell$ and $\boldsymbol{args}(t) := (u_1, \ldots, u_\ell)$.

Definition 2 (HORS). *A (deterministic) higher-order recursion scheme (HORS), written $\mathcal{G}$, is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, \mathcal{S})$, where Σ and $\mathcal{N}$ are as above and both:*

1. *$\mathcal{R}$ is a map from each $F \in \mathcal{N}$ to a term $t \in \mathbf{AppTerms}$ such that if $\mathcal{K}(F)$ has the form $\kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$, then $\mathbf{FV}(t) \subseteq \{x_1, \ldots, x_\ell\}$ and $\mathcal{K}(x_i) = \kappa_i$ and $\boldsymbol{pos}(x_i) = i$ for each $1 \leq i \leq \ell$. We write: $Fx_1 \cdots x_\ell \longrightarrow t \in \mathcal{R}$.*
2. *$\mathcal{S} \in \mathcal{N}$ with $\mathcal{K}(\mathcal{S}) = \mathtt{o}$ is the **initial non-terminal**.*

The *order* of a recursion scheme $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ is $max(\{\mathtt{ord}(F) \mid F \in \mathcal{N}\})$. To simplify our algorithm, *without loss of generality we assume that terms in the image of $\mathcal{R}$ only ever contain $a \in \Sigma$ in a context of the form: $au_1 \cdots u_{\mathtt{ar}(a)}$.*

A HORS $\mathcal{G}$ generates its Σ-labelled *value-tree* $\mathbf{Tree}(\mathcal{G})$ by unfolding $\mathcal{S}$. However, we give a slightly non-standard definition of HORS evaluation, based on the presentation in [6]. This is equivalent to other definitions but more convenient for us. Given a HORS $\mathcal{G}$, $t \in \mathbf{AppTerms}$ with $\mathcal{K}(t) = \mathtt{o}$ and $\mathbf{FV}(t) = \emptyset$:

$$t \longrightarrow t'[u_1/x_1, \ldots, u_\ell/x_\ell] \quad \text{if } t = Fu_1 \cdots u_\ell \text{ with } F \in \mathcal{N} \text{ and } Fx_1 \cdots x_\ell \longrightarrow t' \in \mathcal{R}$$

$$t \xrightarrow{(a,i)} u_i \quad \text{if } t = au_1 \cdots u_{\mathtt{ar}(a)} \text{ with } a \in \Sigma \text{ and } 1 \leq i \leq \mathtt{ar}(a)$$

Let $\longrightarrow^*$ be the transitive reflexive closure of $\longrightarrow$. We write $t \xrightarrow{(a_1,i_1)\cdots(a_m,i_m)a} t'$ to mean that there exist terms $t_0, \ldots, t_{2m}$ such that $t \longrightarrow^* t_0$ and for all $0 \leq j \leq m-1$: $t_{2j} \xrightarrow{(a_{j+1},i_{j+1})} t_{2j+1} \longrightarrow^* t_{2j+2}$ and $t_{2m} = t'$ with $\boldsymbol{hd}(t') = a$. The ***value-tree*** $\mathbf{Tree}(\mathcal{G})$ of the HORS $\mathcal{G}$ is the $\Sigma^\perp$-labelled tree:

$$\mathbf{Tree}(\mathcal{G}) : \{\epsilon\} \cup \left\{ i_1 \cdots i_m \;\middle|\; \mathcal{S} \xrightarrow{(\_,i_1)\cdots(\_,i_m)h} \_ \quad \text{for any } h \in \mathbf{At} \right\} \longrightarrow \Sigma^\perp$$

$$\mathbf{Tree}(\mathcal{G})(i_1 \cdots i_m) := \begin{cases} a \text{ where } \mathcal{S} \xrightarrow{(a_1,i_1)\cdots(a_m,i_m)a} \_ \quad \text{and } a \in \Sigma \\ \perp \text{ when there is no } a \in \Sigma \text{ s.t. } \mathcal{S} \xrightarrow{(a_1,i_1)\cdots(a_{m-1},i_m)a} \_ \end{cases}$$

2.3 Product Systems

Henceforth let us fix a co-trivial alternating tree automaton $\mathcal{A}$ and a recursion scheme $\mathcal{G}$. We are interested in algorithms to determine whether $\mathbf{Tree}(\mathcal{G}) \in \mathcal{L}(\mathcal{A})$. This can be reformulated as a reachability problem on a product system:

Definition 3. *For sets $C, C' \subseteq \mathcal{Q} \times \mathbf{AppTerms}$ we write $C \longrightarrow C'$ if for every $(q,t) \in C$ either there exists $(q,t') \in C'$ such that $t \longrightarrow^* t'$ or else there exists a $(q,a,S) \in \Delta_{\mathcal{A}}$ such that for every $(q',i) \in S$ there is some $(q',t') \in C'$ such that $t \xrightarrow{(a,i)} t'$. We again write $\longrightarrow^*$ for the transitive (reflexive) closure of $\longrightarrow$.*

Lemma 1. *$\mathcal{A}$ accepts $\mathbf{Tree}(\mathcal{G})$ if and only if $\{(q_I, \mathcal{S})\} \longrightarrow^* \emptyset$.*

3 Streamlined Collapsible Pushdown Systems

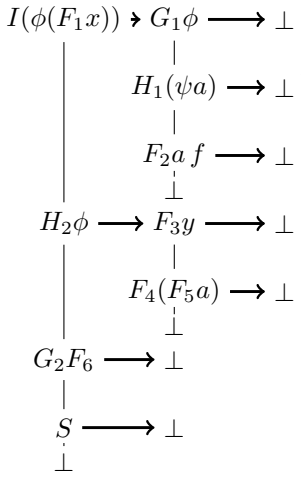


Fig. 1: A stack

Streamlined Collapsible Pushdown Systems restrict collapsible pushdown systems [8] by allowing only the behaviours needed for Carayol and Serre's translation from HORS [6]. They simulate HORS using a form of 'explicit substitution'.

3.1 Stacks

Definition 4. *The set of **argument subterms** of a term t for which $\boldsymbol{args}(t) := (u_1, \ldots, u_{\boldsymbol{w}(t)})$ is: $ArgSTerms(t) := \{t, u_1, \ldots, u_{\boldsymbol{w}(t)}\} \cup \bigcup_{i=1}^{\boldsymbol{w}(t)} ArgSTerms(u_i)$. The **term alphabet** $\mathcal{T}_{\mathcal{G}}$ for $\mathcal{G}$ is: $\mathcal{T}_{\mathcal{G}} := \{\mathcal{S}\} \cup \bigcup_{F \in \mathcal{N}} ArgSTerms(\mathcal{R}(F))$.*

The set of ***stacks*** $\mathbf{Stk}(\mathcal{T}_{\mathcal{G}})$ consists of $\mathcal{T}_{\mathcal{G}}^{\perp}$-labelled trees given in 'term form':

$$\zeta ::= \ \bot \ \mid \ t(\zeta_e, \zeta_\alpha) \qquad \text{where } \zeta_e, \zeta_\alpha \in \mathbf{Stk}(\mathcal{T}_{\mathcal{G}}) \text{ and } t \in \mathcal{T}_{\mathcal{G}}$$

A stack $t(\zeta_e, \zeta_\alpha)$ is represented graphically with ζ_e written below t, joined via a vertical line, and ζ_α written to its right and joined with a horizontal arrow (*e.g.* in Figure 1). Define $top(\bot) := \bot$ and $top(t(\zeta_e, \zeta\alpha)) := t$ and operations:

$$push_t(\zeta) := t(\zeta, \bot) \qquad rew^{\xi}_{t'}(t(\zeta_e, \zeta_\alpha)) := t'(\zeta_e, \xi)$$
$$pop(t(\zeta_e, \zeta_\alpha)) := \zeta_e \qquad collapse(t(\zeta_e, \zeta_\alpha)) := \zeta_\alpha$$

CPDS-Remark 1 *A stack represents the top-most order-1 stack of a CPDS stack according to the 'annotated stack' presentation in [3, 10]. An SCPDS only performs $\theta_1 \cdots \theta_m rew^{\xi}_{t'}$ on ζ when $\xi = \zeta$ and the CPDS does $push_k \theta_1 \cdots \theta_m pop_1 push^k_{t'}$ (identifying '$pop_1 = pop$') where $\theta_1 \ldots \theta_m$ never discard the top_k stack.*

3.2 Term Extraction and Ground-Sorted Stacks

A stack $\zeta \in \mathbf{Stk}(\mathcal{T}_{\mathcal{G}})$ may represent a term $[\![\zeta]\!]_0$. The idea is that $t(\zeta_e, \zeta_\alpha)$ represents a term formed from the 'template' t by replacing free variables in t with terms derived from ζ_e and applying it to terms derived from ζ_α. The following

operation 'locates bindings' in a stack ζ (the notation used here is reminiscent of [18]).

$$\mathit{BindLoc}(i,\zeta) := \begin{cases} (t_i,\zeta) \text{ if } \boldsymbol{args}(\mathit{top}(\zeta)) = (t_1,\ldots,t_{\boldsymbol{w}(\mathit{top}(\zeta))}) \text{ and } 0 < i \leq \boldsymbol{w}(\mathit{top}(\zeta)) \\ \mathit{BindLoc}(i - \boldsymbol{w}(\mathit{top}(\zeta)), \mathit{collapse}(\zeta)) \text{ if } i > \boldsymbol{w}(\mathit{top}(\zeta)) \end{cases}$$

Consider a stack ζ with $\mathit{top}(\zeta) = t \neq \bot$ and $\boldsymbol{hd}(t) = h$.

$$[\![\zeta]\!]_0 := \begin{cases} h[\![\zeta]\!]_1 \cdots [\![\zeta]\!]_m & \text{if } h \in \mathcal{N} \cup \Sigma \\ [\![\mathit{pop}(\zeta)]\!]_{\boldsymbol{pos}(\phi)}[\![\zeta]\!]_1 \cdots [\![\zeta]\!]_m & \text{if } \phi = h \in \mathcal{V} \end{cases}$$

where $m := \min(\mathtt{ar}(h), m')$ where m' is the maximum j s.t. $[\![\zeta]\!]_j$ is defined:

$$[\![\zeta]\!]_j := [\![\mathit{rew}^{\bot}_{t'}(\zeta')]\!]_0 \text{ where } \mathit{BindLoc}(j,\zeta) = (t',\zeta') \text{ for } j \geq 1$$

A stack ζ is ***ground-sorted*** if $[\![\zeta]\!]_0$ is defined with $\mathcal{K}([\![\zeta]\!]_0) = \mathtt{o}$. Figure 1 gives a stack ζ such that $[\![\zeta]\!]_0 = I(F_6(F_1(F_5 a)))(fa)$ (where $\boldsymbol{pos}(\phi) = 1$, $\boldsymbol{pos}(y) = 1$, $\boldsymbol{pos}(x) = 2$ and $\boldsymbol{pos}(\psi) = 2$).

3.3 An SCPDS for the HORS and its Configurations

The ***SCPDS*** $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$ is defined in terms of the APT $\mathcal{A}$ and the HORS $\mathcal{G}$. The ***configurations*** $\mathsf{Config}(\mathsf{KA}_{\mathcal{G},\mathcal{A}})$ of $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$ are pairs $(q,\zeta) \in \mathcal{Q} \times \mathsf{Stk}(\mathcal{T}_{\mathcal{G}})$ consisting of a ***control-state*** (state of the property automaton) and a ***stack***. Since stacks represent terms, $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$ has alternating runs corresponding to runs of the 'product of $\mathcal{A}$ and $\mathcal{G}$' as in Lemma 1.

There is a family of transition relations for $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$. The transition relation applied to a particular configuration is determined by the head symbol of the top element of the stack. First non-terminals and variables:

$$(q,\zeta) \xrightarrow{\mathcal{N}} (q, \mathit{push}_{t'}(\zeta)) \text{ if } \boldsymbol{hd}(\mathit{top}(\zeta)) \in \mathcal{N} \text{ with } \boldsymbol{hd}(\mathit{top}(\zeta)) \longrightarrow t' \in \mathcal{R}$$

$$(q,\zeta) \xrightarrow{\mathcal{V}} (q, \mathit{rew}^{\zeta}_{t'}(\zeta')) \text{ if } \boldsymbol{hd}(\mathit{top}(\zeta)) \in \mathcal{V} \text{ with } \boldsymbol{pos}(\boldsymbol{hd}(\mathit{top}(\zeta))) = i \text{ where } \mathit{BindLoc}(i, \mathit{pop}(\zeta)) = (t',\zeta')$$

To capture alternation in $\mathcal{A}$, the $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$ transitions for terminals go to *sets*:

$$(q,\zeta) \xrightarrow{\Sigma} \{\ (q', \mathit{rew}^{\bot}_{t_i}(\zeta))\ \mid (q',i) \in S\ \}$$

whenever $\mathit{top}(\zeta) = a t_1 \cdots t_{\mathtt{ar}(a)}$ for $a \in \Sigma$ and $(q,a,S) \in \Delta_{\mathcal{A}}$.

A $\xrightarrow{\mathcal{N}}$ transition 'calls a non-terminal' by pushing the right-hand-side of the appropriate rule from $\mathcal{G}$ onto the stack. The stack will now represent the term that would result from applying this rule to the term represented by the original stack. A $\xrightarrow{\mathcal{V}}$ transition 'calls a variable' by locating the binding of a variable in head-position whilst 'remembering the original stack' as an annotation of the top element on the new stack. This is necessary in case the variable has a function sort and the term bound to the variable later calls one of its arguments. In particular the stack before a $\xrightarrow{\mathcal{V}}$ transition represents the same term as the stack after it. We illustrate these transitions in Figure 3 (ignore the symbols in **boldface**).

Definition 5. *Given* $C, C' \subseteq \mathsf{Config}(\mathsf{KA}_{\mathcal{G},\mathcal{A}})$ *define* $C \longrightarrow C'$ *to mean for every* $c \in C$ either *(i)* $c \in C'$ or *(ii) there exists* $D_c \subseteq C'$ *such that either* $c \xrightarrow{\Sigma} D_c$*, or else* $D_c = \{c'\}$ *with* $c \xrightarrow{\mathcal{N}} c'$ *or* $c \xrightarrow{\mathcal{V}} c'$*.* $\longrightarrow^*$ *is reflexive trans. closure of* $\longrightarrow$*.*

Let us define the ***initial configuration*** $c_0 := (q_I, \mathcal{S}(\bot, \bot))$.

A configuration (q, ζ) is ***ground-sorted*** if ζ is ground-sorted. Let GS be the set of ground-sorted configurations. For $C \subseteq \mathsf{GS}$, $[\![C]\!] := \{\ (q, [\![\zeta]\!]_0)\ \mid (q, \zeta) \in C\ \}$.

Theorem 1. *Given* $C \subseteq \mathsf{GS}$*,* $C \longrightarrow^* C'$ *implies that* $C \longrightarrow^* C' \cap \mathsf{GS}$*. Moreover, given* $C, C' \subseteq \mathsf{GS}$*, if* $C \longrightarrow^* C'$*, then* $[\![C]\!] \longrightarrow^* [\![C']\!]$*. Conversely, if* $[\![C]\!] \longrightarrow^* D$*, then there exists* $C' \subseteq \mathsf{GS}$ *such that* $[\![C']\!] \subseteq D$ *and* $C \longrightarrow^* C'$*. Thus by Lemma 1* $\mathcal{A}$ *accepts* $\mathbf{Tree}(\mathcal{G})$ *iff* $\{c_0\} \longrightarrow^* \emptyset$*.*

CPDS-Remark 2 $\mathsf{KA}_{\mathcal{G},\mathcal{A}}$ *works similarly to the CPDS implementing* $\mathcal{G}$ *from the translation in [6].* $BindLoc(i, \zeta)$ *corresponds to multiple CPDS operations involving control-states of the form* (q, i) *keeping track of* i *during the recursion.*

4 A Saturation Algorithm for SCPDS

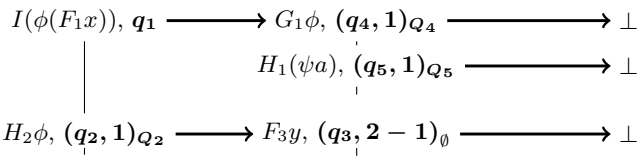
We give a saturation algorithm for SCPDS that computes a *'stack automaton'* (an automaton recognising sets of SCPDS configurations, defined in the next subsection) recognising the set $\mathsf{Pre}^{\star}_{\mathsf{KA}_{\mathcal{G},\mathcal{A}}} := \{\ c \in \mathsf{GS}\ \mid c \longrightarrow^* \emptyset\ \}$. By Lemma 1 and Theorem 1, this is the set of configurations 'generating a tree' accepted by $\mathcal{A}$. We start with a stack-automaton $\boldsymbol{\Delta}_{\mathsf{Acc}}$ recognising the set of configurations c such that $c \longrightarrow \emptyset$ (in a single step). Following the idea common to saturation algorithms, we iteratively add transitions to this stack-automaton to recognise configurations that could reach in a single step a configuration accepted by the previous version of the automaton. This continues until a fixpoint is reached.

SCPDS saturation is a specialisation of the work in [3]. Neither the soundness nor completeness of our refined algorithm follows immediately from *op. cit.* since one would need to check that the 'restriction' of the algorithm is precisely what is required. Nevertheless the argument for completeness is fairly standard, whilst the soundess proof is based on the correspondence with intersection types for HORS. The latter is arguably much simpler than the analogous proof in [3].

4.1 Stack Automata

Stack-automata are a subclass of alternating tree automata acting on stacks and constitute a refined version of the stack-automata used by the CPDS saturation algorithm [3]. The states of a stack-automaton consist of the elements of $\mathcal{Q}$ together with states of the form $(q, i)_Q$ where $q \in \mathcal{Q}$, Q is a set of states and $1 \leq i \leq r$ (where as before r is the maximum arity of any subterm of $\mathcal{G}$).

An automaton is defined by a set of transitions Δ that begin with a state in $\mathcal{Q}$ and propagate sets of states of the form $(q, i)_Q$ down each child of a node in the stack. Transitions beginning from a state of the form $(q, i)_Q$ are *derived* from this set of transitions. This allows us to enforce a 'consistency' of stack-automaton



$\{(q_1, I(\phi(F_1x))), \{(q_2, 1)_{Q_2}, (q_3, 2)_\emptyset\}, \{(q_4, 1)_{Q_4}\})$ and $(q_4, \phi, \{(q_5, 1)_{Q_5}\}, Q_4)\} \subseteq \Delta$

Fig. 2: Part of a 'run' on the stack from Figure 1 of a stack-automaton.

transitions giving rise to a natural interpretation in terms of types; they streamline CPDS stack automata to better fit types just as SCPDS streamline CPDS to better fit HORS. We define $\mathbf{Q}(\mathcal{Q}, r) := \bigcup_{j\in\omega} \mathbf{Q}_j(\mathcal{Q}, r)$ where:

$$\mathbf{Q}_1(\mathcal{Q}, r) := [1, r] \ \times \ (\mathcal{Q} \ \times \ \emptyset) \qquad \mathbf{Q}_{j+1}(\mathcal{Q}, r) := [1, r] \ \times \ \left(\mathcal{Q} \ \times \ 2^{\mathbf{Q}_j(\mathcal{Q},r)}\right) \ \cup \ \mathbf{Q}_j(\mathcal{Q}, r)$$

We write elements of $(i, (q, Q)) \in \mathbf{Q}(\mathcal{Q}, r)$ as $(q, i)_Q$.

A ***stack-automaton*** for $\mathbf{KA}_{\mathcal{G},\mathcal{A}}$ is defined by its transition relation Δ:

$$\Delta \ \subseteq_{\mathit{finite}} \ \mathcal{Q} \ \times \ \mathcal{T}_\mathcal{G} \ \times \ 2^{\mathbf{Q}(\mathcal{Q},r)} \ \times \ 2^{\mathbf{Q}(\mathcal{Q},r)}$$

Similarly to CPDS stack automata [3], we write (q, t, Q_e, Q_α) as $q \xrightarrow{t\ Q_\alpha} Q_e$.

Intuitively we want the automaton to accept from the state $(q, i)_Q$ those stacks ζ such that $\mathit{BindLoc}(i, \zeta) = (t', \zeta')$ and $\mathit{rew}^\xi_{t'}(\zeta')$ is accepted from q if ξ is accepted from every state in Q. This is ensured by the full set of transitions $\Delta^\star$ ***induced*** by Δ:

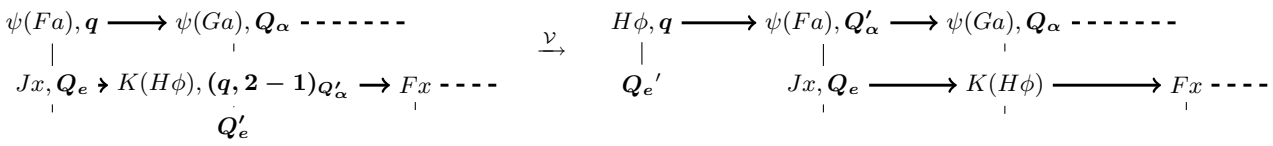
$$\Delta^\star := \Delta \ \cup \ \left\{ \ (q, i)_Q \ \xrightarrow{h\, t_1 \cdots t_\ell\ \emptyset} \ Q_e \ \middle| \ 1 \le i \le \ell \text{ and } q \ \xrightarrow{t_i\ Q} \ Q_e \in \Delta \ \right\}$$
$$\cup \ \left\{ \ (q, i)_Q \ \xrightarrow{h\, t_1 \cdots t_\ell\ \{(q, i-\ell)_Q\}} \ \emptyset \ \mid r \ge i > \ell \text{ and } (q, i)_Q \in \mathbf{Q}(\mathcal{Q}, r) \ \right\}$$

The set of stacks recognised by Δ from a state $p \in \mathcal{Q} \ \cup \ \mathbf{Q}(\mathcal{Q}, r)$ is defined to be the language recognised by $\Delta^\star$ viewed as an alternating tree automaton:

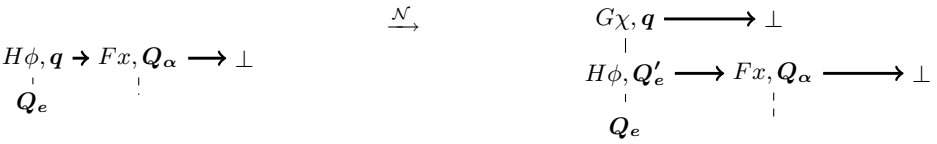
$$\mathcal{L}(\Delta)_p := \left\{ \begin{array}{l} t(\zeta_e, \zeta_\alpha) \\ \in \mathbf{Stk}(\mathcal{T}_\mathcal{G}) \end{array} \middle| \begin{array}{l} \exists p \ \xrightarrow{t\ Q_\alpha} \ Q_e \in \Delta^\star \text{ s.t. } \zeta_e \in \mathcal{L}(\Delta)_{p_e} \text{ and} \\ \zeta_\alpha \in \mathcal{L}(\Delta)_{p_\alpha} \text{ for every } p_e \in Q_e \text{ and every } p_\alpha \in Q_\alpha \end{array} \right\}$$

The set recognised by Δ is: $\mathcal{L}(\Delta) := \{ \ (q, \zeta) \ \mid q \in \mathcal{Q} \text{ and } \zeta \in \mathcal{L}(\Delta)_q \ \}$.

CPDS-Remark 3 *The states of a CPDS encoding a HORS are of the form q and (q, i) (with $q \in \mathcal{Q}$ and $1 \le i \le r$). The states of the* CPDS *stack automaton that read an 'order-1 stack' are of the form $q_{Q_n,\ldots,Q_2}$ or $(q, i)_{Q_n,\ldots,Q_2}$ where Q_j is a set of states analogous to our $\mathbf{Q}_{n-j+1}(\mathcal{Q}, r)$. The CPDS saturation algorithm needs states of this form as it must handle the possibility of the arbitrary creation of order-k links (for $2 \le k \le n$) and arbitrary pop_k operations. Since the*



(a) A 'variable call' step where $\boldsymbol{pos}(\psi) = 2$



(b) A 'non-terminal call' where $H\chi z \longrightarrow G\chi$

Fig. 3: Illustration of 'evaluation steps' of a stack and stack-automata runs

behaviour of SCPDS is much more structured, our SCPDS saturation algorithm is correct despite our simpler stack automata. Indeed the lack of spurious information would have performance benefits. One can also view $\Delta^\star$ as being formed from Δ by applying the CPDS saturation steps for rew and collapse.

4.2 Saturation

Using suggestively similar notation to [3], for $Q \subseteq \mathbf{Q}(\mathcal{Q}, r)$ we write $Q \xrightarrow[\Delta^\star]{t\ Q_\alpha} Q_e$ to mean that for each $p \in Q$ there exists $p \xrightarrow{t\ Q^p_\alpha} Q^p_e \in \Delta^\star$ such that $Q_\alpha = \bigcup_{p\in Q} Q^p_\alpha$ and $Q_e = \bigcup_{p\in Q} Q^p_e$.

Example 1. Suppose that $(q_1, t_1, Q_{\alpha 1}, Q_{e1}), (q_2, t_2, Q_{\alpha 2}, Q_{e2}) \in \Delta$. Then

$$\{(q_1, 1)_{Q_{\alpha 1}}, (q_2, 2)_{Q_{\alpha 2}}, (q_3, 3)_{Q_{\alpha 3}}\} \xrightarrow[\Delta]{F\, t_1\, t_2\ \{(q_3,1)_{Q_{\alpha 3}}\}} Q_{e1} \cup Q_{e2} \text{ when } F \text{ is atomic.}$$

Given a stack automaton Δ we define the following ***saturation steps*** $\boldsymbol{\Phi}^V, \boldsymbol{\Phi}^{\mathcal{N}}, \boldsymbol{\Phi}^\Sigma$, corresponding to each kind of transition an SCPDS can make. Each extends Δ to accept configurations that can reach $\mathcal{L}(\Delta)$ by a single $\xrightarrow{\mathcal{V}}$, $\xrightarrow{\mathcal{N}}$, $\xrightarrow{\Sigma}$ transition.

$$\boldsymbol{\Phi}^V(\Delta) := \Delta \cup \{q \xrightarrow{t\ Q_\alpha} \{(q, \boldsymbol{pos}(x))_{Q'_\alpha}\} \cup Q_e \mid q \xrightarrow[\Delta]{t'\ Q'_\alpha} Q'_e \text{ and } Q'_\alpha \xrightarrow[\Delta]{t\ Q_\alpha} Q_e \text{ for some } t' \in \mathcal{T}_\mathcal{G} \text{ with } \boldsymbol{hd}(t) = x \in \mathcal{V} \text{ and } \mathcal{K}(x) = \mathcal{K}(t')\}$$

$$\boldsymbol{\Phi}^{\mathcal{N}}(\Delta) := \Delta \cup \{q \xrightarrow{t\ Q_\alpha} Q_e \mid q \xrightarrow[\Delta]{t'\ \emptyset} Q'_e \xrightarrow[\Delta]{t\ Q_\alpha} Q_e \text{ where } \boldsymbol{hd}(t) = F \in \mathcal{N} \text{ and } Fx_1 \dots x_{\mathtt{ar}(F)} \longrightarrow t' \in \mathcal{R}\}$$

$$\boldsymbol{\Phi}^\Sigma(\Delta) := \Delta \cup \{q \xrightarrow{at_1\cdots t_{\mathtt{ar}(a)}\ \emptyset} \bigcup_{(q',i)\in S} Q_e^{(q',i)} \mid (q, a, S) \in \Delta_\mathcal{A} \text{ and there exist } q' \xrightarrow[\Delta]{t_i\ \emptyset} Q_e^{(q',i)} \text{ for some family of sets } Q_e^{(q',i)} \text{ where } (q', i) \text{ ranges over } S\}$$

$\boldsymbol{\Phi}(\Delta) := \boldsymbol{\Phi}^{\mathcal{N}}(\Delta) \cup \boldsymbol{\Phi}^{V}(\Delta) \cup \boldsymbol{\Phi}^{\Sigma}(\Delta)$ and $\boldsymbol{\Delta}_{\mathbf{Acc}} := \{\ (q, t, \emptyset, \emptyset)\ \mid (q, \boldsymbol{hd}(t), \emptyset) \in \Delta_{\mathcal{A}}\ \}$.

Figure 3 illustrates the idea for the $\boldsymbol{\Phi}^{V}$ and $\boldsymbol{\Phi}^{\mathcal{N}}$ steps. The stack on the right-hand side of the $\xrightarrow{\mathcal{V}}$ is annotated in **boldface** with the (sets of) states that are propagated by transitions $q \xrightarrow[\Delta]{H\phi\ Q'_\alpha} Q'_e$ and $Q'_\alpha \xrightarrow[\Delta]{\psi(Fa)\ Q_\alpha} Q_e$ in some stack-automaton Δ. The stack on the left-hand side of $\xrightarrow{\mathcal{V}}$ is annotated with (sets of) states illustrating how it must be accepted by $\boldsymbol{\Phi}^{V}(\Delta)$ due to its transition $q \xrightarrow[\boldsymbol{\Phi}^{V}(\Delta)]{\psi(Fa)\ Q_\alpha} Q_e \cup \{(q,2)_{Q'_\alpha}\}$ (we assume $\boldsymbol{pos}(\psi) = 2$). Recall that $q \xrightarrow[\Delta]{H\phi\ Q'_\alpha} Q'_e$ gives (in $\Delta^\star$) transitions $(q,2)_{Q'_\alpha} \xrightarrow[\Delta]{Jx\ \{(q,2-1)\}} \emptyset$ and $(q,1)_{Q'_\alpha} \xrightarrow[\Delta]{K(H\phi)\ \emptyset} Q'_e$. These transitions must also belong to $\boldsymbol{\Phi}^{V}(\Delta)^\star$.

The $\xrightarrow{\mathcal{N}}$ example gives an analogous illustration of $\boldsymbol{\Phi}^{\mathcal{N}}$.

Definition 6. *The **SCPDS-saturation algorithm** repeatedly applies $\boldsymbol{\Phi}$ to $\Delta_0 := \boldsymbol{\Delta}_{\mathbf{Acc}}$ thereby computing a sequence of automata $\Delta_{i+1} := \boldsymbol{\Phi}(\Delta_i)$ until a fixpoint $\Delta_{m+1} = \Delta_m$ is reached, which we denote by $\boldsymbol{\Phi}^* := \Delta_m$.*

Theorem 2. *The algorithm terminates and $\mathcal{L}(\boldsymbol{\Phi}^*) \cap \mathsf{GS} = \mathsf{Pre}^\star_{\mathsf{KA}_{\mathcal{G},\mathcal{A}}}$. Hence, by Theorem 1, $\mathcal{A}$ accepts $\mathbf{Tree}(\mathcal{G})$ if and only if $c_0 \in \mathcal{L}(\boldsymbol{\Phi}^*)$.*

$\mathsf{Pre}^\star_{\mathsf{KA}_{\mathcal{G},\mathcal{A}}} \subseteq \mathcal{L}(\boldsymbol{\Phi}^*)$ comes from $\boldsymbol{\Delta}_{\mathbf{Acc}} \subseteq \boldsymbol{\Phi}^{V}(\boldsymbol{\Phi}^*) = \boldsymbol{\Phi}^{\mathcal{N}}(\boldsymbol{\Phi}^*) = \boldsymbol{\Phi}^{\Sigma}(\boldsymbol{\Phi}^*)$ and the correctness of the intuition above about $\boldsymbol{\Delta}_{\mathbf{Acc}}$ and saturation steps. $\mathcal{L}(\boldsymbol{\Phi}^*) \subseteq \mathsf{Pre}^\star_{\mathsf{KA}_{\mathcal{G},\mathcal{A}}}$. Termination follows from a correspondence with types considered next.

5 Type-Based Interpretation

Δ can represent intersection typing judgements for elements of $\mathcal{T}_{\mathcal{G}}$. $\Delta^\star$ then allows one to build up typing judgements for more complex terms using the assumptions in Δ in a manner that is sound relative to Kobayashi's type system. We reinterpret the algorithm as building up *typing judgements* rather than transitions, yielding a soundness and termination proof. It also means our algorithm unifies a pushdown-saturation approach with a type-based approach—the same algorithm can be interpreted in both ways.

5.1 Review of Kobayashi's Types

We now recall Kobayashi's type-based characterisation of the model-checking problem. Whilst [12] originally considered types describing acceptance by trivial automata, one can also consider types with the dual interpretation [5] corresponding to the co-trivial acceptance condition, which is what we employ here.

The intersection types $\mathbf{ITypes}_{\mathcal{Q}}$ is constructed from the states $\mathcal{Q}$ of $\mathcal{A}$:

$$\tau ::= q \mid \sigma \to \tau \quad \text{with } \sigma ::= \bigwedge\{\tau_1, \ldots, \tau_k\}.$$

Here, q ranges over a set $\mathcal{Q}$ of base types. The ***well-sorted*** types τ are those such that $\tau :: \kappa$ for some sort κ where $q :: \mathtt{o}$ for $q \in \mathcal{Q}$ and if $\tau_1 :: \kappa, \ldots, \tau_k :: \kappa$ and $\tau :: \kappa'$, then $\bigwedge\{\tau_1, \ldots, \tau_k\} \to \tau :: \kappa \to \kappa'$.

We use the following rules, which require each $t : \tau$ to satisfy $\tau :: \mathcal{K}(t)$.

$$\Gamma, x : \tau \vdash_{\mathcal{A}} x : \tau \quad \text{(T-Var)} \qquad \frac{(q, a, \{(q_j, i) \mid 1 \leq i \leq \Sigma(a), j \in I_i\}) \in \Delta_{\mathcal{A}}}{\Gamma \vdash_{\mathcal{A}} a : \bigwedge_{j \in I_1} q_j \to \cdots \to \cdots \bigwedge_{j \in I_{\Sigma(a)}} q_j \to q} \text{(T-Const)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} t_1 : \bigwedge_{i \in I} \tau_i \to \tau \qquad \Gamma' \vdash_{\mathcal{A}} t_2 : \tau_i \text{ (for all } i \in I)}{\Gamma \ \cup \ \Gamma' \vdash_{\mathcal{A}} t_1 t_2 : \tau} \quad \text{(T-App)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} t : q \qquad F x_1 \cdots x_k \longrightarrow t \in \mathcal{R}}{\emptyset \vdash_{\mathcal{A}} F : \bigwedge \{\ \tau_1 \mid x_1 : \tau_1 \in \Gamma\ \} \to \cdots \to \bigwedge \{\ \tau_k \mid x_k : \tau_k \in \Gamma\ \} \to q} \text{(T-NT)}$$

Theorem 3 ([5]). $\mathbf{Tree}(\mathcal{G}) \in \mathcal{L}(\mathcal{A})$ *if and only if* $\emptyset \vdash_{\mathcal{A}} S : q_I$.

5.2 Type Extraction

Let us return to stack automata. For each sort $\kappa = \kappa_1 \to \kappa_2 \to \cdots \kappa_k \to \mathtt{o}$, $q \in \mathcal{Q}$ and $Q \subseteq \mathbf{Q}(\mathcal{Q}, r)$ such that $(\tau, j) \in Q$ implies $1 \leq j \leq k$, we define $[\![q, \emptyset]\!]_{\mathtt{o}} = q$ and $[\![q, Q]\!]_{\kappa} := \sigma_1 \to \sigma_2 \to \cdots \sigma_k \to q$ to each where $\sigma_j := \bigwedge_{(q_j, j)_{Q_j} \in Q} [\![q_j, Q_j]\!]_{\kappa_j}$.

Definition 7. *We say that a stack-automaton transition $(q, t, Q_e, Q_\alpha) \in \Delta$ is* ***well-sorted*** *if $[\![q, Q_\alpha]\!]_{\mathcal{K}(t)}$ is defined and for every $(q', i)_Q \in Q_e$ there is a variable x occurring in t such that $\mathbf{pos}(x) = i$ and $[\![q', Q]\!]_{\mathcal{K}(x)}$ is defined. A stack automaton Δ is* ***well-sorted*** *if every transition therein is well-sorted.*

It is easy to see that every well-sorted stack-automaton transition (q, t, Q_e, Q_α) can be associated with a well-defined (but maybe incorrect) type-judgement:

$$[\![(q, t, Q_e, Q_\alpha)]\!] := \left\{ x : [\![q_e, Q]\!]_{\mathcal{K}(x)} \;\middle|\; \begin{array}{l} x \in \mathbf{FV}(t) \text{ and} \\ (q_e, \mathbf{pos}(x))_Q \in Q_e \end{array} \right\} \Vdash_{\mathcal{A}} t : [\![q, Q_\alpha]\!]_{\mathcal{K}(t)}$$

(see below for rules of $\Vdash_{\mathcal{A}}$). For a well-sorted Δ we define $[\![\Delta]\!] := \{\ [\![J]\!] \mid J \in \Delta\ \}$.

5.3 The Type-Based Algorithm

We will write $\Gamma \Vdash_{\mathcal{A}} t : \tau$ to indicate a ***typing judgement*** derivable by the rules in Figure 4, which we restrict to terms $t \in \mathcal{T}_{\mathcal{G}}$. We mark the turnstyle of a premise with a tilde $\Vdash^{\sim}_{\mathcal{A}}$ when it is not required for soundness but which nevertheless turns out to be imposed by the saturation algorithm.

We write $\boldsymbol{\nabla}_{\mathbf{Acc}}$ to denote the set of typing judgements that are valid due to the rule (Accept). Given a set of judgements $\boldsymbol{\nabla}$, we define $\boldsymbol{\Psi}^{V}(\boldsymbol{\nabla})$, $\boldsymbol{\Psi}^{\mathcal{N}}(\boldsymbol{\nabla})$ and $\boldsymbol{\Psi}^{\Sigma}(\boldsymbol{\nabla})$ for the smallest set containing $\boldsymbol{\nabla}$ and all judgements derivable from $\boldsymbol{\nabla}$ using respectively a single instance of the rule (Head-Var), (Rec) or (Tran).

$$\frac{(q, a, \emptyset) \in \Delta_{\mathcal{A}} \qquad \boldsymbol{hd}(t) = a}{\Vdash_{\mathcal{A}} \ t : q} \text{ (Accept)}$$

$$\frac{\forall j \in [1,l].\ \forall \tau_j \in T_j.\ \exists \Gamma_j^{\tau_j}.\ \Gamma_j^{\tau_j} \ \Vdash_{\mathcal{A}} \ t_j : \tau_j \qquad \begin{array}{c}\text{There exist } \Gamma \text{ and } t \text{ s.t. } \mathcal{K}(t) = \mathcal{K}(\phi) \text{ and} \\ \Gamma \Vdash^{\sim}_{\mathcal{A}} t : \bigwedge T_1 \to \bigwedge T_2 \to \cdots \to \bigwedge T_l \to \tau\end{array}}{\bigcup_{j \in [1,l], \tau_j \in T_j} \Gamma_j^{\tau_j},\ \phi : \bigwedge T_1 \to \bigwedge T_2 \to \cdots \to \bigwedge T_l \to \tau \ \Vdash_{\mathcal{A}} \ \phi t_1 \cdots t_l : \tau} \text{ (Head-Var)}$$

$$\frac{F x_1 \cdots x_{\mathtt{ar}(F)} \longrightarrow t \quad \in \mathcal{R} \qquad \Gamma \ \Vdash_{\mathcal{A}} \ t : q \qquad \forall i \in [1,l].\ \forall x_i : \tau \in \Gamma.\ \exists \Gamma_i^{\tau}.\ \Gamma_i^{\tau} \ \Vdash_{\mathcal{A}} \ t_i : \tau}{\bigcup_{i \in [1,l], x_i : \tau \in \Gamma} \Gamma_i^{\tau} \ \Vdash_{\mathcal{A}} \ F t_1 \cdots t_l : \bigwedge \{\ \tau \mid x_{l+1} : \tau \in \Gamma\ \} \to \cdots \to \bigwedge \{\ \tau \mid x_{\mathtt{ar}(F)} : \tau \in \Gamma\ \} \to q} \text{ (Rec)}$$

$$\frac{(q, a, S) \in \Delta_{\mathcal{A}} \qquad \text{for each } 1 \le i \le \mathtt{ar}(a) \ \text{ and every } \ (q,i) \in S \quad \exists \Gamma_{(q,i)}.\ \Gamma_{(q,i)} \Vdash_{\mathcal{A}} t_i : q}{\bigcup_{(q,i) \in S} \Gamma_{(q,i)} \Vdash_{\mathcal{A}} \ a t_1 \cdots t_{\mathtt{ar}(a)} : q} \text{ (Tran)}$$

Fig. 4: Rules for $\Vdash_{\mathcal{A}}$ (for (Head-Var) $0 \le l \le \mathtt{ar}(\phi)$ and for (Rec) $0 \le l \le \mathtt{ar}(F)$)

Theorem 4 (Type Correspondence). *If Δ is well-sorted, then so are $\boldsymbol{\Phi}^V(\Delta)$, $\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)$ and $\boldsymbol{\Phi}^{\Sigma}(\Delta)$. Moreover $[\![\boldsymbol{\Phi}^V(\Delta)]\!] = \boldsymbol{\Psi}^V([\![\Delta]\!])$, $[\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!] = \boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!])$, and $[\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!] = \boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!])$. Also, $\boldsymbol{\Delta}_{\mathsf{Acc}}$ is well-sorted with $[\![\boldsymbol{\Delta}_{\mathsf{Acc}}]\!] = \boldsymbol{\nabla}_{\mathsf{Acc}}$.*

This theorem tells us that SCPDS saturation can equivalently be seen as repeatedly applying $\Psi(\boldsymbol{\nabla}) := \boldsymbol{\Psi}^V(\boldsymbol{\nabla}) \cup \boldsymbol{\Psi}^{\mathcal{N}}(\boldsymbol{\nabla}) \cup \boldsymbol{\Psi}^{\Sigma}(\boldsymbol{\nabla})$ until a fixpoint $\boldsymbol{\Psi}^*$ is reached. Since there are only finitely many types of a given sort this implies termination.

The algorithm above can also be viewed as a simplified version of HorSat [5] that ignores various optimisations. However, note it is slightly less naïve than naïvest possible type-based algorithm, which would blindly compute *all* valid type judgements. SCPDS saturation restricts its search space slightly with the guard ($\Vdash^{\sim}_{\mathcal{A}}$) on the (Head-Var) rule, which constitutes a rough approximate emptiness check on a type before weakening is allowed. The algorithm also has 'built in' the obvious 'memoisation'—previously derived judgements are retained.

5.4 Soundness

We add two further typing rules that capture (over-approximately) 'runs' of a stack automaton. For a stack $t(\zeta_e, \zeta_\alpha)$, (Cut) corresponds to sending states down ζ_e and (App) corresponds to sending states down ζ_α.

$$\frac{\Gamma \Vdash_{\mathcal{A}} v : \tau' \ \ (\mathbf{FV}(v) = \{x_1, \ldots, x_\ell\}) \qquad \Vdash_{\mathcal{A}} u_i : \tau \ \text{ for } x_i : \tau \in \Gamma}{\Vdash_{\mathcal{A}} v[u_i/x_i]_{i \in [1,\ell]} : \tau'} \text{ (Cut)}$$

$$\frac{\Vdash_{\mathcal{A}} u : \bigwedge T \to \tau' \qquad \Vdash_{\mathcal{A}} v : \tau \ (\text{for all } \tau \in T)}{\Vdash_{\mathcal{A}} (uv) : \tau'} \text{ (App)}$$

All six rules are sound relative to Kobayashi's type system for $\mathcal{A}$.

Lemma 2 (Relative Soundness). *If $\Gamma \Vdash_{\mathcal{A}} t : \tau$ then $\Gamma \vdash t : \tau$*

From this and the following lemma it is then possible to derive $\mathcal{L}(\boldsymbol{\Phi}^*) \subseteq \mathsf{Pre}^{\star}_{\mathsf{KA}_{\mathcal{G},\mathcal{A}}}$.

Lemma 3. *If $(q, \zeta) \in \mathcal{L}(\boldsymbol{\Phi}^*) \cap \mathsf{GS}$, then $\Vdash_{\mathcal{A}} [\![\zeta]\!]_0 : q$.*

6 Conclusion

We have given an account of saturation for higher-order model-checking in terms that make sense both from the perspective of type-based algorithms working directly on HORS as well as from the automata-theoretic perspective. We hope this will contribute towards greater synergy between these two methodologies.

References

1. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, pages 135–150, 1997.
2. A. Bouajjani and A. Meyer. Symbolic Reachability Analysis of Higher-Order Context-Free Processes. In *FSTTCS*, volume 3328 of *LNCS*, 2005, Springer Pub.
3. C. H. Broadbent, A. Carayol, M. Hague, and O. Serre. A saturation method for collapsible pushdown systems. In *ICALP* , *LNCS* 7392, pages 165–176. Springe-Verlag, 2012.
4. C. H. Broadbent, A. Carayol, M. Hague, and O. Serre. A collapsible approach to verifying higher-order programs. submitted to ICFP, 2013, http://cshore.cs.rhul.ac.uk
5. C. H. Broadbent and N. Kobayashi. Saturation-based model checking of higher-order recursion schemes. Submitted to CSL, 2013. (http://www-kb.is.s.u-tokyo.ac.jp/∼koba/horsat/)
6. A. Carayol and O. Serre. Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In *LICS*, IEEE, 2012.
7. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. In *INFINITY*, volume 9, pages 27–37, 1997.
8. M. Hague, A. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *LICS*, IEEE Computer Society, 2008.
9. M. Hague and C.-H. L. Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. *LMCS*, 4(4), 2008.
10. A. Kartzow and P. Parys. Strictness of the collapsible pushdown hierarchy. In *Proceedings of MFCS 2012*, volume 7464 of *LNCS*, pages 566–577. Springer, 2012.
11. N. Kobayashi. Model-checking higher-order functions. In *PPDP*, 2009.
12. N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proc. of POPL*, pages 416–428, 2009.
13. N. Kobayashi. A practical linear time algorithm for trivial automata model checking of higher-order recursion schemes. In *Proc. of FoSSaCS*, pages 260–274, 2011.
14. N. Kobayashi. GTRecS2: A model checker for recursion schemes based on games and types. Tool at http://www-kb.is.s.u-tokyo.ac.jp/∼koba/gtrecs2/, 2012.
15. N. Kobayashi and C.-H. L. Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. *LMCS*, 7(4), 2011.
16. R. P. Neatherway, S. J. Ramsay, and C.-H. L. Ong. A traversal-based algorithm for higher-order model checking. In *ICFP* , pages 353–364, 2012.
17. S. Salvati and I. Walukiewicz. Krivine machines and higher-order schemes. In *ICALP* , *LNCS* 6756 , pages 162–173. Springer, 2011.
18. S. Salvati and I. Walukiewicz. Recursive schemes, krivine machines, and collapsible pushdown automata. In *Reachability Problems*, pages 6–20. Springer, 2012.
19. A. Seth. An alternative construction in symbolic reachability analysis of second order pushdown systems. In *Workshop on Reachability Problems*, 2007.

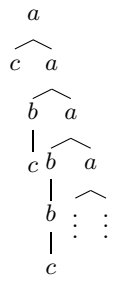


Fig. 5

A An Example of a HORS

Consider $\mathcal{G}_1 = (\Sigma, \mathcal{N}, \mathcal{R}, \mathcal{S})$ where: $\Sigma = \{a, b, c\}$ with $\mathtt{ar}(a) = 2$, $\mathtt{ar}(b) = 1$, $\mathtt{ar}(c) = 0$ and $\mathcal{N} = \{\mathcal{S}, F\}$ with $\mathcal{K}(\mathcal{S}) = \mathtt{o}$ and $\mathcal{K}(F) = \mathtt{o} \to \mathtt{o}$. $\mathcal{R}$ contains: $\mathcal{S} \longrightarrow F\,c$ and $F\,x \longrightarrow a\,x\,(F(b\,x))$. $\mathcal{G}$ generates the tree in Figure 5.

B Preliminaries

Lemma 1 $\mathcal{A}$ accepts $\mathbf{Tree}(\mathcal{G})$ if and only if $\{(q_I, \mathcal{S})\} \longrightarrow^* \emptyset$.

Proof. Taking the product of an automaton expressing a property and the system about which the property is expressed is standard, but for completeness we sketch a proof of this lemma here.

Let us modify the reduction relation to act on sets D of *triples* $(q, t, \upsilon) \in \mathcal{Q} \times \mathbf{AppTerms} \times dom(\mathbf{Tree}(\mathcal{G}))$ rather than just sets of pairs $(q, t) \in \mathcal{Q} \times \mathbf{AppTerms}$. For sets $D, D' \subseteq \mathcal{Q} \times \mathbf{AppTerms} \times dom(\mathbf{Tree}(\mathcal{G}))$ we write $D \longrightarrow D'$ if D' is such that for every $(q, t, \upsilon) \in D$ either there exists $(q, t', \upsilon) \in D'$ such that $t \longrightarrow t'$ or else there exists a $(q, a, S) \in \Delta_{\mathcal{A}}$ such that for every $(q', i) \in S$ there is some $(q', t', \upsilon i) \in D'$ such that $t \xrightarrow{(a,i)} t'$.

One technical difference (for convenience) is that the reduction relation on pairs allows $(q, \upsilon) \in D'$ to be sufficient to handle $(q, \upsilon) \in D$. (This is necessary to get the correspondence with SCPDS). This option is disallowed for triples. However, it is still easy to see that $\{(q_I, \mathcal{S}, \epsilon)\} \longrightarrow^* \emptyset$ if and only if $\{(q_I, \mathcal{S})\} \longrightarrow^* \emptyset$, so we may work with sets of these triples rather than sets of pairs.

Given a subset $U \subseteq \mathcal{Q} \times \mathbf{AppTerms} \times dom(\mathbf{Tree}(\mathcal{G}))$ we define the $2^{\mathcal{Q}}$-labelled tree $R_U : dom(\mathbf{Tree}(\mathcal{G})) \longrightarrow 2^{\mathcal{Q}}$ by $R_U(\upsilon) := \{\ q \mid \text{ there exists some } (q, \_, \upsilon) \in U\ \}$.

Suppose that $\{(q_I, \mathcal{S}, \epsilon)\} \longrightarrow U_1 \longrightarrow U_2 \longrightarrow \cdots \longrightarrow U_m = \emptyset$ for some $U_1, \ldots, U_m$ such that $U = \bigcup_{i=1}^{m} U_i$. We claim that there exists an accepting run-tree R' such that $dom(R') = dom(R_U)$ and $R'(\upsilon) \subseteq R(\upsilon)$ for every $\upsilon \in dom(R')$.

First note that $q_I \in R_U(\epsilon)$ and so we can take $R'(\epsilon) = \{q_I\}$ as required. Now suppose that $q \in R_U(\upsilon)$ for some $\upsilon \in dom(R)$. Let j be the maximum integer such that there exists a triple of the form $(q, t, \upsilon) \in U_j$. Since $U_m = \emptyset$,

we must have that j exists with $j < m$. Since j is maximal it must be the case that there is no triple of the form $(q, t', \upsilon) \in U_{j+1}$. This is only possible if there exists a $(q, a, S_q) \in \Delta_{\mathcal{A}}$ such that for every $(q', i) \in S_q$ there is some $(q', t'_i, \upsilon i) \in U_{j+1}$ such that $t \xrightarrow{(a,i)} t'_i$. Note also (by induction on $|\upsilon|$) that we must have $\mathcal{S} \xrightarrow{(a_1,i_1)\cdots(a_k,i_k)a} t$ where $\upsilon = i_1 \cdots i_k$. Thus by definition $\mathbf{Tree}(\mathcal{G})(\upsilon) = a$. For each $\upsilon i \in dom(R_U)$ we can take $R'(\upsilon i) := \bigcup_{q \in R'(\upsilon)} \{\ q' \mid (q', i) \in S_q\ \}$, which by construction must meet the requirements of a run-tree.

Now we check that $R'(\upsilon) \neq \emptyset$ for only finitely many $\upsilon \in dom(R')$. Suppose for contradiction that this is not the case. By construction this must mean that there is an infinite path $\epsilon, i_1, i_1 i_2, i_1 i_2 i_3, \ldots$ in the tree such that $R'(i_1 \cdots i_j) \neq \emptyset$ for all $j \geq 0$ and in particular for all $1 \leq j \leq m-1$, which implies that there exists $(\_, \_, i_1 \cdots i_j) \in U_m$ for some $1 \leq j \leq m-1$. This is a contradiction.

Conversely suppose that there is an accepting run-tree R of $\mathcal{A}$ on $\mathbf{Tree}(\mathcal{G})$. We can then define $U_0 := \{(q_I, \mathcal{S}, \epsilon)\}$ and $U_{j+1} := \{\ (q, t', \upsilon)\ \mid (q, t, \upsilon) \in U_j \text{ and } t \longrightarrow t'\ \} \cup \{\ (q', t_i, \upsilon i)\ \mid (q, a t_1 \cdots t_{\mathtt{ar}(a)}, \upsilon) \in U_j \text{ and } q' \in R(\upsilon i)\ \}$. Definition chasing tells us that for all $j \in \omega$ we have $U_j \longrightarrow U_{j+1}$. Now suppose for contradiction that $U_j \neq \emptyset$ for all $j \in \omega$. Since $R(\upsilon) \neq \emptyset$ for only finitely many υ (R is accepting), there must exist a $\upsilon \in dom(R)$ and $q \in \mathcal{Q}$ and $j \in \omega$ such that for each $j' \geq j$ there exists a triple of the form $(q, t, \upsilon) \in U_{j'}$ (for some term t). But this implies that there is an infinite reduction sequence $t \longrightarrow t' \longrightarrow t'' \longrightarrow \cdots$ and consequently that $\mathbf{Tree}(\mathcal{G})(\upsilon) = \bot$. But it also implies that $R(\upsilon) \neq \emptyset$ (it contains q), contradicting the assumption that R is a run-tree since by definition $\Delta_{\mathcal{A}}$ contains no element of the form $(\_, \bot, \_)$. Thus $U_0 \longrightarrow U_1 \longrightarrow \cdots \longrightarrow U_m = \emptyset$ for some $m \in \omega$, as required. □

C Streamlined Collapsible Pushdown Systems

Lemma 4. *Let ζ be such that $[\![\zeta]\!]_0 =: t$ is well-defined and an applicative term (*i.e. *well-sorted). Suppose further that $top(\zeta) = u$. Then if $\mathsf{FV}(u) = \{x_1, \ldots, x_k\}$ with $\boldsymbol{pos}(x_i) = i$ (for each $1 \leq i \leq k$), it must be the case that $t = u[[\![pop(\zeta)]\!]_1/x_1, \ldots, [\![pop(\zeta_k)]\!]_k/x_k][\![collapse(\zeta)]\!]_1 \cdots [\![collapse(\zeta)]\!]_{\mathtt{ar}(u)-\mathtt{ar}(t)}$. Moreover t contains no variables.*

Proof. We argue by induction on the structure of ζ and $top(\zeta)$. Suppose that $u = h u_1 \cdots u_\ell$ for $h \in \mathbf{At}$. By definition

$$t = h'[\![\zeta]\!]_1 \cdots [\![\zeta]\!]_\ell [\![\zeta]\!]_{\ell+1} \cdots [\![\zeta]\!]_{\mathtt{ar}(h)-\mathtt{ar}(t)}$$

where $h' = h$ if h is not a variable (and since it is atomic must be a closed term) and $h' = [\![pop(\zeta)]\!]_{\boldsymbol{pos}(x)}$ if $h' = x$ is a variable. In the latter case, the head occurrence of x receives the substitution required by the lemma, and by the induction hypothesis ($pop(\zeta)$ is strictly smaller than ζ) is a closed term.

Now by definition of $[\![\zeta]\!]_i$, for $1 \leq i \leq \ell$ it must be the case that $[\![\zeta]\!]_i = [\![rew^{\perp}_{u_i}(\zeta)]\!]_0$. By the induction hypothesis ($rew^{\perp}_{u_i}(\zeta)$ is either smaller or the same size as ζ, but u_i is a strict subterm of u) it must thus be the case that $[\![\zeta]\!]_i =$

$u_i[[\![pop(\zeta)]\!]_1/x_1, \ldots, [\![pop(\zeta_k)]\!]_k]$ (note that $collapse(rew^{\perp}_{u_i}(\zeta)) = \perp$ and $[\![\perp]\!]_j$ is undefined for any j) and contains no variables.

For $\ell < i \leq \mathtt{ar}(h) - \mathtt{ar}(t)$ it must be the case that $[\![\zeta]\!]_i = [\![collapse(\zeta)]\!]_{i-\ell}$ (since $BindLoc(i, \zeta) = BindLoc(i - \ell, collapse(\zeta))$ in this case). By the induction hypothesis these terms also contain no variables.

Thus

$$t = u[[\![pop(\zeta)]\!]_1/x_1, \ldots, [\![pop(\zeta_k)]\!]_k/x_k][\![collapse(\zeta)]\!]_1 \cdots [\![collapse(\zeta)]\!]_{\mathtt{ar}(u)-\mathtt{ar}(t)}$$

and contains no variables, as required. □

Theorem 1 Given $C \subseteq \mathbf{GS}$, $C \longrightarrow^* C'$ implies that $C \longrightarrow^* C' \cap \mathbf{GS}$. Moreover, given $C, C' \subseteq \mathbf{GS}$, if $C \longrightarrow^* C'$, then $[\![C]\!] \longrightarrow^* [\![C']\!]$. Conversely, if $[\![C]\!] \longrightarrow^* D$, then there exists $C' \subseteq \mathbf{GS}$ such that $[\![C']\!] \subseteq D$ and $C \longrightarrow^* C'$. Thus by Lemma 1 $\mathcal{A}$ accepts $\mathbf{Tree}(\mathcal{G})$ in co-trivial mode iff $\{c_0\} \longrightarrow^* \emptyset$.

Proof. The equivalence between HORS and CPDA given in [6] in some sense could be viewed as a proof of this theorem, but we sketch a proof here to check sanity.

Suppose that $C \subseteq \mathbf{GS}$ and $C \longrightarrow C'$ (single-step). We show that $C \longrightarrow C' \cap \mathbf{GS}$ and that if $C' \subseteq \mathbf{GS}$ then $[\![C]\!] \longrightarrow [\![C']\!]$ by considering each $(q, \zeta) \in C$ in turn. We must have one of the following cases:

'Reflexive' case: $(q, \zeta) \in C'$. Then the requirements are trivially met (recalling that $\longrightarrow$ between sets of terms is defined in terms of the *reflexive* transitive closure of $\longrightarrow$ between terms).

$\xrightarrow{\mathcal{N}}$ **case:** There exists $(q, \zeta') \in C'$ such that $\zeta' = push_{t'}(\zeta)$ where $\boldsymbol{hd}(top(\zeta)) = F \in dom(\mathcal{N})$ with $Fx_1 \cdots x_\ell \longrightarrow t' \in \mathcal{R}$. Since $(q, \zeta) \in \mathbf{GS}$ it must by definition be that $[\![\zeta]\!]_0 = Ft_1 \cdots t_\ell$ with $\mathcal{K}(Ft_1 \cdots t_\ell) = \mathtt{o}$.
Thus $\mathtt{ar}(F) = \ell$ and $\mathbf{FV}(t') \subseteq \{x_1, \ldots, x_\ell\}$ for variables $x_1, \ldots, x_\ell$ such that $\boldsymbol{pos}(x_i) = i$ for $1 \leq i \leq \ell$. Since $[\![\zeta]\!]_i = t_i$ for $1 \leq i \leq \ell$, it must be the case (by Lemma 4) that $[\![\zeta']\!]_0 = t'[t_1/x_1, \ldots, t_\ell/x_\ell]$. Thus $t = Ft_1 \cdots t_\ell \longrightarrow t'$, satisfying the requirement for $[\![C]\!] \longrightarrow [\![C']\!]$, and $\mathcal{K}(t') = \mathtt{o}$, satisfying the requirement for $C \longrightarrow C' \cap \mathbf{GS}$.

$\xrightarrow{\mathcal{V}}$ **case:** We have $top(\zeta) = \phi t_1 \cdots t_\ell$ with $\phi \in \mathcal{V}$. There exists $(q, \zeta') \in C'$ such that $\zeta' = rew^{\zeta''}_{t'}(\zeta)$ where $BindLoc(\boldsymbol{pos}(\phi), pop(\zeta)) = (t', \zeta'')$. Thus, by definition, $[\![rew^{\perp}_{t'}(\zeta'')]\!]_0 = [\![\zeta]\!]_{\boldsymbol{pos}(\phi)}$. We must then have:

$$[\![\zeta]\!]_0 = [\![\zeta]\!]_{\boldsymbol{pos}(\phi)}[\![\zeta]\!]_1 \cdots [\![\zeta]\!]_\ell = [\![rew^{\perp}_{t'}(\zeta'')]\!]_0[\![collapse(\zeta')]\!]_1 \cdots [\![collapse(\zeta')]\!]_\ell = [\![\zeta']\!]_0$$

Thus $\mathcal{K}([\![\zeta']\!]_0) = \mathtt{o}$, satsifying the requirement for $C \longrightarrow C' \cap \mathbf{GS}$ and $[\![\zeta]\!]_0 \in [\![C']\!]$, satisfying the requirement for $[\![C]\!] \longrightarrow [\![C']\!]$ (recalling that the definition of this relation is defined in terms of the *reflexive* transitive closure of reduction on terms).

$\xrightarrow{\Sigma}$ **case:** There must exist $(q, a, S) \in \Delta$ such that $top(\zeta) = at_1 \cdots t_{\mathtt{ar}}(a)$ and $(q', rew^{\perp}_{t_i}(\zeta)) \in C'$ for each $(q', i) \in S$. If $[\![\zeta]\!]_0 = at'_1 \cdots t'_{\mathtt{ar}}(a)$, then $[\![rew^{\perp}_{t_i}(\zeta)]\!]_0 =$

t'_i. Since $\mathcal{K}([\![\zeta]\!]_0) = \mathsf{o}$ we must also have $\mathcal{K}(t'_i) = \mathsf{o}$ and so $C \longrightarrow C' \cap \mathbf{GS}$. Moreover for each $(q', i) \in S$ we must have $(q', t'_i) \in [\![C']\!]$, satisfying the requirement for $[\![C]\!] \longrightarrow [\![C']\!]$.

So by induction on the length of a sequence $C_1 \longrightarrow C_2 \longrightarrow \cdots \longrightarrow C_m$ witnessing $C_1 \longrightarrow^* C_m$ we get that $C_1 \longrightarrow^* C_m \cap \mathbf{GS}$ and that $[\![C_1]\!] \longrightarrow^* [\![C_m]\!]$.

We now show the converse. Suppose that $[\![C]\!] \longrightarrow D$. We build up a C' such that $C \longrightarrow^* C'$ (note the transitive closure) and $[\![C']\!] \subseteq D$, by considering each $(q, \zeta) \in C$ in turn according to the following cases:

'reflexive' case: If $(q, [\![\zeta]\!]_0) \in D$, then we add (q, ζ) to C'.

Non-terminal reduction case: If $(q, t') \in D$ where $[\![\zeta]\!]_0 \longrightarrow t'$, then we must have $[\![\zeta]\!]_0 = Ft_1 \cdots t_\ell$ where $F \in dom(\mathcal{N})$ and $Fx_1 \cdots x_\ell \longrightarrow u \in \mathcal{R}$ with $t' = u[t_1/x_1, \ldots, t_\ell/x_\ell]$. We consider the following two cases:

$\xrightarrow{\mathcal{N}}$ **case:** Suppose that $\boldsymbol{hd}(top(\zeta)) = F$. Then for the same reasons as before we have $(q, \zeta) \xrightarrow{\mathcal{N}} (q, \zeta')$ such that $[\![\zeta']\!]_0 = t'$. So we just add (q, ζ') to C'.

$\xrightarrow{\mathcal{V}}$ **case:** Suppose that $\boldsymbol{hd}(top(\zeta)) = \phi \in \mathcal{V}$. Consider the greatest m such that there is a sequence of $\xrightarrow{\mathcal{V}}$ transitions $(q, \zeta) \xrightarrow{\mathcal{V}} (q, \zeta_1) \xrightarrow{\mathcal{V}} \cdots \xrightarrow{\mathcal{V}} (q, \zeta_m)$. It must be that $m \geq 1$. It must also be that m is finite since otherwise it would be possible to perform an infinite sequence of operations on ζ of the form

$$pop; collapse; \cdots; collapse; rew_{\_}; pop; collapse; \cdots; collapse; rew_{\_}; \cdots$$

which is impossible since stacks are by definition finite. (Note in particular that the structure of the sequence always discards the rewritten element including its annotation.)

But when covering the $\xrightarrow{\mathcal{V}}$ case before, we saw that $[\![\zeta_1]\!]_0 = \cdots = [\![\zeta_m]\!]_0$. Thus in particular $[\![\zeta_m]\!]_0 = Ft_1 \cdots t_\ell$. But since m is maximal, we must have $(q, \zeta_m) \xrightarrow{\mathcal{N}} (q, \zeta')$ such that $[\![\zeta']\!]_0 = t'$. Thus we may add (q, ζ') to C'. (This case is the reason why we can only consider $C \longrightarrow^* C'$ and not $C \longrightarrow C'$).

Terminal reduction case: If $[\![\zeta]\!]_0 = at_1 \cdots t_\ell$ and there exists $(q, a, S) \in \Delta$ such that $(q', t_i) \in D$ for every $(q', i) \in S$, then if $\boldsymbol{hd}(top(\zeta)) = a$, we can add $(q', rew^{\perp}_{t'_i}(\zeta))$ to C' where $top(\zeta) = at'_1 \cdots t'_\ell$. If $\boldsymbol{hd}(top(\zeta))$ is a variable, then we first reason in a similar manner to the case above.

So by induction on the length of a sequence $[\![C]\!] \longrightarrow D_1 \longrightarrow \cdots \longrightarrow D_m$ witnessing $[\![C]\!] \longrightarrow^* D_m$ we can construct a C_m such that $C_1 \longrightarrow^* C_m$ and $[\![C_m]\!] \subseteq D_m$. (We cannot guarantee the existence of a set C_m such that $[\![C_m]\!] = D_m$ due to the technicalities of the way we have formulated the definition of $\longrightarrow$ for terms. It is always possible to add arbitrary additional terms to a set on the right-hand-side of $\longrightarrow$ without violating the definition and in particular terms that may not not be in the image of $\mathbf{GS}$ under $[\![\_]\!]_0$). $\square$

D A Saturation Algorithm for SCPDS

Lemma 5. $\boldsymbol{\Phi^*}$ *satisfies:* $\mathcal{L}(\boldsymbol{\Phi^*}) \cap \mathbf{GS} \supseteq \mathbf{Pre}^\star_{\mathbf{KA}_{\mathcal{G},\mathcal{A}}}$.

Proof. Suppose that $(q,\zeta) \in \mathbf{Pre}^\star_{\mathbf{KA}_{\mathcal{G},\mathcal{A}}}$ (so in particular $(q,\zeta) \in \mathbf{GS}$). We argue by induction on the least m such that:

$$\{(q,\zeta)\} = C_m \longrightarrow \cdots \longrightarrow C_0 \longrightarrow \emptyset$$

(with each $C_i \subseteq \mathbf{GS}$) that $(q,\zeta) \in \mathcal{L}(\Delta_m)$.

If $m = 0$ then we have $\{(q,\zeta)\} \longrightarrow \emptyset$. This is only possible if $(q,\zeta) \xrightarrow{\Sigma} \emptyset$, which in turn is only possible if $top(\zeta) = at_1 \cdots t_{\mathtt{ar}(a)}$ for some $a \in \Sigma$ such that $(q,a,\emptyset) \in \Delta_{\mathcal{A}}$. It follows that $(q,\zeta) \in \mathcal{L}(\boldsymbol{\Delta}_{\mathbf{Acc}})$ and so we must also have $(q,\zeta) \in \mathcal{L}(\boldsymbol{\Phi^*})$.

Suppose now that $m > 0$ with $m = m' + 1$. We consider in turn each possible head $\boldsymbol{hd}(t)$ of $t := top(\zeta)$ where $\boldsymbol{args}(t) = (t_1, \ldots, t_{\boldsymbol{w}(t)})$:

Case $x \in \mathcal{V}$: So $\boldsymbol{hd}(t) = x$. Then we must have $\{(q,\zeta)\} \xrightarrow{\mathcal{V}} C_{m'}$. Thus $(q, rew^{\zeta}_{t'}(\zeta')) \in C_{m'}$ where $BindLoc(\boldsymbol{pos}(x), pop(\zeta)) = (t',\zeta')$. Since $(q,\zeta) \in \mathbf{GS}$ (by assumption) it must also be the case that $\mathcal{K}(x) = \mathcal{K}(t')$. By the induction hypothesis $(q, rew^{\zeta}_{t'}(\zeta')) \in \mathcal{L}(\Delta_{m'})$. Suppose that this is witnessed by a run of the stack-automaton (viewed as a tree-automaton) beginning with the transition $(q, t', Q'_e, Q'_\alpha) \in \Delta_{m'}$. In particular we must have $\zeta \in \mathcal{L}(\Delta_{m'})_p$ for every $p \in Q'_\alpha$.
It must be the case that t' is a term such that $t' = u_i$ for some $1 \leq i \leq \ell$ where $\boldsymbol{args}(top(\zeta')) = (u_1, \ldots, u_\ell)$. By definition we must have $((q,i)_{Q'_\alpha}, top(\zeta'), Q'_e, \emptyset) \in \Delta^\star$ and so it must also be the case that $\zeta' \in \mathcal{L}(\Delta_{m'})_{(q,i)_{Q'_\alpha}}$, since $pop(rew^{\zeta}_{t'}(\zeta')) = pop(\zeta')$ is accepted from every state in Q'_e due to (q, t', Q'_e, Q'_α) being an initial transition of an accepting run for $rew^{\zeta}_{t'}(\zeta')$.
Suppose that unwinding the recursive definition of $BindLoc(\boldsymbol{pos}(x), pop(\zeta))$ gives a sequence of stacks $\zeta_1, \ldots, \zeta_k$ and integers $i_1, \ldots, i_k$ where $\zeta_1 = pop(\zeta)$ with $i_1 = \boldsymbol{pos}(x)$, $\zeta_k = \zeta'$ with $i_k = i$, and for $1 \leq j < k$ we have $\zeta_{j+1} = collapse(\zeta_j)$ and $i_{j+1} = i_j - \boldsymbol{w}(top(\zeta_j)))$. Arguing by induction on $(k-j)$ we can check that for all $1 \leq j \leq k$ we have $\zeta_j \in \mathcal{L}(\Delta_{m'})_{(q,i_j)_{Q'_\alpha}}$. The base case $(j = k)$ was given above, and the induction step follows from the fact that we must have $i_j > \boldsymbol{w}(top(\zeta_j))$ and $((q,i_j)_{Q'_\alpha}, top(\zeta_j), \emptyset, \{(q,i_{j+1})_{Q'_\alpha}\}) \in {\Delta_{m'}}^\star$. Thus in particular $pop(\zeta) \in \mathcal{L}(\Delta_{m'})_{(q,\boldsymbol{pos}(x))_{Q'_\alpha}}$.
Recall that $\zeta \in \mathcal{L}(\Delta_{m'})_p$ for every $p \in Q'_\alpha$. This means that there must exist sets of states Q_α and Q_e such that

$$Q'_\alpha \xrightarrow[\Delta_{m'}]{top(\zeta)\ Q_\alpha} Q_e$$

and $collapse(\zeta) \in \mathcal{L}(\Delta_{m'})_{p_\alpha}$ for every $p_\alpha \in Q_\alpha$, and $pop(\zeta) \in \mathcal{L}(\Delta_{m'})_{p_e}$ for every $p_e \in Q_e$. We may thus conclude that

$$(q,\zeta) \in \mathcal{L}(\Delta_{m'} \cup \{q \xrightarrow{top(\zeta)\ Q_\alpha} Q_e \cup \{(q, \boldsymbol{pos}(x))_{Q'_\alpha}\}\}).$$

But

$$q \xrightarrow{top(\zeta)\ Q_\alpha} Q_e \cup \{(q, \boldsymbol{pos}(x))_{Q'_\alpha}\} \subseteq \boldsymbol{\Phi}^V(\Delta_{m'}) \subseteq \boldsymbol{\Phi}(\Delta_{m'}) = \Delta_m$$

and so $(q, \zeta) \in \mathcal{L}(\Delta_m)$, as required.

Case $F \in \mathcal{N}$: Then we must have $\{(q, \zeta)\} \xrightarrow{\mathcal{N}} C_{m'}$ and so $(q, push_{t'}(\zeta)) \in C_{m'}$ where $Fx_1 \cdots x_{\mathtt{ar}(F)} \longrightarrow t' \in \mathcal{R}$. By the induction hypothesis $(q, push_{t'}(\zeta)) \in \mathcal{L}(\Delta_{m'})$. Suppose that (q, t', Q'_e, Q'_α) is the first transition in a run of the automaton (viewed as at tree automaton) witnessing $(q, push_{t'}(\zeta)) \in \mathcal{L}(\Delta_{m'})$. Since $\perp$ is not read by any transition of $\Delta_{m'}$ and $collapse(push_{t'}(\zeta)) = \perp$, it must be the case that $Q'_\alpha = \emptyset$. Moreover for every $p \in Q'_e$, it must be the case that $\zeta \in \mathcal{L}(\Delta_{m'})_p$. Thus there must exist sets of states Q_α and Q_e such that:

$$q \xrightarrow[\Delta_{m'}]{t'\ \emptyset} Q'_e \xrightarrow[\Delta_{m'}]{top(\zeta)\ Q_\alpha} Q_e$$

and $pop(\zeta) \in \mathcal{L}(\Delta_{m'})_{p_e}$ for every $p_e \in Q_e$, and $collapse(\zeta) \in \mathcal{L}(\Delta_{m'})_{p_\alpha}$ for every $p_\alpha \in Q_\alpha$. Thus:

$$(q, \zeta) \in \mathcal{L}(\Delta_{m'} \cup \{q \xrightarrow{top(\zeta)\ Q_\alpha} Q_e\}).$$

But

$$\Delta_{m'} \cup \{q \xrightarrow{top(\zeta)\ Q_\alpha} Q_e\} \subseteq \boldsymbol{\Phi}^{\mathcal{N}}(\Delta_{m'}) \subseteq \boldsymbol{\Phi}(\Delta_{m'}) = \Delta_m.$$

Case $a \in dom(\Sigma)$: By our generality preserving assumption about the context of terminal symbols we must have $top(\zeta) = at_1 \cdots t_{\mathtt{ar}(a)}$. We must have $\{(q, \zeta)\} \xrightarrow{\Sigma} C_{m'}$. Thus there exists $(q, a, S) \in \Delta_{\mathcal{A}}$ such that for every $(q', i) \in S$ we have $(q', rew^{\perp}_{t_i}(\zeta)) \in C_{m'}$. By the induction hypothesis we have for each of these configurations that $(q', rew^{\perp}_{t_i}(\zeta)) \in \mathcal{L}(\Delta_{m'})$. Let $(q', t_i, Q_{e_{q',i}}, \emptyset)$ be the first transition in a run of the automaton (viewed as a tree automaton) witnessing $(q', rew^{\perp}_{t_i}(\zeta)) \in \mathcal{L}(\Delta_{m'})$. (The $\emptyset$ component is necessary since $collapse(rew^{\perp}_{t_i}(\zeta)) = \perp$ and $\perp$ does not feature in any transition of $\Delta_{m'}$). Thus for every $(q', i) \in S$ and $p \in Q_{e_{q',i}}$ it must be the case that $pop(\zeta) = pop(rew^{\perp}_{t_i}(\zeta)) \in \mathcal{L}(\Delta_{m'})_p$. Thus:

$$(q, \zeta) \in \mathcal{L}(\Delta_{m'} \cup \{q \xrightarrow{at_1 \cdots t_{\mathtt{ar}(a)}\ \emptyset} \bigcup_{(q',i) \in S} Q_{e_{q',i}}\})$$

and

$$\Delta_{m'} \cup \{q \xrightarrow{at_1 \cdots t_{\mathtt{ar}(a)}\ \emptyset} \bigcup_{(q',i) \in S} Q_{e_{q',i}}\} \subseteq \boldsymbol{\Phi}^{\Sigma}(\Delta_{m'}) \subseteq \boldsymbol{\Phi}(\Delta_{m'}) = \Delta_m$$

and so $(q, \zeta) \in \mathcal{L}(\Delta_m)$ as required. □

Lemma 6. *$\boldsymbol{\Phi}^*$ satisfies: $\mathcal{L}(\boldsymbol{\Phi}^*) \cap \mathbf{GS} \subseteq \mathbf{Pre}^{\star}_{\mathbf{KA}_{\mathcal{G},\mathcal{A}}}$.*

Proof. Suppose that $(q, \zeta) \in \mathcal{L}(\boldsymbol{\Phi}^*) \cap \mathbf{GS}$. By Lemma 3 (see Appendix E—note that the forward reference is purely presentational and there is no circularity) it is the case that $\Vdash_{\mathcal{A}} [\![\zeta]\!]_0 : q$. By Lemma 2 it follows that $\vdash_{\mathcal{A}} t : q$.

Let $\mathcal{G}_t$ be the HORS $\mathcal{G}$ modified to have a new initial symbol $\mathcal{S}_t$ with rule $\mathcal{S}_t \longrightarrow t$ and let $\mathcal{A}_q$ be $\mathcal{A}$ modified to have q as its initial state. Note it must also be the case that $\vdash_{\mathcal{A}_q} t : q$ (where $\vdash_{\mathcal{A}_q}$ is the Kobayashi-typing relation for $\mathcal{A}_q$) with respect to the rules of $\mathcal{G}_t$ (since the modifications are conservative with respect to the atoms of $\mathcal{G}$). However we also get $\vdash_{\mathcal{A}_q} \mathcal{S}_t : q$ with respect to $\mathcal{G}_t$.

By Theorem 3 it must thus be the case that $\mathbf{Tree}(\mathcal{G}_t) \in \mathcal{L}(\mathcal{A}_q)$. Thus by Theorem 1 we have $(q, \zeta) \in \mathbf{Pre}^{\star}_{\mathbf{KA}_{\mathcal{G},\mathcal{A}}}$, as required. □

Lemma 7. *The saturation algorithm terminates—that is there exists an $m \in \omega$ such that $\Delta_m = \Delta_{m+1}$.*

Proof. We note that for $q, q' \in \mathcal{Q}$ and $Q, Q' \subseteq \mathbf{Q}(\mathcal{Q}, r)$ and sort κ such that $[\![q, Q]\!]_\kappa$ and $[\![q', Q']\!]_\kappa$ are both defined we have $[\![q, Q]\!]_\kappa = [\![q', Q']\!]_\kappa$ implies $q = q'$ and $Q = Q'$. This is a simple induction on the structure of κ. Thus for well-sorted stack-automaton transitions (q, t, Q_e, Q_α) and $(q', t', Q'_e, Q'_\alpha)$ we have that $[\![(q, t, Q_e, Q_\alpha)]\!] = [\![(q', t', Q'_e, Q'_\alpha)]\!]$ implies $(q, t, Q_e, Q_\alpha) = (q', t', Q'_e, Q'_\alpha)$. Let us call this property *'injectivity of typification'*.

By Theorem 4 every transition added by the saturation algorithm is well-sorted. Since every free variable in a term t has a unique sort and t itself has a unique sort, and also since there are only finitely many intersection types of any given sort (since $\mathcal{Q}$ is finite) there are only finitely many possible well-sorted transitions that read any given term $t \in \mathcal{T}_{\mathcal{G}}$. Since $\mathcal{T}_{\mathcal{G}}$ is also finite there are only finitely many well-sorted transitions overall. Since the saturation algorithm grows the stack-automaton monotonically by adding only well-sorted transitions, it must thus eventually terminate.

□

The three lemmas above combine to give the statement of Theorem 2.

E Type-Based Interpretation

Theorem 4 If Δ is well-sorted, then so are $\boldsymbol{\Phi}^V(\Delta)$, $\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)$ and $\boldsymbol{\Phi}^{\Sigma}(\Delta)$. Moreover $[\![\boldsymbol{\Phi}^V(\Delta)]\!] = \boldsymbol{\Psi}^V([\![\Delta]\!])$, $[\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!] = \boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!])$, and $[\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!] = \boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!])$. Also, $\boldsymbol{\Delta}_{\mathbf{Acc}}$ is well-sorted with $[\![\boldsymbol{\Delta}_{\mathbf{Acc}}]\!] = \boldsymbol{\nabla}_{\mathbf{Acc}}$.

Proof. First recall that $\boldsymbol{\Delta}_{\mathbf{Acc}} := \{\ (q, t, \emptyset, \emptyset) \mid (q, \boldsymbol{hd}(t), \emptyset) \in \Delta_{\mathcal{A}}\ \}$ and that $\boldsymbol{\nabla}_{\mathbf{Acc}}$ is the set of typing judgements postulated by the (Accept) rule. It is immediate from definitions that $[\![\boldsymbol{\Delta}_{\mathbf{Acc}}]\!] = \boldsymbol{\nabla}_{\mathbf{Acc}}$. (In particular recall that we are assuming that all occurrences of terminals are fully applied so $\boldsymbol{hd}(t) \in \Sigma$ implies that $\mathcal{K}(t) = \mathsf{o}$).

Now suppose that Δ is well-sorted. We show first that $\boldsymbol{\Phi}^V(\Delta)$, $\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)$ and $\boldsymbol{\Phi}^{\Sigma}(\Delta)$ are well-sorted and that $[\![\boldsymbol{\Phi}^V(\Delta)]\!] \subseteq \boldsymbol{\Psi}^V([\![\Delta]\!])$, $[\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!] \subseteq \boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!])$, and $[\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!] \subseteq \boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!])$. Let us consider each operator in turn:

$\boldsymbol{\Phi}^V$: Suppose that $q \xrightarrow[\Delta]{t'\ Q'_\alpha} Q'_e$ and $Q'_\alpha \xrightarrow[\Delta]{t\ Q_\alpha} Q_e$ for some term $t' \in \mathcal{T}_\mathcal{G}$ with $\boldsymbol{hd}(t) = \phi \in \mathcal{V}$ and $\mathcal{K}(\phi) = \mathcal{K}(t')$. Indeed suppose that $t = \phi u_1 \cdots u_\ell$ and that $\mathcal{K}(\phi) = \kappa_1 \to \cdots \to \kappa_\ell \to \kappa_{\ell+1} \to \cdots \to \kappa_{\mathtt{ar}(\phi)} \to \mathtt{o}$.
Let $\Gamma'_e := \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q'_e \text{ with } x \in \mathbf{FV}(t')\ \}$. For each $1 \leq i \leq \mathtt{ar}(\phi)$ let $T_i := \{\ [\![q', Q]\!]_{\kappa_i} \mid (q', i)_Q \in Q'_\alpha\ \}$. Due to the well-sortedness of Δ it must be the case that $[\![q \xrightarrow[\Delta]{t'\ Q'_\alpha} Q'_e]\!]$ is defined and (by definition) equal to:

$$\Gamma'_e \Vdash_\mathcal{A} t' : \bigwedge T_1 \to \cdots \to \bigwedge T_\ell \to \bigwedge T_{\ell+1} \to \cdots \to \bigwedge T_{\mathtt{ar}(x)} \to q \qquad (*)$$

Now recall that we are assuming $Q'_\alpha \xrightarrow[\Delta]{\phi u_1 \cdots u_\ell\ Q_\alpha} Q_e$. Since all members of Q'_α must be of the form $(q', i)_Q$, this must be witnessed entirely by transitions in $\Delta^\star - \Delta$. It follows that

$$Q_\alpha = \{\ (q', i-\ell)_Q \mid (q', i)_Q \in Q'_\alpha \text{ and } i > \ell\ \}\,(**)$$

It also follows that for each $p := (q', i)_Q \in Q'_\alpha$ with $1 \leq i \leq \ell$, Δ must contain a transition $q' \xrightarrow[\Delta]{u_i\ Q} Q_{(p,i)}$ for some set of states $Q_{(p,i)}$ such that:

$$\bigcup_{\substack{p = (q', i)_Q \in Q'_\alpha \\ i \in [1,\ell]}} Q_{(p,i)} = Q_e \qquad (***)$$

Again by assumption each of these transitions must be well-sorted. By definition

$$[\![q' \xrightarrow[\Delta]{u_i\ Q} Q_{(p,i)}]\!] = \Gamma_i^\tau \Vdash_\mathcal{A} u_i : \tau$$

where $\Gamma_i^\tau = \{\ x : [\![q'', Q']\!]_{\mathcal{K}(x)} \mid (q'', \boldsymbol{pos}(x))_{Q'} \in Q_{(p,i)} \text{ with } x \in \mathbf{FV}(u_i)\ \}$ Thus for each $i \in [1, \ell]$ and for each $\tau \in T_i$ there exists Γ_i^τ such that $\Gamma_i^\tau \Vdash_\mathcal{A} u_i : \tau \in [\![\Delta]\!]$. Since the judgement $(*)$ also belongs to $[\![\Delta]\!]$ we may apply the (Head-Var) rule to get that $\boldsymbol{\Psi}^V([\![\Delta]\!])$ contains:

$$\bigcup_{i \in [1,\ell], \tau \in T_i} \Gamma_i^\tau, \phi : \bigwedge T_1 \to \cdots \to \bigwedge T_\ell \to \bigwedge T_{\ell+1} \to \cdots \to \bigwedge T_{\mathtt{ar}(x)} \to q$$
$$\Vdash_\mathcal{A} t : \bigwedge T_{\ell+1} \to \cdots \to \bigwedge T_{\mathtt{ar}(x)} \to q$$

(Recall that $t = \phi u_1 \cdots u_\ell$). By $(**)$ we have $\bigwedge T_{\ell+1} \to \cdots \to \bigwedge T_{\mathtt{ar}(x)} \to q = [\![q, Q_\alpha]\!]_{\mathcal{K}(t)}$. Note also that $\bigcup_{i \in [1,\ell], \tau \in T_i} \Gamma_i^\tau = \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q_e \text{ with } x \in \mathbf{FV}(t)\ \}$ due to $(***)$.
It follows that

$$q \xrightarrow{t\ Q_\alpha} \{(q, \boldsymbol{pos}(\phi))_{Q'_\alpha}\} \cup Q_e$$

is well-sorted with the conclusion of (Head-Var) given above as its image under $[\![\_]\!]$. Thus $[\![\boldsymbol{\Phi}^V(\Delta)]\!] \subseteq \boldsymbol{\Psi}^V([\![\Delta]\!])$, as required.

$\boldsymbol{\Phi}^{\mathcal{N}}$: Suppose that $q \xrightarrow[\Delta]{t'\ \emptyset} Q'_e \xrightarrow[\Delta]{t\ Q_\alpha} Q_e$ where $t = Fu_1 \cdots u_\ell$ for $F \in \mathcal{N}$ with $F \longrightarrow t' \in \mathcal{R}$. Since Δ is assumed to be well-sorted, we must have

$$[\![q \xrightarrow[\Delta]{t'\ \emptyset} Q'_e]\!] = \Gamma'_e \Vdash_{\mathcal{A}} t' : q \qquad (\dagger)$$

where

$$\Gamma'_e := \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q'_e \text{ with } x \in \mathbf{FV}(t')\ \}$$

Suppose that $\mathcal{K}(F) = \kappa_1 \to \cdots \kappa_\ell \to \kappa_{\ell+1} \to \cdots \to \kappa_{\mathtt{ar}(F)} \to \mathtt{o}$. For each $1 \leq i \leq \mathtt{ar}(F)$, define $T_i := \{\ [\![q', Q]\!]_{\kappa_i} \mid (q', i)_Q \in Q'_e\ \}$. Observe that for each $1 \leq i \leq \ell$ we have $T_i = \{\ \tau \mid x : \tau \in \Gamma'_e\ \}$ $(\dagger\dagger)$.
Following the same line of reasoning as in the $\boldsymbol{\Phi}^V$-case, we get that for each $i \in [1, \ell]$ and $\tau \in T_i$ there exists Γ_i^τ such that $\Gamma_i^\tau \Vdash_{\mathcal{A}} u_i : \tau$. From the fact that the judgement $(\dagger)$ is in $[\![\Delta]\!]$ and the fact $(\dagger\dagger)$, we may thus apply the (Rec) rule to get that the following judgement belongs to $\boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!])$:

$$\bigcup_{i \in [1,\ell], \tau \in T_i} \Gamma_i^\tau \Vdash_{\mathcal{A}} t : \bigwedge T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(F)} \to q \qquad (\dagger\,\dagger\,\dagger)$$

(recalling that $t = Fu_1 \cdots u_\ell$).
We also get (following the same line of reasoning as in the $\boldsymbol{\Phi}^V$-case) that $[\![q, Q_\alpha]\!]_{\mathcal{K}(t)} = \bigwedge T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(F)} \to q$ and that $\bigcup_{i \in [1,\ell], \tau \in T_i} \Gamma_i^\tau = \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q_e \text{ with } x \in \mathbf{FV}(t)\ \}$. It follows that $q \xrightarrow{t\ Q_\alpha} Q_e$ is well-sorted with $(\dagger\,\dagger\,\dagger)$ its image under $[\![\_]\!]$.
Thus $[\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!] \subseteq \boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!])$, as required.

$\boldsymbol{\Phi}^{\Sigma}$: Suppose that $(q, a, S) \in \Delta_{\mathcal{A}}$ and that there are transitions $q' \xrightarrow[\Delta]{t_i\ \emptyset} Q_e^{(q',i)}$ in Δ for each $(q', i) \in S$.
It must thus be the case that $[\![\Delta]\!]$ contains judgements $\Gamma_{(q',i)} \Vdash_{\mathcal{A}} t_i : q'$ for every $(q', i) \in S$. We may thus apply the rule (Tran) to get that $\boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!])$ contains:

$$\bigcup_{(q',i) \in S} \Gamma_{(q',i)} \Vdash_{\mathcal{A}} at_1 \cdots t_{\mathtt{ar}(a)} : q$$

where we will have

$$\bigcup_{(q',i) \in S} \Gamma_{(q',i)} = \left\{ x : [\![q'', Q]\!]_{\mathcal{K}(x)} \;\middle|\; (q'', \boldsymbol{pos}(x))_Q \in \bigcup_{(q',i) \in S} Q^{(q',i)_e} \text{ with } x \in \bigcup_{i \in [1, \mathtt{ar}(a)]} \mathbf{FV}(t_i) \right\}$$

Thus the transition $q \xrightarrow{at_1 \cdots t_{\mathtt{ar}(a)}\ \emptyset} \bigcup_{(q',i) \in S} Q^{(q',i)_e}$ is well-sorted with the judgement above being its image under $[\![\_]\!]$. Thus $[\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!] \subseteq \boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!])$, as required.

Now we show that $\boldsymbol{\Psi}^V([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^V(\Delta)]\!]$, $\boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!]$, and $\boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!]$. Let us consider each operator in turn:

$\boldsymbol{\Psi}^V$: Suppose that for some terms $t, t_1, \ldots, t_\ell$ and sets of types $T_1, \ldots, T_\ell, T_{\ell+1}, \ldots, T_{\mathtt{ar}(t)}$ it is the case that $[\![\Delta]\!]$ contains judgements $\Gamma_j^\tau \Vdash_{\mathcal{A}} t_j : \tau$ for each $j \in [1, \ell]$ and each $\tau \in T_j$, and also a judgement $\Gamma \Vdash_{\mathcal{A}} t : \bigwedge T_1 \to \cdots \bigwedge T_{\mathtt{ar}(t)} \to q$.
It follows that Δ must contain transitions $q_\tau \xrightarrow[\Delta]{t_j\ Q_\alpha^\tau} Q_j^\tau$ for each $j \in [1, \ell]$ and each $\tau \in T_j$, such that $[\![q_\tau, Q_\alpha^\tau]\!]_{\mathcal{K}(t_j)} = \tau$, and

$$\Gamma_j^\tau = \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q_j^\tau \text{ with } x \in \mathbf{FV}(t)\ \}$$

We must also have a transition:

$$q \xrightarrow[\Delta]{t\ Q'_\alpha} Q'_e$$

such that in particular

$$T_i = \{\ [\![q, Q]\!]_{\kappa_i} \mid (q, i)_Q \in Q'_\alpha\ \}$$

for each $1 \leq i \leq \mathtt{ar}(t)$, where $\mathcal{K}(t) = \kappa_1 \to \cdots \kappa_\ell \to \kappa_{\ell+1} \to \cdots \to \kappa_{\mathtt{ar}(t)} \to \mathtt{o}$. We must also have that $(q', i)_Q \in Q'_\alpha$ implies that $i \in [1, \mathtt{ar}(t)]$.
Let ϕ be a variable such that $\mathcal{K}(\phi) = \mathcal{K}(t)$ and consider the judgement:

$$\bigcup_{j \in [1,\ell], \tau \in T_j} \Gamma_j^\tau, \phi : \bigwedge T_1 \to \cdots \bigwedge T_\ell \to T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(t)} \to q \Vdash_{\mathcal{A}} \phi t_1 \cdots t_\ell : T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(t)} \to q$$

that can be inferred by the (Head-Var) rule from the judgements above assumed to be in $[\![\Delta]\!]$. We need to show that a corresponding transition belongs to $\boldsymbol{\Phi}(\Delta)$.
Since we constrain (by definition) $\Vdash_{\mathcal{A}}$ to make assertions only about elements in $\mathcal{T}_\mathcal{G}$ we must have $\phi t_1 \cdots t_\ell \in \mathcal{T}_\mathcal{G}$. By the definition of $\Delta^\star$ we must then have:

$$\{\ (q_\tau, j)_{Q_\alpha^\tau} \mid j \in [1, \ell], \tau \in T_j\ \} \xrightarrow[\Delta]{\phi t_1 \cdots t_\ell\ \emptyset} \bigcup_{j \in [1,\ell], \tau \in T_j} Q_j^\tau =: Q_e \qquad (*)$$

Also by the definition of $\Delta^\star$ we must have:

$$\{\ (q', j)_Q \mid j \in [\ell + 1, \mathtt{ar}(t)] \text{ and } (q', j)_Q \in Q'_\alpha\ \} \xrightarrow[\Delta]{\phi t_1 \cdots t_\ell\ Q_\alpha} \emptyset \qquad (**)$$

where

$$Q_\alpha = \{\ (q', j - \ell)_Q \mid j \in [\ell + 1, \mathtt{ar}(t)] \text{ and } (q', j)_Q \in Q'_\alpha\ \}$$

Combining $(*)$ and $(**)$ gives us:

$$Q'_\alpha \xrightarrow[\Delta]{\phi t_1 \cdots t_\ell\ Q_\alpha} Q_e$$

We thus have all the transitions required to guarantee that $\boldsymbol{\Phi}^V(\Delta)$ must contain the transition:

$$q \xrightarrow{\phi t_1 \cdots t_\ell \ Q_\alpha} Q_e \cup \{(q, \boldsymbol{pos}(\phi))_{Q'_\alpha}\}$$

which must have as its image under $[\![\_]\!]$ the judgement

$$\bigcup_{j\in[1,\ell],\tau\in T_j} \Gamma_j^\tau, \phi : \bigwedge T_1 \to \cdots \bigwedge T_\ell \to T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(t)} \to q \Vdash_{\mathcal{A}} \phi t_1 \cdots t_\ell : T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(t)} \to q$$

as required. This shows that $\boldsymbol{\Psi}^V([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^V(\Delta)]\!]$.

$\boldsymbol{\Psi}^{\mathcal{N}}$: Suppose that for some terms $t_1, \ldots, t_\ell$ and sets of types $T_1, \ldots, T_\ell, T_{\ell+1}, \ldots, T_{\mathtt{ar}(F)}$ it is the case that $[\![\Delta]\!]$ contains judgements $\Gamma_j^\tau \Vdash_{\mathcal{A}} t_j : \tau$ for each $j \in [1, \ell]$ and each $\tau \in T_j$, and also a judgement $\Gamma \Vdash_{\mathcal{A}} t' : q$ where for $F \in \mathcal{N}$ it is the case that $F \longrightarrow t' \in \mathcal{R}$ and that $\mathcal{K}(F) = \kappa_1 \to \cdots \kappa_\ell \to \kappa_{\ell+1} \to \cdots \to \kappa_{\mathtt{ar}(F)} \to \mathtt{o}$. Suppose further that for each $i \in [1, \mathtt{ar}(F)]$ it is the case that $\Gamma = \{\ x : \tau \mid \tau \in T_i \text{ and } x \in \mathbf{FV}(t') \text{ and } \boldsymbol{pos}(x) = i \ \}$. Using these judgements one can derive from the (Rec) rule the following judgement:

$$\bigcup_{j\in[1,\ell],\tau\in T_j} \Gamma_j^\tau \Vdash_{\mathcal{A}} F t_1 \cdots t_\ell : T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(F)} \to q$$

We must show that a corresponding transition belongs to $\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)$.
From the assumption about judgements in $[\![\Delta]\!]$, it follows that Δ must contain transitions $q_\tau \xrightarrow[\Delta]{t_j \ Q_\alpha^\tau} Q_j^\tau$ for each $j \in [1, \ell]$ and each $\tau \in T_j$, such that $[\![q_\tau, Q_\alpha^\tau]\!]_{\mathcal{K}(t_j)} = \tau$, and

$$\Gamma_j^\tau = \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q_j^\tau \text{ with } x \in \mathbf{FV}(t) \ \}$$

and also a transition:

$$q \xrightarrow[\Delta]{t' \ \emptyset} Q'_e$$

where

$$\Gamma = \{\ x : [\![q', Q]\!]_{\mathcal{K}(x)} \mid (q', \boldsymbol{pos}(x))_Q \in Q'_e \text{ with } x \in \mathbf{FV}(t) \ \}$$

Following the same line of reasoning as in the $\boldsymbol{\Psi}^V$ case (with Q'_e in place of Q'_α) gives us:

$$Q'_e \xrightarrow[\Delta]{F t_1 \cdots t_\ell \ Q_\alpha} \bigcup_{j\in[1,\ell],\tau\in Q_j^\tau} =: Q_e$$

where

$$Q_\alpha = \{\ (q', j-\ell)_Q \mid j \in [\ell+1, \mathtt{ar}(t)] \text{ and } (q', j)_Q \in Q'_e \ \}$$

It must thus be the case that $\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)$ contains:

$$q \xrightarrow{Ft_1 \cdots t_\ell \; Q_\alpha} Q_e$$

which yields

$$\bigcup_{j \in [1,\ell], \tau \in T_j} \Gamma_j^\tau \Vdash_{\mathcal{A}} Ft_1 \cdots t_\ell : T_{\ell+1} \to \cdots \to T_{\mathtt{ar}(F)} \to q$$

under $[\![\_]\!]$, as required. Thus $\boldsymbol{\Psi}^{\mathcal{N}}([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^{\mathcal{N}}(\Delta)]\!]$

$\boldsymbol{\Psi}^{\Sigma}$: Suppose that for some $(q, a, S) \in \Delta_{\mathcal{A}}$ and for each $1 \leq i \leq \mathtt{ar}(a)$ and every $(q, i) \in S$ there exists $\Gamma_{(q',i)}$ such that $\Gamma_{(q,i)} \Vdash_{\mathcal{A}} t_i : q'$ is in $[\![\Delta]\!]$.
It follows that there exist corresponding transitions in Δ of the form:

$$q' \xrightarrow[\Delta]{t_i \; \emptyset} Q_{(q',i)}$$

Thus $\boldsymbol{\Phi}^{\Sigma}(\Delta)$ contains a transition of the form:

$$q \xrightarrow{at_1 \cdots t_{\mathtt{ar}(a)} \; \emptyset} \bigcup_{(q',i) \in S} Q_{(q',i)}$$

that under $[\![\_]\!]$ maps to the judgement

$$\bigcup_{(q',i) \in S} \Gamma_{(q',i)} \Vdash_{\mathcal{A}} at_1 \cdots t_{\mathtt{ar}(a)} : q$$

This is the judgement resulting from those mentioned above by using the (Tran) rule. It thus follows that $\boldsymbol{\Psi}^{\Sigma}([\![\Delta]\!]) \subseteq [\![\boldsymbol{\Phi}^{\Sigma}(\Delta)]\!]$

□

Lemma 2 If $\Gamma \Vdash_{\mathcal{A}} t : \tau$, then $\Gamma \vdash_{\mathcal{A}} t \; : \; \tau$.

Proof. We first give the rules for $\vdash_{\mathcal{A}}$ that can replace the first four $\Vdash_{\mathcal{A}}$ rules (Accept), (Head-Var), (Rec) and (Tran). We note that for any applicative term t we have $\vdash_{\mathcal{A}} t : \bigwedge \emptyset$. We also note that if $\Gamma \vdash_{\mathcal{A}} t : \tau$, then $\Gamma \cup \Gamma' \vdash_{\mathcal{A}} t : \tau$ for any Γ'. We thus without loss of generality allow ourselves to include implicit weakening (adding such a Γ') in the replacements we give below.

The (Accept) rule in which $t = at_1 \cdots t_{\mathtt{ar}(a)}$ can be replaced by:

$$\dfrac{\dfrac{\dfrac{}{(q, a, \emptyset) \in \Delta_{\mathcal{A}}}}{\vdash_{\mathcal{A}} a : \underbrace{\bigwedge \emptyset \to \cdots \to \bigwedge \emptyset}_{\mathtt{ar}(a) \text{ times}} \to q}\text{(T-Const)}}{\vdash_{\mathcal{A}} at_1 : \underbrace{\bigwedge \emptyset \to \cdots \to \bigwedge \emptyset}_{\mathtt{ar}(a)-1 \text{ times}} \to q}\text{(T-App)}$$

$$\dfrac{\dfrac{\vdots}{\vdash_{\mathcal{A}} at_1 \cdots t_{\ell-1} : \bigwedge \emptyset \to q}\text{(T-App)}}{\vdash_{\mathcal{A}} at_1 \cdots t_{\ell-1} t_\ell : \bigwedge \emptyset \to q}\text{(T-App)}$$

The (Head-Var) rule can be replaced by:

$$\dfrac{\dfrac{\dfrac{}{\phi : \bigwedge T_1 \to \cdots \to \bigwedge T_l \to \tau \vdash_{\mathcal{A}} \phi : \bigwedge T_1 \to \cdots \to \bigwedge T_l \to \tau}\,(\text{T-Var}) \qquad \bigcup_{\tau \in T_1} \Gamma_1^\tau \vdash_{\mathcal{A}} t_1 : \tau' \text{ (for all } \tau' \in T_1)}{\bigcup_{\tau \in T_1} \Gamma_1^\tau, \phi : \bigwedge T_1 \to \cdots \to \bigwedge T_l \to \tau \vdash_{\mathcal{A}} \phi t_1 : \bigwedge T_2 \to \cdots \to \bigwedge T_l \to \tau}\,(\text{T-App}) \quad \vdots \quad \vdots}{\bigcup_{\tau \in T_1} \Gamma_1^\tau \cup \cdots \cup \bigcup_{\tau \in T_1} \Gamma_{l-1}^\tau, \phi : \bigwedge T_1 \to \cdots \to \bigwedge T_l \to \tau \vdash_{\mathcal{A}} \phi t_1 \cdots t_{l-1} : \bigwedge T_l \to \tau}\,(\text{T-App}) \qquad \bigcup_{\tau \in T_l} \Gamma_l^\tau \vdash_{\mathcal{A}} t_l : \tau' \text{ (for all } \tau' \in T_l)}{\bigcup_{\tau \in T_1} \Gamma_1^\tau \cup \cdots \cup \bigcup_{\tau \in T_{l-1}} \Gamma_{l-1}^\tau \cup \bigcup_{\tau \in T_l} \Gamma_l^\tau, \phi : \bigwedge T_1 \to \cdots \to \bigwedge T_l \to \tau \vdash_{\mathcal{A}} \phi t_1 \cdots t_{l-1} t_l : \tau}$$

Where $\Gamma = \bigcup_{i=1}^{k} \{\, x_i : \tau \mid \tau \in T_i \,\}$, the (Rec) rule can be replaced by:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash_{\mathcal{A}} t : q \qquad F x_1 \cdots x_k \longrightarrow t \in \mathcal{R}}{\vdash_{\mathcal{A}} F : \bigwedge T_1 \to \cdots \to \bigwedge T_k \to q}\,(\text{T-NT}) \qquad \bigcup_{\tau \in T_1} \Gamma_1^\tau \vdash_{\mathcal{A}} t_1 : \tau' \text{ (for all } \tau \in T_1)}{\bigcup_{\tau \in T_1} \Gamma_1^\tau \vdash_{\mathcal{A}} F t_1 : \bigwedge T_2 \to \cdots \to \bigwedge T_k \to q}\,(\text{T-App}) \quad \vdots \quad \vdots}{\bigcup_{\tau \in T_1} \Gamma_1^\tau \cup \cdots \cup \bigcup_{\tau \in T_{l-1}} \Gamma_{l-1}^\tau \vdash_{\mathcal{A}} F t_1 \cdots t_{l-1} : \bigwedge T_l \to \bigwedge T_{l+1} \to \cdots \to \bigwedge T_k \to q}\,(\text{T-App}) \qquad \bigcup_{\tau \in T_l} \Gamma_l^\tau \vdash_{\mathcal{A}} t_l : \tau' \text{ (for all } \tau' \in T_l)}{\bigcup_{\tau \in T_1} \Gamma_1^\tau \cup \cdots \cup \bigcup_{\tau \in T_{l-1}} \Gamma_{l-1}^\tau \cup \bigcup_{\tau \in T_l} \Gamma_l^\tau \vdash_{\mathcal{A}} F t_1 \cdots t_{l-1} t_l : \bigwedge T_{l+1} \to \cdots \to \bigwedge T_k \to q}$$

The (Tran) rule can be replaced by:

$$\dfrac{\dfrac{(q, a, \{(i, q_j) \mid 1 \le i \le \mathtt{ar}(a), j \in I_i\}) \in \Delta_{\mathcal{A}}}{\vdash_{\mathcal{A}} a : \bigwedge_{j \in I_1} q_j \to \cdots \to \cdots \bigwedge_{j \in I_{\mathtt{ar}(a)}} q_j \to q} \qquad \Gamma_1 \vdash_{\mathcal{A}} t_1 : q_j \ (\forall j \in I_1)}{\Gamma_1 \vdash_{\mathcal{A}} a t_1 : \bigwedge_{j \in I_2} q_j \to \cdots \to \cdots \bigwedge_{j \in I_{\mathtt{ar}(a)}} q_j \to q}\,(\text{T-App})$$

$$\dfrac{\dfrac{\vdots \qquad\qquad \vdots}{\Gamma_1 \cup \cdots \cup \Gamma_{\mathtt{ar}(a)-1} \vdash_{\mathcal{A}} a t_1 \cdots t_{\mathtt{ar}(a)-1} : \bigwedge_{j \in I_{\mathtt{ar}(a)}} q_j \to q}\,(\text{T-App}) \qquad \Gamma_{\mathtt{ar}(a)} \vdash_{\mathcal{A}} t_{\mathtt{ar}(a)} : q_j \ (\forall j \in I_{\mathtt{ar}(a)})}{\Gamma_1 \cup \cdots \cup \Gamma_{\mathtt{ar}(a)-1} \cup \Gamma_{\mathtt{ar}(a)} \vdash_{\mathcal{A}} F t_1 \cdots t_{\mathtt{ar}(a)-1} t_{\mathtt{ar}(a)} : q}$$

The (App) rule can be replaced verbatim with the (T-APP) rule. We can thus convert a $\Vdash_{\mathcal{A}}$ derivation into a derivation using only the $\vdash$ rules together with (Cut). But we can then just perform cut-elimination. It is known (see *e.g.* [12]) that the *substitution lemma* holds for interesection types— that is if $\Gamma, x : \tau_1, \ldots, x : \tau_k \vdash_{\mathcal{A}} u : \tau$ (with x not occurring in Γ and $\Gamma' \vdash_{\mathcal{A}} x : \tau_i$ for every $1 \le i \le k$, then $\Gamma, \Gamma' \vdash_{\mathcal{A}} t[u/x] : \tau$. Using the substitution lemma, a simple induction on a type-derivation with the rules for $\vdash_{\mathcal{A}}$ plus (Cut) gives us the (Cut)-free type derivation. □

Lemma 3 If $(q, \zeta) \in \mathcal{L}(\boldsymbol{\Phi}^*) \cap \mathbf{GS}$, then $\Vdash_{\mathcal{A}} [\![\zeta]\!]_0 : q$.

Proof. We argue by induction on the structure of a stack ζ for the stronger claim that both: (i) if $\zeta \in \mathcal{L}(\boldsymbol{\Phi}^*)_{(q,i)_Q}$ and $[\![\zeta]\!]_i$ is defined and well-sorted, then $\Vdash_{\mathcal{A}} [\![\zeta]\!]_i : [\![q, Q]\!]_{\mathcal{K}([\![\zeta]\!]_i)}$, and (ii) if $\zeta \in \mathcal{L}(\boldsymbol{\Phi}^*)_q$ and is ground sorted, then $\Vdash_{\mathcal{A}} [\![\zeta]\!]_0 : q$.

(i) Suppose first that $\zeta \in \mathcal{L}(\boldsymbol{\Phi}^*)_{(q,i)_Q}$ with $top(\zeta) = h t_1 \cdots t_\ell$ where $h \in \mathbf{At}$. Suppose also that $[\![\zeta]\!]_i$ is defined and well-sorted. Suppose that the first transition in an accepting run of the automaton (viewed as a tree automaton) is $(q, i)_Q \xrightarrow{h t_1 \cdots t_\ell\ \emptyset} Q_e \in \boldsymbol{\Phi}^{**}$ with $1 \le i \le \ell$. This must mean that there is a

transition $q \xrightarrow{t_i\ Q} Q_e \in \boldsymbol{\Phi}^{\star}$. By Theorem 4 it follows that:

$$\left\{ x \ : \ \tau \ \middle| \ \begin{array}{l} x \in \mathbf{FV}(t_i) \text{ and } \tau \in [\![q_e, Q]\!]_{\mathcal{K}(x)} \\ \text{for } (q_e, \boldsymbol{pos}(x))_Q \in Q_e \end{array} \right\} \Vdash_{\mathcal{A}} t_i \ : \ [\![q, Q]\!]_{\mathcal{K}(t_i)}$$

Suppose that $\mathbf{FV}(t_i) = \{x_1, \ldots, x_k\}$. By Lemma 4 and the definition of $[\![\_]\!]_i$ (for $1 \leq i \leq \ell$),

$$[\![\zeta]\!]_i = t_i[[\![pop(\zeta)]\!]_{\boldsymbol{pos}(x_1)}/x_1, \ldots, [\![pop(\zeta)]\!]_{\boldsymbol{pos}(x_k)}/x_k].$$

By the assumption that the given transition is the first transition in an accepting run of ζ, it must be the case that $pop(\zeta) \in \mathcal{L}(\Delta)_{(q',i')_{Q'}}$ for every $(q', i')_{Q'} \in Q_e$. By the induction hypothesis we thus get that $\Vdash_{\mathcal{A}} [\![pop(\zeta)]\!]_i : [\![q', Q']\!]$ for every $(q', i')_{Q'} \in Q_e$. We may therefore apply the (Cut) rule to get $\Vdash_{\mathcal{A}} [\![\zeta]\!]_i : [\![q, Q]\!]_{\mathcal{K}([\![\zeta]\!]_i)}$, as required.

Now suppose that $i > \ell$. The first transition in an accepting run must be of the form $(q, i)_Q \xrightarrow{ht_1 \cdots t_\ell\ (q, i-\ell)_Q} \emptyset \in \boldsymbol{\Phi}^{\star\star}$. Moreover $[\![\zeta]\!]_i = [\![collapse(\zeta)]\!]_{i-\ell}$. So by the induction hypothesis we again get $\Vdash_{\mathcal{A}} [\![\zeta]\!]_i : [\![q, Q]\!]_{\mathcal{K}([\![\zeta]\!]_i)}$, as required.

(ii) Now suppose that $\zeta \in \mathcal{L}(\boldsymbol{\Phi}^{\star})_q$ and is ground sorted. Suppose further that $top(\zeta) = ht_1 \cdots t_\ell$ for $h \in \mathbf{At}$. Suppose that $[\![\zeta]\!]_0 = ht'_1 \cdots t'_\ell t'_{\ell+1} \cdots t'_{\ell'}$. The first transition in an accepting run must belong to $\boldsymbol{\Phi}^{\star}$ and be of the form $q \xrightarrow{ht_1 \cdots t_\ell\ Q_\alpha} Q_e$. Again by Theorem 4, we must have

$$\left\{ x \ : \ \tau \ \middle| \ \begin{array}{l} x \in \mathbf{FV}(ht_1 \cdots t_\ell) \text{ and } \tau \in [\![q_e, Q_\alpha]\!]_{\mathcal{K}(x)} \\ \text{for } (q_e, \boldsymbol{pos}(x))_Q \in Q_e \end{array} \right\} \Vdash_{\mathcal{A}} ht_1 \cdots t_\ell \ : \ [\![q, Q_\alpha]\!]_{\mathcal{K}(ht_1 \cdots t_\ell)} \qquad (*)$$

and so by the same reasoning as before (as in case **(i)**) we may use the induction hypothesis and (Cut) rule to get $\Vdash_{\mathcal{A}} ht'_1 \cdots t'_\ell : [\![q, Q_\alpha]\!]_{\mathcal{K}(ht'_1 \cdots t'_\ell)}$.

By definition we have $t'_i = [\![collapse(\zeta)]\!]_{i-\ell}$ for $i > \ell$. It must also be the case that for each $(q', j)_{Q'} \in Q_\alpha$ it is the case that $collapse(\zeta) \in \mathcal{L}(\boldsymbol{\Phi}^{\star})_{(q',j)_{Q'}}$. So by the induction hypothesis we must have

$$\Vdash_{\mathcal{A}} t'_i : [\![q', Q']\!]_{\mathcal{K}(t'_i)}$$

for every $(q', i-\ell)_{Q'} \in Q_\alpha$. But by the definition of $[\![q, Q_\alpha]\!]_{\mathcal{K}(ht'_1 \cdots t'_\ell)}$ this is precisely what is required to use (*) and the (App) rule to get

$$\Vdash_{\mathcal{A}} ht'_1 \cdots t'_\ell t'_{\ell+1} \cdots t_{\ell'} : q$$

as required. □