TECHNICAL REPORTS IN COMPUTER SCIENCE
Technical University of Dortmund

Bounded Combinatory Logic (Extended Version)

Boris Düdder
Technical University of Dortmund
Department of Computer Science
boris.duedder@cs.tu-dortmund.de

Moritz Martens
Technical University of Dortmund
Department of Computer Science
moritz.martens@cs.tu-dortmund.de

Jakob Rehof
Technical University of Dortmund
Department of Computer Science
jakob.rehof@cs.tu-dortmund.de

Paweł Urzyczyn
University of Warsaw
Institute of Informatics
urzy@mimuw.edu.pl
(Partly supported by grant N N206 355836 from the Ministry of Science and Higher Education)

Number: 840
February 2012

Technical University of Dortmund — Department of Computer Science
Otto-Hahn-Str. 14, 44227 Dortmund

ABSTRACT

In combinatory logic one usually assumes a fixed set of basic combinators (axiom schemes), usually **K** and **S**. In this setting the set of provable formulas (inhabited types) is PSPACE-complete in simple types and undecidable in intersection types. When arbitrary sets of axiom schemes are considered, the inhabitation problem is undecidable even in simple types (this is known as Linial-Post theorem).

Bounded combinatory logic (BCL_k) arises from combinatory logic by imposing the bound k on the depth of types (formulae) which may be substituted for type variables in axiom schemes. We consider the inhabitation (provability) problem for BCL_k: Given an arbitrary set of typed combinators and a type τ, is there a combinatory term of type τ in k-bounded combinatory logic?

Our main result is that the problem is $(k+2)$-EXPTIME complete for BCL_k with intersection types, for every fixed k (and hence non-elementary when k is a parameter). We also show that the problem is EXPTIME-complete for simple types, for all k.

Theoretically, our results give new insight into the expressive power of intersection types. From an application perspective, our results are useful as a foundation for composition synthesis based on combinatory logic.

CONTENTS

1 Introduction 5

2 Preliminaries 9

3 Bounded Combinatory Logic 13

4 Simple types, $\text{BCL}_k(\rightarrow)$ 21

5 Lower bound for intersection types 23

BIBLIOGRAPHY 30

1 INTRODUCTION

In standard combinatory logic (see, e.g., [5]), one usually considers a fixed set of typed combinators (a combinatory basis), for example $\mathbf{S} : (\alpha \to (\beta \to \gamma)) \to (\alpha \to \beta) \to (\alpha \to \gamma)$ and $\mathbf{K} : \alpha \to \beta \to \alpha$. Under the propositions-as-types correspondence, combinator types correspond to axiom schemes of propositional logic in a Hilbert-style proof system, with modus ponens and a rule of axiom scheme instantiation as the principles of deduction. The schematic interpretation of axioms corresponds to implicit polymorphism of combinator types, where type variables ($\alpha, \beta, \gamma, \ldots$) may be instantiated with arbitrary types. Thus, the combinator **K** has types $\tau \to \sigma \to \tau$ for all τ and σ.

In this paper we consider *bounded combinatory logic* (BCL_k), which arises from combinatory logic by imposing the bound k on the depth of types (formulae) which may be substituted for type variables in axiom schemes. For example, in BCL_k the type scheme of the combinator **K** can only be instantiated to $\tau \to \sigma \to \tau$ for τ and σ with depth $\leq k$. By imposing the bound, inhabitation becomes decidable in cases where the unbounded problem is undecidable.

Our interest in bounded combinatory logic is motivated both by theoretical concerns and from the standpoint of applications. Theoretically, we are interested in the complexity and expressive power of the system, depending on the bound. From an application perspective, we consider bounded combinatory logic as a foundation for type-based synthesis, following [9]. In the present paper we generalize from the monomorphic case of [9] to arbitrary bounded levels of polymorphism.

Bounded combinatory logic. In contrast to standard combinatory logic (see, e.g., [5]), we bound the depth of types used to instantiate types of combinators, but rather than considering a *fixed* base of combinators (for example, the base **S**, **K**) as is usual in combinatory logic, we consider the inhabitation problem *relativized* to an arbitrary set Γ of typed combinators, given as part of the input:

Given Γ and τ, is there an applicative term e such that $\Gamma \vdash_k e : \tau$?

The relativized problem is generally much harder than the fixed-base problem. For example, inhabitation in standard (unbounded) simple-typed **SK**-calculus is Pspace-complete [12], whereas the unbounded relativized problem is *undecidable*, even in simple types. We recall that the latter type of problem has been considered since 1948 when Linial and Post [7] initiated a line of work studying decision problems for arbitrary propositional axiom systems (often referred to as partial propositional calculi, abbreviated PPC) answering a question posed by Tarski in 1946. They proved (among other things) that there exists a PPC with an unsolvable decision problem (Linial-Post theorem). Since then, many results have been obtained for various PPC, e.g., Gladstone [3] and Singletary [10] showed that every r.e. degree can be represented by a PPC. In 1974, Singletary [11] showed that the implicational fragment of PPC can represent every r.e. many-one degree. The problem considered there is identical to the unbounded relativized inhabitation problem for simple types.

Our main result is that the relativized inhabitation problems for BCL$_k$ with intersection types form an infinite hierarchy, being $(k+2)$-EXPTIME-complete for each fixed k. A non-elementary lower bound follows for the problem where k is taken as an input parameter. Our lower bound techniques, which may be of independent interest, expose new aspects of the expressive power of intersection types. We generically simulate alternating Turing machines operating in $\exp_{k+1}(n)$-bounded space, where $\exp_m$ denotes the iterated exponential function. For each k, we devise a numeral representation with intersection types in BCL$_k$ for numbers between 0 and $\exp_{k+1}(n)-1$, and we use this system to achieve a succinct representation (exploiting k-bounded polymorphism) of the Turing tape. In contrast, we show that the k-bounded inhabitation problem is EXPTIME-complete for simple types, for all k.

A foundation for composition synthesis With this paper we continue the work begun in [9] on investigating limited systems of combinatory logic as a foundation for type-based synthesis (automatic synthesis of function compositions from a repository of typed functions). In [9], we proved the monomorphic inhabitation problem EXPTIME-complete and devised inhabitation algorithms that we have since implemented and applied to synthesis. In our applications, the set Γ models a repository, the goal type τ is considered as a specification of a desired composition, and the inhabitation algorithm automatically constructs solutions (if any) to the synthesis problem. The relativized inhabitation problem is the natural basis for applications in synthesis, where Γ models a changing repository of functions. As argued in [9], intersection types play a key role in these applications, since they can be used to specify deep semantic properties.

A limited degree of polymorphism has been found to be very useful in applications, since it allows for succinct specifications. In particular, the lowest level (BCL$_0$) of the hierarchy studied here turns out to be already of major importance. At this level, we are able to instantiate type variables with atoms or intersections of such. Since type structure can be atomized by introducing type names (atoms) for structured types through definitions, many interesting problems can be specified and solved in BCL$_0$.

As a simple example of succinctness, consider that we can represent any finite function $f : A \to B$ as an intersection type $\tau_f = \bigcap_{a \in A} a \to f(a)$, where elements of A and B are type constants. Suppose we have combinators $F_i : \tau_{f_i}$ in Γ, and we want to synthesize compositions of such functions represented as types (in some of our applications they could, for example, be refinement types [2]). We might want to introduce composition combinators of arbitary arity, say $g : (A \to A)^n \to (A \to A)$. In the monomorphic system, a function table for g would be exponentially large in n. In BCL$_0$, we can represent g with the single declaration $G : (\alpha_0 \to \alpha_1) \to (\alpha_1 \to \alpha_2) \to \cdots \to (\alpha_{n-1} \to \alpha_n) \to (\alpha_0 \to \alpha_n)$ in Γ. Through level-0 polymorphism, the action of g is thereby fully specified.

Interestingly, by the present results, the complexity of $\textsc{bcl}_0$ is 2-EXPTIME complete and hence comparable in complexity to other known synthesis frameworks (such as, e.g., variants of temporal logic and of propositional dynamic logic). It is also interesting to observe that the lower bound techniques of the present paper appear to reveal a methodology by which inhabitation of intersection types can be used to express a form of logic programming at the type level, which appears to be useful in synthesis. Space limitations preclude us from going into further details here, and we report on our experience in synthesis in a separate paper.

PRELIMINARIES

2

Types: Type expressions, ranged over by τ, σ etc., are defined by

$$\tau \ ::= \ a \mid \tau \to \tau \mid \tau \cap \tau$$

where $a, b, c, \ldots$ range over *atoms* comprising of *type constants*, drawn from a finite set $\mathbb{A}$ including the constant ω, and *type variables*, drawn from a disjoint denumerable set $\mathbb{V}$ ranged over by α, β etc. We let $\mathbb{T}$ denote the set of all types.

As usual, types are taken modulo commutativity ($\tau \cap \sigma = \sigma \cap \tau$), associativity ($(\tau \cap \sigma) \cap \rho = \tau \cap (\sigma \cap \rho)$), and idempotency ($\tau \cap \tau = \tau$). A type *environment* Γ is a finite set of type assumptions of the form $x : \tau$. We let $Dm(\Gamma)$ and $Rn(\Gamma)$ denote the domain and range of Γ. Let $Var(\tau)$, $Cnst(\tau)$ and $At(\tau)$ denote, respectively, the set of variables, the set of constants and the set of atoms occurring in τ, and we extend the definitions to environments, written $Var(\Gamma)$, $Cnst(\Gamma)$ and $At(\Gamma)$ in the standard way.

A type $\tau \cap \sigma$ is said to have τ and σ as *components*. For an intersection of several components we sometimes write $\bigcap_{i=1}^{n} \tau_i$ or $\bigcap_{i \in I} \tau_i$ or $\bigcap\{\tau_i \mid i \in I\}$, where the empty intersection is identified with ω.

Subtyping: Subtyping $\leq$ is the least preorder (reflexive and transitive relation) on $\mathbb{T}$, with

$$\sigma \leq \omega, \quad \omega \leq \omega \to \omega, \quad \sigma \cap \tau \leq \sigma, \quad \sigma \cap \tau \leq \tau, \quad \sigma \leq \sigma \cap \sigma;$$
$$(\sigma \to \tau) \cap (\sigma \to \rho) \leq \sigma \to \tau \cap \rho;$$
$$\text{If } \sigma \leq \sigma' \text{ and } \tau \leq \tau' \text{ then } \sigma \cap \tau \leq \sigma' \cap \tau' \text{ and } \sigma' \to \tau \leq \sigma \to \tau'.$$

We identify σ and τ when $\sigma \leq \tau$ and $\tau \leq \sigma$. The following distributivity properties follow from the axioms of subtyping:

$$(\sigma \to \tau) \cap (\sigma \to \rho) = \sigma \to (\tau \cap \rho) \qquad (\sigma \to \tau) \cap (\sigma' \to \tau') \leq (\sigma \cap \sigma') \to (\tau \cap \tau')$$

Paths: If $\tau = \tau_1 \to \cdots \to \tau_m \to \sigma$, then we write $\sigma = tgt_m(\tau)$ and $\tau_i = arg_i(\tau)$, for $i \leq m$. If $arg_i(\tau) = \rho$ for all i we also write $\tau = \rho^m \to \sigma$. A type of the form $\tau_1 \to \cdots \to \tau_m \to a$, where $a \neq \omega$ is an atom,[1] is called a *path of length m*. A type τ is *organized* if it is a (possibly empty) intersection of paths (those are called *paths in* τ). Note that premises in an organized type do not have to be organized, i.e., organized is not necessarily normalized [4].

Lemma 1 *Every type τ is equal to an organized type $\overline{\tau}$, computable in polynomial time.*

---

[1] Observe that $\tau_1 \to \cdots \to \tau_m \to \omega = \omega$.

Proof: Define $\overline{a} = a$ if a is an atom and let $\overline{\tau \cap \sigma} = \overline{\tau} \cap \overline{\sigma}$. If $\overline{\sigma} = \bigcap_{i \in I} \sigma_i$ then take $\overline{\tau \to \sigma} = \bigcap_{i \in I}(\tau \to \sigma_i)$. □

Sets of paths: For an organized type σ, we let $\mathbb{P}_m(\sigma)$ denote the set of all paths in σ of length m or more. We extend the definition to arbitrary τ by implicitly organizing τ, i.e., we write $\mathbb{P}_m(\tau)$ as a shorthand for $\mathbb{P}_m(\overline{\tau})$.

Type size: The *size* of a type τ, denoted $|\tau|$, is defined to be the number of nodes in the syntax tree of τ (this is identical to the textual size of τ). The *path length* of a type τ is denoted $\|\tau\|$ and is defined to be the maximal length of a path in τ.

Substitutions: A *substitution* is a function $S : \mathbb{V} \to \mathbb{T}$ such that S is the identity everywhere but on a finite subset of $\mathbb{V}$. For a substitution S, we define the *support* of S, written $Supp(S)$, as $Supp(S) = \{\alpha \in \mathbb{V} \mid \alpha \neq S(\alpha)\}$. We may write $S : V \to \mathbb{T}$ when V is a finite subset of $\mathbb{V}$ with $Supp(S) \subseteq V$. We write $At(S)$ to denote the set $\{At(S(\alpha)) \mid \alpha \in Supp(S)\}$. A substitution S is tacitly lifted to a function on types, $S : \mathbb{T} \to \mathbb{T}$, by homomorphic extension. Finally, a *constant-function* is a map $c : \mathbb{A} \to \mathbb{A}$ such that $c(\omega) = \omega$. Constant-functions are tacitly lifted to functions $c : \mathbb{T} \to \mathbb{T}$.

The following property, probably first stated in [1], is often called *beta-soundness*. Note that the converse is trivially true.

Lemma 2 *Let a_j, for $j \in J$, be atoms.*

1. *If $\bigcap_{i \in I}(\sigma_i \to \tau_i) \cap \bigcap_{j \in J} a_j \leq \alpha$ then $\alpha = a_j$, for some $j \in J$.*
2. *If $\bigcap_{i \in I}(\sigma_i \to \tau_i) \cap \bigcap_{j \in J} a_j \leq \sigma \to \tau$, where $\sigma \to \tau \neq \omega$, then the set $\{i \in I \mid \sigma \leq \sigma_i\}$ is nonempty and $\bigcap\{\tau_i \mid \sigma \leq \sigma_i\} \leq \tau$.*

Lemma 3 *Let $\bigcap_{i \in I} \tau_i \leq \beta_1 \to \cdots \to \beta_m \to p$, where τ_i are paths. Then there is an $i \in I$ such that $\tau_i = \alpha_1 \to \cdots \to \alpha_m \to p$ and $\beta_j \leq \alpha_j$, for all $j \leq m$.*

Proof: Induction with respect to m, using the beta soundness (Lemma 2). □

Lemma 4 *Let S be a substitution and let c be a constant-function. Then $\sigma \leq \tau$ implies $S(\sigma) \leq S(\tau)$ and $c(\sigma) \leq c(\tau)$.*

Proof: Induction with respect to the definition of $\sigma \leq \tau$. □

Alternating Turing Machines

An *alternating Turing machine* is a tuple $\mathcal{M} = (\Sigma, Q, q_0, q_a, q_r, \Delta)$. The set of states $Q = Q_\exists \uplus Q_\forall$ is partitioned into a set $Q_\exists$ of existential states and a set $Q_\forall$ of universal states. There is an initial state $q_0 \in Q$, an accepting state $q_a \in Q_\forall$, and a rejecting state $q_r \in Q_\exists$. We take

$\Sigma = \{0, 1, \sqcup\}$, where $\sqcup$ is the blank symbol (used to initialize the tape but not written by the machine). The transition relation Δ satisfies

$$\Delta \subseteq \Sigma \times Q \times \Sigma \times Q \times \{\textsc{l}, \textsc{r}\},$$

where $h \in \{\textsc{l}, \textsc{r}\}$ are the moves of the machine head (left and right). For $b \in \Sigma$ and $q \in Q$, we write $\Delta(b, q) = \{(c, p, h) \mid (b, q, c, p, h) \in \Delta\}$. We assume $\Delta(b, q_a) = \Delta(b, q_r) = \emptyset$, for all $b \in \Sigma$, and $\Delta(b, q) \neq \emptyset$ for $q \in Q \setminus \{q_a, q_r\}$. A *configuration* of $\mathcal{M}$ is a word wqw' with $q \in Q$ and $w, w' \in \Sigma^*$. The *successor* relation $\mathcal{C} \Rightarrow \mathcal{C}'$ on configurations is defined as usual [8], according to Δ. We classify a configuration wqw' as *existential*, *universal*, *accepting* etc., according to q. The notion of *eventually accepting* configuration is defined by induction:[2]

- An accepting configuration is eventually accepting.
- If $\mathcal{C}$ is existential and some successor of $\mathcal{C}$ is eventually accepting then so is $\mathcal{C}$.
- If $\mathcal{C}$ is universal and all successors of $\mathcal{C}$ are eventually accepting then so is $\mathcal{C}$.

---

[2] Formally we define the set of all eventually accepting configurations as the smallest set satisfying the appropriate closure conditions.

BOUNDED COMBINATORY LOGIC 3

Definition 5 *(Levels) Given a type* τ *we define the* level *of* τ*, written* $\ell(\tau)$*, as follows.*

$$\begin{array}{lcl} \ell(a) & = & 0, \textit{ for } a \in \mathbb{A} \cup \mathbb{V}; \\ \ell(\tau \to \sigma) & = & 1 + \max\{\ell(\tau), \ell(\sigma)\}; \\ \ell(\bigcap_{i=1}^{n} \tau_i) & = & \max\{\ell(\tau_i) \mid i = 1, \ldots, n\}. \end{array}$$

The level of a substitution S*, written* $\ell(S)$*, is defined as*

$$\ell(S) = \max\{\ell(S(\alpha)) \mid \alpha \in \mathbb{V}\}.$$

A level-k type *is a type* τ *with* $\ell(\tau) \leq k$*, and a* level-k substitution *is a substitution* S *with* $\ell(S) \leq k$*. For* $k \geq 0$*, we let* $\mathbb{T}_k$ *denote the set of all level-*k *types. For a subset* A *of atomic types, we let* $\mathbb{T}_k(A)$ *denote the set of level-*k *types with atoms (leaves) in the set* A*.* ☐

Notice that the level of a type is independent from the width (number of arguments) of intersections. Notice also that $\ell(S)$ is completely determined by the restriction of S to $Supp(S)$: if $Supp(S) = \emptyset$, then $\ell(S) = 0$, and if $Supp(S) \neq \emptyset$, then $\ell(S) = \max\{\ell(S(\alpha)) \mid \alpha \in Supp(S)\}$. Finally, we have $\ell(S \circ S') \leq \ell(S) + \ell(S')$.

Type assignment: For each $k \geq 0$ the system $\text{BCL}_k(\to, \cap)$ (*k-bounded combinatory logic with intersection types*) is defined by the type assigment rules shown in Figure 3.1. In rule (var), the condition $\ell(S) \leq k$ is understood as a side condition to the axiom $\Gamma, x : \tau \vdash_k x : S(\tau)$. The restriction to *simple types* (types without $\cap$) is called $\text{BCL}_k(\to)$ and is defined by the rules (var), ($\to$E) and ($\leq$), where τ and τ' range over simple types, by dropping all axioms from the subtyping relation that involve $\cap$, and by considering only substitutions S mapping type variables to simple types. Recall from [9] *finite combinatory logic with intersection types*, denoted FCL. This system can be presented as the restriction of BCL_k in which the (var) rule is simplified to the axiom $\Gamma, x : \tau \vdash x : \tau$.

In this paper we are addressing the following *relativized inhabitation problem:*

Given Γ *and* τ*, is there an applicative term* e *such that* $\Gamma \vdash_k e : \tau$*?*

$$\frac{[\ell(S) \leq k]}{\Gamma, x : \tau \vdash_k x : S(\tau)}(\text{var}) \qquad \frac{\Gamma \vdash_k e : \tau \to \tau' \quad \Gamma \vdash_k e' : \tau}{\Gamma \vdash_k (e\, e') : \tau'}(\to\text{E})$$

$$\frac{\Gamma \vdash_k e : \tau_1 \quad \Gamma \vdash_k e : \tau_2}{\Gamma \vdash_k e : \tau_1 \cap \tau_2}(\cap\text{I}) \qquad \frac{\Gamma \vdash_k e : \tau \quad \tau \leq \tau'}{\Gamma \vdash_k e : \tau'}(\leq)$$

Figure 3.1: Bounded combinatory logic BCL_k

Algorithm

In this section we formulate an algorithm to decide the relativized inhabitation problem for BCL_k, and derive the $(k+2)$-EXPTIME upper bound.

Lemma 6 *Let $\Gamma \vdash_k e : \tau$ and let S be a level-m substitution. Then there exists a derivation of $\Gamma \vdash_{k+m} e : S(\tau)$ of the same depth.*

Proof: Induction with respect to the derivation of $\Gamma \vdash_k e : \tau$. In case the derivation is by rule (var), with $\Gamma', x : \sigma \vdash_k x : S_1(\sigma)$ and $S_1(\sigma) = \tau$, we have the derivation: $\Gamma', x : \sigma \vdash_{k+m} x : S \circ S_1(\sigma)$. This is evidently of the same depth as the original derivation. The remaining cases follow easily by induction. $\square$

Lemma 7 *Let $\Gamma \vdash_k e : \tau$ and let c be a constant-function such that c is the identity on $Cnst(\Gamma)$. Then there exists a derivation of $\Gamma \vdash_k e : c(\tau)$ of the same depth.*

Proof: Induction with respect to the derivation of $\Gamma \vdash_k e : \tau$. In case the derivation is by rule (var), assume $\Gamma', x : \sigma \vdash_k x : S(\sigma)$ with $S(\sigma) = \tau$. Since c is the identity on $Cnst(\Gamma)$, it follows that c is the identity on $Cnst(\sigma)$. Therefore, the map $c \circ S$ acts as a type substitution on σ, and consequently we have the derivation: $\Gamma', x : \sigma \vdash_k x : c \circ S(\sigma)$, proving the claim.

In case the derivation is by rule $(\leq)$, we use Lemma 4 and apply the induction hypothesis. The remaining cases follow easily by induction. $\square$

Let $At_\omega(\Gamma, \tau) = At(\Gamma) \cup At(\tau) \cup \{\omega\}$. The following proposition shows that, in order to solve an inhabitation question $\Gamma \vdash_k ? : \tau$, one needs only consider rule (var) restricted to substitutions of the form $S : Var(\Gamma) \to \mathbb{T}_k(At_\omega(\Gamma, \tau))$.

We say that a substitution S *occurs* in a derivation $\mathcal{D}$, whenever S is used in an application of rule (var) in $\mathcal{D}$.

Proposition 8 *If $\Gamma \vdash_k e : \tau$, then there exists a derivation $\mathcal{D}$ of $\Gamma \vdash_k e : \tau$ such that every substitution S occurring in $\mathcal{D}$ satisfies the conditions*

1. $Supp(S) \subseteq Var(\Gamma)$;
2. $At(S) \subseteq At_\omega(\Gamma, \tau)$.

Proof: By induction with respect to the depth d of the derivation of $\Gamma \vdash_k e : \tau$.

Assume $d = 1$. We must have $\Gamma', x : \sigma \vdash_k x : S(\sigma)$ with $S(\sigma) = \tau$, by rule (var). We can restrict S such that $Supp(S) \subseteq Var(\sigma)$, and hence also $Supp(S) \subseteq Var(\Gamma)$. Because $S(\sigma) = \tau$, we then also have $At(S) \subseteq At(\tau)$.

Assume $d > 1$. Suppose we have a derivation

$$\frac{\begin{array}{c}\mathcal{D}_1\\ \Gamma \vdash_k e : \sigma \to \tau\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \Gamma \vdash_k e' : \sigma\end{array}}{\Gamma \vdash_k ee' : \tau}$$

by rule ($\to$E), where $d_i < d$ is the depth of the subderivation $\mathcal{D}_i$ ($i = 1, 2$). Let S be the substitution with $Supp(S) \subseteq Var(\sigma)$ such that S is the identity on $Var(\tau)$ and all variables in $Var(\sigma) \setminus Var(\tau)$ are mapped to ω. Using Lemma 6 we have $\Gamma \vdash_k e : S(\sigma) \to \tau$ and $\Gamma \vdash_k e' : S(\sigma)$ by derivations of depth d_i, respectively. By definition of S, we have

$$Var(S(\sigma)) \subseteq Var(\tau) \tag{3.1}$$

Let c be the constant-function such that c is the identity on $Cnst(\Gamma) \cup Cnst(\tau)$ and mapping all other constants to ω. By Lemma 7, we have $\Gamma \vdash_k e : c(S(\sigma)) \to \tau$ and $\Gamma \vdash_k e' : c(S(\sigma))$ by derivations of depth $d_i < d$, respectively. By definition of c together with (3.1), we have

$$At_\omega(\Gamma, c(S(\sigma)) \subseteq At_\omega(\Gamma, \tau) \tag{3.2}$$

Applying the induction hypothesis, we can conclude that there exist derivations $\mathcal{D}'_1$ and $\mathcal{D}'_2$ such that we have

$$\frac{\begin{array}{c}\mathcal{D}'_1\\ \Gamma \vdash_k e : c(S(\sigma)) \to \tau\end{array} \qquad \begin{array}{c}\mathcal{D}'_2\\ \Gamma \vdash_k e' : c(S(\sigma))\end{array}}{\Gamma \vdash_k ee' : \tau}$$

with every substitution in S occurring in $\mathcal{D}'_1$ or $\mathcal{D}'_2$ satisfying $Supp(S) \subseteq Var(\Gamma)$ and $At(S) \subseteq At_\omega(\Gamma, c(S(\sigma)) \to \tau)$. By (3.2) we have

$$At_\omega(\Gamma, c(S(\sigma)) \to \tau) = At_\omega(\Gamma, \tau)$$

thereby proving the claim.

Suppose we have a derivation by rule ($\leq$) of the form

$$\frac{\overset{\mathcal{D}_1}{\Gamma \vdash_k e : \sigma} \qquad \sigma \leq \tau}{\Gamma \vdash_k e : \tau}$$

where $\mathcal{D}_1$ is of depth $d_1 < d$. We define substitution S and constant-function c as in the previous case and apply Lemma 6 and Lemma 7 to obtain a derivation $\Gamma \vdash_k e : c(S(\sigma))$ of depth $d_1 < d$ and, by Lemma 4, $c(S(\sigma)) \leq \tau$. Applying induction hypothesis, we have a derivation

$$\frac{\overset{\mathcal{D}_1'}{\Gamma \vdash_k e : c(S(\sigma))} \qquad c(S(\sigma)) \leq \tau}{\Gamma \vdash_k e : \tau}$$

satisfying the claim.

The remaining case of rule ($\cap$I) follows easily from the induction hypothesis. □

The following lemma shows that inhabitation in BCL$_k(\to, \cap)$ is equivalent to inhabitation in FCL modulo expansion of the type environment. Given a number k, an environment Γ and a type τ, define for each $x \in Dm(\Gamma)$ the set of substitutions

$$\mathcal{S}_x^{(\Gamma,\tau,k)} = Var(\Gamma(x)) \to \mathbb{T}_k(At_\omega(\Gamma, \tau))$$

and define the environment $\Gamma^{(\tau,k)}$ with domain $Dm(\Gamma)$ so that, for $x \in Dm(\Gamma)$,

$$\Gamma^{(\tau,k)}(x) = \bigcap \{S(\Gamma(x)) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$$

Lemma 9 (Expansion) *One has* $\Gamma \vdash_k e : \tau$ *in* BCL$_k(\to, \cap)$ *iff* $\Gamma^{(\tau,k)} \vdash e : \tau$ *in* FCL.

Proof: If $\Gamma \vdash_k e : \tau$ by a derivation $\mathcal{D}$, consider each application of rule (var) of the form $\Gamma', x : \sigma \vdash_k x : S(\sigma)$, occurring in $\mathcal{D}$. By Proposition 8, we can assume that S is a member of the set $\mathcal{S}_x^{(\Gamma,\tau,k)}$. Hence, one has $\Gamma^{(\tau,k)} \vdash x : S(\sigma)$ in FCL, by an application of rule (var), followed by an application of rule ($\leq$). It follows that $\Gamma^{(\tau,k)} \vdash e : \tau$ holds in FCL.

For the implication in the other direction, consider that one has in BCL$_k(\to, \cap)$

$$\Gamma \vdash_k x : \bigcap \{S(\Gamma(x)) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$$

for all $x \in Dm(\Gamma)$, by multiple applications of rule (var), followed by rule ($\cap$I). □

The following lemma was shown in [9] using a different representation of paths. Here it is formulated and proved in terms of the notion of paths defined in the present paper.

Lemma 10 (Path Lemma for FCL [9]) *The following are equivalent conditions:*

1. $\Gamma \vdash x\, e_1 \ldots e_m : \tau$;
2. *There exists a set P of paths in $\mathbb{P}_m(\Gamma(x))$ such that*
   a) $\bigcap_{\pi \in P} tgt_m(\pi) \;\leq\; \tau$;
   b) $\Gamma \vdash e_i : \bigcap_{\pi \in P} arg_i(\pi)$, *for all* $i \leq m$.

Proof: $(\Rightarrow)$ Induction with respect to the derivation of $\Gamma \vdash (x\, e_1 \ldots e_m) : \tau$. We assume w.l.o.g. that all types considered are organized (we can organize them according to Lemma 1).

If the last rule is (var) then $m = 0$ and $\Gamma(x) = \tau$. Since τ is the intersection of its paths, we can choose $P = \mathbb{P}_0(\tau)$ with $\tau = \bigcap_{\pi \in P} \pi$.

If the last rule of the derivation is $(\leq)$, then the claim follows easily by induction hypothesis and transitivity of $\leq$.

If the last rule is $(\cap\mathrm{I})$, then $\Gamma \vdash (x\, e_1 \ldots e_m) : \tau_1$ and $\Gamma \vdash (x\, e_1 \ldots e_m) : \tau_2$, with $\tau_1 \cap \tau_2 = \tau$. By the induction hypothesis, there are subsets P_1, P_2 of paths in $\mathbb{P}_m(\Gamma(x))$ with $\Gamma \vdash e_j : \bigcap_{\pi \in P_i} arg_j(\pi)$ and $\bigcap_{\pi \in P_i} tgt_m(\pi) \;\leq\; \tau_i$, for $i = 1, 2$ and $j = 1, \ldots, m$. Let $P = P_1 \cup P_2$. Then it follows by $(\cap\mathrm{I})$ that $\Gamma \vdash e_j : \bigcap_{\pi \in P} arg_j(\pi)$, for $j = 1 \ldots m$, and by monotonicity of $\cap$ that $\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau_1 \cap \tau_2 = \tau$, as desired.

If the last rule of the derivation is $(\rightarrow E)$, then we have

$$\Gamma \vdash (x\, e_1 \ldots e_{m-1}) : \tau' \rightarrow \tau \text{ and } \Gamma \vdash e_m : \tau',$$

for some τ'. By the induction hypothesis there exists $P_{m-1} \subseteq \mathbb{P}_{m-1}(\Gamma(x))$ such that

(I1) $\Gamma \vdash e_j : \bigcap_{\pi \in P_{m-1}} arg_j(\pi)$ for $j = 1 \ldots m-1$;

(I2) $\bigcap_{\pi \in P_{m-1}} tgt_{m-1}(\pi) \leq \tau' \rightarrow \tau$.

Let

$$P_m = \{\pi \in P_{m-1} \mid tgt_{m-1}(\pi) = \sigma_\pi \rightarrow \sigma'_\pi \text{ for some } \sigma_\pi, \sigma'_\pi \text{ with } \tau' \leq \sigma_\pi\}.$$

For $\pi \in P_m$, write $tgt_{m-1}(\pi) = \sigma_\pi \rightarrow \sigma'_\pi$. By (I2) and Lemma 2, we have $P_m \neq \emptyset$ and

$$\tau' \leq \bigcap_{\pi \in P_m} \sigma_\pi \tag{3.3}$$

and

$$\bigcap_{\pi \in P_m} \sigma'_\pi \leq \tau \tag{3.4}$$

Since $P_m \subseteq \mathbb{P}_{m-1}(\Gamma(x))$, we have by definition of P_m that $P_m \subseteq \mathbb{P}_m(\Gamma(x))$. Moreover, by definition of P_m we have

$$\bigcap_{\pi \in P_m} arg_m(\pi) = \bigcap_{\pi \in P_m} \sigma_\pi \tag{3.5}$$

and

$$\bigcap_{\pi \in P_m} tgt_m(\pi) = \bigcap_{\pi \in P_m} \sigma'_\pi \tag{3.6}$$

Since $\Gamma \vdash e_m : \tau'$, we have $\Gamma \vdash e_m : \bigcap_{\pi \in P_m} arg_m(\pi)$ by rule $(\leq)$, (3.3) and (3.5). And from (3.4) and (3.6) we get $\bigcap_{\pi \in P_m} tgt_m(\pi) \leq \tau$. It follows that we can choose $P = P_m$.

$(\Leftarrow)$ Since $\Gamma(x)$ is the intersection of its paths, we have

$$\Gamma(x) \leq arg_1(\pi) \to \cdots \to arg_m(\pi) \to tgt_m(\pi),$$

for $\pi \in \mathbb{P}_m(\Gamma(x))$. The claim now follows from the assumption together with obvious applications of the type rules. $\square$

Lemma 11 (Path Lemma for $\text{BCL}_k(\to, \cap)$**)** *The following are equivalent conditions:*

1. $\Gamma \vdash_k x\, e_1 \ldots e_m : \tau$;
2. *There exists a set* P *of paths in* $\mathbb{P}_m(\bigcap\{S(\Gamma(x)) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\})$ *such that*
   a) $\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau$;
   b) $\Gamma \vdash_k e_i : \bigcap_{\pi \in P} arg_i(\pi)$, *for all* $i \leq m$.

Proof: Immediate, by Lemma 9 and Lemma 10. $\square$

The following corollary will be used later.

Corollary 12 *Let* $\Gamma(x) = \bigcap_{j \in J}(\tau_1^j \to \cdots \to \tau_m^j \to \sigma^j)$. *If* $\Gamma \vdash x\, e_1 \ldots e_m : \tau$ *then there are substitutions* S_ℓ, *for* $\ell \in L$, *and numbers* j_ℓ *such that*

1. $\bigcap_{\ell \in L} S_\ell(\sigma^{j_\ell}) \leq \tau$;
2. $\Gamma \vdash_k e_i : \bigcap_{\ell \in L} S_\ell(\tau_i^{j_\ell})$.

Proof: Let the set of paths P be chosen according to Lemma 11. Write $\mathcal{S}_x^{(\Gamma,\tau,k)} = S_1, \ldots, S_r$. Each path $\pi \in P$ is of the form

$$\pi = S_\ell(\tau_1^{j_\ell}) \to \cdots \to S_\ell(\tau_m^{j_\ell}) \to \pi_\ell$$

where π_ℓ is a path in $S_\ell(\sigma^{j_\ell})$, for some ℓ ($1 \leq \ell \leq r$).

Let L be the set of all ℓ ($1 \leq \ell \leq r$) such that for some path $\pi \in P$

$$\pi = S_\ell(\tau_1^{j_\ell}) \to \cdots \to S_\ell(\tau_m^{j_\ell}) \to \pi_\ell$$

Then we have $\bigcap_{\pi \in P} arg_i(\pi) = \bigcap_{\ell \in L} S_\ell(\tau_i^{j_\ell})$, and since $S_\ell(\sigma^{j_\ell})$ is just the intersection of all its paths, we also have $S_\ell(\sigma^{j_\ell}) \leq \pi_\ell$, hence $\bigcap_{\ell \in L} S_\ell(\sigma^{j_\ell}) \leq \bigcap_{\ell \in L} \pi_\ell$. From Lemma 11 we have $\Gamma \vdash_k e_i : \bigcap_{\ell \in L} S_\ell(\tau_i^{j_\ell})$ and $\bigcap_{\ell \in L} \pi_\ell \leq \tau$, from which the claim follows. $\square$

Let $\exp_k$ be the iterated exponential function, given by $\exp_0(n) = n$, $\exp_{k+1}(n) = 2^{\exp_k(n)}$. The lemma below can be shown by an elementary counting argument.

Lemma 13 *For every k, there is a polynomial $p(n)$ such that the number of level-k types over n atoms is at most $\exp_{k+1}(p(n))$, and the size of such types is at most $\exp_k(p(n))$. The number and size of* simple *level-k types (for a fixed k) is respectively bounded by a polynomial and a constant.*

Proof: For general level-k types, we prove by induction with respect to k that the number of level-k types over n atoms can be bounded by $\exp_{k+1}(p(n))$, each of size bounded by $\exp_k(p(n))$. Indeed, at level 0 we have subsets of n possible atoms, yielding 2^n level-0 types each of size bounded by n, and at level $k+1$ we can form intersections (subsets) of types of the form $\tau^{[k]} \to \sigma^{[k]}$, where $\tau^{[k]}$ and $\sigma^{[k]}$ are level-k types. Hence, assuming inductively $\exp_{k+1}(q(n))$ level-k types for a polynomial $q(n)$, the number of level-$(k+1)$ types is bounded by $2^{(\exp_{k+1}(q(n)))^2}$, which can in turn be bounded by $\exp_{k+2}(r(n))$ for a polynomial $r(n)$. Moreover, assuming inductively that the size of a level-k type is bounded by $\exp_k(q(n))$, intersections of at most $\exp_{k+1}(q(n))$ such types are of size bounded by $\exp_k(q(n)) \times \exp_{k+1}(q(n))$, which in turn can be bounded by $\exp_{k+1}(r(n))$.

For simple types bounded by depth k, the claim follows easily by an inductive argument with respect to k. □

Theorem 14 *Inhabitation in* $\text{BCL}_k(\to,\cap)$ *is in* $(k+2)$-EXPTIME.

Proof: The alternating Turing machine shown in Figure 3.2 is a decision procedure for inhabitation in $\text{BCL}_k(\to,\cap)$ for each $k \geq 0$, being a direct alternating implementation of Lemma 11. In Figure 3.2 we use shorthand notation for instruction sequences starting from existential states (CHOOSE ...) and instruction sequences starting from universal states (FORALL$(i = 1 \dots k)\, S_i$). A command of the form CHOOSE $x \in S$ branches from an existential state to successor states in which x gets assigned distinct elements of S. A command of the form FORALL$(i = 1 \dots k)\, S_i$ branches from a universal state to successor states from which each instruction sequence S_i is executed.

The machine operates in bounded space, because, for all Γ, τ, k, x, the set $\mathcal{S}_x^{(\Gamma,\tau,k)}$ is finite. More precisely, it follows from Lemma 13 that the size of $\mathcal{S}_x^{(\Gamma,\tau,k)}$ can be bounded by $\exp_{k+1}(p(n))$. For consider that we have, for some polynomial $r(n)$

$$| \mathcal{S}_x^{(\Gamma,\tau,k)} | = | \mathbb{T}_k(At_\omega(\Gamma,\tau)) |^{|Var(\Gamma)|} \leq (\exp_{k+1}(r(n)))^n$$

which shows that $| \mathcal{S}_x^{(\Gamma,\tau,k)} |$ is bounded by $\exp_{k+1}(p(n))$ for a polynomial $p(n)$.[1] Moreover, we have by Lemma 13 that the size of each level-k type can be bounded by $\exp_k(p(n))$, for some polynomial $p(n)$. It follows that the types σ' (Figure 3.2, line 2) can be written down in space bounded by $\exp_{k+1}(p(n))$, and hence the algorithm is bounded in alternating space $\exp_{k+1}(p(n))$. By the identity ASPACE$(f(n)) =$ DTIME$(2^{\mathcal{O}(f(n))})$ inhabitation is therefore in $(k+2)$-EXPTIME. □

```
$Input:\ \Gamma, \tau, k$
$loop:$
CHOOSE $(x : \sigma) \in \Gamma$;
$\sigma' := \bigcap\{S(\sigma) \mid S \in \mathcal{S}_x^{(\Gamma,\tau,k)}\}$;
CHOOSE $m \in \{0, \ldots, \|\sigma'\|\}$;
CHOOSE $P \subseteq \mathbb{P}_m(\sigma')$;

IF $(\bigcap_{\pi \in P} tgt_m(\pi) \leq \tau)$ THEN
   IF $(m = 0)$ THEN ACCEPT;
   ELSE
         FORALL$(i = 1 \ldots m)$
               $\tau := \bigcap_{\pi \in P} arg_i(\pi)$;
         GOTO $loop$;
```

Figure 3.2: Alternating Turing machine deciding inhabitation in BCL$_k$

---

1 It is worth noting that the number of substitutions at level zero is exponential, even if there are only 2 atoms to substitute for n variables, and that also holds with simple types. The exponential jumps at higher levels are due to intersections.

4

SIMPLE TYPES, $\mathrm{BCL}_k(\to)$

The upper bound for simple types is obtained as a special case of the analysis in Section 3.

Theorem 15 *Inhabitation in $\mathrm{BCL}_k(\to)$ is in* EXPTIME*, for all k.*

Proof: The proof uses the same argument as the proof of Theorem 14. The difference is that now we only substitute simple types. Under this restriction, the machine of Figure 3.2 operates in alternating polynomial space, because all types of the form $S(\sigma)$ are of linear size. $\square$

Theorem 16 *For every $k \geq 0$, the inhabitation problem for $\mathrm{BCL}_k(\to)$ is* EXPTIME*-complete.*

Proof: The argument below is inspired by the EXPTIME-hardness proof in [6]. Take an alternating TM, working in polynomial space $p(n)$. We use fresh type atoms to represent every state and tape symbol. A configuration $\mathcal{C} = wqw'$, where $w = b_1 \ldots b_{m-1}$ and $w' = b_m \ldots b_{p(n)}$ is encoded as a type

$$\varphi_{\mathcal{C}} = b_1 \to \cdots \to b_{m-1} \to q \to b_m \to \cdots \to b_{p(n)}.$$

The initial configuration for input $a_1 \ldots a_n$ is

$$\varphi_0 = q_0 \to a_1 \to \cdots \to a_n \to \sqcup \to \cdots \to \sqcup,$$

with an appropriate number of blanks $\sqcup$. We define an environment Γ so that, for all $\mathcal{C}$,

$$\mathcal{C} \text{ is eventually accepting} \qquad \text{if and only if} \qquad \Gamma \vdash \varphi_{\mathcal{C}}\,. \tag{*}$$

We put into Γ polymorphic patterns for accepting configurations:

$$\alpha_1 \to \cdots \to \alpha_{m-1} \to q_a \to \alpha_m \to \cdots \to \alpha_{p(n)}\,,$$

for all $m = 1, \ldots, p(n) - 1$, and types representing machine moves, as we now define.

Let $\Delta(b, q) = \{(c_j, p_j, h_j) \mid j = 1, \ldots, r\}$. The patterns

$$\zeta_{bqm}(\vec{\alpha}) = \alpha_1 \to \cdots \to \alpha_{m-1} \to q \to b \to \alpha_{m+1} \to \cdots \to \alpha_{p(n)}$$

represent all configurations where $\Delta(b, q)$ is applicable (and m is the head position). For $j = 1, \ldots, r$, type $\eta_{bqmj}(\vec{\alpha})$ represents the j-th successor configuration. If $h_j = \mathsf{R}$ then

$$\eta_{bqmj}(\vec{\alpha}) = \alpha_1 \to \cdots \to \alpha_{m-1} \to c_j \to p_j \to \alpha_{m+1} \to \cdots \to \alpha_{p(n)}\,,$$

otherwise $h^j = \mathsf{L}$, and

$$\eta_{bqmj}(\vec{\alpha}) = \alpha_1 \to \cdots \to \alpha_{m-2} \to p_j \to \alpha_{m-1} \to c_j \to \alpha_{m+1} \to \cdots \to \alpha_{p(n)}\,.$$

The main property of the above types is as follows. If $\mathcal{C} = b_1 \ldots b_{m-1} q b b_{m+1} \ldots b_{p(n)}$ then there exists exactly one substitution S (mapping each α_i to b_i) such that $S(\zeta_{bqm}) = \varphi_{\mathcal{C}}$. In addition, if $\mathcal{D}_1, \ldots, \mathcal{D}_r$ are all the successor configurations of $\mathcal{C}$ then we have $S(\eta_{bqmj}) = \varphi_{\mathcal{D}_j}$. Now if q is an existential state then Γ contains all types of the form

$$\eta_{bqmj} \to \zeta_{bqm}\,,$$

and for universal q there is only one type:

$$\eta_{bqm1} \to \cdots \to \eta_{bqmr} \to \zeta_{bqm} \,.$$

To prove the "only if" part of (*) we proceed by induction with respect to the definition of acceptance. First observe that $\Gamma \vdash \varphi_{\mathcal{C}}$, for all accepting configurations. Then take a non-final existential $\mathcal{C} = wqw'$ with q at position m. If $\mathcal{C}$ is eventually accepting then it is so because one of the successors of $\mathcal{C}$, say $\mathcal{D}$, is eventually accepting. The induction hypothesis applies to $\mathcal{D}$, thus $\Gamma \vdash \varphi_{\mathcal{D}}$ holds. Now, observe that $\varphi_{\mathcal{D}} = \eta_{bqmj}[S]$ and $\varphi_{\mathcal{C}} = \zeta_{bqm}[S]$, for some b, j, and an appropriate substitution S. It follows that $\Gamma \vdash \varphi_{\mathcal{C}}$.

If $\mathcal{C}$ is universal then all its successors $\mathcal{D}_1, \ldots, \mathcal{D}_r$ must be eventually accepting, whence $\varphi_{\mathcal{D}_1}, \ldots, \varphi_{\mathcal{D}_k}$ are all derivable. The substitution S turns $\eta_{bqm1} \to \cdots \to \eta_{bqmk} \to \zeta_{bqm}$ into $\varphi_{\mathcal{D}_1} \to \cdots \to \varphi_{\mathcal{D}_r} \to \varphi_{\mathcal{C}}$, so we can derive $\varphi_{\mathcal{C}}$.

For the "if" part, we use induction with respect to proofs. The base case is only possible when $\mathcal{C}$ is accepting. Otherwise the only way to derive $\Gamma \vdash \varphi_{\mathcal{C}}$, for an existential $\mathcal{C}$, is by using one of the formulas $\eta_{bqmj} \to \zeta_{bqm}$, so for some S we must have $\Gamma \vdash \eta_{bqmj}[S]$ and $\zeta_{bqm}[S] = \varphi_{\mathcal{C}}$. Hence $\eta_{bqmj}[S]$ represents a possible next configuration, and we can use induction.

In the universal case, to derive $\varphi_{\mathcal{C}}$ we must use the type $\eta_{bqm1} \to \cdots \to \eta_{bqmk} \to \zeta_{bqm}$, and a substitution S such that $\Gamma \vdash \eta_{bqmj}[S]$, for all j, and $\zeta_{bqm}[S] = \varphi_{\mathcal{C}}$. But then $\eta_{bqmj}[S] = \varphi_{\mathcal{D}_j}$, where $\mathcal{D}_j$ are the successors of $\mathcal{C}$. By the induction hypothesis, all $\mathcal{D}_j$ are eventually accepting, and therefore so is $\mathcal{C}$. □

5

LOWER BOUND FOR INTERSECTION TYPES

In this section we fix a number K and an $\exp_{K+1}(n)$-space bounded alternating Turing machine $\mathcal{M}$. In what follows it is assumed that $k \leq K$, whenever level k is considered. The basic idea is to represent a configuration of $\mathcal{M}$ by, essentially, a type of the form

$$\bigcap_{i=0}^{\exp_{K+1}(n)-1} \mathsf{Cell}(a_i, q, \langle m \rangle_K, \langle i \rangle_K),$$

where $a_i \in \Sigma$, $q \in Q$, $0 \leq m \leq \exp_{K+1}(n) - 1$. Each component $\mathsf{Cell}(a_i, q, \langle m \rangle_K, \langle i \rangle_K)$ represents one of the tape cells, where a_i represents the symbol in the i-th cell, q represents the current state, type $\langle m \rangle_K$ represents the address (number) of the cell which is under the current ATM head position, and $\langle i \rangle_K$ represents the address of the cell itself. Notice that the types q and $\langle m \rangle_K$ are identical across all the components of the type (i.e., across all indexes i). The adresses $\langle i \rangle_K$ impose a numerical order on the cell representations, so that we can represent a tape consisting of a sequence of cells. Moreover, we can use these addresses to compute the head position of the ATM (moving left or right of the current cell address).

Since we need a representation which is polynomial bounded in the size of the ATM input, we cannot represent such types explicitly in our reduction. In order to achieve a succinct (polynomial sized) representation, we exploit polymorphism. The basic insight in the reduction is to represent the large configuration types *implicitly*, as polymorphic types $\mathsf{Cell}(\alpha, q, \beta, \gamma)$, and to arrange the environment Γ coding the behavior of $\mathcal{M}$ in such a way that large expansions (under polymorphic instantiation) of such types become forced into the explicit form shown. As in the proof of Theorem 16, the basic strategy for coding the ATM behavior is to represent a computation sequence $\mathcal{C}_1\mathcal{C}_2 \cdots \mathcal{C}_m$ by a sequence of forced inhabitation goals in reverse order of implication, by (essentially) having the implications $[\mathcal{C}_{i+1}] \to [\mathcal{C}_i]$ in Γ such that asking for inhabitation of $[\mathcal{C}_i]$ forces the inhabitation of $[\mathcal{C}_{i+1}]$ (letting $[\mathcal{C}]$ denote the type representing the configuration $\mathcal{C}$).

Predicates

The *predicates* we use are certain type patterns serving as “containers” for their arguments. The idea is that a predicate like $F(\tau, \sigma)$ encodes a pair of types τ and σ and a “flag” F in a unique way. This is achieved by making sure that type $F(\tau, \sigma)$ is large enough to never be substituted for a variable. In addition, τ and σ are placed inside $F(\tau, \sigma)$ several times to avoid unwanted subtyping.

Some auxiliary notation for the beginning. Write $F^{[1]}$ for F and $F^{[n+1]}$ for $F^{[n]} \to F$. For instance, $F^{[4]} = ((F \to F) \to F) \to F$. Also let $\Omega_\tau = (\tau \to \tau) \to \tau \to \tau$.

Let $N > K$ be a fixed number. Type $F(\tau_1, \tau_2, \tau_3, \tau_4)$ (a predicate of four arguments) is defined using a dedicated type constant F (the *predicate identifier*), as follows:

$$F(\tau_1, \tau_2, \tau_3, \tau_4) = (((F^{[N]} \to \Omega_{\tau_1}) \to \Omega_{\tau_2}) \to \Omega_{\tau_3}) \to \Omega_{\tau_4}.$$

Predicates of fewer arguments are defined by repeating the last one, e.g. $G(\tau, \sigma)$ will stand for $G(\tau, \sigma, \sigma, \sigma)$. In what follows, the word "predicate" may refer to any $F(\tau_1, \ldots, \tau_4)$.
The level of $F(\tau_1, \ldots, \tau_4)$ is larger than K, and therefore types of the form $F(\tau_1, \ldots, \tau_4)$ never occur in the range of a substitution. Further properties are as follows:

Lemma 17 *For all types τ, σ and all predicates Φ_i and Φ:*

1. *If $\bigcap_{i \in I} \Omega_{\tau_i} \leq \Omega_\sigma$ then $\tau_i = \sigma$, for some i.*
2. *If $\bigcap_{i \in I} \Phi_i \leq \Phi$ then $\Phi = \Phi_i$, for some i.*

Proof: Use Lemma 2. Details omitted. □

In our construction we use the following forms of predicates (for $k \leq K$ and $j \leq n$):

- Unary: $\mathsf{Zero}_k(\alpha)$, $\mathsf{z}_k(\alpha)$, $\mathsf{m}_k(\alpha)$, $\mathsf{Max}_k(\alpha)$, $\mathsf{Num}_k(\alpha)$, $\mathsf{n}^k(\alpha)$, $\mathsf{Num}^j(\alpha)$, $\mathsf{Bit}(\alpha)$, $\mathsf{Tape}^j(\alpha)$.
- Binary: $\mathsf{Succ}_k(\alpha, \beta)$, $\mathsf{Diff}_k(\alpha, \beta)$, $\mathsf{d}_k(\alpha, \beta)$, $\mathsf{n}_k(\alpha, \beta)$.
- Ternary: $\mathsf{R}_k(\alpha, \beta, \gamma)$, $\mathsf{L}_k(\alpha, \beta, \gamma)$.
- Quaternary: $\mathsf{Cell}(\alpha, \beta, \gamma, \delta)$.

In addition to that we also have the following constants (for $j \leq n$):

- 0, 1, 0_j, 1_j, $\bullet$.

and special constants for all internal states and tape symbols of the machine.

Intersection type numerals

Fix a natural number n. Let $\mathbb{B}[n]$ denote the union of n copies of $\mathbb{B} = \{0, 1\}$, written $\mathbb{B}[n] = \{0_1, \ldots, 0_n\} \cup \{1_1, \ldots, 1_n\}$. We let b range over $\mathbb{B}$ and we let $\mathfrak{b}$ range over $\mathbb{B}[n]$. The sets of *level-k numerals* ($k \geq 0$), denoted $\mathcal{N}_k$, are constructed from $\mathbb{B}[n]$ by induction:

- $\mathcal{N}_0 = \{\bigcap_{i=1}^{n} \mathfrak{b}_i \mid \mathfrak{b}_i \in \{0_i, 1_i\} \text{ for } i = 1 \ldots n\}$
- $\mathcal{N}_{k+1} = \{\bigcap_{\tau \in \mathcal{N}_k} (\tau \to b_\tau) \mid b_\tau \in \{0, 1\}, \text{ for } \tau \in \mathcal{N}_k\}$

Clearly, the size of $\mathcal{N}_k$ is $\exp_{k+1}(n)$. The value of a numeral $\sigma \in \mathcal{N}_k$ is denoted $[\![\sigma]\!]$ and is defined by induction with respect to k:

- $k = 0$: $[\![\bigcap_{i=1}^{n} \mathfrak{b}_i]\!] = \sum_{i=1}^{n} [\![\mathfrak{b}_i]\!] \times 2^{i-1}$, with $[\![0_i]\!] = 0$ and $[\![1_i]\!] = 1$
- $k > 0$: $[\![\bigcap_{\sigma \in \mathcal{N}_k} (\tau \to b_\tau)]\!] = \sum_{\tau \in \mathcal{N}_k} b_\tau \times 2^{[\![\tau]\!]}$

For instance, if $n = 4$ then the value of $0_1 \cap 1_2 \cap 0_3 \cap 1_4$ is $2 + 8 = 10$. And if $n = 2$ then the value of $((0_1 \cap 0_2) \to 0) \cap ((0_1 \cap 1_2) \to 1) \cap ((1_1 \cap 0_2) \to 0) \cap ((1_1 \cap 1_2) \to 1)$ is 10 as well.

It is easy to prove by induction that for $\sigma \in \mathcal{N}_k$ we have $0 \leq [\![\sigma]\!] \leq \exp_{k+1}(n) - 1$, and for $k > 0$ we can write σ canonically as $\sigma = \bigcap_{i=0}^{\exp_k(n)-1} (\tau_i \to b_i)$, where $[\![\tau_i]\!] = i$ and $b_i \in \mathbb{B}$, and with $[\![\sigma]\!] = \sum_{i=0}^{\exp_k(n)-1} b_i \times 2^i$.

It is also straightforward to see that, for any x between 0 and $\exp_{k+1}(n) - 1$, there is *exactly one* $\sigma \in \mathcal{N}_k$ with $[\![\sigma]\!] = x$. We use the notation $\sigma = \langle x \rangle_k$.

The encoding

Our goal is to define a $\textsc{bcl}_K$ type environment Γ, representing the behavior of the machine $\mathcal{M}$. The environment Γ consists of several groups of declarations, to handle predicates over numerals, the tape, and the transition function. Note that each type σ in Γ is an intersection which has a component of the form $(\bullet^m \to \bullet)$, for some m, and that all other components are arrows of m arguments, ending with predicates of the same identifier F. We then say that σ, and the corresponding combinator, is *m-ary*, and that F is the *target identifier* of σ.

Lemma 18 *If x is m-ary and $\Gamma \vdash_K xe_1 \dots e_r : \bullet$ then $r = m$.*

Proof: If $\Gamma \vdash_K xe_1 \dots e_r : \bullet$ then by Lemma 11 we have $\bigcap_{\pi \in P} tgt_r(\pi) \leq \bullet$, for some set P of paths in types of the form $S(\Gamma(x))$. The only such path is $\bullet^m \to \bullet$, whence $m = r$. $\square$

Lemma 19 *Let $\Gamma \vdash_K e : F(\tau_1, \dots, \tau_4) \cap \bullet$, where $F(\tau_1, \dots, \tau_4)$ is a predicate. Then $e = xe_1 \dots e_m$, for some m-ary combinator x with target identifier F. More precisely, $\Gamma(x)$ has the form $\xi \cap (\zeta_1 \to \dots \to \zeta_m \to F(\rho_1, \dots, \rho_4))$, and there is a substitution S such that $S(\rho_i) = \tau_i$, for $i = 1, \dots, 4$, and $\Gamma \vdash_K e_i : S(\zeta_i)$, for $i = 1, \dots, m$.*

Proof: The term e must be of the form $e = xe_1 \dots e_r$, where x is a combinator of some arity m in Γ. It follows from Lemma 18 that $m = r$, and from Corollary 12 we obtain that $\bigcap_{\ell \in L} \Phi_\ell \cap \bullet \leq F(\tau_1, \dots, \tau_4) \cap \bullet$, where Φ_ℓ are predicates with the same target G. Since $\bullet$ is a constant, we actually have $\bigcap_{\ell \in L} \Phi_\ell \leq F(\tau_1, \dots, \tau_4)$. By Lemma 17, one of Φ_ℓ must be equal to $F(\tau_1, \dots, \tau_4)$, in particular $F = G$. Note that Φ_ℓ is obtained as $S(tgt_m(\phi))$, for some component ϕ of $\Gamma(x)$, and this S is the substitution required by the lemma. $\square$

Numeral predicates

The declarations shown in Figure 5.1 and Figure 5.2 are included in Γ, for every $k < K$. Together they specify the way numerals are handled at each level k. The predicates are defined inductively with respect to k. Thus, in Figure 5.1 we define the base predicates for numerals in $\mathcal{N}_0$, whereas Figure 5.2 contains definitions for predicates at all higher levels $k+1$. These latter definitions may inductively refer to definitions at lower levels (for example, in Figure 5.2, the declaration for the combinator $\mathbf{N}_{k+1}$ refers to the lower level predicate Zero_k).

$$
\begin{array}{lcl}
\mathbf{Z}_0 & : & \mathsf{Zero}_0(0_1 \cap 0_2 \cap \cdots \cap 0_n) \cap \bullet \\
\mathbf{M}_0 & : & \mathsf{Max}_0(1_1 \cap 1_2 \cap \cdots \cap 1_n) \cap \bullet \\
 & & \\
\mathbf{N}_0 & : & [\mathsf{n}^2(\alpha) \to \mathsf{Num}_0(1_1 \cap \alpha)] \cap [\mathsf{n}^2(\alpha) \to \mathsf{Num}_0(0_1 \cap \alpha)] \cap [\bullet \to \bullet] \\
\mathbf{n}_0^2 & : & [\mathsf{n}^3(\alpha) \to \mathsf{n}^2(1_2 \cap \alpha)] \cap [\mathsf{n}^3(\alpha) \to \mathsf{n}^2(0_2 \cap \alpha)] \cap [\bullet \to \bullet] \\
\ldots & : & \ldots \\
\mathbf{n}_0^n & : & \mathsf{n}^n(1_n) \cap \mathsf{n}^n(0_n) \cap \bullet \\
 & & \\
\mathbf{D}_0 & : & [\mathsf{d}_0(\alpha,\beta) \to \mathsf{Num}_0(\alpha) \to \mathsf{Num}_0(\beta) \to \mathsf{Diff}_0(\alpha,\beta)] \cap [\bullet \to \bullet \to \bullet \to \bullet] \\
\mathbf{d}_0 & : & \bigcap_{i=1}^{n}(\mathsf{d}_0(0_i \cap \alpha, 1_i \cap \beta) \cap \mathsf{d}_0(1_i \cap \alpha, 0_i \cap \beta)) \cap \bullet \\
 & & \\
\mathbf{S}_0 & : & [\mathsf{Num}_0(0_1 \cap \alpha) \to \mathsf{Num}_0(1_1 \cap \alpha) \to \mathsf{Succ}_0(0_1 \cap \alpha, 1_1 \cap \alpha)] \cap \\
 & & [\mathsf{Num}_0(1_1 \cap 0_2 \cap \alpha) \to \mathsf{Num}_0(0_1 \cap 1_2 \cap \alpha) \to \mathsf{Succ}_0(1_1 \cap 0_2 \cap \alpha, 0_1 \cap 1_2 \cap \alpha)] \cap \\
 & & \ldots \cap \\
 & & [\mathsf{Num}_0(1_1 \cap 1_2 \cap \cdots \cap 1_{n-1} \cap 0_n) \to \\
 & & \quad \mathsf{Num}_0(0_1 \cap 0_2 \cap \cdots \cap 0_{n-1} \cap 1_n) \to \\
 & & \qquad \mathsf{Succ}_0(1_1 \cap 1_2 \cap \cdots \cap 1_{n-1} \cap 0_n, 0_1 \cap 0_2 \cap \cdots \cap 0_{n-1} \cap 1_n)] \cap \\
 & & [\bullet \to \bullet \to \bullet]
\end{array}
$$

Figure 5.1: Numeral predicates, level 0

Turing machine

Now we turn to the actual machine simulation. Declarations in Figure 5.3 are used to "create" the initial configuration with input word $a_1 \ldots a_n$ and with further tape cells filled with blanks up to length $\exp_{K+1}(n)$. Tape cells are identified by numbers from 0 to $\exp_{K+1}(n) - 1$.

Before we define the core part of our coding, we introduce one more notational convention. A multiple implication $\tau_1 \to \tau_2 \to \cdots \to \tau_m \to \tau$ is sometimes written as $(\tau_1, \ldots, \tau_m) \to \tau$. We extend this style by using informal abbreviations for sequences of premises. For instance,

$$
\begin{array}{lcl}
\mathbf{B} & : & \mathsf{Bit}(0) \cap \mathsf{Bit}(1) \cap \bullet \\
\mathbf{Z}_{k+1} & : & [\mathsf{Num}_{k+1}(\alpha) \to \mathsf{z}_{k+1}(\alpha) \to \mathsf{Zero}_{k+1}(\alpha)] \cap [\bullet \to \bullet \to \bullet] \\
\mathbf{z}_{k+1} & : & [\mathsf{z}_{k+1}(\alpha) \to \mathsf{z}_{k+1}((\beta \to 0) \cap \alpha)] \cap [\bullet \to \bullet] \\
\mathbf{z}'_{k+1} & : & \mathsf{z}_{k+1}(\beta \to 0) \cap \bullet \\
 & & \\
\mathbf{M}_{k+1} & : & [\mathsf{Num}_{k+1}(\alpha) \to \mathsf{m}_{k+1}(\alpha) \to \mathsf{Max}_{k+1}(\alpha)] \cap [\bullet \to \bullet \to \bullet] \\
\mathbf{m}_{k+1} & : & [\mathsf{m}_{k+1}(\alpha) \to \mathsf{m}_{k+1}((\beta \to 1) \cap \alpha)] \cap [\bullet \to \bullet] \\
\mathbf{m}'_{k+1} & : & \mathsf{m}_{k+1}(\beta \to 1) \cap \bullet \\
 & & \\
\mathbf{N}_{k+1} & : & [\mathsf{Bit}(\gamma) \to \mathsf{n}_{k+1}(\beta \to \gamma, \alpha) \to \mathsf{Zero}_{k}(\beta) \to \mathsf{Num}_{k+1}((\beta \to \gamma) \cap \alpha)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \\
\mathbf{n}_{k+1} & : & [\mathsf{Bit}(\varepsilon) \to \mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{n}_{k+1}(\delta \to \varepsilon, \alpha) \to \mathsf{n}_{k+1}(\beta \to \gamma, (\delta \to \varepsilon) \cap \alpha)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \\
\mathbf{n}'_{k+1} & : & [\mathsf{Bit}(\varepsilon) \to \mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{Max}_{k}(\delta) \to \mathsf{n}_{k+1}(\beta \to \gamma, \delta \to \varepsilon)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \\
 & & \\
\mathbf{D}_{k+1} & : & [\mathsf{d}_{k+1}(\alpha, \beta) \to \mathsf{Num}_{k+1}(\alpha) \to \mathsf{Num}_{k+1}(\beta) \to \mathsf{Diff}_{k+1}(\alpha, \beta)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \\
\mathbf{d}_{k+1} & : & \mathsf{d}_{k+1}((\gamma \to 1) \cap \alpha, (\gamma \to 0) \cap \beta) \cap \mathsf{d}_{k+1}((\delta \to 0) \cap \alpha, (\delta \to 1) \cap \beta) \cap \bullet \\
 & & \\
\mathbf{S}_{k+1} & : & [\mathsf{R}_{k+1}(\beta, \alpha, \gamma) \to \mathsf{Zero}_{k}(\beta) \to \mathsf{Succ}_{k+1}((\beta \to 0) \cap \alpha, (\beta \to 1) \cap \gamma)] \cap \\
 & & [\mathsf{L}_{k+1}(\beta, \alpha, \gamma) \to \mathsf{Zero}_{k}(\beta) \to \mathsf{Succ}_{k+1}((\beta \to 1) \cap \alpha, (\beta \to 0) \cap \gamma)] \cap \\
 & & [\bullet \to \bullet \to \bullet] \\
\mathbf{s}_{k+1} & : & [\mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{L}_{k+1}(\delta, \alpha, \gamma) \to \mathsf{L}_{k+1}(\beta, (\delta \to 1) \cap \alpha, (\delta \to 0) \cap \gamma)] \cap \\
 & & [\mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{R}_{k+1}(\delta, \alpha, \gamma) \to \mathsf{L}_{k+1}(\beta, (\delta \to 0) \cap \alpha, (\delta \to 1) \cap \gamma)] \cap \\
 & & [\mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{R}_{k+1}(\delta, \alpha, \gamma) \to \mathsf{R}_{k+1}(\beta, (\delta \to 0) \cap \alpha, (\delta \to 0) \cap \gamma)] \cap \\
 & & [\mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{R}_{k+1}(\delta, \alpha, \gamma) \to \mathsf{R}_{k+1}(\beta, (\delta \to 1) \cap \alpha, (\delta \to 1) \cap \gamma)] \cap \\
 & & [\bullet \to \bullet \to \bullet] \\
\mathbf{s}'_{k+1} & : & [\mathsf{Max}_{k}(\delta) \to \mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{R}_{k+1}(\beta, \delta \to 0, \delta \to 0)] \cap \\
 & & [\mathsf{Max}_{k}(\delta) \to \mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{R}_{k+1}(\beta, \delta \to 1, \delta \to 1)] \cap \\
 & & [\mathsf{Max}_{k}(\delta) \to \mathsf{Succ}_{k}(\beta, \delta) \to \mathsf{L}_{k+1}(\beta, \delta \to 0, \delta \to 1)] \cap \\
 & & [\bullet \to \bullet \to \bullet]
\end{array}
$$

Figure 5.2: Numeral predicates, level $k+1$

type $\tau_1 \to \tau_2 \to \tau_3 \to \sigma_1 \to \sigma_2 \to \sigma_3 \to \tau$ may be written as $A \to B \to \tau$, where $A = (\tau_1, \tau_2, \tau_3)$ and $B = (\sigma_1, \sigma_2, \sigma_3)$.

$$\begin{array}{lcl}
\mathbf{Init} & : & [\mathsf{Zero}_K(\alpha) \to \mathsf{Cell}(a_1,\ q_0,\ \alpha,\ \alpha) \cap \mathsf{Tape}^1(\alpha) \to \mathsf{Tape}] \cap [\bullet \to \bullet \to \bullet] \\
\mathbf{init}^i & : & [\mathsf{Zero}_K(\gamma) \to \mathsf{Succ}_K(\alpha,\beta) \to \mathsf{Tape}^{i+1}(\beta) \cap \mathsf{Cell}(a_{i+1},\ q_0,\ \gamma,\ \beta) \to \mathsf{Tape}^i(\alpha)] \cap \\
 & & [\mathsf{Zero}_K(\gamma) \to \mathsf{Succ}_K(\alpha,\beta) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \qquad\qquad\qquad\qquad\qquad\qquad \text{(for all } i < n\text{)} \\
\mathbf{init}^n & : & [\mathsf{Zero}_K(\gamma) \to \mathsf{Succ}_K(\alpha,\beta) \to \mathsf{Tape}^n(\beta)\ \cap\ \mathsf{Cell}(\sqcup,\ q_0,\ \gamma,\ \beta)\ \to \mathsf{Tape}^n(\alpha)] \cap \\
 & & [\mathsf{Zero}_K(\gamma) \to \mathsf{Succ}_K(\alpha,\beta) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon)] \cap \\
 & & [\bullet \to \bullet \to \bullet \to \bullet] \\
\mathbf{finit} & : & [\mathsf{Max}_K(\alpha) \to \bullet \to \mathsf{Tape}^n(\alpha)] \cap \\
 & & [\mathsf{Max}_K(\alpha) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon) \to \mathsf{Cell}(\eta,\ q_0,\ \delta,\ \varepsilon)]\ \cap [\bullet \to \bullet \to \bullet]
\end{array}$$

Figure 5.3: Initial configuration under construction

Given q and b, let $\Delta(b,q) = \{(c_i,p_i,h_i) \mid i = 1,\ldots,r\}$. By $\mathsf{V}^{qbi}(\delta)$ and $\mathsf{U}^{qbi}(\alpha,\delta,\gamma)$ we abbreviate triples of types used to represent the transition defined by (c_i,p_i,h_i). The role of $\mathsf{V}^{qbi}(\delta)$ is to encode the action at the presently scanned tape cell, while $\mathsf{U}^{qbi}(\alpha,\delta,\gamma)$ applies to all other tape cells. Assume first that $h_i = \textsc{l}$. Then we define:

$$\begin{array}{lcl}
\mathsf{V}^{qbi}(\delta) & = & (\mathsf{Succ}_K(\beta,\delta),\ \mathsf{Diff}_K(\xi,\zeta),\ \mathsf{Cell}(c_i,p_i,\beta,\delta)), \\
\mathsf{U}^{qbi}(\alpha,\delta,\gamma) & = & (\mathsf{Succ}_K(\beta,\delta),\ \mathsf{Diff}_K(\gamma,\delta),\ \mathsf{Cell}(\alpha,p_i,\beta,\gamma)).
\end{array}$$

If $h_i = \textsc{r}$ then the definition is altered as follows:

$$\begin{array}{lcl}
\mathsf{V}^{qbi}(\delta) & = & (\mathsf{Succ}_K(\delta,\beta),\ \mathsf{Diff}_K(\xi,\zeta),\ \mathsf{Cell}(c_i,p_i,\beta,\delta)), \\
\mathsf{U}^{qbi}(\alpha,\delta,\gamma) & = & (\mathsf{Succ}_K(\delta,\beta),\ \mathsf{Diff}_K(\gamma,\delta),\ \mathsf{Cell}(\alpha,p_i,\beta,\gamma)).
\end{array}$$

Now, if q is an existential state then for every $i \leq r$ there is a combinator

$$\begin{array}{lcl}
\mathbf{Step}^{qbi} & : & [\mathsf{V}^{qbi}(\delta) \ \to\ \mathsf{Cell}(b,q,\delta,\delta)] \cap \\
 & & [\mathsf{U}^{qbi}(\alpha,\delta,\gamma) \to \mathsf{Cell}(\alpha,q,\delta,\gamma)] \cap \\
 & & [\bullet^3 \ \to\ \bullet]
\end{array}$$

For universal q, we declare one combinator $\mathbf{Step}^{qb}$:

$$\begin{array}{lcl}
\mathbf{Step}^{qb} & : & [\mathsf{V}^{qb1}(\delta) \ \to\ \cdots \ \to\ \mathsf{V}^{qbr}(\delta) \ \to\ \mathsf{Cell}(b,q,\delta,\delta)] \cap \\
 & & [\mathsf{U}^{qb1}(\alpha,\delta,\gamma) \to \cdots \to \mathsf{U}^{qbr}(\alpha,\delta,\gamma) \to \mathsf{Cell}(\alpha,q,\delta,\gamma)] \cap \\
 & & [\bullet^3 \ \to\ \cdots \ \to\ \bullet^3 \ \to\ \bullet]
\end{array}$$

Properties of the coding

We now collect the main properties of our coding. The first two lemmas state that our numeral system works properly.

Lemma 20 *For every $k \leq K$ there are terms $Zero_k$, Max_k, Num_k, $Diff_k$, $Succ_k$, such that for all types σ and τ:*

1. *If* $\sigma = \langle\, 0 \,\rangle_k$ *then* $\Gamma \vdash_K Zero_k : \mathsf{Zero}_k(\sigma) \cap \bullet$.
2. *If* $\sigma = \langle\, \exp_{k+1}(n) - 1 \,\rangle_k$ *then* $\Gamma \vdash_K Max_k : \mathsf{Max}_k(\sigma) \cap \bullet$.
3. *If* $\sigma \in \mathcal{N}_k$ *then* $\Gamma \vdash_K Num_k : \mathsf{Num}_k(\sigma) \cap \bullet$.
4. *If* $\sigma, \tau \in \mathcal{N}_k$, *and* $[\![\sigma]\!] \neq [\![\tau]\!]$ *then* $\Gamma \vdash_K Diff_k : \mathsf{Diff}_k(\sigma, \tau) \cap \bullet$.
5. *If* $\sigma, \tau \in \mathcal{N}_k$, *and* $[\![\sigma]\!] + 1 = [\![\tau]\!]$ *then* $\Gamma \vdash_K Succ_k : \mathsf{Succ}_k(\sigma, \tau) \cap \bullet$.

Proof: Beginning with $k = 0$, we have $Num_0 = \mathbf{N}_0(\mathbf{n}_0^2(\mathbf{n}_0^3(\ldots(\mathbf{n}_0^{n-1}(\mathbf{n}_0^n))\ldots)))$, $Zero_0 = \mathbf{Z}_0$, $Max_0 = \mathbf{M}_0$, $Diff_0 = \mathbf{D}_0\mathbf{d}_0 Num_0 Num_0$, and $Succ_0 = \mathbf{S}_0 Num_0 Num_0$. Take $max = \exp_{k+1}(n)$ and for $k \geq 0$ define $Num_{k+1} = \mathbf{N}_{k+1}\mathbf{B}((\mathbf{n}_{k+1}\mathbf{B}Succ_k)^{max-2}(\mathbf{n}'_{k+1}\mathbf{B}Succ_k Max_k))Zero_k$, $Zero_{k+1} = \mathbf{Z}_{k+1}Num_{k+1}(\mathbf{z}_{k+1}^{max-1}(\mathbf{z}'_{k+1}))$, and $Max_{k+1} = \mathbf{M}_{k+1}Num_{k+1}(\mathbf{m}_{k+1}^{max-1}(\mathbf{m}'_{k+1}))$. Now we can define successor $Succ_{k+1} = \mathbf{S}_{k+1}((\mathbf{s}_{k+1}Succ_k)^{max-2}(\mathbf{s}'_{k+1}Max_k Succ_k))Zero_k$, and the last term we need is $Diff_{k+1} = \mathbf{D}_{k+1}\mathbf{d}_{k+1}Num_{k+1}Num_{k+1}$. □

Lemma 21 *For every $k \leq K$ and every e:*

1. *If* $\Gamma \vdash_K e : \mathsf{Zero}_k(\sigma) \cap \bullet$ *then* $\sigma = \langle\, 0 \,\rangle_k$.
2. *If* $\Gamma \vdash_K e : \mathsf{Max}_k(\sigma) \cap \bullet$ *then* $\sigma = \langle\, \exp_{k+1}(n) - 1 \,\rangle_k$.
3. *If* $\Gamma \vdash_K e : \mathsf{Num}_k(\sigma) \cap \bullet$ *then* $\sigma \in \mathcal{N}_k$.
4. *If* $\Gamma \vdash_K e : \mathsf{Diff}_k(\sigma, \tau) \cap \bullet$ *then* $\sigma, \tau \in \mathcal{N}_k$, *and* $[\![\sigma]\!] \neq [\![\tau]\!]$.
5. *If* $\Gamma \vdash_K e : \mathsf{Succ}_k(\sigma, \tau) \cap \bullet$ *then* $\sigma, \tau \in \mathcal{N}_k$, *and* $[\![\sigma]\!] + 1 = [\![\tau]\!]$.

Proof: The proof is by induction with respect to k, and we show the five claims in the order of their numbers. Of the ten possible cases we consider $\Gamma \vdash_K e : \mathsf{Num}_{k+1}(\sigma) \cap \bullet$ as an example. It follows from Lemma 19 that $e = \mathbf{N}_{k+1}e_1e_2e_3$, and $\sigma = S((\beta \to \gamma) \cap \alpha)$ and we can derive $\Gamma \vdash_K e_1 : \mathsf{Bit}(S(\gamma)) \cap \bullet$, $\Gamma \vdash_K e_2 : \mathbf{n}_{k+1}(S(\beta \to \gamma), S(\alpha)) \cap \bullet$, and $\Gamma \vdash_K e_3 : \mathbf{Zero}_{k+1}(S(\beta)) \cap \bullet$, for some S. Then $S(\beta) = \langle\, 0 \,\rangle_k$ and $S(\gamma)$ is 0 or 1. We prove by induction that $\Gamma \vdash_K e' : \mathbf{n}_{k+1}(\varphi, \tau)$ implies $\varphi = \langle\, i \,\rangle_k \to \varphi'$ and $\tau = \bigcap_{j>i}\langle\, j \,\rangle_k \to b_j$, for some i, and conclude that $\sigma = \bigcap_{j\geq 0}\langle\, j \,\rangle_k \to b_j$, i.e., that σ is indeed a numeral. □

Let $\mathcal{C} = wqw'$ be a configuration of our machine $\mathcal{M}$. Assume that $w = b_0 \ldots b_{h-1}$ and $w' = b_h \ldots b_{\exp_{K+1}(n)-1}$. That is, the address of the currently scanned tape cell is h. We take the following type to be the encoding of $\mathcal{C}$:

$$[\mathcal{C}] \;=\; \bigcap_{i=0}^{\exp_{K+1}(n)-1} \mathsf{Cell}(b_i, q, \langle h \rangle_K, \langle i \rangle_K).$$

Now let $\mathcal{C}_0$ be the initial configuration for input $a_1 \ldots a_n$. (Thus $b_i = a_{i+1}$, for $i < n$.)

Lemma 22 *The intersection* $\mathsf{Tape}\cap\bullet$ *is inhabited in* Γ *iff so is* $[\mathcal{C}_0]\cap\bullet$.

Proof: Suppose that $\Gamma\vdash e : [\mathcal{C}_0]\cap\bullet$. If $T_i = \mathbf{init}^i Zero_K Succ_K$, for $i = 1,\dots,n$, then

$$\Gamma\vdash \mathbf{Init}Zero_K(T_1(T_2(\cdots(T_{n-1}(T_n^{m-n}(\mathbf{finit}Max_K\, e)))\cdots))) : \mathsf{Tape}\cap\bullet\,.$$

On the other hand, if $\Gamma\vdash e : \mathsf{Tape}\cap\bullet$ then $e = xe_1\dots e_m$, where x is m-ary (Lemma 18). Since $\Gamma\vdash e : \mathsf{Tape}$, the only possibility is that $e = \mathbf{Init}\,e_1e_2$, where $\Gamma\vdash e_1 : Zero_K(\langle 0\rangle_K)$ and $\Gamma\vdash e_2 : \mathsf{Tape}^1(\langle 0\rangle_K)\cap\mathsf{Cell}(a_1, q_0, \langle 0\rangle_K, \langle 0\rangle_K\rangle$. We prove by induction wrt r that $e_2 = T_1(T_2(\cdots(e')\cdots))$, where e' has type $\bullet\cap\mathsf{Tape}^{\ell}(\langle r\rangle)\cap\bigcap_{i\le r}\mathsf{Cell}(b_i, q_0, \langle 0\rangle, \langle i\rangle)$, and $\ell = \min\{r+1, n\}$. For $r = \exp_{K+1}(n)-1$, term e' is of type $\bullet\cap\mathsf{Tape}^n(\langle r\rangle_K)\cap[\mathrm{C}_0]$. □

Lemma 23 *A configuration* $\mathcal{C}$ *is eventually accepting iff* $\Gamma\vdash[\mathcal{C}]\cap\bullet$.

Proof: The "only if" part goes by induction with respect to the definition of acceptance. If $\mathcal{C}$ is an accepting configuration (universal without successors) then we have a declaration

$$\mathbf{Step}^{q_ab}\ :\ \mathsf{Cell}(b, q_a, \delta, \delta)\ \cap\ \mathsf{Cell}(\alpha, q_a, \delta, \gamma)\ \cap\ \bullet,$$

for appropriate b, whence $\Gamma\vdash\mathbf{Step}^{q_ab} : [\mathcal{C}]\cap\bullet$. Let $\mathcal{C} = wqbw'$ be existential, with q at address t. If $\mathcal{C}\to\mathcal{C}'$, with $\mathcal{C}'$ eventually accepting then, by the induction hypothesis, $[\mathcal{C}']\cap\bullet$ is inhabited. Assume for example that $\mathcal{C}'$ is obtained from $\mathcal{C}$ using a triple $(c_i, p_i, h_i)\in\Delta(b, q)$, with $h_i = \textsc{l}$. Then $[\mathcal{C}']$ differs from $[\mathcal{C}]$ in that we have $\mathsf{Cell}(c_i, p_i, \langle t-1\rangle, \langle t\rangle)$ instead of $\mathsf{Cell}(b, q, \langle t\rangle, \langle t\rangle)$ and $\mathsf{Cell}(b_j, p_i, \langle t-1\rangle, \langle j\rangle)$ instead of $\mathsf{Cell}(b_j, q, \langle t\rangle, \langle j\rangle)$, for all $j\neq t$. It follows that $\Gamma\vdash\mathbf{Step}^{qbi}Succ_kDiff_k\, e : [\mathcal{C}]\cap\bullet$, where $Succ_k$ and $Diff_k$ are defined as in Lemma 20 for appropriate k and e is an inhabitant of $[\mathcal{C}']\cap\bullet$.

In the universal case, we build an inhabitant of $[\mathcal{C}]\cap\bullet$ as $\mathbf{Step}^{qb}Succ_kDiff_k\, e_1\dots Succ_kDiff_k\, e_r$, where $Succ_k$ and $Diff_k$ are as above $e_1,\dots,e_r$ prove the codes of all successor configurations.

The proof from right to left is by induction with respect to length of inhabitants. Let $\Gamma\vdash e : [\mathcal{C}]\cap\bullet$. If e is a single combinator then $e = \mathbf{Step}^{q_ab}$, by Lemma 19. Otherwise $e = xe_1\dots e_m$, for an m-ary x. It is possible that $e = \mathbf{init}^ie_0e_1e_2$ or $\mathbf{finit}e_1e_2$, but then e_2 also proves $[\mathcal{C}]\cap\bullet$. Therefore the shortest inhabitant must begin with $\mathbf{Step}^{qbi}$ or $\mathbf{Step}^{qb}$, and we proceed as in the proof of Lemmas 21 and 22, using Lemma 19 as a basic tool. □

Theorem 24 *For every* $k\ge 0$*, the relativized inhabitation problem for* $\textsc{bcl}_k(\to,\cap)$ *is complete in* $(k{+}2)$*-*Exptime.

Proof: By a routine padding argument[1] it suffices to prove that the halting problem for $\exp_{k+1}(n)$-space bounded ATM's is reducible to inhabitation in $\textsc{bcl}_k(\to,\cap)$. The latter claim follows from Lemmas 22 and 23: to determine if $\mathcal{M}$ accepts the input it is enough to ask if $\Gamma\vdash\bullet\cap\mathsf{Tape}$. □

---

[1] If $L\in\textsc{Dtime}(\exp_{k+1}(p(n)))$ then $L\le_{\log}\{w\#^{p(n)-|w|}\mid w\in L\}\in\textsc{Dtime}(\exp_{k+1}(n))$.

BIBLIOGRAPHY

[1] Barendregt, H., Coppo, M., and Dezani-Ciancaglini, M. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic 48*, 4 (1983), 931–940.

[2] Freeman, T., and Pfenning, F. Refinement types for ML. In *ACM Conference on Programming Language Design and Implementation (PLDI)* (1991), ACM, pp. 268–277.

[3] Gladstone, M. D. Some ways of constructing a propositional calculus of any required degree of unsolvability. *Transactions of the American Mathematical Society 118* (1965), 195–210.

[4] Hindley, J. R. The simple semantics for Coppo-Dezani-Sallé types. In *International Symposium on Programming* (1982), M. Dezani-Ciancaglini and U. Montanari, Eds., vol. 137 of *LNCS*, Springer, pp. 212–226.

[5] Hindley, J. R., and Seldin, J. P. *Lambda-calculus and Combinators, an Introduction*. Cambridge University Press, 2008.

[6] Kuśmierek, D. The inhabitation problem for rank two intersection types. In *TLCA* (2007), S. Ronchi Della Rocca, Ed., vol. 4583 of *LNCS*, Springer, pp. 240–254.

[7] Linial, L., and Post, E. L. Recursive unsolvability of the deducibility, Tarski's completeness and independence of axioms problems of propositional calculus. *Bulletin of the American Mathematical Society 55* (1949), 50.

[8] Papadimitriou, C. H. *Computational Complexity*. Addison-Wesley, 1994.

[9] Rehof, J., and Urzyczyn, P. Finite combinatory logic with intersection types. In *TLCA* (2011), C.-H. L. Ong, Ed., vol. 6690 of *Lecture Notes in Computer Science*, Springer, pp. 169–183.

[10] Singletary, W. E. Recursive unsolvability of a complex of problems proposed by Post. *Journal of the Faculty of Science, University of Tokyo 14* (1967), 25–58.

[11] Singletary, W. E. Many-one degrees associated with partial propositional calculi. *Notre Dame Journal of Formal Logic XV*, 2 (1974), 335–343.

[12] Statman, R. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science 9* (1979), 67–72.

Forschungsberichte
der Fakultät für Informatik
der Technischen Universität Dortmund

ISSN 0933-6192

Anforderungen an:
Dekanat Informatik | TU Dortmund
D-44221 Dortmund