New Topos Constructions for Time Complexity Classes

Jonas Frey
Department of Computer Science
University of Copenhagen, Denmark
jofr@di.ku.dk

Jakob Grue Simonsen
Department of Computer Science
University of Copenhagen, Denmark
simonsen@di.ku.dk

Abstract

We present a method to associate Krivine realizability toposes to superpolynomial time complexity classes. The main technical stepping stone is a proof that the Krivine machine with input and output instructions and continuations (IOKAM) is a *reasonable machine*: counting reduction steps gives a complexity measure that is polynomially equivalent to established models of computation like the Turing machine.

Keywords Realizability, Computational Complexity, Topos Theory

1. Introduction

The *effective topos* (Hyland 1982) is the categorical manifestation of Kleene's *number realizability* (Kleene 1945). It can be considered the universe of computable mathematics, or to be more precise of Markov's *constructive recursive mathematics* (Troelstra and van Dalen 1988), since its constructive internal logic validates statements like *Church's thesis* (stating that all functions of type $\mathbb{N} \to \mathbb{N}$ are recursive) and *Markov's* principle (which is a classical tautology and intuitively states that an unbounded search that does not diverge returns a result).

Variations of the realizability technique in which the computational complexity of the realizers is bounded or at least controlled have been given e.g. by (Crossley et al. 1994; Dal Lago and Hofmann 2011), but these techniques do not give rise to *toposes*.

In the present work we present an approach to associate realizability toposes to complexity classes, which is not based on the traditional Kleene realizability, but on Krivine's *classical realizability*. As a variation of a technique introduced in (Frey), the approach does not impose any conditions on the computational complexity of realizers, but instead the complexity bound plays a role in the definition of the *pole*, which is a parameter in Krivine realizability that is used to define negation.

More specifically, we make the link to asymptotic complexity by augmenting the syntax of Krivine realizability with input and output instructions, which we give an operational semantics in an extension of the Krivine machine called *IOKAM*. To be able to talk about runtime complexity in the framework of the IOKAM, we show that it is a *reasonable machine* in the sense of (van Emde Boas 1990), meaning that it simulates – and is simulated by – established reasonable machine models like the Turing machine or the random access machine (RAM) with polynomial overhead.

To show that the IOKAM is a reasonable machine, we show in Section 3 that the IOKAM can be simulated on the RAM with quadratic overhead, and in Section 4 that Turing machines can be simulated on the IOKAM with linear overhead. For the first part, we define a compilation procedure of IOKAM terms into RAM programs (Table 4, inspired by (Selinger 2003)), and to make the link between IOKAM and RAM execution, we consider as intermediate formalism a variant of the Krivine machine that uses *explicit environments*, and which we abbreviate EIOKAM (Section 2.2). For the simulation of Turing machines on the IOKAM, we adapt a technique of (Accattoli and Dal Lago 2012).

In Section 5 we describe how to associate Krivine realizability toposes to complexity classes, and discuss future work.

To follow the exposition, the reader should ideally have at least a basic understanding about category theory, λ-calculus, and computational models and computational complexity.

1.1 Related work

Hofmann (Hofmann 1997) associates toposes to complexity classes by taking *sheaves* on the monoid of polynomial time computable functions on $\mathbb{N}$. This approach is quite different from ours in that (i) our method is *intensional* in that it allows to associate a set algorithms (in the IOKAM) with a topos, and (ii) we do not expect the toposes arising from our construction to be *Grothendieck toposes.*

Accattoli et al. (Accattoli et al. 2014) show that several abstract machines, including the KAM, can be *distilled* to the Linear Substitution Calculus; in particular, the length of reductions in the KAM without side effects is polynomially related to the Turing machine time usage). The main difference with our work is that, following (Frey), the variant of the KAM we use is equipped with both explicit notions of I/O and continuations, and thus the results of (Accattoli et al. 2014) do not apply without reworking their technical development substantially.

The procedure we use for associating realizability toposes to (sets of) computable functions is from (Frey); the main difference in approach is that in (Frey) toposes are associated to *extensional* specifications (represented by sets of processes closed under observational equivalence), whereas the poles that we associate to complexity classes in the present

CONF 'yy Month d–d, 20yy, City, ST, Country
Copyright © 20yy ACM 978-1-nnnn-nnnn-n/yy/mm. . . $15.00
DOI: http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

(push)	$(tu \star \pi \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star u{\cdot}\pi \ , \iota, \ \omega)$
(pop)	$(\lambda t \star u{\cdot}\pi \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t[u] \star \pi \ , \iota, \ \omega)$
(save)	$(\mathsf{cc} \star t{\cdot}\pi \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star \mathsf{k}_\pi{\cdot}\pi, \iota, \ \omega)$
(restore)	$(\mathsf{k}_\pi \star t{\cdot}\rho \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star \pi \ , \iota, \ \omega)$
(r0)	$(\mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi, 0\iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star \pi \ , \iota, \ \omega)$
(r1)	$(\mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi, 1\iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(u \star \pi \ , \iota, \ \omega)$
(rε)	$(\mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi, \varepsilon, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(v \star \pi \ , \varepsilon, \ \omega)$
(w0)	$(\mathsf{w0} \star t{\cdot}\pi \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star \pi \ , \iota, 0\omega)$
(w1)	$(\mathsf{w1} \star t{\cdot}\pi \quad , \iota, \omega)$	$\rightsquigarrow_{\text{iok}}$	$(t \star \pi \ , \iota, 1\omega)$

Table 1. The IOKAM transition relation.

work should be viewed as *intensional* specifications (because they are not closed under observational equivalence).

1.2 De Bruijn indices

We use de Bruijn indices in Sections 2 and 3, since they are the natural choice of formalism to handle environments in the EIOKAM. Our usage is close to (Pierce 2002, Section 6) (e.g. we start numbering free variables by 0, not 1 like (Barendregt 1984)).

2. Three machines

In this section, we introduce the IOKAM, the EIOKAM, and the RAM.

We use binary strings $\iota, \omega \in \{0,1\}^*$ as input and output data for the (E)IOKAM, whereas in the RAM model, input and output is read and written from and to designated memory areas that store non-negative integers and are addressed by non-negative integers, and are therefore represented by functions $s : \mathbb{N} \to \mathbb{N}$ (we assume $0 \in \mathbb{N}$). When simulating the EIOKAM on the RAM we read and write input and output sequentially, and it therefore is convenient to view the memory areas as *streams*, and use stream-like and function-like notations at the same time. Specifically, we write $s[n]$ for the final segment of a stream $s : \mathbb{N} \to \mathbb{N}$ starting at index n, i.e. the function given by $s[n](i) = s(n+i)$. Since our streams (and also the binary lists of course) store only ever one-digit integers, we write stream and list constructions simply by juxtaposition, i.e. for $\sigma \in \{0,1\}^*$ and $s : \mathbb{N} \to \mathbb{N}$, 0σ and $1s$ are the list and stream that are obtained by adding a 0 or 1 respectively to the beginning of σ and s. We write 0^∞ for the stream/function that is constant 0.

2.1 The IOKAM

The IOKAM is obtained by equipping the Krivine machine (Krivine 2007) with instructions for reading, writing, and termination. The formal definition is as follows.

Definition 2.1 The sets of IOKAM *terms* and *stacks* are inductively defined as follows.

Terms:	$t ::= n \mid \lambda t \mid t\,t \mid \mathsf{cc} \mid \mathsf{k}_\pi \mid \mathsf{r} \mid \mathsf{w0} \mid \mathsf{w1} \mid \mathsf{end}$	$n \in \mathbb{N}$
Stacks:	$\pi ::= \varepsilon \mid t{\cdot}\pi$	t closed

We denote the set of closed terms by Λ, and the set of stacks by Π.

A *process* is a pair of a term t and a stack π, denoted by $t \star \pi$.

An *IOKAM configuration* is a triple $A = (t \star \pi, \iota, \omega)$ of a process $t \star \pi$, and two binary strings $\iota, \omega \in \{0,1\}^*$.

The binary *transition relation* $\rightsquigarrow_{\text{iok}}$ on IOKAM configurations is defined in Table 1. A configuration $(s \star \pi, \iota, \omega)$ is said to be *blocked* if $s \neq \mathsf{end}$ and no rule of $\rightsquigarrow_{\text{iok}}$ applies. $\diamondsuit$

The first three term constructors are those of the untyped λ-calculus, the corresponding rules (push) and (pop) of the IOKAM implement *weak head reduction*. cc is the control operator commonly known as call/cc, when it comes in head position it creates a copy of the stack that is stored in a *continuation term* k_π (rule (save)). We call a term t *continuation-free*, if it contains no subterm of the form k_π.

The remaining term constructors and rules (which are not part of the 'core' Krivine machine) handle reading and writing from and to the input and output strings ι and ω. In the following we will refer to those operations jointly as I/O.

For transitions that do not involve I/O, we will often abbreviate by writing transitions between processes instead of configurations. Thus, e.g.

$$t \star \pi \rightsquigarrow_{\text{iok}} u \star \rho$$

is a shorthand for $(t \star \pi, \iota, \omega) \rightsquigarrow_{\text{iok}} (u \star \rho, \iota, \omega)$ for arbitrary ι and ω. Sometimes we will use the notation suppressing state even for I/O transitions, and record the I/O actions by *labeling* the transition. Thus, we may for example write

$$t \star \pi \overset{\mathsf{r0}}{\rightsquigarrow}_{\text{iok}} u \star \rho$$

to say that $t \star \pi$ passes to $u \star \rho$ while reading a '0'. This is equivalent to

$$\forall \iota, \omega \in \{0,1\}^* \,.\, (t \star \pi, 0{\cdot}\iota, \omega) \rightsquigarrow_{\text{iok}} (u \star \rho, \iota, \omega).$$

In a similar way, we use labels $\mathsf{r1}, \mathsf{r}\varepsilon, \mathsf{w0}, \mathsf{w1}$ to mark transitions that read '1', fail to read on empty input, and write '0' and '1', respectively.

Remark 2.2 Inspection of the rules of $\rightsquigarrow_{\text{iok}}$ shows that the transition relation on configurations is deterministic[1]. The rule that applies to a configuration $(s \star \pi, \iota, \rho)$ is determined by the form of s, and by ι in case $s = \mathsf{r}$.

This means that any configuration $S = (s \star \pi, \iota, \omega)$ has a unique, maximal execution sequence

$$S \rightsquigarrow_{\text{iok}} S_1 \rightsquigarrow_{\text{iok}} \cdots S_j \rightsquigarrow_{\text{iok}} \cdots .$$

If the execution sequence is finite, say, ending at $S_m = (s' \star \pi', \iota', \omega')$, then either $s' = \mathsf{end}$ or S_m is blocked (by definition).

A configuration is blocked if $s \neq \mathsf{end}$ but the stack has too few elements for a rule to match; specifically a configuration is blocked iff one of the following is true:

1. $|\pi| = 0$ and $s \in \{\mathsf{cc}, \mathsf{w0}, \mathsf{w1}\}$,
2. $|\pi| = 0$ and there is a term t with $s = \lambda t$,
3. $|\pi| = 0$ and there is a stack ρ with $s = \mathsf{k}_\rho$,
4. $|\pi| < 3$ and $s = \mathsf{r}$. $\diamondsuit$

Substitution in de Bruijn indices

In Definition 3.1 we translate EIOKAM syntax to IOKAM syntax, which requires simultaneous substitutions, and the natural choice of convention here is to *decrease* higher indices when substituting – e.g. the expression $t[u, v]$ denotes the result of simultaneously substituting u and v for indices 0 and 1, and decreasing the higher indices by 2. The formal definition and central lemma are as follows.

Definition 2.3 The simultaneous substitution $t[u_0, \ldots, u_{n-1}]$ of λ-terms $u_0 \ldots u_{n-1}$ for indices $0, \ldots, n-1$ of a λ-term t

[1] However, the *labeled* transition system on *processes* that we introduced as an informal shorthand is not, since e.g. a process $\mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi$ can make three different transitions labeled by $\mathsf{r0}, \mathsf{r1}, \mathsf{r}\varepsilon$.

is defined as follows.

$$t[u_0,\dots,u_{n-1}] = t[u_0,\dots,u_{n-1}]_0, \quad \text{where}$$

$$i[u_0,\dots,u_{n-1}]_k = \begin{cases} i & \text{if } i<k \\ u_{i-k} & \text{if } k \le i < k+n \\ i-n & \text{if } i \ge k+n \end{cases}$$

$$(st)[u_0,\dots,u_{n-1}]_k = s[u_0,\dots,u_{n-1}]_k\, t[u_0,\dots,u_{n-1}]_k$$

$$(\lambda t)[u_0,\dots,u_{n-1}]_k = \lambda(t[\uparrow^1 u_0,\dots,\uparrow^1 u_{n-1}]_{k+1})$$

$$t[u_0,\dots,u_{n-1}]_k = t \qquad \text{for } t \in \{\mathsf{cc},\mathsf{k}_\pi,\mathsf{r},\mathsf{w0},\mathsf{w1},\mathsf{end}\}$$

$$\uparrow^k_j i = \begin{cases} i & \text{if } i<j \\ i+k & \text{if } i \ge j \end{cases}$$

$$\uparrow^k_j (st) = \uparrow^k_j s \uparrow^k_j t$$

$$\uparrow^k_j (\lambda t) = \lambda(\uparrow^k_{j+1} t)$$

$$\uparrow^k_j t = t \qquad \text{for } t \in \{\mathsf{cc},\mathsf{k}_\pi,\mathsf{r},\mathsf{w0},\mathsf{w1},\mathsf{end}\}$$

Lemma 2.4 *For arbitrary terms $s, u_0, \dots, u_n$ we have:*

1. $\uparrow^l_j \uparrow^k_j s = \uparrow^{k+l}_j s$
2. $(\uparrow^{k+1}_j s)[u]_{j+k} = \uparrow^k_j s$
3. $s[\uparrow^{k+1} u_1,\dots,\uparrow^{k+1} u_n]_{k+1}\,[\uparrow^k u_0]_k = s[\uparrow^k u_0,\dots,\uparrow^k u_n]_k$.

2.2 The EIOKAM

The Krivine machine in the form with explicit environments was first described by Crégut (Crégut 1990), who cites an unpublished 'bruillon' (note) of Krivine from 1985. Krivine himself published a description of his machine only in 2007 (Krivine 2007), and the description is quite informal (e.g., there are no BNF grammars), but he includes continuations (which Crégut and the 1985 note did not). In the definition of the EIOKAM we follow Crégut's presentation (it is slightly simpler than Krivine's description which includes an optimization that is irrelevant for us) and extend it by to handle continuations and I/O. The syntax of the EIOKAM has almost the same terms as the KAM, but adds the syntactic classes of *closures* and *environments*.

Definition 2.5 The terms, closures, environments, and stacks of the EIOKAM are defined by the grammar

Terms: $t ::= n \mid \lambda t \mid t\,t \mid \mathsf{cc} \mid \mathsf{k}_\pi \mid \mathsf{r} \mid \mathsf{w0} \mid \mathsf{w1} \mid \mathsf{end} \quad (n \in \mathbb{N})$
Closures: $c ::= (t, E) \qquad (|E| \ge \mathrm{FI}(t))$
Environments: $E ::= \varepsilon \mid c{\cdot}E$
Stacks: $\pi ::= \varepsilon \mid c{\cdot}\pi$

where in the rule for closures $|E|$ is the length of the environment viewed as list (in particular $|\varepsilon| = 0$), and $\mathrm{FI}(t)$ is the number of *free indices* of the term t, which is defined inductively as follows.

$$\mathrm{FI}(n) = n+1$$
$$\mathrm{FI}(\lambda t) = \mathrm{FI}(t) \dot{-} 1 = \max(\mathrm{FI}(t)-1, 0)$$
$$\mathrm{FI}(tu) = \max(\mathrm{FI}(t), \mathrm{FI}(u))$$
$$\mathrm{FI}(t) = 0 \qquad \text{for } t \in \{\mathsf{cc},\mathsf{r},\mathsf{w0},\mathsf{w1},\mathsf{end},\mathsf{k}_\pi\}$$
◇

Observe that stacks and environments are both lists of closures – however, the distinction is important for the notion of *depth* that is introduced in Section 2.2.1 and used in the proof of Lemma 3.4.

As for the IOKAM, the operational semantics of the EIOKAM is defined as a transition relation on *configurations*, to which we come now.

Definition 2.6 An *EIOKAM configuration* is a quintuple $B = (t, E, \pi, \iota, \omega)$ consisting of a term t, an environment E with $|E| \ge \mathrm{FI}(t)$, a stack π, and binary strings $\iota, \omega \in \{0,1\}^*$.

The *transition relation* $\rightsquigarrow_{\mathrm{eiok}}$ on EIOKAM configurations is defined in Table 2. To further differentiate it, we write $\rightsquigarrow_{\mathrm{eiokv}}$ for transitions of type (v0) and (vS) (called *variable transitions*), and $\rightsquigarrow_{\mathrm{eiok}\bullet}$ for all other types of transitions (called *genuine transitions*).

A configuration $(s, E, \pi, \iota, \omega)$ is said to be *blocked* if $s \neq \mathsf{end}$ and no rule of $\rightsquigarrow_{\mathrm{eiok}}$ applies. ◇

The nine *genuine transition rules* have the same names as the nine rules of the IOKAM and perform analogous operations, with the notable difference of (pop) which performs a substitution on the IOKAM and moves a closure to the environment on the EIOKAM. The environment can be viewed as a mechanism to *delay* substitutions (whose cost can not easily be estimated when implemented on a machine model like the RAM), and instantiate variables only once they come in head position. This is the role of the *variable rules* (v0) and (vS).

Remarks 2.7
1. Inspection of the rules show that the condition $|E| \ge \mathrm{FI}(t)$ on configurations $B = (t, E, \pi, \iota, \omega)$ is preserved by the transition relation.
2. Just like $\rightsquigarrow_{\mathrm{iok}}$, $\rightsquigarrow_{\mathrm{eiok}}$ is deterministic and thus has unique maximal execution sequences which are either infinite, or conclude with end in term position, or a blocked state.
3. Just as the IOKAM, the EIOKAM blocks if there are not enough arguments on the stack (the condition $|E| \ge \mathrm{FI}(t)$ on configurations $(t, E, \pi, \iota, \omega)$ rules out blocking on empty environment, i.e. the EIOKAM can never block with a variable n in head position). ◇

2.2.1 Depth

The concept of *depth* of environments, will be used in the proof of Lemma 3.4 to estimate the length of EIOKAM derivations against IOKAM derivations.

Definition 2.8
- The *depth* of environments is defined by

$$\mathrm{depth}(\varepsilon) = 0$$
$$\mathrm{depth}((t, E){\cdot}F) = 1 + \max(\mathrm{depth}(E), \mathrm{depth}(F))$$

- The *maximal depth* of an EIOKAM configuration $B = (t, E, \pi, \iota, \omega)$ – denoted $\mathrm{mdepth}(B)$ is by definition the maximum of the depths of *all* environments occurring in B (besides E also those in the closures in π, and in the closures in stacks saved in continuation terms k_ρ). ◇

Lemma 2.9 *For configurations $A = (t, E, \pi, \iota, \omega)$, $A' = (u, F, \rho, \iota', \omega')$, B, B', we have*

1. $\mathrm{depth}(E) \le \mathrm{mdepth}(A)$
2. *If $A \rightsquigarrow_{\mathrm{eiokv}} A'$, then we have* $\mathrm{depth}(F) < \mathrm{depth}(E)$ *and* $\mathrm{mdepth}(A') \le \mathrm{mdepth}(A)$
3. *If $B \rightsquigarrow_{\mathrm{eiok}\bullet} B'$, then* $\mathrm{mdepth}(B') \le \mathrm{mdepth}(B) + 1$

Proof. The first claim is obvious by definition of $\mathrm{mdepth}(A)$.

The first half of the second claim follows from the definition of rules (v0), (vS), and of $\mathrm{depth}(-)$. The second half follows since no new environments are created by variable transitions.

For the third claim, observe that only rules that create new environments can increase the maximal depth, and the only such rule is (pop):

$$B = (\lambda s, F, (t, E){\cdot}\pi, \iota, \omega) \rightsquigarrow_{\mathrm{eiok}\bullet} (s, (t, E){\cdot}F, \pi, \iota, \omega) = B'$$

(push)	$(tu$,	$E,$	$\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$E,$	$(u, E)\cdot\pi,$	$\iota,$	$\omega)$
(pop)	$(\lambda s$,	$F,$	$(t, E)\cdot\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(s$,	$(t, E)\cdot F,$	$\pi,$	$\iota,$	$\omega)$
(v0)	$(0$,	$(t, E)\cdot F,$	$\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$E,$	$\pi,$	$\iota,$	$\omega)$
(vS)	$(n{+}1,$	$(t, E)\cdot F,$	$\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(n,$	$F,$	$\pi,$	$\iota,$	$\omega)$
(save)	$(\mathsf{cc}$,	$E,$	$(t, F)\cdot\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$(\mathsf{k}_\pi, \varepsilon)\cdot\pi,$	$\iota,$	$\omega)$
(restore)	$(\mathsf{k}_\pi$,	$E,$	$(t, F)\cdot\rho,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\iota,$	$\omega)$
(r0)	$(\mathsf{r}$,	$E,$	$(t, F)\cdot c\cdot d\cdot\pi,$	$0\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\iota,$	$\omega)$
(r1)	$(\mathsf{r}$,	$E,$	$c\cdot(t, F)\cdot d\cdot\pi,$	$1\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\iota,$	$\omega)$
(rε)	$(\mathsf{r}$,	$E,$	$c\cdot d\cdot(t, F)\cdot\pi,$	$\varepsilon, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\varepsilon,$	$\omega)$
(w0)	$(\mathsf{w0}$,	$E,$	$(t, F)\cdot\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\iota,$	$0\omega)$
(w1)	$(\mathsf{w1}$,	$E,$	$(t, F)\cdot\pi,$	$\iota, \omega)$	$\rightsquigarrow_{\mathrm{eiok}}$	$(t$,	$F,$	$\pi,$	$\iota,$	$1\omega)$

Table 2. The EIOKAM transition relation.

The new environment created here is $(t, E)\cdot F$, and we have

$$\begin{aligned}\mathrm{depth}((t, E)\cdot F) &= 1 + \max(\mathrm{depth}(E), \mathrm{depth}(F)) \\ &\leq 1 + \mathrm{mdepth}(T),\end{aligned}$$

whence $\mathrm{mdepth}(B') \leq 1 + \mathrm{mdepth}(B)$ (since all environments in B' except $(t, E)\cdot F$ are already contained in B). ■

2.3 The RAM

We use the notation of the classical RAM model from (van Emde Boas 1990; Jones 1997).

Informally, a *RAM program* is a sequence of instructions manipulating data stored in memory. We equip the RAM with the following memory structure.

- Six registers `SP` (stack pointer), `EP` (environment pointer), `MH` (upper boundary of allocated memory in the heap area), `IP` (input pointer), `OP` (output pointer), `TV` (temporary value).
- Three infinite arrays $\mathtt{H}[-]$ (for the heap), $\mathtt{I}[-]$ (for input data, read-only during execution), $\mathtt{O}[-]$ (output) of memory cells.

The three memory areas in the arrays above can be encoded in standard representations of the RAM with one memory area only, using techniques involving modular arithmetic (Jones 1997), in a way that increases program size and runtime at most by a constant factor.

In the next two definitions, we introduce the RAM syntax and the transition relation that a *RAM program* induces on the set of *RAM configurations*. We use the following substitution-like notation $s\{i = n\}$ for memory updates:

$$s\{i = n\}(j) = \begin{cases} n & \text{if } i = j \\ s(j) & \text{else,} \end{cases}$$

where $s : \mathbb{N} \to \mathbb{N}$ represents an array of storage cells, and $i, j, n \in \mathbb{N}$.

Definition 2.10 A *value* is either

- a constant $n \in \mathbb{N}$, or
- a sum $r + a$ of a register $r \in \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\}$ and a constant $a \in \mathbb{Z}$, or
- a sum $\mathtt{H}[r{+}n]{+}a$, $\mathtt{I}[r{+}n]{+}a$, or $\mathtt{O}[r{+}n]{+}a$ of a memory cell and a constant, where $r \in \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\}$, $n \in \mathbb{N}$ and $a \in \mathbb{Z}$

A RAM program is a sequence $P = P_0 \ldots P_{n-1}$ of instructions, where each instruction P_i is of one of the following forms.

- *assignment to register:* $r := v$
- *assignment to heap:* $\mathtt{H}[r + n] := v$
- *assignment to output:* $\mathtt{O}[r + n] := v$
- *jump on zero*: $\mathtt{JZ}(v, w)$

where $r \in \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\}$, $n \in \mathbb{N}$, and v, w are values. ◇

Definition 2.11 A *RAM configuration* is a tuple $C = (l, f, h, i, o)$ with $l \in \mathbb{N}$ (representing the program counter), $f : \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\} \to \mathbb{N}$ a valuation on the registers, and $h, i, o : \mathbb{N} \to \mathbb{N}$ representing heap, input, and output.

The *semantics* $[\![v]\!]_C$ of a value v relative to a configuration $C = (l, f, h, i, o)$ is defined as follows:

- $[\![n]\!]_C = n$ for $n \in \mathbb{N}$
- $[\![r + a]\!]_C = \max(f(r){+}a, 0)$ for $r \in \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\}$ and $a \in \mathbb{Z}$.
- $[\![\mathtt{H}[r + n] + a]\!]_C = \max(h(f(r) + n) + a, 0)$, and similarly for $\mathtt{I}[-]$ and $\mathtt{O}[-]$.

Let P be a RAM program. The transition relation $\rightsquigarrow_P$ on RAM configurations $C = (l, f, h, i, o)$ is defined in Table 3.◇

In the following we will sometimes write $\begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}$ for the function $f : \{\mathtt{EP}, \mathtt{SP}, \mathtt{MH}, \mathtt{TV}, \mathtt{IP}, \mathtt{OP}\} \to \mathbb{N}$ with

$$\begin{array}{ll} f(\mathtt{SP}) = \mathrm{sp} & f(\mathtt{TV}) = \mathrm{tv} \\ f(\mathtt{EP}) = \mathrm{ep} & f(\mathtt{IP}) = \mathrm{ip}\ . \\ f(\mathtt{MH}) = \mathrm{mh} & f(\mathtt{OP}) = \mathrm{op} \end{array}$$

Using this notation, the RAM configuration (l, f, h, i, o) can be written as

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o).$$

3. Simulating the IOKAM on the RAM

In this section we compare execution sequences of the IOKAM and the EIOKAM (Section 3.1) and give a simulation of the EIOKAM on the RAM (Section 3.2). Together, we obtain a simulation result of the IOKAM on the RAM (Section 3.3).

3.1 IOKAM and EIOKAM

To relate IOKAM and EIOKAM execution sequences, we define a mapping of EIOKAM syntax (with environments) to IOKAM syntax (without environments). The basic idea is that an EIOKAM closure (t, E) gets mapped to an IOKAM term by inductively and simultaneously substituting the

C	$\rightsquigarrow_P$	$(l+1, f\{r = [\![v]\!]_C\}, h, i, o)$	if	$P_l \equiv r := v$		
C	$\rightsquigarrow_P$	$(l+1, f, h\{f(r){+}n = [\![v]\!]_C\}, i, o)$	if	$P_l \equiv \mathtt{H}[r+n] := v$		
C	$\rightsquigarrow_P$	$(l+1, f, h, i, o\{f(r){+}n = [\![v]\!]_C\})$	if	$P_l \equiv \mathtt{O}[r+n] := v$		
C	$\rightsquigarrow_P$	$([\![w]\!]_C, f, h, i, o)$	if	$P_l \equiv \mathtt{JZ}(v,w)$	and	$[\![v]\!]_C = 0$
C	$\rightsquigarrow_P$	$(l+1, f, h, i, o)$	if	$P_l \equiv \mathtt{JZ}(v,w)$	and	$[\![v]\!]_C \neq 0$

Table 3. The transition relation on RAM configurations induced by a RAM program P

terms induced by the closures in E for the free variables in t. Since closures can also appear *inside* of terms (in continuations), we have to define the mapping by induction over the entire syntax.

Definition 3.1 The mapping from EIOKAM [terms, closures, environments, and stacks] to IOKAM [terms, lists of terms, and stacks] is mutually inductively defined as follows.

$$\begin{array}{lll} n^\top = n & \mathsf{w0}^\top = \mathsf{w0} & (t,E)^\top = t^\top[E^\top] \\ (\lambda t)^\top = \lambda(t^\top) & \mathsf{w1}^\top = \mathsf{w1} & \varepsilon^\top = \varepsilon \\ (tu)^\top = t^\top u^\top & \mathsf{r}^\top = \mathsf{r} & (c{\cdot}E)^\top = c^\top{\cdot}E^\top \\ \mathsf{cc}^\top = \mathsf{cc} & \mathsf{end}^\top = \mathsf{end} & (c{\cdot}\pi)^\top = c^\top{\cdot}\pi^\top \\ \mathsf{k}_\pi^\top = \mathsf{k}_{(\pi^\top)} & & \end{array}$$

The rules for stacks and environments use the same notation for different intended meanings: while EIOKAM stacks are simply translated to IOKAM stacks, EIOKAM environments (for which there is no corresponding syntactic IOKAM class) are translated to 'lists of IOKAM terms', to be simultaneously substituted in the rule for closures.

Finally, EIOKAM configurations are mapped to IOKAM configurations as follows:

$$B = (t, E, \pi, \iota, \omega) \quad \mapsto \quad B^\top = ((t,E)^\top \star \pi^\top, \iota, \omega) \qquad \diamondsuit$$

Observe that we have $t^\top = t$ for continuation-free terms.

It is now straightforward to prove:

Lemma 3.2 *If B is a blocked configuration of the EIOKAM, then $B^\top$ is blocked.*

The following lemma shows that the EIOKAM transition relation may be simulated straightforwardly by the IOKAM transition relation:

Lemma 3.3 *If $B \rightsquigarrow_{eiok\bullet} B'$ is a genuine EIOKAM transition, then $B^\top \rightsquigarrow_{\text{iok}} B'^\top$ by an IOKAM rule of the same name. If $B \rightsquigarrow_{\text{eiokv}} B'$, then $B^\top = B'^\top$.*

For the next lemma – which is the main lemma of the subsection – recall from Remarks 2.2 and 2.7 that IOKAM and EIOKAM configurations admit unique maximal execution sequences.

Lemma 3.4 *Let t be a term not containing any continuations k_π, and let $\iota \in \{0,1\}^*$. Then we have*

$$(t, \varepsilon, \varepsilon, \iota, \varepsilon) \rightsquigarrow^n_{\text{eiok}} (\mathsf{end}, \_, \_, \_, \omega)$$

if and only if

$$(t \star \varepsilon, \iota, \varepsilon) \rightsquigarrow^m_{\text{iok}} (\mathsf{end} \star \_, \_, \omega),$$

that is, t executed with input ι on the IOKAM terminates iff it does so on the EIOKAM, in which case the output is the same. Moreover, $m \leq n \leq m(m+3)/2$.

Proof. First assume that we have an EIOKAM execution sequence

$$\begin{aligned} (t, \varepsilon, \varepsilon, \iota, \varepsilon) = B_0 \rightsquigarrow_{\text{eiok}} B_1 \rightsquigarrow_{\text{eiok}} \dots \\ \dots \rightsquigarrow_{\text{eiok}} B_n = (\mathsf{end}, E, \pi, \iota', \omega). \end{aligned}$$

Then by Lemma 3.3, applying $(-)^\top$ to all components yields an IOKAM execution sequence

$$\begin{aligned} (t^\top \star \varepsilon, \iota, \varepsilon) = B_0^\top \rightsquigarrow^{\leq 1}_{\text{iok}} B_1^\top \rightsquigarrow^{\leq 1}_{\text{iok}} \dots \\ \dots \rightsquigarrow^{\leq 1}_{\text{iok}} B_n^\top = (\mathsf{end} \star \pi^\top, \iota', \omega) \end{aligned}$$

whose length is obviously bounded by n.

Conversely assume that we have an IOKAM execution sequence

$$\begin{aligned} (t \star \varepsilon, \iota, \varepsilon) = A_0 \rightsquigarrow_{\text{iok}} A_1 \rightsquigarrow_{\text{iok}} \dots \\ \dots \rightsquigarrow_{\text{iok}} A_m = (\mathsf{end} \star \_, \_, \omega) \end{aligned} \tag{3.1}$$

and consider consider the following enumeration scheme of the unique maximal EIOKAM execution sequence of $B_0^0 = (t, \varepsilon, \varepsilon, \iota, \varepsilon)$ with pairs of numbers.

$$\begin{array}{l} B_0^0 \\ \vdots \\ \rightsquigarrow_{\text{eiok}\bullet} B_k^0 \rightsquigarrow_{\text{eiokv}} \dots \rightsquigarrow_{\text{eiokv}} B_k^{j_k} \\ \rightsquigarrow_{\text{eiok}\bullet} B_{k+1}^0 \rightsquigarrow_{\text{eiokv}} \dots \rightsquigarrow_{\text{eiokv}} B_{k+1}^{j_{k+1}} \\ \vdots \end{array} \tag{3.2}$$

Variable transitions $\rightsquigarrow_{\text{eiokv}}$ increase the superscript, and genuine transitions $\rightsquigarrow_{\text{eiok}\bullet}$ reset the superscript to 0 and increase the subscript. Each genuine transition is potentially followed by several variable transitions, but the number of successive variable transitions can be bounded as follows. Firstly $\text{mdepth}(B_k^0) \leq k$, since $\text{mdepth}(B_0^0) = 0$ and only genuine transitions can increase the maximal depth, at most by 1 (Lemma 2.9-2,3). Secondly, denoting the active environment in B_k^i by E_k^i, we have

$$\begin{aligned} \text{mdepth}(B_k^0) &\geq \text{depth}(E_k^0) && \text{by 2.9-1} \\ &> \text{depth}(E_k^1) && \text{by 2.9-2} \\ &> \dots \\ &> \text{depth}(E_k^{j_k}) \geq 0 \end{aligned}$$

and since the left hand side is bounded by k, we can deduce that $j_k \leq k$, i.e. the k-th genuine transition is followed by at most k variable transitions.

From Lemma 3.3 and $B_0^{0\top} = A_0$ it follows that $B_k^{i\top} = A_k$ for all appropriate i, k, i.e. (3.2) is projected to an initial segment of (3.1), whereby EIOKAM configurations in the same line in (3.2) are mapped to the same IOKAM configuration. The attained initial segment of (3.1) cannot be a strict one, since the lines of (3.2) are finite, and $(-)^\top$

maps blocked EIOKAM configurations to blocked IOKAM configurations by Lemma 3.2. Thus, (3.1) has to terminate with a configuration B_m^i for some i. This B_m^i has end in term position (as it cannot be blocked), and from ${B_m^i}^\top = A^m$ we can deduce that B_m^i has ω in output position, as claimed.

Finally, the length of the sequence

$$\begin{array}{l} \qquad\qquad B_0^0 \\ \qquad\qquad \vdots \\ \rightsquigarrow_{\mathrm{eiok}\bullet} B_k^0 \rightsquigarrow_{\mathrm{eiokv}} \dots \rightsquigarrow_{\mathrm{eiokv}} B_k^{j_k} \\ \qquad\qquad \vdots \\ \rightsquigarrow_{\mathrm{eiok}\bullet} B_m^0 \rightsquigarrow_{\mathrm{eiokv}} \dots \rightsquigarrow_{\mathrm{eiokv}} B_m^i \end{array}$$

can be estimated by

$$m + \sum_{k<m} j_k + i \le m + \sum_{k \le m} j_k \le m + \sum_{k \le m} k = m(m+3)/2. \qquad \blacksquare$$

3.2 EIOKAM and RAM

To relate EIOKAM and RAM, we define a compilation procedure that translates continuation-free EIOKAM terms (which coincide with continuation-free IOKAM terms) to RAM programs. The approach is inspired by Selinger (Selinger 2003), but using linked lists instead of arrays to represent closures (hence, our construction admits more efficient (push), but Selingers admits more efficient variable lookup).

An IOKAM configuration $(t, E, \pi, \iota, \omega)$ is mapped to a RAM configuration as follows:

- t is represented by the state of the *program counter*, which points to an instruction/line in the code area.
- E and π are both lists of closures on the abstract machine. On the RAM, they are represented as linked lists on the *heap* $\mathtt{H}[-]$. A node in any of these lists is a *frame* consisting of thee cells $\mathtt{H}[i], \mathtt{H}[i+1], \mathtt{H}[i+2]$, where $\mathtt{H}[i]$ points to the beginning of the compilation $[\![t]\!]$ of t in the code area, $\mathtt{H}[i+1]$ points to an environment in the heap, and $\mathtt{H}[i+1]$ points to the remainder of the list. Values of 0 for $\mathtt{H}[i+1]$ or $\mathtt{H}[i+2]$ mean that the corresponding part of the data structure is empty (to make this work, we put the first frame at address 1, leaving $\mathtt{H}[0]$ always uninitialized).

  The registers $\mathtt{EP}$ and $\mathtt{SP}$ point to the current environment and stack, respectively.
- The remaining input ι is represented by a separate memory area $\mathtt{I}[-]$ (where the input is written at the beginning as a string of 0's and 1's, terminated by a 2), and a pointer $\mathtt{IP}$ which keeps track of how much of the input has already been read.

The exception for the 'subterm principle' mentioned above is the $\mathfrak{c}$ rule. This rule *does* create a new term of the form k_π which is not a subterm of anything on the left. The compilation of $\mathfrak{c}$ has to push a frame onto the stack that represents k_π, where π is the current stack. Do do this, we change the meaning of the first and second cell of the frame: in the first cell we write the address of a special routine that restores continuations, and in the second cell we write the address of π (and not the address of an environment as we would usually do – the term k_π does not require an environment since it is closed). In the third cell we write the address of π as well, since that becomes the remainder of the stack.

The compilation $[\![t]\!]$ of terms t is defined in Table 4. As can be seen at the beginning, $[\![t]\!]$ consists of a part $(\!|t|\!)$ which is defined inductively on the term structure, preceded by five instructions of 'global' code to handle restoring of continuations. We call this global part the *prelude*. The prelude is marked by the symbolic label $\boldsymbol{K}$ which we use to refer to it in jumps. It is to be understood that these symbolic labels are replaced by line numbers during linking; we use upper case letters for global labels and lower case letters for local labels.

To understand the compilation process, observe that except for the $\mathfrak{c}$ rule, the execution rules for the EIOKAM (Table 2) never creates new terms – terms occurring in closures or in head position on the right are almost always subterms of terms occurring on the left. This means that we do not have to create terms dynamically during execution, and can simply use addresses of code corresponding to subterms of the original program where the operational semantics uses terms.

We have the following lemma:

Lemma 3.5 *Assume t is a continuation-free term, that $\iota \in \{0,1\}^*$, and let $P = [\![t]\!]$.*

We have

$$\exists n\,.\,(t, \varepsilon, \varepsilon, \iota, \varepsilon) \rightsquigarrow^n_{\mathrm{eiok}} (\mathsf{end}, E, \pi, \iota', \omega)$$

if and only if there exists an $m \in \mathbb{N}$ such that

$$\left(\boldsymbol{S}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, 0^\infty, \iota 2 0^\infty, 0^\infty\right) \rightsquigarrow^m_P \left(\boldsymbol{E}, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & |\omega| \end{bmatrix}, h, i, \omega^r 0^\infty\right),$$

i.e. the term t executed on the EIOKAM with input ι terminates successfully in n steps with output ω iff the program $[\![t]\!]$ executed on the RAM with input ι terminates successfully after m steps with the same output ω.

Moreover, the length of the execution sequences can be estimated by

$$n \le m \le 13n + 1.$$

3.3 IOKAM and RAM

We now combine the simulation results of Sections 3.1 and 3.2 into a simulation of the IOKAM on the RAM.

Lemma 3.6 *Assume t is a continuation-free term, and let $P = [\![t]\!]$. Then for strings $\iota, \omega \in \{0,1\}^*$ we have*

$$\exists n\,.\,(t \star \varepsilon, \iota, \varepsilon) \rightsquigarrow^n_{\mathrm{iok}} (\mathsf{end} \star \_\_, \_\_, \omega)$$

if and only if there exists an $m \ge 0$ such that

$$\left(\boldsymbol{S}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, 0^\infty, \iota 2 0^\infty, 0^\infty\right) \rightsquigarrow^m_P \left(\boldsymbol{E}, \begin{bmatrix} \_ & \_ \\ \_ & \_ \\ \_ & |\omega| \end{bmatrix}, \_\_, \_\_, \omega^r 0^\infty\right)$$

That is, the term t executed on the IOKAM with input ι terminates successfully in n steps with output ω iff the program $[\![t]\!]$ executed on the RAM with input ι terminates successfully after m steps with the same output ω.

Moreover,

$$n \le m \le \frac{13}{2}n^2 + \frac{39}{2}n + 1.$$

Proof. Direct combination of Lemmas 3.4 and 3.5. $\blacksquare$

4. Simulating Turing machines on IOKAM

The final stepping stone is to simulate Turing machines on the IOKAM.

Recall that *weak head reduction* in λ-calculus is leftmost outermost β-reduction that does not contract redexes below abstractions. We denote weak head reduction with the arrow

```
⟦t⟧   ≡ K : JZ(SP, F)
            TV := SP
            SP := EP
            EP := H[TV + 1]
            JZ(0, H[TV])
        S : ⦇t⦈
        E : (success)
        F : (failure)

⦇λt⦈  ≡     JZ(SP, F)
            H[MH] := H[SP]
            H[MH + 1] := H[SP + 1]
            H[MH + 2] := EP
            EP := MH
            MH := MH + 3
            SP := H[SP + 2]
            ⦇t⦈

⦇tu⦈  ≡     H[MH] := j
            H[MH + 1] := EP
            H[MH + 2] := SP
            SP := MH
            MH := MH + 3
            ⦇t⦈
        j : ⦇u⦈

⦇0⦈   ≡     TV := H[EP]
            EP := H[EP + 1]
            JZ(0, TV)
```

```
⦇cc⦈    ≡ H[MH] := K
          H[MH + 1] := H[SP + 2]
          H[MH + 2] := H[SP + 2]
          JZ(SP, F)
          TV := SP
          SP := MH
          MH := MH + 3
          EP := H[TV + 1]
          JZ(0, H[TV])

⦇w0⦈    ≡ JZ(SP, F)
          O[OP] := 0
          OP := OP + 1
          TV := H[SP]
          EP := H[SP + 1]
          SP := H[SP + 2]
          JZ(0, TV)

⦇w1⦈    ≡ JZ(SP, F)
          O[OP] := 1
          OP := OP + 1
          TV := H[SP]
          EP := H[SP + 1]
          SP := H[SP + 2]
          JZ(0, TV)

⦇n + 1⦈ ≡ EP := H[EP + 2]
          ⦇n⦈
```

```
⦇end⦈ ≡     JZ(0, E)

⦇r⦈   ≡     JZ(SP, F)
            TV := H[SP + 2]
            JZ(TV, F)
            JZ(H[TV + 2], F)
            JZ(I[IP], m)
            SP := H[SP + 2]
            JZ(I[IP] − 1, n)
            SP := H[SP + 2]
            TV := H[SP]
            EP := H[SP + 1]
            SP := H[SP + 2]
            JZ(0, TV)
        n : TV := H[SP]
            EP := H[SP + 1]
            SP := H[SP + 2]
            SP := H[SP + 2]
            IP := IP + 1
            JZ(0, TV)
        m : TV := H[SP]
            EP := H[SP + 1]
            SP := H[SP + 2]
            SP := H[SP + 2]
            SP := H[SP + 2]
            IP := IP + 1
            JZ(0, TV)
```

The compilation ⟦t⟧ of a term consists of a global context and a code segment ⦇t⦈ that is defined by induction on the term structure. The global context is annotated by global labels $\boldsymbol{S}, \boldsymbol{K}, \boldsymbol{E}, \boldsymbol{F}$. Here $\boldsymbol{S}$ is the entry point for execution and is used to initialize the program counter. $\boldsymbol{K}$ points to a code segment that is used to restore continuations. $\boldsymbol{E}$ and $\boldsymbol{F}$ serve as final values for the program counter, and denote orderly termination and failure (blocking on empty context), respectively.

Table 4. Compilation

symbol $\rightsquigarrow_{\text{wh}}$, thus, a weak head reduction step is a β-reduction step of the form

$$(\lambda x.s)tt_1 \dots t_n \rightsquigarrow_{\text{wh}} s[t/x]t_1 \dots t_n.$$

Accattoli and Dal Lago (Accattoli and Dal Lago 2012, Section 6) give a simulation of Turing machines in λ-calculus with weak head reduction and unit cost measure that only incurs linear time overhead: If a Turing machine halts in n steps, the corresponding term in λ-calculus will reach a weak head normal form via weak head reduction in $O(n)$ steps.

In this section we adapt the approach of Accatoli and Dal Lago to show how Turing machines can be simulated efficiently on the IOKAM. There are two obstructions:

- Although the execution relation of the Krivine machine is very similar to weak head reduction, there is no one-to-one correspondence between steps on the Krivine machine and weak head reduction steps. For example, setting $I = \lambda x.x$ we have $II \rightsquigarrow_{\text{wh}} I$ in one step, but $II \star \varepsilon \rightsquigarrow^2_{\text{iok}} I$ in two steps. Even worse, we have $III \rightsquigarrow_{\text{wh}} II$ in one step as well, but the KAM configuration $III \star \varepsilon$ cannot be reduced to $II \star \varepsilon$.
- While Accattoli and Dal Lago encode input and output as terms in their computation model, the IOKAM reads and writes data using side effects.

The first item is handled by Lemma 4.1 which shows how the number of steps on the Krivine machine can be bounded by a linear function the number of weak head reduction steps.

To treat for the second item, we define terms that take care of reading and preparing an initial configuration, and of extracting the result of a final configuration for subsequent writing.

4.1 A sequence of ancillary lemmas

The following lemma – which is a refinement of (Krivine 2009, Lemma 14) – states that the length of IOKAM execution sequences can be bounded by a linear function in the length of weak head reduction sequences.

The substitution in the phrasing of the lemma is necessary since in the assumption it is important that the reduct starts with a *variable*, but the conclusion is about IOKAM execution, which is only defined for *closed* terms.

Lemma 4.1 *Assume that $s[k], u_n[k], \dots, u_1[k]$ are terms containing at most a free variable k, that $v_m, \dots, v_1$ are* *closed terms, and that*

$$s[k]u_n[k]\dots u_1[k] \quad \rightsquigarrow^{l}_{\mathrm{wh}} \quad kv_m\dots v_1$$

for some $l \in \mathbb{N}$. Then for any stack π and closed term t we have

$$s[t]\star u_n[t]\cdot\ldots\cdot u_1[t]\cdot\pi \quad \rightsquigarrow^{2l+m-n}_{\mathrm{iok}} \quad t\star v_m\cdot\ldots\cdot v_1\cdot\pi.$$

Following (Accattoli and Dal Lago 2012), we use the following encoding of binary strings $s \in \{0,1\}^*$.

$$\lceil 0\cdot\sigma\rceil = \lambda xyz.x\lceil\sigma\rceil$$
$$\lceil 1\cdot\sigma\rceil = \lambda xyz.y\lceil\sigma\rceil$$
$$\lceil\varepsilon\rceil = \lambda xyz.z$$

Furthermore, we need the *Turing fixed-point combinator*

$$\Theta = (\lambda xy.y(xxy))(\lambda xy.y(xxy))$$

which satisfies

$$\Theta t \rightsquigarrow^{2}_{\mathrm{wh}} t(\Theta t).$$

Because of our earlier choices of convention, strings in CPS must be reverse encoded. This is done as follows. For a binary string $\sigma = \sigma_1\dots\sigma_n \in \{0,1\}^*$ we write $\sigma^r = \sigma_n\dots\sigma_1$ for the reversed string. With the terms V, W, B defined as

$$V = \lambda wkst.s(\lambda s'.wks'(\lambda xyz.xt))(\lambda s'.wks'(\lambda xyz.yt))(kt)$$
$$W = \Theta V$$
$$B = \lambda ks.Wks\lceil\varepsilon\rceil,$$

we can prove the following lemma.

Lemma 4.2 *Let $\sigma = \sigma_1\dots\sigma_n \in \{0,1\}^*$. We then have $Bk\lceil\sigma\rceil \rightsquigarrow^{10n+11}_{\mathrm{wh}} k\lceil\sigma^r\rceil$.*

We now recall the relevant part of Accattoli and Dal Lago's treatment of simulating Turing machines in λ-calculus with weak head reduction. Our notation is based on theirs, but simplified (omitting some subscripts and superscripts).

Throughout the rest of this section we assume that $\mathcal{M}$ is a Turing machine computing a partial function $f : \{0,1\}^* \rightharpoonup \{0,1\}^*$, and that $g : \{0,1\}^* \rightharpoonup \mathbb{N}$ is the partial function that assigns to each $\iota \in \{0,1\}^*$ the number of computation steps that $\mathcal{M}$ performs on input ι, whenever it terminates.

Accattoli and Dal Lago define an encoding of general Turing machine configurations in λ-terms, but for us it suffices to know that for binary strings $\iota, \omega \in \{0,1\}^*$ there are uniquely determined initial and final configurations that represent input ι and output ω. We denote these configurations by C_ι and D_ω, respectively.

Based on these encodings, Accattoli and Dal Lago define terms I, T, F that simulate $\mathcal{M}$ in CPS style, in the following sense.

Lemma 4.3 *There are closed λ-terms I, T, F such that for any binary strings ι, ω and any term t we have:*

1. *$It\lceil\iota\rceil \rightsquigarrow^{*}_{\mathrm{wh}} tC_\iota$ in $O(|\iota|)$ steps.*
2. *If $f(\iota)$ is defined, then $TtC_\iota \rightsquigarrow^{*}_{\mathrm{wh}} tD_{f(\iota)}$ in $O(g(\iota))$ steps.*
3. *If $f(\iota)$ is undefined, then $Tt\lceil\iota\rceil$ diverges.*
4. *$FtD_\omega \rightsquigarrow^{*}_{\mathrm{wh}} t\lceil\omega\rceil$ in $O(|\omega|)$ steps.*

Proof. Lemmas 33-35 in (Accattoli and Dal Lago 2012). ∎

Composing the terms I, T, F yields a full simulation of $\mathcal{M}$:

Lemma 4.4 *Let $M = \lambda ks.I(T(Fk))s$.*

Then, for any term t and input string $\iota \in \{0,1\}^*$, we have

$$Mt\lceil\iota\rceil \rightsquigarrow^{*}_{\mathrm{wh}} t\lceil f(\iota)\rceil$$

in a number of steps in $O(|\iota|+g(\iota))$ whenever $f(\iota)$ is defined. If $f(\iota)$ is undefined, $Mt\lceil\iota\rceil$ diverges.

Proof. If $f(\iota)$ is defined, we have

$$\begin{array}{lll} Mt\lceil\iota\rceil = (\lambda ks.I(T(Fk))s)t\lceil\iota\rceil & & \\ \rightsquigarrow^{2}_{\mathrm{wh}} I(T(Ft))\lceil\iota\rceil & & \\ \rightsquigarrow^{*}_{\mathrm{wh}} T(Ft)C_\iota & \text{in } O(|\iota|) \text{ steps} & \text{by Lemma 4.3-1} \\ \rightsquigarrow^{*}_{\mathrm{wh}} FtD_{f(\iota)} & \text{in } O(g(\iota)) \text{ steps} & \text{by Lemma 4.3-2} \\ \rightsquigarrow^{*}_{\mathrm{wh}} t\lceil f(\iota)\rceil & \text{in } O(|f(\iota)|) \text{ steps} & \text{by Lemma 4.3-4} \end{array}$$

Thus, the entire reduction above has length $O(|\iota| + g(\iota) + |f(\iota)|)$. Note that $|f(\iota)| \leq |\iota|+g(\iota)$ – the length of the output of a Turing machine is bounded by the length of the input and the number of steps, because the symbols in the output that are not in already in the input have to be written one by one. Hence, the number of steps in the reduction is $O(|\iota| + g(\iota))$, as claimed.

If $f(\iota)$ is undefined, then the execution diverges by Lemma 4.3-3. ∎

Accatoli and Dal Lago prove a similar result to the above (Accattoli and Dal Lago 2012, Lemma 36), but we cannot use it directly since they instantiate the continuation variable and thereby give up compositionality.

The following two lemmas describe how to combine the term M with terms for reading and writing to obtain a simulation of Turing machines on the IOKAM.

Reading is done using the terms

$$U = \lambda vks\,.\,\mathsf{r}(vk(\lambda wxyz\,.\,ws))(vk(\lambda wxyz\,.\,xs))(ks)$$
$$R = \lambda k\,.\,\Theta Uk\lceil\varepsilon\rceil,$$

as shown in the following lemma.

Lemma 4.5 *For any term $\iota \in \{0,1\}^*$ and $t \in \Lambda$ we have*

$$(Rt\star\varepsilon, \iota, \varepsilon) \rightsquigarrow^{*}_{\mathrm{iok}} (t\lceil\iota^r\rceil\star\varepsilon, \varepsilon, \varepsilon).$$

in a number of steps that is linear in $|\iota|$.

Writing is done using the term

$$W = \Theta(\lambda wks.s(\lambda s'.\mathtt{w0}wks')(\lambda s'.\mathtt{w1}wks')k),$$

as described in the following lemma.

Lemma 4.6 *For t a term and $\omega \in \{0,1\}^*$ we have*

$$(Wt\lceil\omega\rceil\star\varepsilon, \varepsilon, \varepsilon) \rightsquigarrow^{*}_{\mathrm{iok}} (t\star\varepsilon, \varepsilon, \omega^r)$$

in a number of steps that is linear in $|\omega|$,

4.2 Main simulation result

Using the ancillary lemmas, we can prove the main result.

Theorem 4.7 *Let $N = R(B(M(B(W\mathsf{end}))))$. Then for any $\iota \in \{0,1\}^*$ the execution sequence of $(N\star\varepsilon, \iota, \varepsilon)$ is finite if and only if $f(\iota)$ is defined, and in this case we have*

$$(N\star\varepsilon, \iota, \varepsilon) \rightsquigarrow_{\mathrm{iok}} (\mathsf{end}\star\varepsilon, \varepsilon, f(\iota))$$

in a number of steps that is linear in $|\iota| + g(\iota)$.

Proof. The execution sequence is

$$\begin{array}{ll} (R(B(M(B(W\mathsf{end}))))\star\varepsilon, \iota, \varepsilon) & \\ \rightsquigarrow^{*}_{\mathrm{iok}} (B(M(B(W\mathsf{end})))\lceil\iota^r\rceil\star\varepsilon, \varepsilon, \varepsilon) & \text{by Lemma 4.5} \\ \rightsquigarrow^{*}_{\mathrm{iok}} (M(B(W\mathsf{end}))\star\lceil\iota\rceil, \varepsilon, \varepsilon) & \text{by Lemma 4.2} \\ \rightsquigarrow^{*}_{\mathrm{iok}} (B(W\mathsf{end})\star\lceil f(\iota)\rceil, \varepsilon, \varepsilon) & \text{by Lemma 4.4} \\ \rightsquigarrow^{*}_{\mathrm{iok}} (W\mathsf{end}\star\lceil f(\iota)^r\rceil, \varepsilon, \varepsilon) & \text{by Lemma 4.2} \\ \rightsquigarrow^{*}_{\mathrm{iok}} (\mathsf{end}\star\varepsilon, \varepsilon, f(\iota)) & \text{by Lemma 4.6} \end{array}$$

whenever $f(\iota)$ is defined, otherwise it diverges in the step with M in head position. The first two transitions take a number of steps that is linear in $|\iota|$. The number of steps in the third transition is linear in $|\iota|+g(\iota)$, and the last two are linear in $|f(\iota)|$, which in turn is bounded by $|\iota|+g(\iota)$. This gives a total number of steps that is linear in $|\iota|+g(\iota)$. ■

5. Toposes from complexity classes

We now present a construction that associates Krivine realizability toposes to time complexity classes. Since the IOKAM as presented in Section 2.1 computes functions from $\{0,1\}^*$ to $\{0,1\}^*$, it is most adequate to consider *functional complexity classes* (Papadimitriou 1994, Section 10.3), which we identify with subsets of the set $\mathrm{Fun}(\{0,1\}^*, \{0,1\}^*)$ of endofunctions on $\{0,1\}^*$ (the more well-known classes of *decision problems* are then obtained by restricting to functions with codomain $\{0,1\}^*$).

In the following we will use **FP** – the class of functions computable in polynomial time – as running example, but the arguments and constructions generalize to any *super-polynomial* class, i.e. a set of functions defined by a set of time bounds closed under taking polynomials (e.g., the class **FEXP** of functions computable in exponential time).

As the IOKAM is a reasonable model, we can define **FP** as the set of functions $f : \{0,1\}^* \to \{0,1\}^*$ that are computable by an IOKAM process $t\star\pi$ in polynomial time, in the sense that

$$\forall\iota\in\{0,1\}^*\,.\,(t\star\pi,\iota,\varepsilon)\rightsquigarrow^*_{\mathrm{iok}}(\mathsf{end},\_,\_,f(\iota)) \qquad (5.1)$$

in a number of steps that can be bounded by a polynomial $P\in\mathbb{N}[X]$ in $|\iota|$. Corresponding to the *extensional* class **FP** of polynomial time *functions*, we can even consider th *intentional* class $\mathfrak{P}$ of polynomial time *algorithms*, which we define as the set of IOKAM processes that terminate in polynomial time in the length of the input, i.e.

$$\mathfrak{P}=\{t\star\pi\mid\exists P\in\mathbb{N}\,\forall\iota\in\{0,1\}^*$$
$$(t\star\pi,\iota,\varepsilon)\rightsquigarrow^{\le P(|\iota|)}_{\mathrm{iok}}(\mathsf{end}\star\_,\_,\_)\}$$

Writing $t\star\pi\rhd f$ as a shorthand for (5.1), the statement that **FP** is the class of functions that can be implemented in polynomial time on the IOKAM can now be expressed as

$$\mathbf{FP}=\{f:\{0,1\}^*\to\{0,1\}^*\mid\exists t\star\pi\in\mathfrak{P}\,.\,t\star\pi\rhd f\}.$$

In the following, we explain how to build a Krivine realizability model around the set $\mathfrak{P}$ of polynomial time algorithms. To start we recall basic facts and ideas about realizability and realizability toposes, and we outline the construction of Krivine realizability toposes as presented in (Frey).

5.1 Realizability and realizability toposes

Realizability was introduced in (Kleene 1945) as a proof theoretic model building tool for constructive logical theories like Heyting arithmetic. Formalizing ideas of the so-called *Brouwer-Heyting-Kolmogov* (BHK) interpretation (van Atten 2014), the role of *truth values* in realizability is taken by sets of so-called *realizers*, where realizers are to be seen as abstractions of mathematical proofs. Depending on the desired properties of the ensuing model, various kinds of mathematical objects can be chosen as realizers, for example the set of closed terms of the untyped λ-calculus.

Realizability toposes – the category theoretic incarnation of realizability – were introduced in (Hyland et al. 1980; Hyland 1982), building on Lawvere's insights in the connection between topos theory and logic (Lawvere 1970). Informally, a (Kleene) realizability topos can be understood as a completion of the category **Set** of sets and functions w.r.t. a *non-standard logic*, whose *semantic predicates* on a set J are functions $\varphi,\psi : J\to P(\mathcal{A})$ into the power set of the set $\mathcal{A}$ of realizers. The set $P(\mathcal{A})^J$ of semantic predicates on J carries a preorder $\le$ whose definition is a formalization of the BHK interpretation of logical *implication* – intuitively, $\varphi\le\psi$ if there exists a construction (itself represented by a realizer) that transforms elements of $\varphi(j)$ into elements of $\psi(j)$ uniformly in j. Formally, the assignment $J\mapsto(P(\mathcal{A})^J,\le)$ can be viewed as a *functor* of type $\mathbf{Set}^{\mathrm{op}}\to\mathbf{Ord}$ (i.e. it contravariantly maps sets to preorders). This functor satisfies a list of axioms that make it a *tripos* in the terminology of (Hyland et al. 1980), from which the realizability topos $\mathbf{Set}[\mathcal{A}]$ is obtained via the *tripos-to-topos construction.*

5.2 Krivine realizability triposes and toposes

In the beginning of the 2000s, Krivine (Krivine 2001, 2003, 2009) introduced a realizability interpretation validating *classical logic*, whose first category theoretic formulation was given in (Streicher 2013) (in the following exposition we follow the presentation of (Frey) which is more direct).

Contrary to Kleene realizability, whose realizers are an abstraction of the notion of 'proof', Krivine realizability is built around a duality of *proofs* and *refutations*, which are represented by *closed terms*, and *stacks* (of the IOKAM in our case), respectively. The interplay of proofs and refutations is orchestrated by a set of processes $\perp\!\!\!\perp\subseteq\Lambda\times\Pi$ known as *pole*, which plays the role of 'falsity' in a negative translation, or the *type of responses* in (Streicher and Reus 1998). The strength and flexibility of Krivine realizability lies in the variability of the pole, which only has to satisfy the following saturatedness condition.

Definition 5.1 A *pole* is a set $\perp\!\!\!\perp\subseteq\Lambda\times\Pi$ of processes which is closed under inverse *I/O-free* reduction, i.e. for all processes $t\star\pi$ and $u\star\rho$ we have

$$t\star\pi\rightsquigarrow_{\mathrm{iok}}u\star\rho\ \wedge\ u\star\rho\in\perp\!\!\!\perp\ \Rightarrow\ t\star\pi\in\perp\!\!\!\perp,$$

where $\rightsquigarrow_{\mathrm{iok}}$ is one of (push), (pop), (save), (restore). ◇

A *truth value* in Krivine realizability is a set $S\subseteq\Pi$ of stacks[2], and for a fixed pole $\perp\!\!\!\perp$ we call a term t a *realizer* of S – and write $t\Vdash S$ – if $\forall\pi\in S\,.\,t\star\pi\in\perp\!\!\!\perp$.

Semantic predicates on a set J are again families $\varphi,\psi : J\to P(\Pi)$ of truth values, and as for Kleene realizability, the semantic predicates on J carry a *preorder structure*, which in this case is defined in terms of the pole, and a special class of terms called *proof-like.*

Definition 5.2
- The set of *proof-like terms* is the set $\mathsf{PL}\subseteq\Lambda$ of IOKAM terms that do not contain end or any of the I/O instructions $\mathsf{r},\mathsf{w0},\mathsf{w1}$ as subterms[3].
- The ordering on the set $P(\Pi)^J$ of semantic predicates on J is defined by $\varphi\le\psi$ if and only if

$$\exists t\in\mathsf{PL}\,\forall j\in J\,\forall u\Vdash\varphi(j)\,\forall\pi\in\psi(i)\,.\,t\star u\in\perp\!\!\!\perp \qquad ◇$$

The ordering on predicates can be understood as a combination of the definition in the case of *Kleene* realizability with a negative translation; this is explained in slightly different

[2] In (Miquel 2011), a set S of stacks is called a *falsity value*, and the set of its realizers is called truth value.

[3] Krivine uses a slightly different notion of proof-like term, the difference is discussed in (Frey, Section 5.2).

words in (Miquel 2011). As in the case of Kleene realizability, the construction of $J \mapsto (P(\Pi)^J, \leq)$ is functorial in J, and one can show the following.

Theorem 5.3 *The assignment* $J \mapsto (P(\Pi)^J, \leq)$ *gives rise to a* Boolean tripos $\mathcal{K}_{\perp\!\!\!\perp} : \mathbf{Set}^{\mathsf{op}} \to \mathbf{Ord}$ *– and via the tripos-to-topos construction to a* Boolean topos $\mathbf{Set}[\perp\!\!\!\perp]$ *– for any pole* $\perp\!\!\!\perp$*.*

Proof. (Frey, Theorem 22). ∎

While the outlined construction always produces triposes and toposes, these may be *degenerate* if we choose the pole too big. This can be ruled out with the following criterion.

Lemma 5.4 *The topos* $\mathbf{Set}[\perp\!\!\!\perp]$ *is* non-degenerate *– i.e. not equivalent to the terminal category* 1 *– if and only if every process* $t \star \pi \in \perp\!\!\!\perp$ *contains at least one of the 'non-logical' instructions* $\mathsf{end}, \mathsf{r}, \mathsf{w0}, \mathsf{w1}$ *as subterms.*

Proof. (Frey, Lemma 26). ∎

5.3 Complexity classes are poles

The final result of this paper is now:

Theorem 5.5 *The set* $\mathfrak{P}$ *of polynomial time algorithms is a pole, and the induced topos* $\mathbf{Set}[\mathfrak{P}]$ *is non-degenerate.*

Proof. If the length of execution sequences of a process $u \star \rho$ is bounded by a polynomial P and $t \star \pi \rightsquigarrow_{\text{iok}} u \star \rho$ is an I/O-free transition, then then the polynomial $P + 1$ bounds the length of execution sequences of $t \star \pi$. Hence $\mathfrak{P}$ is a pole. To see that $\mathbf{Set}[\mathfrak{P}]$ is non-degenerate we appeal to Lemma 5.4 and observe that every process $t \star \rho$ in $\mathfrak{P}$ contains at least the non-logical instruction end (since it is assumed to terminate for all inputs). ∎

As noted above, the fact that the IOKAM is a reasonable model implies that $\mathfrak{P}$ computes exactly the polynomial-time computable functions; hence, Theorem 5.5 associates a topos to the complexity class **FP**. Similarly, we obtain a topos for **P** by restricting the co-domains of functions, and a topos for every superpolynomial class, e.g. the classes of exponential, or elementary, or primitive recursive functions.

6. Future work

In future work we intend to study the toposes arising from complexity classes via the presented construction, in particular the internal logic of these toposes, and variations of the construction. One obvious variation is to use instead of read instructions a distinguished variable for which input values can be substituted (e.g. coded as Church numerals). This seems to be a promising approach since it would allow to access and use the same input in different parts of a process without having to pass it around explicitly after reading. Another possibility is to change the definition of proof-like terms – for the non-degeneracy argument we only have to forbid end as subterm of proof-like terms, but could allow read and write instructions.

References

B. Accattoli and U. Dal Lago. On the invariance of the unitary cost model for head reduction (long version). *CoRR*, abs/1202.1641, 2012.

B. Accattoli, P. Barenbaum, and D. Mazza. Distilling abstract machines. In *ICFP'14—Proceedings of the 2014 ACM SIGPLAN International Conference on Functional Programming*, pages 363–376. ACM, New York, 2014.

H. Barendregt. *The lambda calculus, Its syntax and semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, revised edition, 1984.

P. Crégut. An abstract machine for lambda-terms normalization. In *LISP and Functional Programming*, pages 333–340, 1990.

J. Crossley, G. Mathai, and R. G. Seely. A logical calculus for polynomial-time realizability. *Methods Logic Comput. Sci.*, 1 (3):279–298, 1994.

U. Dal Lago and M. Hofmann. Realizability models and implicit complexity. *Theoret. Comput. Sci.*, 412(20):2029–2047, 2011.

J. Frey. Realizability toposes from specifications. In *13th International Conference on Typed Lambda Calculi and Applications, TLCA 2015, July 1-3, 2015, Warsaw, Poland*, pages 196–210.

M. Hofmann. An application of category-theoretic semantics to the characterisation of complexity classes using higher-order function algebras. *Bulletin of Symbolic Logic*, 3(4):469–486, 1997.

J. Hyland. The effective topos. In *The L.E.J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*, volume 110 of *Stud. Logic Foundations Math.*, pages 165–216. North-Holland, Amsterdam, 1982.

J. Hyland, P. Johnstone, and A. Pitts. Tripos theory. *Math. Proc. Cambridge Philos. Soc.*, 88(2):205–231, 1980. ISSN 0305-0041.

N. Jones. *Computability and complexity - from a programming perspective.* Foundations of computing series. MIT Press, 1997.

S. Kleene. On the interpretation of intuitionistic number theory. *J. Symb. Log.*, 10(4):109–124, 1945.

J. Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. *Archive for Mathematical Logic*, 40(3):189–205, 2001.

J. Krivine. Dependent choice, 'quote' and the clock. *Theoret. Comput. Sci.*, 308(1-3):259–276, 2003.

J. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

J. Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.

F. Lawvere. Quantifiers and sheaves. In *Actes du congres international des mathematiciens, Nice*, volume 1, pages 329–334, 1970.

A. Miquel. Existential witness extraction in classical realizability and via a negative translation. *Log. Methods Comput. Sci.*, 7 (2):2:2, 47, 2011.

C. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994.

B. Pierce. *Types and programming languages.* MIT Press, Cambridge, MA, 2002.

P. Selinger. From continuation passing style to Krivine's abstract machine. Manuscript, 23 pages., 2003.

T. Streicher. Krivine's classical realisability from a categorical perspective. *Mathematical Structures in Computer Science*, 23(06):1234–1256, 2013.

T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of functional programming*, 8 (06):543–572, 1998.

A. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction*, volume 1. Elsevier Science & Technology, 1988.

M. van Atten. The development of intuitionistic logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2014 edition, 2014.

P. van Emde Boas. Machine models and simulations. In *Handbook of theoretical computer science, Vol. A*, pages 1–66. Elsevier, Amsterdam, 1990.

A. Details and proofs omitted from the main text

A.1 De Bruijn indices

Proof. (of Lemma 2.4) Number *1.* is by induction on s.

- Variables $i < j$:

$$\uparrow^l_j \uparrow^k_j i = \uparrow^k_j i = i = \uparrow^{k+l}_j i$$

- Variables $i \geq j$:

$$\uparrow^l_j \uparrow^k_j i = \uparrow^l_j (i+k) = i + k + l = \uparrow^{k+l}_j i$$

- Application:

$$\begin{aligned}\uparrow^l_j \uparrow^k_j (st) &= \uparrow^l_j (\uparrow^k_j s \uparrow^k_j t)\\ &= \uparrow^l_j \uparrow^k_j s \uparrow^l_j \uparrow^k_j t\\ &= \uparrow^{k+l}_j s \uparrow^{k+l}_j t\\ &= \uparrow^{k+l}_j (st)\end{aligned}$$

- Abstraction:

$$\uparrow^l_j \uparrow^k_j \lambda s = \uparrow^l_j \lambda \uparrow^k_{j+1} s = \lambda \uparrow^l_{j+1} \uparrow^k_{j+1} s = \lambda \uparrow^{k+l}_{j+1} s = \uparrow^{k+l}_j \lambda s$$

- $s \in \{\mathsf{cc}, \mathsf{k}_\pi, \mathsf{r}, \mathsf{w0}, \mathsf{w1}, \mathsf{end}\}$:

$$\uparrow^l_j \uparrow^k_j s = \uparrow^l_j s = s = \uparrow^{k+l}_j s$$

Number *2.* by induction on s.

- Variables $i < j$:

$$\begin{aligned}(\uparrow^{k+1}_j i)[u]_{j+k} &= i[u]_{j+k} && \text{since } i < j\\ &= i && \text{since } i < j+k\\ &= \uparrow^k_j (i) && \text{since } i < j+k.\end{aligned}$$

- Variables $i \geq j$:

$$\begin{aligned}&(\uparrow^{k+1}_j i)[u]_{j+k}\\ &= (i+k+1)[u]_{j+k} && \text{since } i \geq j\\ &= i+k && \text{since } i+k+1 \geq j+k+1\\ &= \uparrow^k_j (i) && \text{since } i \geq j.\end{aligned}$$

- Application:

$$\begin{aligned}&(\uparrow^{k+1}_j (st))[u]_{j+k}\\ &= (\uparrow^{k+1}_j s \uparrow^{k+1}_j t)[u]_{j+k}\\ &= (\uparrow^{k+1}_j s)[u]_{j+k} (\uparrow^{k+1}_j t)[u]_{j+k}\\ &= \uparrow^k_j s \uparrow^k_j t && \text{by hypothesis}\\ &= \uparrow^k_j (st).\end{aligned}$$

- Abstraction:

$$\begin{aligned}&(\uparrow^{k+1}_j \lambda s)[u]_{j+k}\\ &= (\lambda \uparrow^{k+1}_{j+1} s)[u]_{j+k}\\ &= \lambda((\uparrow^{k+1}_{j+1} s)[\uparrow^1 u]_{j+k+1})\\ &= \lambda \uparrow^k_{j+1} s && \text{by hypothesis}\\ &= \uparrow^k_j \lambda s\end{aligned}$$

- $s \in \{\mathsf{cc}, \mathsf{k}_\pi, \mathsf{r}, \mathsf{w0}, \mathsf{w1}, \mathsf{end}\}$:

$$(\uparrow^{k+1}_j s)[u]_{j+k} = s = \uparrow^k_j s.$$

Number *3.* by induction on s:

- Variables $i < k$:

$$\begin{aligned}&i[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k\\ &\qquad = i[\uparrow^k u_0]_k = i = i[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

- Variables $i = k$:

$$\begin{aligned}&k[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k\\ &\qquad = k[\uparrow^k u_0]_k = \uparrow^k u_0 = k[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

- Variables i with $k < i \leq k+n$:

$$\begin{aligned}&i[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k = (\uparrow^{k+1} u_{i-k})[\uparrow^k u_0]_k\\ &\qquad = \uparrow^k u_{i-k} = i[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

- Variables $i > k+n$:

$$\begin{aligned}&i[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k\\ &\qquad = i[\uparrow^k u_0]_k = i - n - 1 = i[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

- Abstraction:

$$\begin{aligned}&\lambda(s)[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k\\ &= \lambda(s[\uparrow^{k+2} u_1, \dots, \uparrow^{k+2} u_n]_{k+2})[\uparrow^k u_0]_k && \text{by } \mathit{1.}\\ &= \lambda(s[\uparrow^{k+2} u_1, \dots, \uparrow^{k+2} u_n]_{k+2} [\uparrow^{k+1} u_0]_{k+1}) && \text{by } \mathit{1.}\\ &= \lambda(s[\uparrow^{k+1} u_0, \dots, \uparrow^{k+1} u_n]_{k+1}) && \text{by hypothesis}\\ &= \lambda(s[\uparrow^{k+1} u_0, \dots, \uparrow^{k+1} u_n]_{k+1})\\ &= \lambda(s)[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

- $s \in \{\mathsf{cc}, \mathsf{k}_\pi, \mathsf{r}, \mathsf{w0}, \mathsf{w1}, \mathsf{end}\}$:

$$\begin{aligned}&s[\uparrow^{k+1} u_1, \dots, \uparrow^{k+1} u_n]_{k+1} [\uparrow^k u_0]_k\\ =&s[\uparrow^k u_0]_k = s = s[\uparrow^k u_0, \dots, \uparrow^k u_n]_k\end{aligned}$$

A.2 Correctness of simulation between IOKAM and EIOKAM

Proof. (Lemma 3.2) Write $B = (t, E, \pi, \iota, \omega)$ and thus

$$B^\top = ((t, E)^\top \star \pi^\top, \iota, \omega) = (t^\top [E^\top] \star \pi^\top, \iota, \omega)$$

Split on cases according to the form of t:

- $t = s\, u$ is impossible, as B would not be blocked.
- $t = \lambda s$. As B is blocked, $\pi = \varepsilon$. Hence, $B^\top = ((\lambda(s^\top))[E^\top] \star \varepsilon^\top, \iota, \omega) = (\lambda s'' \star \varepsilon, \iota, \omega)$ for some IOKAM term s''. But then $B^\top$ is also blocked because no IOKAM rule applies to a configuration on the form $(\lambda s'' \star \varepsilon, \iota, \omega)$.
- $t = n$ can not lead to blocking, as pointed out in Remark 2.7.
- $t = \mathsf{cc}$. As B is blocked, $\pi = \varepsilon$. Thus,

$$B^\top = (\mathsf{cc}^\top [E^\top] \star \varepsilon^\top, \iota, \omega) = (\mathsf{cc} \star \varepsilon, \iota, \omega)$$

  whence also $B^\top$ is blocked as the IOKAM rule for $t = \mathsf{cc}$ requires a non-empty stack.
- $t = \mathsf{k}_\rho$. As B is blocked, $\pi = \varepsilon$, and thus: $B^\top = (\mathsf{k}_\rho^\top [E^\top] \star \varepsilon^\top, \iota, \omega) = (\mathsf{k}_{\rho^\top} \star \varepsilon, \iota, \omega)$, and thus $B^\top$ us blocked.
- $t = \mathsf{r}$. As B is blocked, $|\pi| < 3$. Thus, $|\pi^\top| < 3$ and hence $B^\top = (\mathsf{r}^\top [E^\top] \star \pi^\top, \iota, \omega) = (\mathsf{r} \star \pi^\top, \iota, \omega)$ is blocked.
- $t = \mathsf{w1}$. As B is blocked, $\pi = \varepsilon$ and hence $B^\top = (\mathsf{w1}^\top [E^\top] \star \varepsilon^\top, \iota, \omega) = (\mathsf{w1} \star \varepsilon, \iota, \omega)$ and thus $B^\top$ is blocked.
- $t = \mathsf{w0}$. As B is blocked, $\pi = \varepsilon$ and hence $B^\top = (\mathsf{w0}^\top [E^\top] \star \varepsilon^\top, \iota, \omega) = (\mathsf{w0} \star \varepsilon, \iota, \omega)$ and thus $B^\top$ is blocked.

- $t = \mathsf{end}$ is impossible, as B would not be blocked.

Proof. (Lemma 3.3) By cases on the rule used in $B \rightsquigarrow_{\text{eiok}} B'$:

(push) consider $(tu, E, \pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, E, (u, E)\cdot\pi, \iota, \omega)$, and apply $(-)^\top$ to the configurations on both sides. We have

$$\begin{aligned}(tu, E, \pi, \iota, \omega)^\top &= ((tu)^\top[E^\top] \star \pi^\top, \iota, \omega)\\ &= (t^\top[E^\top]\; u^\top[E^\top] \star \pi^\top, \iota, \omega)\end{aligned}$$

and

$$\begin{aligned}(t, E, (u, E)\cdot\pi, \iota, \omega)^\top &= (t^\top[E^\top] \star (u, E)^\top\cdot\pi^\top, \iota, \omega)\\ &= (t^\top[E^\top] \star u[E^\top]\cdot\pi^\top, \iota, \omega)\end{aligned}$$

and as, using (push),

$$(t^\top[E^\top]u^\top[E^\top]\star\pi^\top, \iota, \omega) \rightsquigarrow_{\text{iok}} (t^\top[E^\top]\star u^\top[E^\top]\cdot\pi^\top, \iota, \omega)$$

we obtain $(tu, E, \pi, \iota, \omega)^\top \rightsquigarrow_{\text{iok}} (t, E, (u, E)\cdot\pi, \iota, \omega)^\top$, using (push).

(pop) Assume $(\lambda t, E, c\cdot\pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, c\cdot E, \pi, \iota, \omega)$, and apply $(-)^\top$ to the configurations on both sides. Write $E = c_1 \cdots c_n$. We obtain:

$$(\lambda t, E, c\cdot\pi, \iota, \omega)^\top = ((\lambda t^\top)[E^\top], c^\top\cdot\pi^\top, \iota, \omega)$$

and $(t, c\cdot E, \pi, \iota, \omega)^\top = (t^\top[c^\top\cdot E^\top] \star \pi^\top, \iota, \omega)$. Observe that

$$\begin{aligned}(\lambda t^\top)[E^\top] &= (\lambda t^\top)[c_1^\top \cdots c_n^\top]_0\\ &= \lambda(t^\top[\uparrow^1(c_1^\top)\cdots\uparrow^1(c_n^\top)]_1)\end{aligned}$$

and thus, by application of rule (pop),

$$\begin{aligned}&(\lambda t, E, c\cdot\pi, \iota, \omega)^\top\\ &\quad\rightsquigarrow_{\text{iok}} (t^\top[\uparrow^1(c_1^\top)\cdots\uparrow^1(c_n^\top)]_1[c^\top]_0 \star \pi^\top, \iota, \omega)\end{aligned}$$

By Lemma 2.4, we obtain

$$t^\top[\uparrow^1(c_1^\top)\cdots\uparrow^1(c_n^\top)]_1[c^\top]_0 = t^\top[c^\top\cdot c_1^\top\cdots c_n^\top]_0 = t^\top[c^\top\cdot E^\top]$$

and thus $(\lambda t, E, c\cdot\pi, \iota, \omega)^\top \rightsquigarrow_{\text{iok}} (t^\top[c^\top\cdot E^\top]\star\pi^\top, \iota, \omega) = (t, c\cdot E, \pi, \iota, \omega)^\top$, as desired.

(save) Assume $(\mathsf{cc}, E, (t, F)\cdot\pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, F(\mathsf{k}_\pi, \varepsilon)\cdot\pi, \iota, \omega)$ and apply $^\top$ to both sides to obtain

$$(\mathsf{cc}, E, (t, F)\cdot\pi, \iota, \omega)^\top = (\mathsf{cc} \star (t^\top[F^\top])\cdot\pi^\top, \iota, \omega)$$

and

$$(t, F, (\mathsf{k}_\pi, \varepsilon)\cdot\pi, \iota, \omega)^\top = (t^\top[F^\top] \star \mathsf{k}_{(\pi^\top)}\cdot\pi^\top, \iota, \omega).$$

We have

$$(\mathsf{cc} \star (t^\top[F^\top])\cdot\pi^\top, \iota, \omega) \rightsquigarrow_{\text{iok}} (t^\top[F^\top] \star \mathsf{k}_{(\pi^\top)}\cdot\pi^\top, \iota, \omega)$$

by application of rule (save), as desired.

(restore) Assume that

$$(\mathsf{k}_\pi, E, (t, F)\cdot\rho, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, F, \pi, \iota, \omega)$$

and apply $(-)^\top$ to both sides to obtain

$$(\mathsf{k}_\pi, E, (t, F)\cdot\rho, \iota, \omega)^\top = (\mathsf{k}_{(\pi^\top)} \star (t^\top[F^\top])\cdot\rho^\top, \iota, \omega)$$

and

$$(t, F, \pi, \iota, \omega)^\top = (t^\top[F^\top] \star \pi^\top, \iota, \omega).$$

Then,

$$(\mathsf{k}_{\pi^\top} \star (t^\top[F^\top])\cdot\rho^\top, \iota, \omega) \rightsquigarrow_{\text{iok}} (t^\top[F^\top] \star \pi^\top, \iota, \omega)$$

by application of rule (restore), proving the desideratum.

(r0), (r1), (rε), (w0), (w1) : Straightforward, as none of these rules involve matching or change of the environment E.

(v0) For $(0, (t, E)\cdot F, \pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, E, \pi, \iota, \omega)$, we have to show that applying $(-)^\top$ to both sides yields the same result. Indeed we have

$$\begin{aligned}(0, (t, E)\cdot F, \pi, \iota, \omega)^\top &= (0[(t^\top[E^\top])\cdot F^\top] \star \pi^\top, \iota, \omega)\\ &= (t^\top[E^\top] \star \pi^\top, \iota, \omega)\\ &= (t, E, \pi, \iota, \omega)^\top\end{aligned}$$

as desired.

(vS) We have to show that that $(-)^\top$ yields the same result when applied to both sides of a transition $(n+1, c\cdot E, \pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (n, E, \pi, \iota, \omega)$. Since the right hand side is a valid EIOKAM configuration we have $|E| \geq \mathrm{FI}(n) = n + 1$, hence we have $E = c_0 \cdots c_n \cdot F$ for some environment F. With this we can argue

$$\begin{aligned}(n{+}1, c\cdot E, \pi, \iota, \omega)^\top &= ((n{+}1)[c^\top\cdot c_0^\top \cdots c_n^\top\cdot F^\top] \star \pi^\top, \iota, \omega)\\ &= (c_n^\top \star \pi^\top, \iota, \omega)\\ &= (n[c_0^\top \cdots c_n^\top\cdot F^\top] \star \pi^\top, \iota, \omega)\\ &= (n, E, \pi, \iota, \omega)^\top,\end{aligned}$$

as desired. ∎

A.3 Correctness of compilation

To relate RAM and EIOKAM configurations, we mutually inductively define the following relations which express how EIOKAM syntax is encoded in RAM configurations.

Definition A.1 Let P be a program with prelude, let $l, a, M \in \mathbb{N}$, and let $h : \mathbb{N} \to \mathbb{N}$.

- $(l, a, M, h) \Vdash_P (t, E)$ for t not a continuation term means that
  - l points to $(\!|t|\!)$ in P (i.e. $(\!|t|\!)$ is contained in P as a subsequence, starting at line l)
  - $(a, M, h) \Vdash_P E$
- $(l, a, M, h) \Vdash_P (\mathsf{k}_\pi, E)$ means
  - $l = \boldsymbol{K}$
  - $(a, M, h) \Vdash_P \pi$
- $(a, M, h) \Vdash_P \varepsilon$ means
  - $a = 0$
  - $M > 0$
- $(a, M, h) \Vdash_P c\cdot E$ is defined to mean
  - $0 < a < M - 2$
  - $(h(a), h(a+1), M, h) \Vdash_P c$
  - $(h(a+2), M, h) \Vdash_P E$
- Finally, for a RAM configuration $(l, \begin{bmatrix}\text{ep} & \text{sp}\\ \text{mh} & \text{tv}\\ \text{ip} & \text{op}\end{bmatrix}, h, i, o)$ and an EIOKAM configuration $(t, E, \pi, \iota, \omega)$,

$$(l, \begin{bmatrix}\text{ep} & \text{sp}\\ \text{mh} & \text{tv}\\ \text{ip} & \text{op}\end{bmatrix}, h, i, o) \Vdash_P (t, E, \pi, \iota, \omega)$$

is defined to mean
  - $(l, \text{ep}, \text{mh}, h) \Vdash_P (t, E)$
  - $(\text{sp}, \text{mh}, h) \Vdash_P \pi$
  - $i[\text{ip}] = \iota 2 0^\infty$
  - $\text{op} = |\omega|$ and $o = \omega^r 0^\infty$ ◇

Lemma A.2

- *If $(l, a, M, h) \Vdash_P c$ then $(l, a, M{+}n, h') \Vdash_P c$ for any $n \in \mathbb{N}$ and any h' with $h(i) = h'(i)$ for all $i < M$.*
- *If $(a, M, h) \Vdash_P E$ then $(a, M{+}n, h') \Vdash_P E$ for any $n \in \mathbb{N}$ and any h' with $h(i) = h'(i)$ for all $i < M$.*

Proof. Straightforward simultaneous induction on the derivation of $(l, a, M, h) \Vdash_P c$ and $(a, M, h) \Vdash_P E$. ∎

Lemma A.3 *Assume P is a RAM program with prelude (such that references to $\boldsymbol{K}$ make sense), that B is an EIOKAM configuration, and that C is a RAM configuration with $C \Vdash_P B$. Then*

- *either $B = (\mathsf{end}, E, \pi, \iota, \omega)$ for some E, π, ι, ω, and there exists a RAM configuration $C' = (\boldsymbol{E}, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & |\omega| \end{bmatrix}, h, i, \omega^r 0^\infty)$ with $C \rightsquigarrow_P C'$*
- *or B is blocked and there exists a RAM configuration $C' = (\boldsymbol{F}, f, h, i, o)$ with $C \rightsquigarrow_P^{\leq 4} C'$,*
- *or there is a transition $B \rightsquigarrow_{\text{eiok}} B'$ and a sequence $C \rightsquigarrow_P^{\leq 13} C'$ of transitions with $C' \Vdash_P B'$.*

Proof. By cases on the form of B.

The proof is by cases on the term in head position, sometimes distinguishing additional sub-cases. We first list all possible cases; the proof of each subcase follows afterwards.

1. Head position n (variable) – in this case the closedness condition in Definition 2.6 of EIOKAM configuration implies that the environment is non-empty, and we distinguish the following to sub-cases.
   (a) $B = (0, (t, E){\cdot}F, \pi, \iota, \omega)$
   (b) $B = (n+1, (t, E){\cdot}F, \pi, \iota, \omega)$
2. Head position λt – sub-cases
   (a) $B = (\lambda t, E, \varepsilon, \iota, \omega)$ (blocking)
   (b) $B = (\lambda t, E, c{\cdot}\pi, \iota, \omega)$
3. Application in head position, i.e. $B = (t\,u, E, \pi, \iota, \omega)$
4. Head position cc – sub-cases
   (a) $B = (\mathsf{cc}, E, \varepsilon, \iota, \omega)$ (blocking)
   (b) $B = (\mathsf{cc}, E, (t, F){\cdot}\pi, \iota, \omega)$
5. Head position k_π – sub-cases
   (a) $B = (\mathsf{k}_\pi, E, \varepsilon, \iota, \omega)$ (blocking)
   (b) $B = (\mathsf{k}_\pi, E, (t, F){\cdot}\pi, \iota, \omega)$
6. Head position r – sub-cases
   (a) stack contains less than three closures (blocking)
   (b) stack contains at least three closures and input starts with 0
   (c) stack contains at least three closures and input starts with 1
   (d) stack contains at least three closures and input is empty
7. Head position $\mathsf{w0}$ – sub-cases
   (a) $B = (\mathsf{w0}, E, \varepsilon, \iota, \omega)$ (blocking)
   (b) $B = (\mathsf{w0}, E, (t, F){\cdot}\pi, \iota, \omega)$
8. Head position $\mathsf{w1}$ – sub-cases
   (a) $B = (\mathsf{w1}, E, \varepsilon, \iota, \omega)$ (blocking)
   (b) $B = (\mathsf{w1}, E, (t, F){\cdot}\pi, \iota, \omega)$
9. Head position end – no sub-cases

Here are the proofs of the individual cases:

Case (1a): The statement

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (0, (t, E){\cdot}F, \pi, \iota, \omega)$$

unfolds to the list

- l points to $(\!|0|\!)$ in P
- $0 < \text{ep} < \text{mh} - 2$
- $(h(\text{ep}), h(\text{ep}+1), \text{mh}, h) \Vdash_P (t, E)$
- $(h(\text{ep}+2), \text{mh}, h) \Vdash_P F$
- $(\text{sp}, \text{mh}, h) \Vdash_P \pi$
- conditions on i/o (we do not spell them out because the relevant part of the configurations is changed neither for RAM nor for IOKAM in this case)

of conditions, where $(\!|0|\!)$ is the code segment

$$\begin{aligned} l : {}& \mathtt{TV} := \mathtt{H}[\mathtt{EP}] \\ & \mathtt{EP} := \mathtt{H}[\mathtt{EP}+1] \\ & \mathtt{JZ}(0, \mathtt{TV}) \end{aligned}$$

The EIOKAM makes the transition

$$(0, (t, E){\cdot}F, \pi, \iota, \omega) \rightsquigarrow_{\text{iok}} (t, E, \pi, \iota, \omega),$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \quad \rightsquigarrow_P^3 \quad (h(\text{ep}), \begin{bmatrix} h(\text{ep}{+}1) & \text{sp} \\ \text{mh} & h(\text{ep}) \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o).$$

We have to show that

$$(h(\text{ep}), \begin{bmatrix} h(\text{ep}{+}1) & \text{sp} \\ \text{mh} & h(\text{ep}) \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (t, E, \pi, \iota, \omega),$$

which amounts to the following list of conditions.

- $(h(\text{ep}), h(\text{ep}+1), \text{mh}, h) \Vdash_P (t, E)$
- $(\text{sp}, \text{mh}, h) \Vdash_P \pi$
- conditions on i/o

These conditions are all contained in the previous list.

Case (1b): The statement

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (n+1, (t, E){\cdot}F, \pi, \iota, \omega)$$

unfolds to the list

- l points to $(\!|n+1|\!)$ in P
- $0 < \text{ep} < \text{mh} - 2$
- $(h(\text{ep}), h(\text{ep}+1), \text{mh}, h) \Vdash_P (t, E)$
- $(h(\text{ep}+2), \text{mh}, h) \Vdash_P F$
- $(\text{sp}, \text{mh}, h) \Vdash_P \pi$
- conditions on i/o

of conditions, where $(\!|n+1|\!)$ is the code segment

$$\begin{aligned} l : {}& \mathtt{EP} := \mathtt{H}[\mathtt{EP}+2] \\ & (\!|n|\!) \end{aligned}$$

The EIOKAM configuration performs the transition

$$(n+1, (t, E){\cdot}F, \pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (n, F, \pi, \iota, \omega),$$

and the RAM configuration performs the following transition.

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \rightsquigarrow_P (l+1, \begin{bmatrix} h(\text{ep}{+}2) & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o)$$

We have to show that

$$(l+1, \begin{bmatrix} h(\text{ep}{+}2) & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (n, F, \pi, \iota, \omega)$$

and this unfolds to the following list of conditions.

- $l+1$ points to $(\!|n|\!)$ in P
- $(h(\text{ep}+2), \text{mh}, h) \Vdash_P F$

- $(\mathrm{sp}, \mathrm{mh}, h) \Vdash_P \pi$
- conditions on i/o

The last three conditions are identical to the last three conditions of the previous list, and the fact that $l+1$ points to $(\!|n|\!)$ can be seen in the code segment.

Case (2a): The statement

$$(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \Vdash_P (\lambda t, E, \varepsilon, \iota, \omega)$$

unfolds to the list

- l points to $(\!|\lambda t|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $\mathrm{sp} = 0$
- $\mathrm{mh} > 0$
- conditions on i/o

of conditions, where $(\!|\lambda t|\!)$ is the code segment (labels l, l' added for this proof)

```
 l : JZ(SP, F)
     H[MH] := H[SP]
     H[MH + 1] := H[SP + 1]
     H[MH + 2] := EP
     EP := MH
     MH := MH + 3
     SP := H[SP + 2]
l' : (|t|).
```

The EIOKAM configuration $(\lambda t, E, \varepsilon, \iota, \omega)$ is blocked with empty stack, and since the RAM goes to an $\boldsymbol{F}$-state after one transition (since $\mathrm{sp} = 0$):

$$(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \rightsquigarrow_P (\boldsymbol{F}, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o)$$

Case (2b): The statement

$$(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \Vdash_P (\lambda t, E, c{\cdot}\pi, \iota, \omega)$$

unfolds to the list

- l points to $(\!|\lambda t|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $0 < \mathrm{sp} < \mathrm{mh} - 2$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P c$
- $(h(\mathrm{sp}+2), \mathrm{mh}, h) \Vdash_P \pi$
- conditions on i/o

of conditions.

The EIOKAM makes the transition

$$(\lambda t, E, c{\cdot}\pi, \iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, c{\cdot}E, \pi, \iota, \omega)$$

and the RAM makes the following transitions.

$$\begin{aligned}&(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \\ &\qquad \rightsquigarrow_P^7 (l', \left[\begin{smallmatrix} \mathrm{mh} & h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}3 & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h\left\{\begin{smallmatrix} \mathrm{mh} & = h(\mathrm{sp}) \\ \mathrm{mh}{+}1 & = h(\mathrm{sp}{+}1) \\ \mathrm{mh}{+}2 & = \mathrm{ep} \end{smallmatrix}\right\}, i, o)\end{aligned}$$

We have to show that

$$\begin{aligned}&(l', \left[\begin{smallmatrix} \mathrm{mh} & h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}3 & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h\left\{\begin{smallmatrix} \mathrm{mh} & = h(\mathrm{sp}) \\ \mathrm{mh}{+}1 & = h(\mathrm{sp}{+}1) \\ \mathrm{mh}{+}2 & = \mathrm{ep} \end{smallmatrix}\right\}, i, o) \\ &\qquad\qquad \Vdash_P (t, c{\cdot}E, \pi, \iota, \omega),\end{aligned}$$

which unfolds (in several steps) to the following list of conditions (note that t is not a continuation term since those can never occur as subterms).

- l' points to $(\!|t|\!)$ in P
- $0 < \mathrm{mh} < \mathrm{mh} + 1$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}+3, h\left\{\begin{smallmatrix} \mathrm{mh} & = h(\mathrm{sp}) \\ \mathrm{mh}{+}1 & = h(\mathrm{sp}{+}1) \\ \mathrm{mh}{+}2 & = \mathrm{ep} \end{smallmatrix}\right\}) \Vdash_P c$
- $(\mathrm{ep}, \mathrm{mh}+3, h\left\{\begin{smallmatrix} \mathrm{mh} & = h(\mathrm{sp}) \\ \mathrm{mh}{+}1 & = h(\mathrm{sp}{+}1) \\ \mathrm{mh}{+}2 & = \mathrm{ep} \end{smallmatrix}\right\}) \Vdash_P E$
- $(h(\mathrm{sp}+2), \mathrm{mh}+3, h\left\{\begin{smallmatrix} \mathrm{mh} & = h(\mathrm{sp}) \\ \mathrm{mh}{+}1 & = h(\mathrm{sp}{+}1) \\ \mathrm{mh}{+}2 & = \mathrm{ep} \end{smallmatrix}\right\}) \Vdash_P \pi$

The fact that l' points to $(\!|t|\!)$ is evident from the code segment. The fact that $\mathrm{mh} > 0$ follows for example from the third item in the previous list of conditions. The third condition follows from $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P c$ and Lemma A.2, and analogous arguments establish the two last items.

Case (3): The statement

$$(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \Vdash_P (tu, E, \pi, \iota, \omega)$$

unfolds to the list

- l points to $(\!|tu|\!)$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $(\mathrm{sp}, \mathrm{mh}, h) \Vdash_P \pi$
- conditions on i/o

of conditions, where $(\!|tu|\!)$ is the code segment

```
 l : H[MH] := j
     H[MH + 1] := EP
     H[MH + 2] := SP
     SP := MH
     MH := MH + 3
l' : (|t|)
 j : (|u|)
```

The EIOKAM makes the transition

$$(tu, E, \pi, \iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, E, (u, E){\cdot}\pi, \iota, \omega),$$

and the RAM makes the transitions

$$(l, \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h, i, o) \rightsquigarrow_P^5 (l', \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{mh} \\ \mathrm{mh}{+}3 & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h\left\{\begin{smallmatrix} \mathtt{MH} & = \boldsymbol{j} \\ \mathrm{mh}{+}1 & = \mathrm{ep} \\ \mathrm{mh}{+}2 & = \mathrm{sp} \end{smallmatrix}\right\}, i, o).$$

We have to show that

$$(l', \left[\begin{smallmatrix} \mathrm{ep} & \mathrm{mh} \\ \mathrm{mh}{+}3 & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{smallmatrix}\right], h\left\{\begin{smallmatrix} \mathrm{mh} & = \boldsymbol{j} \\ \mathrm{mh}{+}1 & = \mathrm{ep} \\ \mathrm{mh}{+}2 & = \mathrm{sp} \end{smallmatrix}\right\}, i, o) \Vdash_P (t, E, (u, E){\cdot}\pi, \iota, \omega)$$

which amounts to the following list of conditions (making use of the fact that neither t nor u are continuation terms, since they appear as subterms in the application).

- l' points to $(\!|t|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}+3, h\left\{\begin{smallmatrix} \mathrm{mh} & = \boldsymbol{j} \\ \mathrm{mh}{+}1 & = \mathrm{ep} \\ \mathrm{mh}{+}2 & = \mathrm{sp} \end{smallmatrix}\right\}) \Vdash_P E$
- $0 < \mathrm{mh} < \mathrm{mh} + 1$
- $\boldsymbol{j}$ points to $(\!|u|\!)$ in P
- $(\mathrm{sp}, \mathrm{mh}+3, h\left\{\begin{smallmatrix} \mathrm{mh} & = \boldsymbol{j} \\ \mathrm{mh}{+}1 & = \mathrm{ep} \\ \mathrm{mh}{+}2 & = \mathrm{sp} \end{smallmatrix}\right\}) \Vdash_P \pi$
- conditions on i/o

The conditions on l' and $\boldsymbol{j}$ can be read off the code segment, and the conditions on E and π follow from the hypothesis and Lemma A.2. The condition on mh is implicit in both the conditions on E and on π in the hypothesis.

Case (4a): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{cc}, E, \varepsilon, \iota, \omega)$$

unfolds to the list

- l points to $(\!|\mathsf{cc}|\!)$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $\mathrm{sp} = 0$
- $\mathrm{mh} > 0$
- conditions on i/o

of conditions, where $(\!|\mathsf{cc}|\!)$ is the code segment

$$\begin{aligned} l :\ & \mathtt{H[MH]} := \boldsymbol{K} \\ & \mathtt{H[MH}+1\mathtt{]} := \mathtt{H[SP}+2\mathtt{]} \\ & \mathtt{H[MH}+2\mathtt{]} := \mathtt{H[SP}+2\mathtt{]} \\ & \mathtt{JZ(SP}, \boldsymbol{F}) \\ & \mathtt{TV} := \mathtt{SP} \\ & \mathtt{SP} := \mathtt{MH} \\ & \mathtt{MH} := \mathtt{MH} + 3 \\ & \mathtt{EP} := \mathtt{H[TV}+1\mathtt{]} \\ & \mathtt{JZ(0, H[TV])} \end{aligned}$$

The EIOKAM configuration $(\mathsf{cc}, E, \varepsilon, \iota, \omega)$ is blocked, and the RAM goes to $\boldsymbol{F}$ state after 4 steps, since $\mathrm{sp} = 0$.

Case (4b): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{cc}, E, (t, F)\cdot\pi, \iota, \omega)$$

unfolds to the list

- l points to $(\!|\mathsf{cc}|\!)$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $0 < \mathrm{sp} < \mathrm{mh} - 2$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P (t, F)$
- $(h(\mathrm{sp}+2), \mathrm{mh}, h) \Vdash_P \pi$
- conditions on i/o

of conditions.

The EIOKAM makes the transition

$$(\mathsf{cc}, E, (t, F)\cdot\pi, \iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, F, (k_\pi, \varepsilon)\cdot\pi, \iota, \omega)$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \rightsquigarrow_P^9$$
$$(h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & \mathrm{mh} \\ \mathrm{mh}{+}3 & \mathrm{sp} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h\begin{Bmatrix} \mathrm{mh} = \boldsymbol{K} \\ \mathrm{mh}{+}1 = h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}2 = h(\mathrm{sp}{+}2) \end{Bmatrix}, i, o)$$

We have to show that

$$(h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & \mathrm{mh} \\ \mathrm{mh}{+}3 & \mathrm{sp} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h\begin{Bmatrix} \mathrm{mh} = \boldsymbol{K} \\ \mathrm{mh}{+}1 = h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}2 = h(\mathrm{sp}{+}2) \end{Bmatrix}, i, o)$$
$$\Vdash_P (t, F, (k_\pi, \varepsilon)\cdot\pi, \iota, \omega)$$

which unfolds to the following list of conditions.

- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}+3, h\begin{Bmatrix} \mathrm{mh} = \boldsymbol{K} \\ \mathrm{mh}{+}1 = h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}2 = h(\mathrm{sp}{+}2) \end{Bmatrix}) \Vdash_P (t, F)$
- $0 < \mathrm{mh} < \mathrm{mh} + 1$
- $(h(\mathrm{sp}+2), \mathrm{mh}+3, h\begin{Bmatrix} \mathrm{mh} = \boldsymbol{K} \\ \mathrm{mh}{+}1 = h(\mathrm{sp}{+}2) \\ \mathrm{mh}{+}2 = h(\mathrm{sp}{+}2) \end{Bmatrix}) \Vdash_P \pi$
- conditions on i/o

These conditions all follow from the hypothesis, in the same way as in the previous cases.

Case (5a): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{k}_\pi, E, \varepsilon, \iota, \omega)$$

unfolds to the list

- $l = \boldsymbol{K}$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P \pi$
- $\mathrm{sp} = 0$
- $\mathrm{mh} > 0$
- conditions on i/o

of conditions, where $\boldsymbol{K}$ points to the code segment

$$\begin{aligned} \boldsymbol{K} :\ & \mathtt{JZ(SP}, \boldsymbol{F}) \\ & \mathtt{TV} := \mathtt{SP} \\ & \mathtt{SP} := \mathtt{EP} \\ & \mathtt{EP} := \mathtt{H[TV}+1\mathtt{]} \\ & \mathtt{JZ(0, H[TV])} \end{aligned}$$

The EIOKAM is blocked, and the RAM jumps to $\boldsymbol{F}$ in the first line.

Case (5b): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{k}_\pi, E, (t, F)\cdot\rho, \iota, \omega)$$

unfolds to the list

- $l = \boldsymbol{K}$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P \pi$
- $0 < \mathrm{sp} < \mathrm{mh} - 2$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P (t, F)$
- $(h(\mathrm{sp}+2), \mathrm{mh}, h) \Vdash_P \rho$
- conditions on i/o

of conditions.

The EIOKAM makes the transition

$$(\mathsf{k}_\pi, E, (t, F)\cdot\rho, \iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, F, \pi, \iota, \omega),$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \rightsquigarrow_P^5 (h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & \mathrm{ep} \\ \mathrm{mh} & \mathrm{sp} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o).$$

We have to show that

$$(h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & \mathrm{ep} \\ \mathrm{mh} & \mathrm{sp} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (t, F, \pi, \iota, \omega),$$

which unfolds to the following list of conditions.

- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P (t, F)$
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P \pi$
- conditions on i/o

These conditions are all already contained in the hypothesis.

Case (6a): $(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{r}, E, \pi, \iota, \omega)$ with $|\pi| < 3$.

In this case the IOKAM blocks, and we have the following list of conditions.

- l points to $(\!|\mathsf{r}|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $(\mathrm{sp}, \mathrm{mh}, h) \Vdash_P \pi$
- $i[\mathrm{ip}] = \iota 2 0^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions, where $(\!|\mathsf{r}|\!)$ is the code segment

$$\begin{array}{rl}
l: & \mathtt{JZ}(\mathtt{SP},\boldsymbol{F}) \\
 & \mathtt{TV} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{JZ}(\mathtt{TV},\boldsymbol{F}) \\
 & \mathtt{JZ}(\mathtt{H}[\mathtt{TV}+2],\boldsymbol{F}) \\
 & \mathtt{JZ}(\mathtt{I}[\mathtt{IP}],\boldsymbol{m}) \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{JZ}(\mathtt{I}[\mathtt{IP}]-1,\boldsymbol{n}) \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{TV} := \mathtt{H}[\mathtt{SP}] \\
 & \mathtt{EP} := \mathtt{H}[\mathtt{SP}+1] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{JZ}(0,\mathtt{TV}) \\
\boldsymbol{n}: & \mathtt{TV} := \mathtt{H}[\mathtt{SP}] \\
 & \mathtt{EP} := \mathtt{H}[\mathtt{SP}+1] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{IP} := \mathtt{IP}+1 \\
 & \mathtt{JZ}(0,\mathtt{TV}) \\
\boldsymbol{m}: & \mathtt{TV} := \mathtt{H}[\mathtt{SP}] \\
 & \mathtt{EP} := \mathtt{H}[\mathtt{SP}+1] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{SP} := \mathtt{H}[\mathtt{SP}+2] \\
 & \mathtt{IP} := \mathtt{IP}+1 \\
 & \mathtt{JZ}(0,\mathtt{TV})
\end{array}$$

We have to show that the RAM goes to an $\boldsymbol{F}$-state in each of the cases $\pi = \varepsilon$, $\pi = c{\cdot}\varepsilon$, and $\pi = c{\cdot}d{\cdot}\varepsilon$ (since we assumed $|\pi| < 3$). Inspecting the definition of $\Vdash_P$ we see that these imply $\mathrm{sp} = 0$, $h(\mathrm{sp}+2) = 0$, or $h(h(\mathrm{sp}+2)+2) = 0$, respectively, which yield jumps to $\boldsymbol{F}$ in the first, third, or fourth line.

Case (6b): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{r}, E, (t,F){\cdot}c{\cdot}d{\cdot}\pi, 0{\cdot}\iota, \omega)$$

unfolds to the list

- l points to $(\!|\mathsf{r}|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P (t,F)$
- $(h(h(\mathrm{sp}+2)), h(h(\mathrm{sp}+2)+1), \mathrm{mh}, h) \Vdash_P c$
- $(h(h(h(\mathrm{sp}+2)+2)), h(h(h(\mathrm{sp}+2)+2)+1), \mathrm{mh}, h) \Vdash_P d$
- $(h(h(h(\mathrm{sp}+2)+2)+2), \mathrm{mh}, h) \Vdash_P \pi$
- $0 = i(\mathrm{ip})$
- $i[\mathrm{ip}+1] = \iota 20^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions.

The EIOKAM makes the transition

$$(\mathsf{r}, E, (t,F){\cdot}c{\cdot}d{\cdot}\pi, 0{\cdot}\iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, F, \pi, \iota, \omega)$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o)$$
$$\rightsquigarrow_P^{12} (h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & h(h(h(\mathrm{sp}{+}2){+}2){+}2) \\ \mathrm{mh} & h(\mathrm{sp}) \\ \mathrm{ip}{+}1 & \mathrm{op} \end{bmatrix}, h, i, o).$$

We have to show that

$$(h(\mathrm{sp}), \begin{bmatrix} h(\mathrm{sp}{+}1) & h(h(h(\mathrm{sp}{+}2){+}2){+}2) \\ \mathrm{mh} & h(\mathrm{sp}) \\ \mathrm{ip}{+}1 & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (t, F, \pi, \iota, \omega),$$

which amounts to the following list of conditions.

- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P (t,F)$
- $(h(h(h(\mathrm{sp}{+}2){+}2){+}2), \mathrm{mh}, h) \Vdash_P \pi$
- $i[\mathrm{ip}+1] = \iota 20^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

These conditions are all among the hypotheses.

Case (6c): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{r}, E, c{\cdot}(t,F){\cdot}d{\cdot}\pi, 1\iota, \omega)$$

unfolds to the list

- l points to $(\!|\mathsf{r}|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P c$
- $(h(h(\mathrm{sp}+2)), h(h(\mathrm{sp}+2)+1), \mathrm{mh}, h) \Vdash_P (t,F)$
- $(h(h(h(\mathrm{sp}+2)+2)), h(h(h(\mathrm{sp}+2)+2)+1), \mathrm{mh}, h) \Vdash_P d$
- $(h(h(h(\mathrm{sp}+2)+2)+2), \mathrm{mh}, h) \Vdash_P \pi$
- $1 = i(\mathrm{ip})$
- $i[\mathrm{ip}+1] = \iota 20^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions.

The EIOKAM makes the transition

$$(\mathsf{r}, E, c{\cdot}(t,F){\cdot}d{\cdot}\pi, 1{\cdot}\iota, \omega) \rightsquigarrow_{\mathrm{eiok}} (t, F, \pi, \iota, \omega)$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o)$$
$$\rightsquigarrow_P^{13} (h(h(\mathrm{sp}{+}2)), \begin{bmatrix} h(h(\mathrm{sp}{+}2){+}1) & h(h(h(\mathrm{sp}{+}2){+}2){+}2) \\ \mathrm{mh} & h(h(\mathrm{sp}{+}2)) \\ \mathrm{ip}{+}1 & \mathrm{op} \end{bmatrix}, h, i, o).$$

We have to show that

$$(h(h(\mathrm{sp}+2)), \begin{bmatrix} h(h(\mathrm{sp}{+}2){+}1) & h(h(h(\mathrm{sp}{+}2){+}2){+}2) \\ \mathrm{mh} & h(h(\mathrm{sp}{+}2)) \\ \mathrm{ip}{+}1 & \mathrm{op} \end{bmatrix}, h, i, o)$$
$$\Vdash_P (t, F, \pi, \iota, \omega),$$

which amounts to the list

- $(h(h(\mathrm{sp}{+}2)), h(h(\mathrm{sp}{+}2){+}1), \mathrm{mh}, h) \Vdash_P (t,F)$
- $(h(h(h(\mathrm{sp}{+}2){+}2){+}2), \mathrm{mh}, h) \Vdash_P \pi$
- $i[\mathrm{ip}+1] = \iota 20^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions, all of which are among the hypotheses.

Case (6d): The statement

$$(l, \begin{bmatrix} \mathrm{ep} & \mathrm{sp} \\ \mathrm{mh} & \mathrm{tv} \\ \mathrm{ip} & \mathrm{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{r}, E, c{\cdot}d{\cdot}(t,F){\cdot}\pi, \varepsilon, \omega)$$

unfolds to the list

- l points to $(\!|\mathsf{r}|\!)$ in P
- $(\mathrm{ep}, \mathrm{mh}, h) \Vdash_P E$
- $(h(\mathrm{sp}), h(\mathrm{sp}+1), \mathrm{mh}, h) \Vdash_P c$
- $(h(h(\mathrm{sp}+2)), h(h(\mathrm{sp}+2)+1), \mathrm{mh}, h) \Vdash_P d$
- $(h(h(h(\mathrm{sp}+2)+2)), h(h(h(\mathrm{sp}+2)+2)+1), \mathrm{mh}, h) \Vdash_P (t,F)$
- $(h(h(h(\mathrm{sp}+2)+2)+2), \mathrm{mh}, h) \Vdash_P \pi$
- $i[\mathrm{ip}] = 20^\infty$
- $\mathrm{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions.

The EIOKAM makes the transition

$$(\mathsf{r}, E, c{\cdot}d{\cdot}(t,F){\cdot}\pi, \varepsilon, \omega) \rightsquigarrow_{\text{eiok}} (t, F, \pi, \varepsilon, \omega)$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \rightsquigarrow_P^{12}$$

$$(h(h(h(\text{sp}{+}2){+}2)), \begin{bmatrix} h(h(h(\text{sp}{+}2){+}2){+}1) & h(h(h(\text{sp}{+}2){+}2){+}2) \\ \text{mh} & h(h(h(\text{sp}{+}2){+}2)) \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o).p$$

We have to show that

$$(h(h(h(\text{sp}{+}2){+}2)), \begin{bmatrix} h(h(h(\text{sp}{+}2){+}2){+}1) & h(h(h(\text{sp}{+}2){+}2){+}2) \\ \text{mh} & h(h(h(\text{sp}{+}2){+}2)) \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o)$$

$$\Vdash_P (t, F, \pi, \varepsilon, \omega),$$

which amounts to the list

- $(h(h(h(\text{sp}{+}2){+}2)), h(h(h(\text{sp}{+}2){+}2){+}1), \text{mh}, h) \Vdash_P (t, F)$
- $(h(h(h(\text{sp}{+}2){+}2){+}2), \text{mh}, h) \Vdash_P \pi$
- $i[\text{ip}] = 20^\infty$
- $\text{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions, all of which are among the hypotheses.

Case (7a): The statement

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{w0}, E, \varepsilon, \iota, \omega)$$

unfolds to the list

- l points to $\llparenthesis \mathsf{w0} \rrparenthesis$ in P
- $(\text{ep}, \text{mh}, h) \Vdash_P E$
- $\text{sp} = 0$
- $\text{mh} > 0$
- $i[\text{ip}] = \iota 20^\infty$
- $\text{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions, where $\llparenthesis \mathsf{w0} \rrparenthesis$ is the code segment

$$\begin{array}{l} \texttt{JZ}(\texttt{SP}, \boldsymbol{F}) \\ \texttt{O}[\texttt{OP}] := 0 \\ \texttt{OP} := \texttt{OP} + 1 \\ \texttt{TV} := \texttt{H}[\texttt{SP}] \\ \texttt{EP} := \texttt{H}[\texttt{SP} + 1] \\ \texttt{SP} := \texttt{H}[\texttt{SP} + 2] \\ \texttt{JZ}(0, \texttt{TV}) \end{array}$$

The EIOKAM is blocked on empty input, and the RAM jumps to $\boldsymbol{F}$ in the first line.

Case (7b): The statement

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{w0}, E, (t,F){\cdot}\pi, \iota, \omega)$$

unfolds to the list

- l points to $\llparenthesis \mathsf{w0} \rrparenthesis$ in P
- $(\text{ep}, \text{mh}, h) \Vdash_P E$
- $0 < \text{sp} < \text{mh} - 2$
- $(h(\text{sp}), h(\text{sp}+1), \text{mh}, h) \Vdash_P (t, F)$
- $(h(\text{sp}+2), \text{mh}, h) \Vdash_P \pi$
- $i[\text{ip}] = \iota 20^\infty$
- $\text{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions. The EIOKAM makes the transition

$$(\mathsf{w0}, E, (t,F){\cdot}\pi, \iota, \omega) \rightsquigarrow_{\text{eiok}} (t, F, \pi, \iota, 0\omega),$$

and the RAM makes the transitions

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o)$$

$$\rightsquigarrow_P^7 (h(\text{sp}), \begin{bmatrix} h(\text{sp}{+}1) & h(\text{sp}{+}2) \\ \text{mh} & h(\text{sp}) \\ \text{ip} & \text{op}{+}1 \end{bmatrix}, h, i, o\{\text{op} = 0\}).$$

We have to show that

$$(h(\text{sp}), \begin{bmatrix} h(\text{sp}{+}1) & h(\text{sp}{+}2) \\ \text{mh} & h(\text{sp}) \\ \text{ip} & \text{op}{+}1 \end{bmatrix}, h, i, o\{\text{op} = 0\})$$

$$\Vdash_P (t, F, \pi, \iota, 0{\cdot}\omega),$$

which amounts to the following list of conditions.

- $(h(\text{sp}), h(\text{sp}+1), \text{mh}, h)) \Vdash_P (t, F)$
- $(h(\text{sp}+2), \text{mh}, h) \Vdash_P \pi$
- $i[\text{ip}] = \iota 20^\infty$
- $\text{op} = |\omega| + 1$ and $o = \omega^r 00^\infty$

The fist three conditions are already among the assumptions, and the fourth one follows from the last assumption since $o\{\text{op} = 0\}(\text{op}) = 0$ and sp was incremented.

Cases (8a) and *(8b)* are shown in exactly the same way as *(7a)* and *(7b)*.

Case (9): The statement

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \Vdash_P (\mathsf{end}, E, \pi, \iota, \omega)$$

unfolds to the list

- l points to $\llparenthesis \mathsf{end} \rrparenthesis$ in P
- $(\text{ep}, \text{mh}, h) \Vdash_P E$
- $(\text{sp}, \text{mh}, h) \Vdash_P \pi$
- $i[\text{ip}] = \iota 20^\infty$
- $\text{op} = |\omega|$ and $o = \omega^r 0^\infty$

of conditions, where

$$\llparenthesis \mathsf{end} \rrparenthesis \;\equiv\; \texttt{JZ}(0, \boldsymbol{E})$$

The EIOKAM is in end state, and the RAM makes the transition

$$(l, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o) \rightsquigarrow_P (\boldsymbol{E}, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & \text{op} \end{bmatrix}, h, i, o).$$

The claims about op and o are already among the hypotheses. ∎

Proof. (Lemma 3.5) First observe that

$$(\boldsymbol{S}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, 0^\infty, \iota 20^\infty, 0^\infty) \Vdash_P (t, \varepsilon, \varepsilon, \iota, \varepsilon).$$

Lemma A.3 by iteration implies that

$$(t, \varepsilon, \varepsilon, \iota, \varepsilon) \rightsquigarrow_{\text{eiok}}^* (\mathsf{end}, E, \pi, \iota', \omega)$$

iff

$$(\boldsymbol{S}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, 0^\infty, \iota 20^\infty, 0^\infty) \rightsquigarrow_P^* (\boldsymbol{E}, \begin{bmatrix} \text{ep} & \text{sp} \\ \text{mh} & \text{tv} \\ \text{ip} & |\omega| \end{bmatrix}, h, i, \omega^r 0^\infty),$$

and the estimate of the length of execution sequences follows directly from Lemma A.3 as well. ∎

A.4 Simulating Turing machines on the IOKAM

Proof. (Lemma 4.1) By induction on l.

For $l = 0$ we have $s[k]u_n[k] \dots u_1[k] = kv_m \dots v_1$, and hence $n \leq m$, $s[k] = kv_m \dots v_{n+1}$, and $u_i[k] = v_i$ for

$n \geq i \geq 1$. The claim follows as

$$\begin{aligned} & s[t] \star u_n[t] \cdot \ldots \cdot u_1[t] \cdot \pi \\ = \quad & t v_m \ldots v_{n+1} \star v_n \cdot \ldots \cdot v_1 \cdot \pi \\ \rightsquigarrow_{\text{wh}}^{m-n} \; & t \star v_m \cdot \ldots \cdot v_1 \end{aligned}$$

by $(m-n)$-fold application of (push).

For $l = l'+1$ we know that $s[k]u_n[k] \ldots u_1[k]$ can perform at least one weak head reduction step, whence we can write $s[k]$ as

$$s[k] = (\lambda x.s'[x,k])u_{n+i}[k] \ldots u_{n+1}[k],$$

where $i \geq 0$ (and $n+i > 0$ since we need at least one argument). We can hence decompose the weak head reduction sequence as

$$\begin{aligned} s[k]u_n[k] \ldots u_1[k] \quad = \quad & (\lambda x.s'[x,k])u_{n+i}[k] \ldots u_1[k] \\ \rightsquigarrow_{\text{wh}} \; & s'[u_{n+i}[k], k]u_{n+i-1}[k] \ldots u_1[k] \\ \rightsquigarrow_{\text{wh}}^{l'} \; & k v_m \ldots v_1. \end{aligned}$$

Employing the induction hypothesis, we now obtain the IOKAM execution sequence

$$\begin{aligned} & s[t] \star u_n[t] \cdot \ldots \cdot u_1[t] \cdot \pi \\ = \quad & (\lambda x.s'[x,t])u_{n+i}[t] \ldots u_{n+1}[t] \star u_n[t] \cdot \ldots \cdot u_1[t] \cdot \pi \\ \rightsquigarrow_{\text{iok}}^{i} \quad & (\lambda x.s'[x,t]) \star u_{n+i}[t] \cdot \ldots \cdot u_1[t] \cdot \pi \\ \rightsquigarrow_{\text{iok}} \quad & s'[u_{n+i}[t], t] \star u_{n+i-1}[t] \cdot \ldots \cdot u_1[t] \cdot \pi \\ \rightsquigarrow_{\text{iok}}^{2l'+m-(n+i-1)} \quad & t \star v_m \cdot \ldots \cdot v_1. \end{aligned}$$

The total number of steps is $2l' + m - (n+i-1) + i + 1 = 2l + m - n$ as claimed. ■

Proof. (Lemma 4.2) It is easy to see that

$$\begin{aligned} Wk\lceil 0 \cdot \sigma \rceil \lceil \tau \rceil & \rightsquigarrow_{\text{wh}}^{10} Wk\lceil \sigma \rceil \lceil 0 \cdot \tau \rceil \\ Wk\lceil 1 \cdot \sigma \rceil \lceil \tau \rceil & \rightsquigarrow_{\text{wh}}^{10} Wk\lceil \sigma \rceil \lceil 1 \cdot \tau \rceil \\ Wk\lceil \varepsilon \rceil \lceil \tau \rceil & \rightsquigarrow_{\text{wh}}^{9} k\lceil \tau \rceil, \end{aligned}$$

for $\sigma, \tau \in \{0,1\}^*$. Now given $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^*$, we have

$$\begin{aligned} Bk\lceil \sigma_1 \ldots \sigma_n \rceil & \rightsquigarrow_{\text{wh}}^{2} Wk\lceil \sigma_1 \ldots \sigma_n \rceil \lceil \varepsilon \rceil \\ & \rightsquigarrow_{\text{wh}}^{10} Wk\lceil \sigma_2 \ldots \sigma_n \rceil \lceil \sigma_1 \rceil \\ & \quad \vdots \\ & \rightsquigarrow_{\text{wh}}^{10} Wk\lceil \varepsilon \rceil \lceil \sigma_n \ldots \sigma_1 \rceil \\ & \rightsquigarrow_{\text{wh}}^{9} k\lceil \sigma_n \ldots \sigma_1 \rceil \end{aligned}$$

and the total number of steps used above is $10n + 11$ as claimed. ■

Proof. (Lemma 4.5) Recall the notation from Section 2.1 that $t \star \pi \rightsquigarrow_{\text{iok}} u \star \rho$ is shorthand for $(t \star \pi, \iota, \omega) \rightsquigarrow_{\text{iok}} (u \star \rho, \iota, \omega)$ for arbitrary ι and ω.

For any σ, t we then have:

$$\begin{aligned} & \Theta U t \lceil \sigma \rceil \star \varepsilon \\ \rightsquigarrow_{\text{iok}}^{10} \; & \mathsf{r}(\Theta U t\lceil \sigma \cdot 0 \rceil)(\Theta U t \lceil \sigma \cdot 1 \rceil)(t\lceil \sigma \rceil) \star \varepsilon \end{aligned}$$
$$\begin{cases} \overset{\mathsf{r0}}{\rightsquigarrow}{}_{\text{iok}}^{4} \; \Theta U t\lceil 0 \cdot \sigma \rceil \star \varepsilon \\ \overset{\mathsf{r1}}{\rightsquigarrow}{}_{\text{iok}}^{4} \; \Theta U t\lceil 1 \cdot \sigma \rceil \star \varepsilon \\ \overset{\mathsf{r}\varepsilon}{\rightsquigarrow}{}_{\text{iok}}^{4} \; t\lceil \sigma \rceil \star \varepsilon \end{cases}$$

Iterating this argument gives the execution sequence

$$\begin{aligned} & (Rt \star \varepsilon, b_1 \ldots b_n, \varepsilon) \\ \rightsquigarrow_{\text{iok}}^{2} & (\Theta U t\lceil \varepsilon \rceil \star \varepsilon, b_1 \ldots b_n, \varepsilon) \\ \rightsquigarrow_{\text{iok}}^{14} & (\Theta U t\lceil b_1 \rceil \star \varepsilon, b_2 \ldots b_n, \varepsilon) \\ & \quad \vdots \\ \rightsquigarrow_{\text{iok}}^{14} & (\Theta U t\lceil b_n \ldots b_1 \rceil \star \varepsilon, \varepsilon, \varepsilon) \\ \rightsquigarrow_{\text{iok}}^{14} & (t\lceil b_n \ldots b_1 \rceil \star \varepsilon, \varepsilon, \varepsilon) \end{aligned}$$

for $\iota = b_1 \ldots b_n$, with a total number of $14n + 16$ steps. ■

Proof. (Lemma 4.6) We have

$$\begin{aligned} W \star t \cdot \lceil 0 \cdot \omega \rceil \quad & \overset{\mathsf{w0}}{\rightsquigarrow}{}_{\text{iok}}^{*} \qquad \overset{\mathsf{w1}}{\rightsquigarrow}{}_{\text{iok}}^{*} \quad W \star t \cdot \lceil \omega \rceil \\ W \star t \cdot \lceil \varepsilon \rceil \quad & \rightsquigarrow_{\text{iok}}^{*} \quad t \star \varepsilon, \end{aligned}$$

in the first two lines in a number of steps that is independent of ω. Iterating we get

$$\begin{aligned} (Wt\lceil \omega_1 \ldots \omega_n \rceil \star \varepsilon, \varepsilon, \varepsilon) & \rightsquigarrow_{\text{iok}}^{*} (W \star t \cdot \lceil \omega_1 \ldots \omega_n \rceil, \varepsilon, \varepsilon) \\ & \rightsquigarrow_{\text{iok}}^{*} (W \star t \cdot \lceil \omega_2 \ldots \omega_n \rceil, \varepsilon, \omega_1) \\ & \quad \vdots \\ & \rightsquigarrow_{\text{iok}}^{*} (W \star t \cdot \lceil \varepsilon \rceil, \varepsilon, \omega_n \ldots \omega_1) \\ & \rightsquigarrow_{\text{iok}}^{*} (t \star \varepsilon, \varepsilon, \omega_n \ldots \omega_1) \end{aligned}$$

as required, where $\omega = \omega_1 \ldots \omega_n$. The number of steps is linear in n, since there is an upper bound on the number of steps in each line. ■