Syntactically and semantically regular languages of λ-terms coincide through logical relations

Vincent Moreau ✉ ⌂ ⓘD
IRIF & Université Paris Cité & Inria Paris, France

Lê Thành Dũng (Tito) Nguyễn ✉ ⌂ ⓘD
Laboratoire de l'informatique du parallélisme (LIP), École normale supérieure de Lyon, France

───── **Abstract** ─────

A fundamental theme in automata theory is regular languages of words and trees, and their many equivalent definitions. Salvati has proposed a generalization to regular languages of simply typed λ-terms, defined using denotational semantics in finite sets.

We provide here some evidence for its robustness. First, we give an equivalent syntactic characterization that naturally extends the seminal work of Hillebrand and Kanellakis connecting regular languages of words and syntactic λ-definability. Second, we exhibit a class of categorical models of the simply typed λ-calculus, which we call finitely pointable, and we show that, when used in Salvati's definition, they all recognize exactly the same class of languages of λ-terms as the category of finite sets does.

The proofs of these two results rely on logical relations and can be seen as instances of a more general construction of a categorical nature, inspired by previous categorical accounts of logical relations using the gluing construction.

1 Introduction

Much work has been devoted to the study of regular languages of words and trees – also called recognizable – and their equivalent characterizations, typically in terms of automata, algebra, and logic. The remarkable robustness of this notion of regularity has led to attempts to extend it to several other structures, such as infinite words/trees or graphs of bounded treewidth – many examples can be found, for instance, in [3].

This paper focuses on a less studied extension: recognizable languages of simply typed λ-terms, introduced by Salvati [28]. They are a conservative generalization [28, §3]:[1] using the Church encoding, finite words and trees can be represented in the simply typed λ-calculus, and recognizability for Church encodings coincides with the usual notion of regularity.

---

[1] Alternatively, see [9, Proposition 7.1] for the case of words.

Furthermore, as Salvati explains in his habilitation thesis [29], regular languages of λ-terms can be used to shed light on several classical topics concerning the simply typed λ-calculus, such as higher-order matching [28, §6.2] and higher-order grammars [18].

However, not many characterizations of the class of recognizable languages of simply typed λ-terms are known, and it would be desirable to have more evidence that it is robust. Moreover, if we know that this class of languages is a truly canonical object, then so is its Stone dual, namely the recently introduced space of profinite λ-terms [9]. Currently, there exist two definitions of recognizable languages of λ-terms, both provided in [28]:

- The first one [28, Definition 1] uses denotational semantics into the finite standard models of the simply typed λ-calculus. This may be understood as generalizing to higher orders the computational aspects of deterministic finite automata. In the same vein, the concrete construction of profinite λ-terms in [9] depends on specific properties of the category **FinSet** of finite sets and functions between them.
- The second one, grounded in intersection types [28, §4], turns out to admit an equivalent presentation in terms of a denotational semantics in finite domains [29, Theorem 25]. Indeed, the connection between intersection types and semantics is standard, see e.g. [27].

Both definitions can be seen as an instance of the following pattern: the interpretation of a simply typed λ-term in some denotational model with a *finitary* flavor should suffice to know whether the term belongs to our language of interest. This is closely analogous to the algebraic definition of regular word languages. Indeed, viewing the set Σ^* of words over a finite alphabet Σ as the free monoid generated by Σ, a language $L \subseteq \Sigma^*$ is regular if and only if, for some homomorphism $\varphi \colon \Sigma^* \to M$ to a finite monoid M, the "interpretation" $\varphi(w)$ determines[2] whether $w \in L$ for each $w \in \Sigma^*$. Asking for φ to be a homomorphism parallels the compositionality property of denotational semantics.

We may therefore ask:

- What kind of semantics yield the same notion of recognizable language? More precisely, with a definition of **C**-recognizable language for any cartesian closed category (CCC) **C**, i.e. any categorical model of the simply typed λ-calculus with products, the question becomes: when do recognizability by **C** and by **FinSet** coincide?
- Alternatively, is there any characterization of regular languages of λ-terms that does not involve denotational semantics?

For the latter, we propose a positive answer inspired by the following result of Hillebrand and Kanellakis [14, Theorem 3.4]: a language of words is regular if and only if it can be decided by a simply typed λ-term operating on Church-encoded words. By replacing the type of Church encodings with any simple type A, we get a natural notion of syntactically regular language of λ-terms, defined by means purely internal to the simply typed λ-calculus.

Contributions and proof methods.   The main results of this paper are as follows:

Theorem 3.2 any non-degenerate cartesian closed category can recognize at least every language which is syntactically regular.

Theorem 5.14 every language recognized by a CCC which verifies the property of being finitely pointable is recognized by **FinSet**.

Theorem 6.5 every language recognized by **FinSet** is syntactically regular.

These three theorems, taken together, show that all non-degenerate finitely pointable CCCs yield the same notion of regular language of λ-terms, which is the same as the syntactic one.

---

[2] More formally, $L = \varphi^{-1}(P)$ for some $P \subseteq M$.

To achieve this goal, we introduce a construction on CCCs, which we call squeezing, and apply it to logical relations. Its versatility allows us to apply it to prove both Theorem 5.14 and Theorem 6.5.

Related work. Morally, the study of regular languages of λ-terms amounts to understanding what information can be extracted by evaluating simply typed λ-terms in finitary models. A seminal result in this spirit is Statman's finite completeness theorem [31], which can be rephrased as the regularity of all singleton languages of λ-terms – a perspective that has led to a simplified proof [30]. The idea of using another CCC than **FinSet**, easier to use to show Statman's theorem, has been exploited in [18]. This shows the advantage to use an appropriate CCC to recognize a given language, a possibility which is extended to the vast clas of finitely pointable CCCs in this paper.

Finitary semantics are powerful tools, in particular, for understanding the computational power of the simply typed λ-calculus. For instance, in [14], Hillebrand and Kanellakis use the finite set semantics to prove their aforementioned theorem on regular word languages; as for finite Scott domains presented as intersection types, they have been applied by Terui [34] to study the complexity of normalizing simply typed λ-terms.

As can be seen *inter alia* from rather surprising results of Statman [32] and Plotkin [26], finitary models are also useful to tame the infinitary aspects of an extension of the simply typed λ-calculus with a fixed-point operator, called the λY-calculus. Furthermore, the well-studied higher-order model checking problem is about testing regular properties on infinite trees that can be Church-encoded in the λY-calculus; it sits at the interface between automata and programming languages, with applications to the formal verification of functional programs (see e.g. [17]). Decidability of higher-order model checking, first established by Ong through game semantics [24], now admits proofs based on intersection types [16, 25] and on finitary semantics [35, 10]. Drawing on this line of work, higher-order parity automata [20] generalize to λY-terms the recognizable languages of simply typed λ-terms.

The theme of syntactic recognizability *à la* Hillebrand and Kanellakis, for its part, has been recently revived in Nguyễn and Pradic's implicit[3] automata theory. They use substructural λ-calculi and Church encodings to characterize star-free languages [23] and classes of string-to-string functions computed by transducers [22].

Plan of the paper. We start by recalling in Section 2 the semantics of the simply typed λ-calculus, the notion of language recognized by a CCC as defined in [28] for finite sets, and by introducing the notion of syntactically regular language, generalizing recognition as defined in [14]. In Theorem 3.2 of Section 3, we show that every non-degenerate CCC recognizes all syntactically regular languages. In Section 4, we recall the definition of logical relations and introduce the squeezing construction **Sqz**$(-)$ which will be a crucial tool for the two next sections. In Section 5, we introduce finitely pointable CCCs and show in Theorem 5.14 that they do not recognize more languages than finite sets do. In Theorem 6.5 of Section 6, we show that languages recognized by finite sets are syntactically regular. We finish this paper by giving some consequences of the equivalence established by these three theorems in Section 7.

---

[3] The name is a nod to implicit computational complexity, a field concerned with alternative definitions of complexity classes that avoid low-level machine models and explicit resource bounds. As an example, in addition to their result on regular word languages in the simply typed λ-calculus, Hillebrand and Kanellakis's paper [14] also contains characterizations of the k-EXPTIME and k-EXPSPACE hierarchies based on λ-terms, that are again proved by evaluation in finite sets.

2   Languages of λ-terms

Syntax and semantics

We first specify the syntax we are working with. The grammars of types and preterms are

$$A, B ::= \mathrm{o} \mid A \Rightarrow B \mid A \times B \mid 1 \quad \text{and} \quad t, u ::= x \mid \lambda(x : A). t \mid t\, u \mid \langle t, u \rangle \mid t_i \text{ for } i = 1, 2$$

and we consider the usual typing rules and $\beta\eta$-conversion rules, see e.g. [2, §4.1]. We extend the notation of t_i, for the projection to the ith coordinate, to the case where t is of type $A_1 \times \cdots \times A_n$ and i is between 1 and n. As the λ-abstractions are annotated, a closed λ-term has at most one type derivation. For any simple type A, we write $\Lambda(A)$ for the set of closed simply typed λ-terms of type A, taken modulo $\beta\eta$-conversion.

   We recall the semantics of the simply typed λ-calculus into cartesian closed categories, abbreviated as CCC, see [2, Chapter 4] for more details. For any CCC $\mathbf{C}$, object c of $\mathbf{C}$ and simple type A, we define an object $[\![A]\!]_c$ of $\mathbf{C}$ by induction on A as follows:

$$[\![\mathrm{o}]\!]_c := c \qquad [\![A \Rightarrow B]\!]_c := [\![A]\!]_c \Rightarrow [\![B]\!]_c \qquad [\![A \times B]\!]_c := [\![A]\!]_c \times [\![B]\!]_c \qquad [\![1]\!]_c := 1$$

Using the CCC structure of $\mathbf{C}$, one can define a family of set-theoretic functions

$$[\![-]\!]_c \quad : \quad \Lambda(A) \longrightarrow \mathbf{C}(1, [\![A]\!]_c) \qquad \text{for every simple type } A$$

called semantic brackets, sending closed λ-terms to points of the objects $[\![A]\!]_c$.

   These assignments can be described in another way. Let $\mathbf{Lam}$ be the category whose objects are simple types and whose set of morphisms from A to B is $\Lambda(A \Rightarrow B)$, with the expected composition. This category is the free CCC on one object, i.e., for every CCC $\mathbf{C}$ and object c of $\mathbf{C}$, there exists a unique CCC functor $[\![-]\!]_c : \mathbf{Lam} \to \mathbf{C}$ such that $[\![\mathrm{o}]\!]_c = c$. This can be represented by the commutativity of the following diagram:

$$
\begin{array}{ccc}
\mathbf{Lam} & & \\
{\scriptstyle \mathrm{o}} \uparrow \quad \diagdown {\scriptstyle [\![-]\!]_c} & & \\
1 \xrightarrow[\quad c \quad]{} & \mathbf{C} &
\end{array}
\qquad\qquad (1)
$$

   In this paper, the CCCs come with specified terminal object, cartesian products, and exponentials, and CCC functors are required to respect these structures strictly, *on the nose*. In that way, the unicity in the universal property of $\mathbf{Lam}$ depicted in Equation (1) holds up to equality, and not merely isomorphism.

   We write $\mathbf{FinSet}$ for the cartesian closed category of finite sets. The semantics of the simply typed λ-calculus in this CCC corresponds to its naive set-theoretic interpretation. For ease of notation, we identify the finite set Q with the set of functions $\mathbf{FinSet}(1, Q)$.

Recognizable languages of λ-terms, semantically

The notion of regular language of simply typed λ-terms has been introduced in [28, Definition 1] as the instance of the following definition for $\mathbf{C} = \mathbf{FinSet}$. We now introduce the notion of $\mathbf{C}$-recognizable language of λ-terms, for any CCC $\mathbf{C}$.

▶ **Definition 2.1.** *Let $\mathbf{C}$ be a CCC and c be an object of $\mathbf{C}$. For every simple type A and subset $F \subseteq \mathbf{C}(1, [\![A]\!]_c)$, the language L_F of λ-terms of type A is defined as*

$$L_F := \{t \in \Lambda(A) \mid [\![t]\!]_c \in F\} .$$

We define the set $\text{Rec}_c(A)$ *of languages of* λ*-terms of type A recognized by c as*

$$\text{Rec}_c(A) \quad := \quad \{L_F : F \subseteq \mathbf{C}(1, [\![A]\!]_c)\} .$$

Finally, a language L of λ*-terms of type A is* **C***-recognizable if there exists an object c of* **C** *such that L belongs to the set* $\text{Rec}_c(A)$*.*

▶ **Example 2.2.** For any natural number n, we define the associated simple type

$$\text{Church}_n \quad := \quad (\mathbb{o} \Rightarrow \mathbb{o})^n \Rightarrow \mathbb{o} \Rightarrow \mathbb{o} .$$

There is a bijection between the sets $\Lambda(\text{Church}_n)$ and $\{1, \dots, n\}^*$, the set of finite words over an alphabet with n letters, called the *Church encoding*. For example, the word 12212 over the two letter-alphabet $\{1, 2\}$ is encoded as the λ-term

$$\lambda(a : (\mathbb{o} \Rightarrow \mathbb{o})^2). \lambda(e : \mathbb{o}). a_2 \ (a_1 \ (a_2 \ (a_2 \ (a_1 \ e)))) \quad \in \quad \Lambda(\text{Church}_2) .$$

Under this bijection, a language of λ-terms of type Church_n is **FinSet**-recognizable, in the sense of Definition 2.1, if and only if the language of words associated by the Church encoding is a regular language of finite words, see [9, Proposition 7.1].

▶ **Example 2.3.** We give a detailed example using the Church encoding. We show that the language L of λ-terms of type Church_2 which are encodings of words in $\{1, 2\}^*$ that contain an even number of 1s and an odd number of 2s, is **FinSet**-recognizable.

Let Q be the finite set $\{q_\top, q_\perp\}$ and F be the subset of $[\![\text{Church}_2]\!]_Q$ defined as

$$F \quad := \quad \{f \in (Q \Rightarrow Q) \times (Q \Rightarrow Q) \Rightarrow Q \Rightarrow Q \mid f(\text{not}, \text{Id}_Q)(q_\top) = f(\text{Id}_Q, \text{not})(q_\perp) = q_\top\}$$

where $\text{not} : Q \to Q$ is the function defined as $\text{not}(q_\top) = q_\perp$ and $\text{not}(q_\perp) = q_\top$. The language L is equal to L_F which belongs to $\text{Rec}_Q(\text{Church}_2)$, so L is **FinSet**-recognizable.

The idea is that, given the semantic interpretation $f \in [\![\text{Church}_2]\!]_Q$ of the encoding of a word $w \in \{1, 2\}^*$, the states $f(\text{not}, \text{Id}_Q)(q_\top)$ and $f(\text{Id}_Q, \text{not})(q_\perp)$ are the states reached respectively, after reading w, in the two following deterministic finite automata:
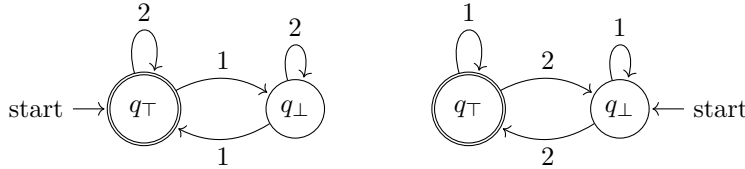
The language L is the intersection of the two languages recognized by these automata.

▶ **Example 2.4.** We consider the simple type

$$\text{UntypedTerms} \quad := \quad ((\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o}) \Rightarrow (\mathbb{o} \Rightarrow \mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o} .$$

There is a canonical bijection between $\Lambda(\text{UntypedTerms})$ and the set of closed untyped λ-terms modulo α-renaming, i.e. syntax trees with binders, without β-conversion. Here are examples of encodings of the latter into the former (for the general definition, see Appendix A):

$$\lambda x. x\, x \quad \rightsquigarrow \quad \lambda(\ell : (\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o}). \lambda(a : \mathbb{o} \Rightarrow \mathbb{o} \Rightarrow \mathbb{o}).$$
$$\ell \ (\lambda(x : \mathbb{o}). a\, x\, x)$$
$$(\lambda x. x\, x)\, (\lambda x. x\, x) \quad \rightsquigarrow \quad \lambda(\ell : (\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o}). \lambda(a : \mathbb{o} \Rightarrow \mathbb{o} \Rightarrow \mathbb{o}).$$
$$a \ (\ell \ (\lambda(x : \mathbb{o}). a\, x\, x)) \ (\ell \ (\lambda(x : \mathbb{o}). a\, x\, x))$$
$$\lambda f. (\lambda x. x\, x)\, (\lambda x. f\, (x\, x)) \quad \rightsquigarrow \quad \lambda(\ell : (\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o}). \lambda(a : \mathbb{o} \Rightarrow \mathbb{o} \Rightarrow \mathbb{o}).$$
$$\ell \ (\lambda(f : \mathbb{o}). a \ (\ell \ (\lambda(x : \mathbb{o}). a\, x\, x))$$
$$(\ell \ (\lambda(x : \mathbb{o}). a\, f\, (a\, x\, x))))$$

This can be seen as an extension of Church encodings to higher-order abstract syntax: indeed, the variable ℓ plays the role of a constructor and introduces a bound variable.

A closed untyped term is *affine* if and only if every bound variable occurs at most once. We now give the outline of the proof, detailed in Appendix A, that the encodings in $\Lambda(\mathsf{UntypedTerms})$ of closed untyped affine terms form a **FinSet**-recognizable language.

Let Q be the finite set $\{0, 1, \infty\} \times \{\top, \bot\}$, where $\{0, 1, \infty\}$ is seen as the additive monoid $\mathbb{N}$ truncated to $2 = 3 = \cdots = \infty$. We consider the two set-theoretic functions $f_{\mathsf{app}} : Q \to (Q \Rightarrow Q)$ and $f_{\mathsf{abs}} : (Q \Rightarrow Q) \to Q$ defined as

$$f_{\mathsf{app}}(k, b)(k', b') := (k + k', \; b \wedge b') \quad \text{and} \quad f_{\mathsf{abs}}(g) := (g_1(0, \top), \; g_2(0, \top) \wedge (g_1(1, \top) \leq 1))$$

where $g_1 : Q \to \{0, 1, \infty\}$ and $g_2 : Q \to \{\top, \bot\}$ are the compositions of $g : Q \to Q$ with the two projections. We verify that, for any closed untyped term t, we have

$$[\![t]\!]_Q \, (f_{\mathsf{abs}})(f_{\mathsf{app}}) \quad = \quad (0, b) \qquad \text{where } b \text{ is } \top \text{ if and only if } t \text{ is affine.}$$

Therefore, if F is the subset of $[\![\mathsf{UntypedTerms}]\!]_Q$ defined as

$$F := \{s \in ((Q \Rightarrow Q) \Rightarrow Q) \Rightarrow (Q \Rightarrow Q \Rightarrow Q) \Rightarrow Q \mid s(f_{\mathsf{abs}})(f_{\mathsf{app}}) = (0, \top)\}$$

then the **FinSet**-recognizable language L_F of terms of type $\mathsf{UntypedTerms}$ is the language of affine terms.

To the best of our knowledge, this regularity result is original. It also strongly suggests that the notion of **FinSet**-recognizable language, when applied to the type $\mathsf{UntypedTerms}$ of syntax trees with binders, differs from the recognizability of these syntax trees by the nominal tree automata of [13, §3.1]. Indeed, nominal automata cannot recognize the language of data words whose letters are all differents [4, Proof of Lemma 5.4].

▶ Remark 2.5. Let $\mathbf{C}$ and $\mathbf{D}$ be CCCs and $G : \mathbf{C} \to \mathbf{D}$ be a CCC functor. By the universal property of **Lam**, see Equation (1), for every object c of $\mathbf{C}$, the following diagram commutes:

In particular, this means that for every simple type A, the objects $[\![A]\!]_{G(c)}$ and $G([\![A]\!]_c)$ are equal, and that for every simply typed λ-terms t and t' of type A,

$$\text{if} \quad [\![t]\!]_c \; = \; [\![t']\!]_c \qquad \text{then} \quad [\![t]\!]_{G(c)} \; = \; [\![t']\!]_{G(c)} \; .$$

This means that interpreting the simply typed λ-calculus at the object c will always be at least as fine as interpreting it at $G(c)$. In particular, this entails that $\mathrm{Rec}_{G(c)}(A) \subseteq \mathrm{Rec}_c(A)$. If G is moreover faithful, we then have that $\mathrm{Rec}_{G(c)}(A) = \mathrm{Rec}_c(A)$ for any object c of $\mathbf{C}$. Therefore, all languages which are $\mathbf{C}$-recognizable are $\mathbf{D}$-recognizable.

Recognizable languages of λ-terms, syntactically

A syntactic approach to recognition is described in [14]. This syntactic approach uses the type substitution, also called cast, whose definition we now recall.

▶ **Definition 2.6.** *If A and B are simple types, we define a simple type $A[B] = A\{\mathbf{o} := B\}$ by replacing every occurence of $\mathbf{o}$ in A by B. We extend this to λ-terms by induction:*

$$x[B] \; := \; x \qquad (\lambda(x : A). \, t)[B] \; := \; \lambda(x : A[B]). \, t[B] \qquad (t \; u)[B] \; := \; t[B] \; u[B]$$

$$\langle t, u \rangle[B] \; := \; \langle t[B], u[B] \rangle \qquad\qquad t_i[B] \; := \; (t[B])_i$$

▷ **Claim 2.7.** For every simple types A and B and $t \in \Lambda(A)$, we have $t[B] \in \Lambda(A[B])$.

▶ **Remark 2.8.** A more categorical way to understand casting is to see it as the unique CCC functor $(-)[B] : \mathbf{Lam} \to \mathbf{Lam}$ such that the following diagram commutes:

$$
\begin{array}{ccc}
\mathbf{Lam} & & \\
\scriptstyle{\mathbb{o}} \uparrow & \diagdown \; {\scriptstyle (-)[B]} & \\
1 & \xrightarrow{\;\; B \;\;} & \mathbf{Lam}
\end{array}
$$

As such, it is the semantic bracket functor $[\![-]\!]_B : \mathbf{Lam} \to \mathbf{Lam}$.

Finally, we recall the encoding of Booleans into the simply typed λ-calculus.

▶ **Definition 2.9.** *Let* Bool *be the simple type* $\mathbb{o}^2 \Rightarrow \mathbb{o}$. *Its two inhabitants are the* λ-*terms*

$$\mathsf{true} := \lambda(x : \mathbb{o}^2).\, x_1 \qquad and \qquad \mathsf{false} := \lambda(x : \mathbb{o}^2).\, x_2 \;.$$

The following definition naturally generalizes the one given in [14] for $A = \mathsf{Church}_n$.

▶ **Definition 2.10.** *For any simple type* A, *a language* $L \subseteq \Lambda(A)$ *is* syntactically regular *if there exists a simple type* B *and a* λ-*term* r *of type* $A[B] \Rightarrow$ Bool *such that*

$$L \;=\; \{t \in \Lambda(A) \mid r \; t[B] =_{\beta\eta} \mathsf{true}\} \;.$$

▶ **Theorem 2.11** (Hillebrand & Kanellakis, [14, Theorem 3.4]). *A language of* λ-*terms of type* Church$_n$ *is* syntactically regular *if and only if the associated language of finite words by the* Church encoding *is regular in the usual sense.*

▶ **Example 2.12.** The Church encodings of words in $\{1, 2\}^*$ with an even number of 1s and an odd number of 2s is syntactically regular. Indeed, we consider the following λ-terms

$$
\begin{aligned}
\mathsf{and} &:= \lambda(p : \mathsf{Bool} \times \mathsf{Bool}).\, \lambda(x : \mathbb{o} \times \mathbb{o}).\, p_1 \; \langle p_2 \; x, x_2 \rangle \\
\mathsf{id} &:= \lambda(b : \mathsf{Bool}).\, b \\
\mathsf{not} &:= \lambda(b : \mathsf{Bool}).\, \lambda(x : \mathbb{o} \times \mathbb{o}).\, b \; \langle x_2, x_1 \rangle
\end{aligned}
$$

and choose, as in Definition 2.10, the type B to be Bool and the simply typed λ-term r to be

$$\lambda(w : \mathsf{Church}_2[\mathsf{Bool}]).\, \mathsf{and} \; \langle w \; \mathsf{not} \; \mathsf{id} \; \mathsf{true}, \; w \; \mathsf{id} \; \mathsf{not} \; \mathsf{false} \rangle \quad : \quad \mathsf{Church}_2[\mathsf{Bool}] \Rightarrow \mathsf{Bool} \;.$$

Just as in Example 2.3, this term can be seen as running two DFAs, both having two states, over the encoding of an input word.

When restricted to types Church$_n$ for any natural number n, the notion of **FinSet**-recognizable and syntactically regular languages coincide, as they both boil down to the usual notion of regular language of finite words, as seen in Example 2.2 and Theorem 2.11 respectively. One of the contributions of the present paper is to show that these two notions coincide at every simple type.

3   Syntactic recognition implies semantic recognition

In this section, we state Theorem 3.2 and prove it by extending the semantic evaluation argument, described by Hillebrand and Kanellakis in their proof of Theorem 2.11 for the case of languages of words, to the more general case of languages of any type.

▶ **Definition 3.1.** *A CCC* **C** *is said to be* non-degenerate *if there exist two objects c', c of* **C** *such that there exist two distinct morphisms $f, g \colon c' \to c$.*

▶ **Theorem 3.2.** *If* **C** *is a non-degenerate CCC, then any language of λ-terms which is* syntactically regular *is* **C**-recognizable.

Proof. We observe first that the non-degeneracy assumption means that the two projections $\pi_1, \pi_2 \colon c \times c \to c$ are not equal, since they yield different results when pre-composed with the morphism $c' \to c \times c$ obtained by pairing f and g.

Let A be a simple type and L be a language of λ-terms of type A which is syntactically regular. There exists a simple type B together with a λ-term $r \in \Lambda(A[B] \Rightarrow \mathsf{Bool})$ such that

$$L \quad = \quad \{t \in \Lambda(A) \mid r\ t[B] =_{\beta\eta} \mathsf{true}\} \ .$$

In order to show that L belongs to $\mathrm{Rec}_{[\![B]\!]_c}(A)$, we work with the interpretation of the simply typed λ-calculus at the object $[\![B]\!]_c$ of **C**. By the universal property of the CCC **Lam**, the following diagram commutes

More concretely, this states that, for every simply typed λ-term t, the two morphisms $[\![t[B]]\!]_c$ and $[\![t]\!]_{[\![B]\!]_c}$ are equal. By viewing r as a morphism from $A[B]$ to Bool in the category **Lam**, the compositionality of the semantic interpretation gives us that

$$[\![r\ t[B]]\!]_c \quad = \quad [\![r]\!]_c \circ [\![t[B]]\!]_c \quad = \quad [\![r]\!]_c \circ [\![t]\!]_{[\![B]\!]_c} \ .$$

By non-degeneracy, the semantic interpretation $[\![-]\!]_c \colon \Lambda(\mathsf{Bool}) \to \mathbf{C}(1, [\![\mathsf{Bool}]\!]_c)$ is injective: indeed, $[\![\mathsf{true}]\!]_c = \pi_1 \neq \pi_2 = [\![\mathsf{false}]\!]_c$. Therefore, we get the following equivalences

$$t \in L \quad \Longleftrightarrow \quad r\ t[B] =_{\beta\eta} \mathsf{true} \quad \Longleftrightarrow \quad [\![r\ t[B]]\!]_c = [\![\mathsf{true}]\!]_c \quad \Longleftrightarrow \quad [\![t]\!]_{[\![B]\!]_c} \in F$$

where F is the subset of $\mathbf{C}(1, [\![A]\!]_{[\![B]\!]_c})$ defined as $\{q \in \mathbf{C}(1, [\![A]\!]_{[\![B]\!]_c}) \mid [\![r]\!]_c \circ q = [\![\mathsf{true}]\!]_c\}$. This shows that L belongs to $\mathrm{Rec}_{[\![B]\!]_c}(A)$, hence that it is a **C**-recognizable language. ◀

4   Logical relations and the squeezing construction

Sconing in a nutshell

In this paragraph, we recall the construction of a CCC of logical relations from one CCC to another. We first recall the construction of logical predicates, also called sconing, which will be general enough to give logical relations as a special case. This method is well-known, see for instance [21] for an introductory account.

▶ **Definition 4.1.** *Let* **C** *be a CCC. The category of logical predicates over* **C**, *that we denote by* **P**(**C**), *is defined as follows:*
- *its objects are the pairs (c, S) of an object c of* **C** *together with a subset $S \subseteq \mathbf{C}(1, c)$,*
- *its morphisms from (c, S) to (c', S') are the morphisms $f : c \to c'$ of* **C** *such that $f \circ (-)$ restricts to a set-theoretic function $S \to S'$.*

▷ Claim 4.2. This category $\mathbf{P}(\mathbf{C})$ is a CCC, with exponentiation given by

$$(c, S) \Rightarrow (c', S') \quad = \quad (c \Rightarrow c', \{f \in \mathbf{C}(1, c \Rightarrow c') \mid \forall s \in S, \mathrm{ev}_{c,c'} \circ \langle f, s \rangle \in S'\})$$

The forgetful functor $(c, S) \mapsto c$ is a CCC functor.

Proof. See [21, p. 5], where the notation $\tilde{\mathbf{C}}$ is used for the category $\mathbf{P}(\mathbf{C})$. ◁

▶ **Definition 4.3.** *Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two CCCs. The* CCC of logical relations *from $\mathbf{C}_1$ to $\mathbf{C}_2$ is the CCC $\mathbf{P}(\mathbf{C}_1 \times \mathbf{C}_2)$, which admits a CCC functor $\mathbf{P}(\mathbf{C}_1 \times \mathbf{C}_2) \to \mathbf{C}_1 \times \mathbf{C}_2$.*

▶ Remark 4.4. We have defined the CCC of logical relations in terms of the logical predicate construction $\mathbf{P}(-)$ of Definition 4.1. More concretely, this construction gives a category that can be described in the following way:

- its objects are triples $(c_1, c_2, \Vdash)$ where c_i is an object of $\mathbf{C}_i$ for $i = 1, 2$ and $\Vdash$ is a subset of $\mathbf{C}_1(1, c_1) \times \mathbf{C}_2(1, c_2)$, and is thus a relation between the points of c_1 and of c_2,
- its morphisms from $(c_1, c_2, \Vdash)$ to $(c'_1, c'_2, \Vdash')$ are pairs $(f_1, f_2) \in \mathbf{C}_1(c_1, c'_1) \times \mathbf{C}_2(c_2, c'_2)$ such that for every pair $(x_1, x_2) \in \mathbf{C}_1(1, c_1) \times \mathbf{C}_2(1, c_2)$,

$$\text{if } x_1 \Vdash x_2 \ , \quad \text{then } f_1 \circ x_1 \Vdash' f_2 \circ x_2 \ .$$

For the proof that this category is a CCC, see [21, Proposition 4.3].

▶ Remark 4.5. The CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$ comes with two projections to $\mathbf{C}_1$ and to $\mathbf{C}_2$ which are CCC functors. By Remark 2.5, we get that for any relation $(c_1, c_2, \Vdash)$,

$$[\![A]\!]_{(c_1,c_2,\Vdash)} \quad = \quad ([\![A]\!]_{c_1}, [\![A]\!]_{c_2}, \Vdash^A) \quad \text{for some } \Vdash^A \subseteq \mathbf{C}_1(1, [\![A]\!]_{c_1}) \times \mathbf{C}_2(1, [\![A]\!]_{c_2}).$$

The interpretation of a λ-term $t \in \Lambda(A)$ at an object $(c_1, c_2, \Vdash)$ is a morphism of the form

$$([\![t]\!]_{c_1}, [\![t]\!]_{c_2}) \quad : \quad 1 \ \longrightarrow \ [\![A]\!]_{(c_1,c_2,\Vdash)} \qquad \text{which means that} \qquad ([\![t]\!]_{c_1}, [\![t]\!]_{c_2}) \in \Vdash^A \ .$$

This is the *fundamental lemma of logical relations*, see e.g. [2, Lemma 4.5.3].

▶ Remark 4.6. At this stage, the categories $\mathbf{C}_1$ and $\mathbf{C}_2$ play a symmetric role. However, this will not always be the case in the rest of the paper, and we therefore say CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$ to emphasize the order.

The squeezing construction

We describe here a construction, which we call squeezing, which produces a CCC $\mathbf{Sqz}(\mathbf{C})$ from a CCC $\mathbf{C}$ equipped with an additional structure that we call a squeezing structure. Intuitively, the objects of $\mathbf{Sqz}(\mathbf{C})$ are objects of $\mathbf{C}$ coming with bounds induced by the structure, inspired by the squeeze theorem of calculus. This construction can be seen as the proof-irrelevant counterpart to the twisted gluing construction described in [1, Definition 5]. The notion of squeezing structure that is used is related to the hypotheses of [7, Lemma 6]; this pattern also occurs in the older proof theory tradition, see for instance [8, §8.A].

Throughout this paragraph, we fix any CCC $\mathbf{C}$. We recall that a wide subcategory is a subcategory containing all objects, and can hence be seen as a predicate on morphisms, closed under finite compositions.

▶ **Definition 4.7.** *A* squeezing structure *on $\mathbf{C}$ is the data of*

- *two wide subcategories $\mathbf{C}_{\text{left}}$ and $\mathbf{C}_{\text{right}}$ of $\mathbf{C}$ with associated notations $\xrightarrow{\text{l}}$ and $\xrightarrow{\text{r}}$ for morphisms, which are stable under finite cartesian products and such that for all $u : c_l \xrightarrow{\text{l}} c'_l$ and $v : c_r \xrightarrow{\text{r}} c'_r$,*

$$v \Rightarrow u \ : \ c'_r \Rightarrow c_l \xrightarrow{\text{l}} c_r \Rightarrow c'_l \qquad and \qquad u \Rightarrow v \ : \ c'_l \Rightarrow c_r \xrightarrow{\text{r}} c_l \Rightarrow c'_r \ .$$

- *for every object c of $\mathbf{C}$, two objects L_c and R_c of $\mathbf{C}$ such that there exists morphisms:*

$$
\begin{array}{llll}
L_1 \xrightarrow{\text{l}} 1 & L_{c \times c'} \xrightarrow{\text{l}} L_c \times L_{c'} & L_{c \Rightarrow c'} \xrightarrow{\text{l}} R_c \Rightarrow L_{c'} & \\
1 \xrightarrow{\text{r}} R_1 & R_c \times R_{c'} \xrightarrow{\text{r}} R_{c \times c'} & L_c \Rightarrow R_{c'} \xrightarrow{\text{r}} R_{c \Rightarrow c'} \ .
\end{array}
\tag{2}
$$

▶ **Remark 4.8.** As we work in a proof-irrelevant setting, we are merely interested in the existence of these morphisms. Nonetheless, knowing that they belong to $\mathbf{C}_{\text{left}}$ or $\mathbf{C}_{\text{right}}$ gets us back some precious information, as we will see in Lemma 4.11 and Section 6.

▶ **Definition 4.9.** *Given a squeezing structure on $\mathbf{C}$, the category $\mathbf{Sqz}(\mathbf{C})$ is the full subcategory of $\mathbf{C}$ whose objects are the objects c of $\mathbf{C}$ for which there exist both a left morphism $L_c \xrightarrow{\text{l}} c$ and a right morphism $c \xrightarrow{\text{r}} R_c$.*

Notice that we write $\mathbf{Sqz}(\mathbf{C})$ even though this construction depends both on the CCC $\mathbf{C}$ and on a squeezing structure on $\mathbf{C}$.

▶ **Theorem 4.10.** *Given a squeezing structure on $\mathbf{C}$, the category $\mathbf{Sqz}(\mathbf{C})$ is a sub-CCC of $\mathbf{C}$.*

The proof is in Appendix B.

Partial surjections

Logical relations which are partial surjections, i.e. both functional and surjective relations, can be a useful tool to obtain partial equivalence relations and to prove semantic results, see [5, §3], [6, §1.4.2] and [9, Theorem A]. We now show that the squeezing construction can be applied to get partial surjections for free.

▶ **Lemma 4.11.** *Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two CCCs and $\mathbf{R}$ be the CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$, whose objects are triples containing relations. Suppose that we are given a squeezing structure on $\mathbf{R}$ such that*
- *the relations in the objects L_c are surjective,*
- *the relations in the objects R_c are functional,*
- *the morphisms (u_1, u_2) in $\mathbf{R}_{\text{left}}$ are such that $\mathbf{C}_2(1, u_2)$ is a surjective function,*
- *the morphisms (v_1, v_2) in $\mathbf{R}_{\text{right}}$ are such that $\mathbf{C}_2(1, v_2)$ is an injective function,*
where, for $u \in \mathbf{C}(a, b)$, the function $\mathbf{C}(1, u) : \mathbf{C}(1, a) \to \mathbf{C}(1, b)$ is the composition $u \circ (-)$.
Then, the relation of any object belonging to $\mathbf{Sqz}(\mathbf{R})$ is a partial surjection.

The proof is in Appendix C. We end this section by giving a definition which will appear in squeezing structures in Proposition 6.4 and in the proof of Proposition 5.12.

▶ **Definition 4.12.** *Let $\mathbf{R}$ be the CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$. We say that a morphism $(f_1, f_2) : (c_1, c_2, \Vdash) \to (c'_1, c'_2, \Vdash')$ is a target-identity if c_2 and c'_2 are the same object and if f_2 is the identity morphism.*

▶ **Remark 4.13.** We remark that the target-identities form a wide subcategory and are stable under products and exponentiation.

5   From finitely pointable CCCs to finite sets

Finitely pointable CCCs and well-pointed CCCs

In a CCC **C**, a *point* of an object c is a morphism $1 \to c$ from the terminal object to c. A CCC is said to be well-pointed if every morphism is determined by its action on points of its domain. A CCC is said to be locally finite if all its hom-sets are finite sets.

▶ **Definition 5.1.** *A CCC* **C** *is said to be* finitely pointable *if there exists a product-preserving and faithful functor* $P : \mathbf{C} \to \mathbf{FinSet}$.

▶ **Example 5.2.** For every finite category **J**, the category $\mathbf{C} = [\mathbf{J}^{\mathrm{op}}, \mathbf{FinSet}]$ of presheaves of finite sets on **J** is a locally finite CCC which is not well-pointed in general. However, it is finitely pointable, by considering the functor $P : [\mathbf{J}^{\mathrm{op}}, \mathbf{FinSet}] \to \mathbf{FinSet}$ defined as

$$P \quad : \quad F \quad \longmapsto \quad \prod_{j \in \mathbf{J}^{\mathrm{op}}} F(j) \ .$$

Every locally finite well-pointed CCC is finitely pointable. Moreover, from every finitely pointable CCC, it is possible to construct a locally finite well-pointed CCC. To do so, we start by recalling a construction used in [15, Definition 9] and [9, §5].

▶ **Definition 5.3.** *Let* **C** *be a CCC and* $P : \mathbf{C} \to \mathbf{Set}$ *be a product-preserving functor. We define the CCC* $\mathbf{C}[P]$ *as the category whose objects are the objects of* **C** *and whose hom-sets are defined as*

$$\mathbf{C}[P](c, c') \quad := \quad P(c \Rightarrow c') \qquad \text{where } c \Rightarrow c' \text{ is the exponential in } \mathbf{C}.$$

The composition in $\mathbf{C}[P]$ *comes from taking the image by* P *of the canonical morphism* $(c' \Rightarrow c'') \times (c \Rightarrow c') \to c \Rightarrow c''$ *given by the CCC structure. There is an identity-on-objects CCC functor from* **C** *to* $\mathbf{C}[P]$ *which is faithful if* P *is.*

▶ Remark 5.4. This construction can be seen as a change of base: indeed, every CCC is enriched over itself and the product-preserving functors amounts to changing the enrichment.

The following proposition gives an equivalent characterization of finitely pointable CCCs.

▶ **Proposition 5.5.** *A CCC* **C** *is* finitely pointable *if and only if it is a sub-CCC of a locally finite well-pointed CCC.*

Proof. If **C** is finitely pointable, then there exists a product-preserving and faithful functor $P : \mathbf{C} \to \mathbf{FinSet}$. Using Definition 5.3, **C** is a sub-CCC of $\mathbf{C}[P]$ which is locally finite. Moreover, for every object c of $\mathbf{C}[P]$, the point functor $\mathbf{C}[P](1, -) = P(1 \Rightarrow (-))$ is naturally isomorphic to P which is faithful. Therefore, the category $\mathbf{C}[P]$ is well-pointed.

Conversely, if **C** is a sub-CCC of a locally finite well-pointed CCC **E**, then the composition of the inclusion $\mathbf{C} \hookrightarrow \mathbf{E}$ and the point functor $\mathbf{E} \to \mathbf{FinSet}$ is a product-preserving and faithful functor from **C** to $\mathbf{FinSet}$, which proves that **C** is finitely pointable.   ◀

Regular languages in finitely pointable CCCs

We first focus on the case of recognizable languages of λ-terms in well-pointed CCCs. We start by introducing the following constructions, which will help us to use partial surjections.

▶ **Definition 5.6.** *Let* **C** *be a category with a terminal object. We define the following full subcategories of* **C***:*

- $\mathbf{C}_{\geq 1}$ *containing the objects c such that* $\mathbf{C}(1, c)$ *contains at least an element; we call these objects* inhabited,
- $\mathbf{C}_{\leq 1}$ *containing the objects c such that* $\mathbf{C}(1, c)$ *contains at most an element,*
- $\mathbf{C}_{=1}$ *containing the objects c such that* $\mathbf{C}(1, c)$ *is a singleton.*

When instantiated to the CCC **Lam** of simple types and λ-terms, the notion of inhabited object coincides with the usual notion of inhabited simple type.

▶ **Proposition 5.7.** *If* **C** *is a CCC, then the category* $\mathbf{C}_{\geq 1}$ *is a sub-CCC of* **C**.

The proof is in Appendix B.

▶ **Proposition 5.8.** *If* **E** *is a well-pointed CCC, then the category* $\mathbf{E}_{\leq 1}$ *is a sub-CCC of* **E**. *Moreover, the category* $\mathbf{E}_{=1}$ *is equivalent to the terminal category.*

The proof is in Appendix B. We now prove the interesting fact that inhabited objects characterize the recognized languages of λ-terms in a well-pointed CCC.

▶ **Proposition 5.9.** *If* **E** *is a well-pointed CCC, then a language of* λ-terms *is* **E**-*recognizable if and only if it is* $\mathbf{E}_{\geq 1}$-*recognizable.*

Proof. Let **E** be a well-pointed CCC. By Proposition 5.7, $\mathbf{E}_{\geq 1}$ is a sub-CCC of **E** so any language which is $\mathbf{E}_{\geq 1}$-recognizable is **E**-recognizable, as explained in Remark 2.5.

We now show that the only language recognized by $\mathbf{E}_{\leq 1}$ is the empty and full languages. Let A be a simple type and c be an object of $\mathbf{E}_{\leq 1}$. As **E** is well-pointed and by Proposition 5.8, we know that $[\![A]\!]_c$ belongs to $\mathbf{E}_{\leq 1}$, which means that $\mathbf{C}(1, [\![A]\!]_c)$ is empty or a singleton, so $\mathrm{Rec}_c(A)$ contains at most the empty and full languages, which are the same when A is not inhabited.

The empty and full languages are recognized by any CCC, so in particular by $\mathbf{E}_{\geq 1}$. Therefore, all **E**-recognizable languages are $\mathbf{E}_{\geq 1}$-recognizable. ◀

▶ **Example 5.10.** ▬ The category $\mathbf{FinSet}_{\geq 1}$ is the CCC of *non-empty* finite sets.
▬ An implicative semilattice is a meet-semilattice such that each meet operation has an upper adjoint. Implicative semilattices are CCCs, however, they are degenerate and so never distinguish different λ-terms of the same type.
  Another way to understand this fact is to remark that their full subcategory of inhabited objects is the terminal category.

We now introduce the basic partial relations at which we will interpret the λ-calculus.

▶ **Definition 5.11.** *Let* **E** *be a well-pointed locally finite CCC and* **R** *be the* CCC of logical relations *from* **FinSet** *to* **E**. *For any object e of* **E**, *we consider the triple*

$$T_e \quad := \quad (\mathbf{E}(1, e), e, \sim_e) \qquad \text{where } \sim_e \text{ is the identity relation of } \mathbf{E}(1, e).$$

which extends to a functor $T : \mathbf{E} \to \mathbf{R}$.

We now prove a converse to Lemma 4.11 in the present case.

▶ **Proposition 5.12.** *Let* **E** *be a well-pointed locally finite CCC whose objects are all* inhabited *and* **R** *be the* CCC of logical relations *from* **FinSet** *to* **E**. *Then, the full subcategory of partial surjections is a sub-CCC of* **R**.

We provide two proofs of this proposition. The first one constructs the CCC by hand, see Appendix B. The other one uses the squeezing technique, see Appendix D.

The following result was first stated by Salvati in [29, Lemma 20]. It appears there without proof, but with the claim that it can be easily established via logical relations.

▶ **Proposition 5.13.** *For every well-pointed locally finite CCC* **E**, *any language which is* **E**-*recognizable is* **FinSet**-*recognizable.*

Proof. Let A be a simple type and L be a language of λ-terms of type A which is **E**-recognizable. By Proposition 5.9, it is $\mathbf{E}_{\geq 1}$-recognizable. Let e be an object of $\mathbf{E}_{\geq 1}$ such that $L \in \mathrm{Rec}_e(A)$. We consider the object $T_e = (\mathbf{E}(1, e), e, \sim_e)$ from Definition 5.11, whose relation is a partial surjection. The object $[\![A]\!]_{T_e}$ is of the form $([\![A]\!]_{\mathbf{E}(1,e)}, [\![A]\!]_e, \sim_e^A)$, where the relation $\sim_e^A$ is a partial surjection, as explained in Remark 2.5.

Let F be a subset of $\mathbf{E}(1, [\![A]\!]_e)$ such that L is L_F. We consider the subset F' of $[\![A]\!]_{\mathbf{E}(1,e)}$ defined as the inverse image

$$F' \quad := \quad \{q \in [\![A]\!]_{\mathbf{E}(1,e)} \mid \exists q' \in F \text{ s.t. } q \sim_e^A q'\}$$

By the fundamental lemma of logical relations, for λ-term t of type A, we have

$$[\![t]\!]_{\mathbf{E}(1,e)} \sim_e^A [\![t]\!]_e \ .$$

which proves that $L_F \subseteq L_{F'}$. Moreover, as $\sim_e^A$ is a functional relation, we get the converse inclusion. This proves that L is **FinSet**-recognizable. ◀

▶ **Theorem 5.14.** *For every finitely pointable CCC* **C**, *any language which is* **C**-*recognizable is* **FinSet**-*recognizable.*

Proof. Let **C** be a finitely pointable CCC. By Proposition 5.5, there is a well-pointed locally finite CCC **E** and a faithful functor $G : \mathbf{C} \to \mathbf{E}$. As explained in Remark 2.5, this shows that all languages which are **C**-recognizable are also **E**-recognizable. By Proposition 5.13, we get that all languages which are **C**-recognizable are also **FinSet**-recognizable. ◀

6 From finite sets to λ-terms

In this section, we apply the squeezing construction of Definition 4.9 on a CCC of logical relations to show that every **FinSet**-recognizable language is syntactically regular, through an encoding of finite sets into the simply typed λ-calculus. To achieve that, we need to change slightly of setting, by moving from finite sets to finite ordinals. This will make it possible to define the functor $\mathrm{Fin}(-)$ without ambiguity.

Therefore, we consider the category **FinOrd** whose objects are natural numbers and whose morphisms are the set-theoretic maps between the associated finite cardinals $\langle n \rangle := \{1, \ldots, n\}$. The inclusion of **FinOrd** in **FinSet** is a fully faithful functor that is essentially surjective, henceforth we get an equivalence between **FinOrd** and **FinSet** using the axiom of choice. In particular, **FinOrd** is a CCC that recognizes the same languages as **FinSet**.

The encoding of finite sets and its squeezing structure

We take **R** to be the CCC of logical relations from **Lam** to **FinOrd**. We recall that the objects of **R** are therefore triples $R = (B, n, \Vdash)$ where B is a simple type, n is a natural number and $\Vdash$ is a subset of the product $\Lambda(B) \times \langle n \rangle$. A morphism of **R** is a pair of morphisms of **Lam** and **FinOrd** which respect the relations.

▶ **Definition 6.1.** *We define the functor* $\mathrm{Fin}(-) : \mathbf{FinOrd} \to \mathbf{Lam}$ *as* $\mathrm{Fin}(n) := \mathrm{o}^n \Rightarrow \mathrm{o}$ *and, for every* $f : n \to n'$, *the* λ-*term* $\mathrm{Fin}(f) : \mathrm{Fin}(n) \to \mathrm{Fin}(n')$ *is*

$$\lambda(p : \mathrm{Fin}(n)).\, \lambda(x : \mathrm{o}^{n'}).\, p\, \langle x_{f(1)}, \ldots, x_{f(n)} \rangle\ .$$

▶ Remark 6.2. As $\mathbf{FinOrd}$ is equivalent to the free cocartesian category and $\mathbf{Lam}^{\mathrm{op}}$ is cocartesian, we get a functor $n \mapsto \mathrm{o}^n$. The composition of the two functors

$$\mathbf{FinOrd} \xrightarrow{\ n \mapsto \mathrm{o}^n\ } \mathbf{Lam}^{\mathrm{op}} \xrightarrow{\ (-) \Rightarrow \mathrm{o}\ } \mathbf{Lam}$$

is precisely the functor $\mathrm{Fin}(-)$.

Our goal is now to exhibit a squeezing structure on $\mathbf{R}$ in order to show Theorem 6.5. We consider the target-identities for the two wide subcategories of the structure. We now define the following family of objects.

▶ **Definition 6.3.** *For any natural number* n, *we define the object* Bij_n *as*

$$\mathrm{Bij}_n \quad := \quad (\mathrm{Fin}(n), n, \Vdash_n)$$

where $\Vdash_n$ *is the bijection between the sets* $\Lambda(\mathrm{Fin}(n))$ *and* $\langle n \rangle$ *defined as*

$$\Vdash_n \quad := \quad \{(\pi_i, i) : 1 \le i \le n\} \qquad \text{with } \pi_i \text{ the } \lambda\text{-term } \lambda(x : \mathrm{o}^n).\, x_i\ .$$

This assignment extends to a functor $\mathrm{Bij}_{(-)} : \mathbf{FinOrd} \to \mathbf{R}$.

▶ **Proposition 6.4.** *There is a squeezing structure on the CCC* $\mathbf{R}$ *such that:*
- *the left and right morphisms are the target-identities of* $\mathbf{R}$,
- *for any object* $c = (B, n, \Vdash)$ *of* $\mathbf{R}$, *the objects* L_c *and* R_c *are both equal to* Bij_n.

The proof is in Appendix D. Proposition 6.4 shows that we have a sub-CCC $\mathbf{Sqz}(\mathbf{C})$ of the CCC of logical relations from $\mathbf{Lam}$ to $\mathbf{FinOrd}$, whose objects are tuples $(B, n, \Vdash)$ such that there exists λ-terms $u : \mathrm{Fin}(n) \to B$ and $v : B \to \mathrm{Fin}(n)$ lifting to the two following target-identities:

$$(\mathrm{Fin}(n), n, \Vdash_n) \xrightarrow{\ (u, \mathrm{Id}_n)\ } (B, n, \Vdash) \quad \text{and} \quad (B, n, \Vdash) \xrightarrow{\ (v, \mathrm{Id}_n)\ } (\mathrm{Fin}(n), n, \Vdash_n)\ .$$

Encoding recognizability by finite sets

We have shown in Proposition 6.4 that we have a squeezing structure on the CCC of logical relations $\mathbf{R}$ from $\mathbf{Lam}$ to $\mathbf{FinOrd}$. We now show how to use this structure, culminating in the link established in Theorem 6.5 between $\mathbf{FinOrd}$-recognizable and syntactically regular languages.

▶ **Theorem 6.5.** *If a language is* $\mathbf{FinSet}$-*recognizable, then it is syntactically regular.*

Proof. Let A be a simple type and $L \subseteq \Lambda(A)$ be any $\mathbf{FinSet}$-recognizable language. There exists a finite set Q and a subset $F \subseteq \langle [\![A]\!]_n \rangle$ such that

$$L \;=\; \{t \in \Lambda(A) \mid [\![t]\!]_n \in F\}\ .$$

We take n to be the cardinality of Q and note $\chi : [\![A]\!]_n \to 2$ the characteristic function associated to the subset F. By applying the functor $\mathrm{Bij}_{(-)}$, we get a morphism of relations

$$\mathrm{Bij}_\chi := (\mathrm{Fin}(\chi),\, \chi) \quad : \quad \left(\mathrm{Fin}([\![A]\!]_n),\, [\![A]\!]_n,\, \Vdash_{[\![A]\!]_n} \right) \;\longrightarrow\; (\mathrm{Fin}(2),\, 2,\, \Vdash_2)\ .$$

The interpretation $[\![A]\!]_{\mathrm{Bij}_n}$ is of the form $(A[\mathrm{Fin}(n)], [\![A]\!]_n, \Vdash_n^A)$ as explained in Remark 2.5. As it is an object of $\mathbf{Sqz}(\mathbf{R})$, it has a target-identity into $\mathrm{Bij}_{[\![A]\!]_n}$. By composing this morphism with Bij_χ, we obtain a morphism

$$(r, \chi) \quad : \quad (A[\mathrm{Fin}(n)], [\![A]\!]_n, \Vdash_n^A) \longrightarrow (\mathrm{Fin}(2), 2, \Vdash_2) \ .$$

By the fundamental lemma of logical relations, we get that, for every λ-term t of type A,

$$t[\mathrm{Fin}(n)] \Vdash_n^A [\![t]\!]_n \qquad \text{on which we apply } (r, \chi) \text{ to get} \qquad r\, t[\mathrm{Fin}(n)] \Vdash_2 \chi([\![t]\!]_n)$$

which states that $r\, t[\mathrm{Fin}(n)] =_{\beta\eta} \mathsf{true}$ if and only if $\chi([\![t]\!]_n)$ is 1. This proves that r recognizes the language L_F given by $F \subseteq \langle [\![A]\!]_n \rangle$, and so that L is syntactically regular. ◀

7 Regular languages

In this section, we want to point out a few consequences of the equivalence previously proved through Theorem 3.2, Theorem 5.14 and Theorem 6.5. Using these theorems, the following definition of regular languages is well-defined.

▶ **Definition 7.1.** *Let A be a simple type. A* regular language *of λ-terms of type A is a subset $L \subseteq \Lambda(A)$ such that one of the following equivalent propositions holds:*
- *L is syntactically regular;*
- *L is $\mathbf{C}$-recognizable, for some finitely pointable CCC $\mathbf{C}$;*
- *L is $\mathbf{FinSet}$-recognizable.*

We note $\mathrm{Reg}(A)$ the set of regular languages of λ-terms of type A.

▶ Remark 7.2. Note that $\mathbf{FinSet}$ recognizes all the regular languages of λ-terms. In that sense, it plays the same role as the monoid

$$M \ := \ \{f : \mathbb{N} \to \mathbb{N} \mid \exists N \in \mathbb{N}, \forall n \geq N, f(n) = n\} \quad \text{with the composition of functions}$$

which recognizes all the regular languages of finite words as all finite monoids can be embedded into M. Such a monoid cannot be finite; however, M is a *locally finite monoid*, i.e. all its finitely generated submonoids are finite (this is a standard notion, see e.g. [11, §V.5]).

In the case of finite words, recognizability by finite monoids and locally finite monoids are equivalent when the alphabet is finite. In the case of λ-terms however, finite CCCs are all degenerate whereas the finitely pointable case yields regular languages of λ-terms.

▶ **Proposition 7.3.** *The set $\mathrm{Reg}(A)$ of regular languages of λ-terms of some simple type A is a Boolean algebra.*

This fact boils down to stability by union or intersection. It is proved in [28, Theorem 8] using intersection types and in [9, Proposition 2.5] using logical relations. We provide another proof, showing that it is a direct consequence of our results and the structure of finitely pointable CCCs.

Proof. Using any of the three conditions of Definition 7.1, it is clear that regular languages are stable under complement.

Let $\mathbf{C}$ and $\mathbf{C}'$ be two finitely pointable CCCs, c be an object of $\mathbf{C}$ and c' be an object of $\mathbf{C}'$. The product CCC $\mathbf{C} \times \mathbf{C}'$ is still finitely pointable and comes with the associated projections $\mathbf{C} \times \mathbf{C}' \to \mathbf{C}$ and $\mathbf{C} \times \mathbf{C}' \to \mathbf{C}'$ which are CCC functors. We consider the object (c, c') of $\mathbf{C} \times \mathbf{C}'$. As explained in Remark 2.5, for any simple type A, we get that $\mathrm{Rec}_c(A) \subseteq \mathrm{Rec}_{(c,c')}(A)$ and $\mathrm{Rec}_{c'}(A) \subseteq \mathrm{Rec}_{(c,c')}(A)$.

This shows that the intersection of two languages in $\mathrm{Rec}_c(A)$ and $\mathrm{Rec}_{c'}(A)$ is still a regular language. Therefore, $\mathrm{Reg}(A)$ is a Boolean algebra. ◀

We now point out two other consequences of the equivalence in Definition 7.1.

- A stronger version of Statman's finite completeness theorem can be understood as the fact that singleton languages, taken modulo $\beta\eta$-conversion, are regular languages. It has multiple proofs, see [33] for proof directly in the finite standard model, [28] using intersection types, [18] in the model of complete lattices and [30] using Böhm trees.
  These results are usually proved in one CCC. Using Theorem 3.2, Theorem 5.14 and Theorem 6.5, we get that the singleton languages are recognized by any non-degenerate finitely pointable CCC and are also syntactically regular.
- Some finitely pointable CCCs are the coKleisli categories of a model of linear logic, see [19]. In [20], a notion of higher-order automaton is presented, which recognizes a language of λ-terms of a given simple type. The run-trees for these non-deterministic automata are defined using an intersection type system, which is an equivalent way of presenting the semantic interpretation $[\![-]\!]$ of the simply typed λ-calculus in the coKleisli category **ScottL**$_!$ of the Scott model of linear logic. Using the equivalence proved in the present article, a language is recognize by a higher-order automaton, i.e. **ScottL**$_!$-recognizable, if and only if it satisfies one of the conditions of Definition 7.1.

8 Conclusion and future perspectives

In this article, we have introduced the notion of finitely pointable CCC and we have shown that all the non-degenerate ones recognize the same languages of λ-terms, which are also syntactically regular. This is evidence for the robustness of the notion of regular language of λ-terms as introduced by Salvati in [28], and the dual notion of profinite λ-term introduced in [9] which is therefore shown to be independent of **FinSet**.

In the case of finite words and trees, regularity is closely related to recognition by finitary structures, and this paper shows that all non-degenerate and finitely pointable CCCs recognize exactly regular languages of λ-terms. As all finitely pointable CCCs are locally finite, one question that remains open is the following: is there a locally finite CCC that recognizes languages of λ-terms that are not regular, i.e. not recognizable by **FinSet**?

As explained in Example 2.2, the regular languages of λ-terms of type Church$_n$ for some natural number n are exactly the usual regular languages of the finite words associated by the Church encoding. It is possible to encode words in other calculi, like the non-commutative affine λ-calculus in which case a syntactic approach analogous to Definition 2.10 yields the star-free languages, see [23]. Moreover, gluing techniques have been studied for other calculi, see [12] for the linear case. One can therefore wonder whether it is possible to develop a semantic approach *à la* Salvati, analogous to Definition 2.1, for other calculi.

──── **References** ────

1   Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Categorical reconstruction of a reduction free normalization proof. In David H. Pitt, David E. Rydeheard, and Peter T. Johnstone, editors, *Category Theory and Computer Science, 6th International Conference, CTCS '95, Cambridge, UK, August 7-11, 1995, Proceedings*, volume 953 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 1995. `doi:10.1007/3-540-60164-3_27`.

2   Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998. `doi:10.1017/CBO9780511983504`.

3   Mikołaj Bojańczyk. Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic. *CoRR*, abs/2008.11635, 2020. `arXiv:2008.11635`.

4   Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014. `doi:10.2168/LMCS-10(3:4)2014`.

5   Antonio Bucciarelli. Logical relations and lambda theories. In *Advances in Theory and Formal Methods of Computing, proceedings of the 3rd Imperial College Workshop*, pages 37–48, 1996.

6   Thomas Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012. `doi:10.1016/j.tcs.2011.11.027`.

7   Marcelo Fiore. Semantic analysis of normalisation by evaluation for typed lambda calculus. *Mathematical Structures in Computer Science*, 32(8):1028–1065, 2022. `doi:10.1017/S0960129522000263`.

8   Jean-Yves Girard. *The Blind Spot: Lectures on logic.* European Mathematical Society, September 2011. `doi:10.4171/088`.

9   Sam van Gool, Paul-André Melliès, and Vincent Moreau. Profinite lambda-terms and parametricity, 2023. To appear in the proceedings of MFPS 2023. `arXiv:2301.12475`.

10 Charles Grellois. *Semantics of linear logic and higher-order model-checking.* PhD thesis, Université Paris 7, April 2016. URL: `https://hal.science/tel-01311150`.

11 Pierre A. Grillet. *Semigroups. An introduction to the structure theory.* Chapman & Hall/CRC Pure and Applied Mathematics. Dekker, 1995. `doi:10.4324/9780203739938`.

12 Masahito Hasegawa. Logical predicates for intuitionistic linear type theories. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, volume 1581 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 1999. `doi:10.1007/3-540-48959-2_15`.

13 Gerco van Heerdt, Tobias Kappé, Jurriaan Rot, Matteo Sammartino, and Alexandra Silva. Tree Automata as Algebras: Minimisation and Determinisation. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*, volume 139 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.CALCO.2019.6`.

14 Gerd G. Hillebrand and Paris C. Kanellakis. On the expressive power of simply typed and let-polymorphic lambda calculi. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 253–263. IEEE Computer Society, 1996. `doi:10.1109/LICS.1996.561337`.

15 Clément Jacq and Paul-André Melliès. Categorical combinatorics for non deterministic strategies on simple games. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 39–70. Springer, 2018. `doi:10.1007/978-3-319-89366-2_3`.

16 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3):20:1–20:62, 2013. `doi:10.1145/2487241.2487246`.

17 Naoki Kobayashi. 10 years of the higher-order model checking project (extended abstract). In Ekaterina Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 2:1–2:2. ACM, 2019. `doi:10.1145/3354166.3354167`.

18 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Information and Computation*, 243:205–221, 2015. `doi:10.1016/j.ic.2014.12.015`.

19 Paul-André Melliès. Categorical Semantics of Linear Logic. In P.-L. Curien, H. Herbelin, J.-L. Krivine, and P.-A. Melliès, editors, *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et Synthèses*. Société Mathématique de France, 2009. URL: `https://smf.emath.fr/publications/semantique-categorielle-de-la-logique-lineaire`.

20   Paul-André Melliès. Higher-order parity automata. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, Reykjavik, Iceland, June 2017. IEEE. `doi:10.1109/LICS.2017.8005077`.

21   John C. Mitchell and Andre Scedrov. Notes on sconing and relators. In Egon Börger, Gerhard Jäger, Hans Kleine Büning, Simone Martini, and Michael M. Richter, editors, *Computer Science Logic, 6th Workshop, CSL '92, San Miniato, Italy, September 28 - October 2, 1992, Selected Papers*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378. Springer, 1992. `doi:10.1007/3-540-56992-8_21`.

22   Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory*. PhD thesis, Université Paris XIII, 2021. URL: `https://hal.science/tel-04132636`.

23   Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.135`.

24   C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90. IEEE Computer Society, 2006. `doi:10.1109/LICS.2006.38`.

25   C.-H. Luke Ong. Higher-order model checking: An overview. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 1–15. IEEE Computer Society, 2015. `doi:10.1109/LICS.2015.9`.

26   Gordon D. Plotkin. Recursion does not always help. In Fairouz Kamareddine, editor, *A Century since Principia's Substitution Bedazzled Haskell Curry. In Honour of Jonathan Seldin's 80th Anniversary*. College Publications, 2023. `arXiv:2206.08413`.

27   Simona Ronchi Della Rocca. Intersection Types and Denotational Semantics: An Extended Abstract (Invited Paper). In Silvia Ghilezan, Herman Geuvers, and Jelena Ivetić, editors, *22nd International Conference on Types for Proofs and Programs (TYPES 2016)*, volume 97 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:7, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.TYPES.2016.2`.

28   Sylvain Salvati. Recognizability in the simply typed lambda-calculus. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2009. `doi:10.1007/978-3-642-02261-6_5`.

29   Sylvain Salvati. *Lambda-calculus and formal language theory*. Habilitation à diriger des recherches, Université de Bordeaux, 2015. URL: `https://hal.science/tel-01253426`.

30   B. Srivathsan and Igor Walukiewicz. An alternate proof of Statman's finite completeness theorem. *Information Processing Letters*, 112(14-15):612–616, 2012. `doi:10.1016/j.ipl.2012.04.014`.

31   Richard Statman. Completeness, invariance and lambda-definability. *Journal of Symbolic Logic*, 47(1):17–26, 1982. `doi:10.2307/2273377`.

32   Richard Statman. On the λY calculus. *Annals of Pure and Applied Logic*, 130(1-3):325–337, 2004. `doi:10.1016/j.apal.2004.04.004`.

33   Richard Statman and Gilles Dowek. On Statman's finite completeness theorem, 1992. `arXiv:2309.03602`.

34   Kazushige Terui. Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, pages 323–338, 2012. `doi:10.4230/LIPIcs.RTA.2012.323`.

35   Igor Walukiewicz. Automata theory and higher-order model-checking. *ACM SIGLOG News*, 3(4):13–31, 2016. `doi:10.1145/3026744.3026745`.

A The regular language of affine untyped terms

The goal of this appendix is to provide a detailed explaination of Example 2.4. We first introduce a grammar for the simply typed λ-terms of type UntypedTerms through the following notion of scoped term.

▶ **Definition A.1.** *We consider untyped terms with de Bruijn indices, given by the grammar*

$$u, v \quad ::= \quad \mathtt{var}_i \text{ for } i \in \mathbb{N}^* \mid \mathtt{abs}(u) \mid \mathtt{app}(u, v)$$

where we write $\mathtt{abs}(-)$ *for the abstraction to distinguish it from the simply typed λ-abstraction.*

For any natural number n and untyped term u, we define the judgment $n \vdash u$ by induction with the rules

$$\frac{}{n \vdash \mathtt{var}_i} \ 1 \le i \le n \qquad \frac{n+1 \vdash u}{n \vdash \mathtt{abs}(u)} \qquad \frac{n \vdash u \quad n \vdash v}{n \vdash \mathtt{app}(u, v)} \ .$$

The judgment $n \vdash u$ has at most one derivation. We call a scoped term any pair of n and u such that $n \vdash u$ is derivable.

In the rest of the appendix, we will simply say that $n \vdash u$ is a scoped term whenever this judgment is derivable. We now give the encoding of scoped terms into the simply typed λ-calculus.

▶ **Definition A.2.** *Let us consider a fixed sequence of simply typed variables $x_k : \mathtt{o}$ for $k \in \mathbb{N}$. We define the context Γ_n as $\ell : (\mathtt{o} \Rightarrow \mathtt{o}) \Rightarrow \mathtt{o}, a : \mathtt{o} \Rightarrow \mathtt{o} \Rightarrow \mathtt{o}, x_1 : \mathtt{o}, \ldots, x_n : \mathtt{o}.$*

For any natural number n and untyped term u, we consider the encoding $\overline{n \vdash u}$ of a scoped term defined by induction as

$$
\begin{aligned}
\overline{n \vdash \mathtt{var}_i} \quad &:= \quad x_{n+1-i} \\
\overline{n \vdash \mathtt{abs}(u)} \quad &:= \quad \ell \ (\lambda(x_{n+1} : \mathtt{o}). \ \overline{n+1 \vdash u}) \\
\overline{n \vdash \mathtt{app}(u, v)} \quad &:= \quad a \ (\overline{n \vdash u}) \ (\overline{n \vdash v})
\end{aligned}
$$

which is such that $\Gamma_n \vdash \overline{n \vdash u} : \mathtt{o}.$

Using normalization for the simply typed λ-calculus, we can claim the following fact.

▷ Claim A.3. For every natural number n, the encoding of scoped terms $n \vdash u$ into the open λ-terms $\overline{n \vdash u}$ of type $\mathtt{o}$ in context Γ_n, taken modulo $\beta\eta$-conversion, is bijective.

In particular, the encoding induces a bijection between the set $\Lambda(\mathsf{UntypedTerms})$ and the set of closed untyped terms, i.e. the u such that $0 \vdash u$ is a scoped term.

▶ **Definition A.4.** *For any scoped term $n \vdash u$ and $1 \le i \le n$, we define the natural number $\mathtt{occ}_i(n \vdash u)$, the number of occurences of the ith variable, by induction as:*

$$
\begin{aligned}
\mathtt{occ}_i(n \vdash \mathtt{var}_j) \quad &:= \quad 1 \text{ if } i = j \text{ otherwise } 0 \\
\mathtt{occ}_i(n \vdash \mathtt{abs}(u)) \quad &:= \quad \mathtt{occ}_{i+1}(n+1 \vdash u) \\
\mathtt{occ}_i(n \vdash \mathtt{app}(u, v)) \quad &:= \quad \mathtt{occ}_i(n \vdash u) + \mathtt{occ}_i(n \vdash v) \ .
\end{aligned}
$$

When the scoped term $n \vdash u$ is clear from context, we will simply write $\mathtt{occ}_i$.

▶ **Definition A.5.** *For any finite set Q and functions $f_{\mathtt{abs}} : (Q \Rightarrow Q) \to Q$ and $f_{\mathtt{app}} : Q \to Q \Rightarrow Q$, we interpret any scoped term $n \vdash u$ as its semantics*

$$n \vdash u \qquad \rightsquigarrow \qquad (\!|n \vdash u|\!) \quad : \quad Q^n \longrightarrow Q$$

which is the set-theoretic function defined, for $q_1, \ldots, q_n \in Q$, by induction as:

$$
\begin{aligned}
(\!|n \vdash \mathtt{var}_i|\!)[q_1, \ldots, q_n] &\;:=\; q_i \\
(\!|n \vdash \mathtt{abs}(u)|\!)[q_1, \ldots, q_n] &\;:=\; f_{\mathsf{abs}}(q \mapsto (\!|n+1 \vdash u|\!)[q, q_1, \ldots, q_n]) \\
(\!|n \vdash \mathtt{app}(u,v)|\!)[q_1, \ldots, q_n] &\;:=\; f_{\mathsf{app}}((\!|n \vdash u|\!)[q_1, \ldots, q_n])((\!|n \vdash v|\!)[q_1, \ldots, q_n]) \;.
\end{aligned}
$$

We write the arguments of the function $(\!|n \vdash u|\!)$ between square brackets $[$ and $]$.

▶ Remark A.6. The semantics of Definition A.5 factor through the encoding of Definition A.2 in the simply typed λ-calculus and its semantic interpretation as for all $q_1, \ldots, q_n \in Q$,

$$
(\!|n \vdash u|\!)(q_1, \ldots, q_n) \;=\; [\![\,\overline{n \vdash u}\,]\!]_Q\,(f_{\mathsf{abs}})(f_{\mathsf{app}})(q_n, \ldots, q_1) \;.
$$

We now instantiate Definition A.5 with the following values of Q, f_{abs} and f_{app}:

- Q is the set $\{0, 1, \infty\} \times \{\bot, \top\}$, with its product monoid structure. For any $q \in Q$, we write $q_1 \in \{0, 1, \infty\}$ and $q_2 \in \{\bot, \top\}$ for its two components.
- f_{abs} is the function $(Q \Rightarrow Q) \to Q$ defined as

$$
g \;\longmapsto\; (g(0, \top)_1 \,,\; g(0, \top)_2 \wedge (g(1, \top)_1 \neq \infty))
$$

- f_{app} is the curried monoid product of Q, i.e. the function $Q \to (Q \Rightarrow Q)$ defined as

$$
(n, b) \;\longmapsto\; (n', b') \mapsto (n + n', b \wedge b') \;.
$$

▶ **Proposition A.7** (Left part of the tuple). *For any scoped term $n \vdash u$ and any elements $k_1, \ldots k_n$ of $\{0, 1, \infty\}$, we have*

$$
(\!|n \vdash u|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \;=\; \mathsf{occ}_1 \cdot k_1 + \cdots + \mathsf{occ}_n \cdot k_n
$$

where the product $\cdot : \mathbb{N} \times \{0, 1, \infty\} \to \{0, 1, \infty\}$ comes from the monoid structure of $\{0, 1, \infty\}$.

Proof. We verify this by induction on the scoped term $n \vdash u$.

$$
\begin{aligned}
&(\!|n \vdash \mathtt{var}_i|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \\
&:=\; k_i \\
&=\; \mathsf{occ}_1 \cdot k_1 + \cdots + \mathsf{occ}_n \cdot k_n
\end{aligned}
$$

$$
\begin{aligned}
&(\!|n \vdash \mathtt{abs}(u)|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \\
&:=\; f_{\mathsf{abs}}((k, b) \mapsto (\!|n+1 \vdash u|\!)[(k, b), (k_1, \top), \, \ldots, \, (k_n, \top)])_1 \\
&=\; (\!|n+1 \vdash u|\!)[(0, \top), (k_1, \top), \, \ldots, \, (k_n, \top)]_1 \\
&=\; \mathsf{occ}_1 \cdot k_1 + \cdots + \mathsf{occ}_n \cdot k_n
\end{aligned}
$$

$$
\begin{aligned}
&(\!|n \vdash \mathtt{app}(u,v)|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \\
&:=\; f_{\mathsf{app}}((\!|n \vdash u|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)])((\!|n \vdash v|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)])_1 \\
&=\; (\!|n \vdash u|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \;+\; (\!|n \vdash v|\!)[(k_1, \top), \, \ldots, \, (k_n, \top)]_1 \\
&=\; \mathsf{occ}_1 \cdot k_1 + \cdots + \mathsf{occ}_n \cdot k_n
\end{aligned}
$$

In the $\mathtt{abs}$ case, the last equality is obtained by remarking that $\mathsf{occ}_1(n+1 \vdash u)$ is multiplied by 0 and that $\mathsf{occ}_{i+1}(n+1 \vdash u) = \mathsf{occ}_i(n \vdash \mathtt{abs}(u))$ for $i \geq 1$. ◀

We introduce the following definition.

▶ **Definition A.8.** *We define the property of a scoped term to be* affine in its bound variables *by induction as follows:*

- $n \vdash \mathtt{var}_i$ *is always* affine in its bound variables,
- $n \vdash \mathtt{abs}(u)$ *is* affine in its bound variables *if and only if*

$$\mathtt{occ}_1(n+1 \vdash u) \ \leq \ 1 \quad and \quad n+1 \vdash u \ is \ affine \ in \ its \ bound \ variables$$

- $n \vdash \mathtt{app}(u,v)$ *is* affine in its bound variables *if and only if*

$$n \vdash u \ and \ n \vdash v \ are \ both \ affine \ in \ their \ bound \ variables.$$

A λ-term $t \in \Lambda(\mathsf{UntypedTerms})$ will said to be affine *when the closed untyped term $0 \vdash u$ bijectively associated to t by Claim A.3 is* affine in its bound variables.

▶ **Proposition A.9** (Right part of the tuple)**.** *For any scoped term $n \vdash u$, we have*

$$(\!| n \vdash u |\!)[(0, \top), \dots, (0, \top)]_2 \quad = \quad b$$

where b is $\top$ if and only if the scoped term $n \vdash u$ is affine in its bound variables.

Proof. We prove this by induction on the scoped term $n \vdash u$.

- For any $1 \leq i \leq n$, $n \vdash \mathtt{var}_i$ is always affine in its bound variables, and we always have

$$(\!| n \vdash \mathtt{var}_i |\!)[(0, \top), \dots, (0, \top)]_2 \ = \ \top .$$

- For any scoped term $n+1 \vdash u$, we have

$$
\begin{aligned}
&(\!| n \vdash \mathtt{abs}(u) |\!)[(0, \top), \dots, (0, \top)]_2 \\
&:= \ f_{\mathsf{abs}}((k, b) \mapsto (\!| n+1 \vdash u |\!)[(k, b), (0, \top), \dots, (0, \top)])_2 \\
&= \ (\!| n+1 \vdash u |\!)[(0, \top), (0, \top), \dots, (0, \top)]_2 \\
&\qquad \wedge \quad (\!| n+1 \vdash u |\!)[(1, \top), (0, \top), \dots, (0, \top)]_1 \neq \infty \\
&= \ (\!| n+1 \vdash u |\!)[(0, \top), (0, \top), \dots, (0, \top)]_2 \\
&\qquad \wedge \quad \mathtt{occ}_1(n+1 \vdash u) \leq 1 .
\end{aligned}
$$

  where the last step comes from Proposition A.7. By the induction hypothesis on the scoped term $n+1 \vdash u$, we get that $n \vdash \mathtt{abs}(u)$ is affine in its bound variables if and only if $(\!| n \vdash \mathtt{abs}(u) |\!)[(0, \top), \dots, (0, \top)]_2$ is $\top$.

- For any scoped terms $n \vdash u$ and $n \vdash v$, we have

$$
\begin{aligned}
&(\!| n \vdash \mathtt{app}(u,v) |\!)[(0, \top), \dots, (0, \top)]_2 \\
&:= \ f_{\mathsf{app}}((\!| n \vdash u |\!)[(0, \top), \dots, (0, \top)])((\!| n \vdash v |\!)[(0, \top), \dots, (0, \top)])_2 \\
&= \ (\!| n \vdash u |\!)[(0, \top), \dots, (0, \top)]_2 \ \wedge \ (\!| n \vdash v |\!)[(0, \top), \dots, (0, \top)]_2
\end{aligned}
$$

  which shows, by the induction hypotheses on $n \vdash u$ and $n \vdash v$, that $n \vdash \mathtt{app}(u,v)$ is affine in its bound variables if and only if $(\!| n \vdash \mathtt{app}(u,v) |\!)[(0, \top), \dots, (0, \top)]_2$ is $\top$. ◀

▶ **Theorem A.10.** *The language of closed affine untyped terms is regular in* **FinSet***.*

Proof. We consider the language

$$L \ := \ \{t \in \Lambda(\mathsf{UntypedTerms}) \mid t \text{ is affine}\} .$$

By definition, $t \in \Lambda(\mathsf{UntypedTerms})$ is affine if and only if $0 \vdash u$ is affine in its bound variables, where $0 \vdash u$ is the unique scoped term such that $t = \overline{0 \vdash u}$, given by Claim A.3. Moreover, by Proposition A.9, we have that $0 \vdash u$ is affine in its bound variables if and only if

$$(\! (0 \vdash u) \!)_2 \quad = \quad \top$$

and, by Remark A.6, $(\! (0 \vdash u) \!)_2 = [\![t]\!]_Q (f_{\mathsf{abs}})(f_{\mathsf{app}})$. Therefore, if we define the subset F of $[\![\mathsf{UntypedTerms}]\!]_Q$ as

$$F \quad := \quad \{ s \in [\![\mathsf{UntypedTerms}]\!]_Q \mid s(f_{\mathsf{abs}})(f_{\mathsf{app}})_2 = \top \}$$

we get that $L = L_F$ and therefore that L is **FinSet**-recognizable. ◀

B   Sub-CCC proofs

Throughout the article, we construct new CCCs as full subcategories of other CCCs which verify some stability properties. A sub-CCC is a sub-category whose inclusion functor is a CCC functor. More precisely, this construction is justified by the following claim:

▷ Claim B.1.   Let $\mathbf{C}$ be a CCC and $\mathbf{D}$ be a full subcategory of $\mathbf{C}$. If:
- the terminal object of $\mathbf{C}$ belongs to $\mathbf{D}$,
- the objects of $\mathbf{D}$ are stable under cartesian products,
- the objects of $\mathbf{D}$ are stable under exponentiation,

then $\mathbf{D}$ is a sub-CCC of $\mathbf{C}$.

Proof of Theorem 4.10

By definition of a squeezing structure, we have the following facts:
- The terminal object 1 of $\mathbf{C}$ is an object of $\mathbf{Sqz}(\mathbf{C})$.
- If c and c' belong to $\mathbf{Sqz}(\mathbf{C})$, then $c \times c'$ still does. Indeed, we have the following morphisms:

$$L_{c \times c'} \xrightarrow{\ 1\ } L_c \times L_{c'} \xrightarrow{\ 1\ } c \times c' \qquad \text{and} \qquad c \times c' \xrightarrow{\ \mathsf{r}\ } R_c \times R_{c'} \xrightarrow{\ \mathsf{r}\ } R_{c \times c'} \ .$$

- If c and c' belong to $\mathbf{Sqz}(\mathbf{C})$, then $c \Rightarrow c'$ still does. Indeed, we have the following morphisms:

$$L_{c \Rightarrow c'} \xrightarrow{\ 1\ } R_c \Rightarrow L_{c'} \xrightarrow{\ 1\ } c \Rightarrow c' \qquad \text{and} \qquad c \Rightarrow c' \xrightarrow{\ \mathsf{r}\ } L_c \Rightarrow R_{c'} \xrightarrow{\ \mathsf{r}\ } R_{c \Rightarrow c'} \ .$$

By Claim B.1, we get that $\mathbf{Sqz}(\mathbf{C})$ is a sub-CCC of $\mathbf{C}$.

Proof of Proposition 5.7

The following facts hold about the category $\mathbf{C}_{\geq 1}$ of inhabited objects:
- The terminal object 1 of $\mathbf{C}$ is in $\mathbf{C}_{\geq 1}$.
- If c and c' belong to $\mathbf{C}_{\geq 1}$, which means that there exists $f \in \mathbf{C}(1, c)$ and $f' \in \mathbf{C}(1, c')$, then the existence of the morphism $\langle f, f' \rangle \in \mathbf{C}(1, c \times c')$ shows that $c \times c'$ belongs to $\mathbf{C}_{\geq 1}$.
- If c is any object of $\mathbf{C}$ (so, it may in particular be an object of $\mathbf{C}_{\geq 1}$) and c' is an object of $\mathbf{C}_{\geq 1}$, which means that there exists a morphism $f' \in \mathbf{C}(1, c')$, then the curryfication of the composition $f \circ !_c$, where $!_c \in \mathbf{C}(c, 1)$, shows that $c \Rightarrow c'$ belongs to $\mathbf{C}_{\geq 1}$. Notice that the argument actually shows that $\mathbf{C}_{\geq 1}$ is an exponential ideal.

By Claim B.1, this proves that $\mathbf{C}_{\geq 1}$ is a sub-CCC of $\mathbf{C}$.

First proof of Proposition 5.12

Let R be the CCC of logical relations from **FinSet** to **E**. We show that the full subcategory $\mathbf{R}_{\mathrm{p.s.}}$ of objects $(Q, e, \Vdash)$ such that the relation $\Vdash$ is a partial surjection is a sub-CCC of **R**. We have the following facts:

- The terminal object of **R** belongs to $\mathbf{R}_{\mathrm{p.s.}}$.
- Let $(Q, e, \Vdash)$ and $(Q', e', \Vdash')$ be objects of $\mathbf{R}_{\mathrm{p.s.}}$, and $\Vdash^\times$ be the relation of their product.
  - Let $\langle q, q' \rangle \in Q \times Q'$ and $\langle x_i, x_i' \rangle \in \mathbf{E}(1, e \times e')$ such that $\langle q, q' \rangle \Vdash^\times \langle x_i, x_i' \rangle$ for $i = 1, 2$. This means that $q \Vdash x_i$ and $q' \Vdash' x_i'$ for $i = 1, 2$. By functionality of $\Vdash$ and $\Vdash'$, we get that $x_1 = x_2$ and $x_1' = x_2'$. Therefore, $\langle x_1, x_1' \rangle = \langle x_2, x_2' \rangle$, so $\Vdash^\times$ is functional.
  - Let $\langle x, x' \rangle$ be any element of $\mathbf{E}(1, e \times e')$. By surjectivity of $\Vdash$ and $\Vdash'$, there exists $q \in Q$ and $q' \in Q'$ such that $q \Vdash x$ and $q \Vdash x'$. Therefore, $\langle q, q' \rangle \Vdash^\times \langle x, x' \rangle$, so $\Vdash^\times$ is surjective.
- Let $(Q, e, \Vdash)$ and $(Q', e', \Vdash')$ be objects of $\mathbf{R}_{\mathrm{p.s.}}$, and $\Vdash^\Rightarrow$ be the relation of their arrow.
  - Let $f \in Q \Rightarrow Q'$ and $g_i \in \mathbf{E}(1, e \Rightarrow e')$ such that $f \Vdash^\Rightarrow g_1$ and $f \Vdash^\Rightarrow g_2$. Let x be any element of $\mathbf{E}(1, e)$. By surjectivity of $\Vdash$, there exists a $q \in Q$ such that $q \Vdash x$, so $f(q) \Vdash' g_i \circ x$ for $i = 1, 2$. By functionality of $\Vdash'$, we get $g_1 \circ x = g_2 \circ x$. As this equality holds for all $x \in \mathbf{E}(1, e)$ *and* **E** *is well-pointed*, we get that $g_1 = g_2$, which proves that $\Vdash^\Rightarrow$ is functional.
  - Let $g \in \mathbf{E}(1, e \Rightarrow e')$. By surjectivity of R', we can choose a function $\rho \colon \mathbf{E}(1, X') \to Q'$ such that $(\rho(x'), x') \in R'$ for every x'. Let $f'(q) = \rho(g \circ (R(q)))$ where R, as a functional relation, can seen as a partial function $Q \rightharpoonup \mathbf{E}(1, X)$. Then f' is also a partial function; since its codomain Q' is non-empty (indeed, $\mathbf{E}(1, X') \neq \varnothing$ by assumption and $\rho(\mathbf{E}(1, X')) \subseteq Q'$) it can be extended to a total function $f \colon Q \to Q'$. This construction ensures that $(f, g) \in R \Rightarrow R'$, so we have shown that $R \Rightarrow R'$ is surjective.

By Claim B.1, we get that $\mathbf{R}_{\mathrm{p.s.}}$ is a sub-CCC of **R**.

Proof of Proposition 5.8

We remark that the following facts hold about $\mathbf{E}_{\leq 1}$:

- The terminal object 1 of **E** is in $\mathbf{E}_{\leq 1}$.
- If e and e' belong to $\mathbf{E}_{\leq 1}$, then $\mathbf{E}(1, e \times e')$ is in bijection with $\mathbf{E}(1, e) \times \mathbf{E}(1, e')$, so $e \times e'$ belongs to $\mathbf{E}_{\leq 1}$.
- If e and e' belong to $\mathbf{E}_{\leq 1}$, as **E** is well-pointed, we have an injection of $\mathbf{E}(1, e \Rightarrow e')$ into $\mathbf{E}(1, e) \Rightarrow \mathbf{E}(1, e')$, so $e \Rightarrow e'$ belongs to $\mathbf{E}_{\leq 1}$.

By Claim B.1, this proves that $\mathbf{E}_{\leq 1}$ is a sub-CCC of **E**.

Let e and e' be two objects of $\mathbf{E}_{=1}$. As the object $e \Rightarrow e'$ belongs to both $\mathbf{E}_{\geq 1}$ and $\mathbf{E}_{\leq 1}$, there exists a unique morphism from e to e'. This shows that $\mathbf{E}_{=1}$ is equivalent to the terminal category.

C Partial surjections

Proof of Lemma 4.11

Let $c = (c_1, c_2, \Vdash)$ be an object of $\mathbf{Sqz}(\mathbf{R})$. There exist morphisms

$$(u_1, u_2) \ : \ L_c \xrightarrow{\ \mathrm{l}\ } c \qquad \text{and} \qquad (v_1, v_2) \ : \ c \xrightarrow{\ \mathrm{r}\ } R_c \ .$$

We show that the relation $\Vdash$ is functional and surjective.

- Let $x_1 \in \mathbf{C}_1(1, c_1)$ and $x_2, x'_2 \in \mathbf{C}_2(1, c_2)$ such that $x_1 \Vdash x_2$ and $x_1 \Vdash x'_2$.
  As (v_1, v_2) is a morphism, we get $v_1 \circ x_1 \Vdash v_2 \circ x_2$ and $v_1 \circ x_1 \Vdash v_2 \circ x'_2$. As the relation of R_c is functional, we get that $v_2 \circ x_2 = v_2 \circ x'_2$. By injectivity of $\mathbf{C}_2(1, v_2)$, we get that $x_2 = x'_2$, so the relation $\Vdash$ is functional.
- Let $x_2 \in \mathbf{C}_2(1, c_2)$. We write $(c_{L,1}, c_{L,2} \Vdash_L)$ for the object L_c.
  By surjectivity of $\mathbf{C}_2(1, v_2)$, there exists $x_{L,2} \in \mathbf{C}_2(1, c_{L,2})$ such that $x_2 = v_2 \circ x_{L,2}$. By surjectivity of the relation $\Vdash_L$, there exists $x_{L,1}$ such that $x_{L,1} \Vdash_L x_{L,2}$. As (u_1, u_2) is a morphism, we have $u_1 \circ x_{L,1} \Vdash x_2$, so the relation $\Vdash$ is surjective.

This shows that all objects in $\mathbf{Sqz}(\mathbf{R})$ are partial surjections.

D Squeezing structures

Second proof of Proposition 5.12

There exists a squeezing structure such that
- for any object $R = (Q, e, \Vdash)$ of $\mathbf{R}$, the objects L_R and R_R are both equal to T_e;
- the two wide subcategories $\mathbf{R}_{\text{left}}$ and $\mathbf{R}_{\text{right}}$ are both taken to be the wide subcategory of target-identities.

The fact that the point functor T is product-preserving gives us all the target-identities of the squeezing structure, except for the case of the morphism

$$(\mathbf{E}(1, e), e, \sim_e) \Rightarrow (\mathbf{E}(1, e'), e', \sim_{e'}) \longrightarrow (\mathbf{E}(1, e \Rightarrow e'), e \Rightarrow e', \sim_{e \Rightarrow e'}) .$$

For this morphism, we will crucially use the fact that we relate $\mathbf{FinSet}$ and $\mathbf{E}$, which is well-pointed. We have a set-theoretic function

$$i \quad : \quad \mathbf{E}(1, e \Rightarrow e') \longrightarrow \mathbf{E}(1, e) \Rightarrow \mathbf{E}(1, e')$$

which is injective as $\mathbf{E}$ is well-pointed and lifts to a target-identity. As the object $e \Rightarrow e'$ of $\mathbf{E}$ is inhabited, the set $\mathbf{E}(1, e \Rightarrow e')$ is non-empty and the set-theoretic function i admits a retraction

$$r \quad : \quad \mathbf{E}(1, e) \Rightarrow \mathbf{E}(1, e') \longrightarrow \mathbf{E}(1, e \Rightarrow e')$$

which lifts to a target-identity $T_e \Rightarrow T_{e'} \to T_{e \Rightarrow e'}$. By Lemma 4.11, we know that the objects of $\mathbf{Sqz}(\mathbf{R})$ are partial surjections. Conversely, suppose that $R = (Q, e, \Vdash)$ is such that $\Vdash$ is a partial surjection. We the two following target-identities:
- The fact that the relation $\Vdash$ is surjective yields a set-theoretic function $\mathbf{E}(1, e) \to Q$ which lifts to a target-identity $T_e \to R$.
- As the relation $\Vdash$ is functional and $\mathbf{E}(1, e)$ is non-empty as e is inhabited, there exists a set-theoretic function $Q \to \mathbf{E}(1, e)$ extending $\Vdash$, and any such set-theoretic function lifts to a target-identity $R \to T_e$.

This shows that the objects of $\mathbf{Sqz}(\mathbf{R})$ are exactly the partial surjections, which then form a sub-CCC of $\mathbf{R}$ by Theorem 4.10.

Proof of Proposition 6.4

It is clear that target-identities are composable and stable under finite products and exponentials, which is what is asked given that left and right morphisms are the same in the present case.

We are left with the task to show the existence of the morphisms as described in Equation (2) in our particular setting. As there exists at most one target-identity whose $\mathbf{Lam}$ component is a given λ-term, we give these λ-terms.

- **Case** $\mathrm{Bij}_1 \to 1$**:** The unique morphism $\mathrm{Bij}_1 \to 1$ is a target-identity.
- **Case** $1 \to \mathrm{Bij}_1$**:** The λ-term $\lambda(y : 1). \lambda(x : \mathrm{o}). x$ lifts to a target-identity

$$1 \quad \longrightarrow \quad (\mathrm{Fin}(1), 1, \Vdash_1) \ .$$

- **Case** $\mathrm{Bij}_{n \times n'} \to \mathrm{Bij}_n \times \mathrm{Bij}_{n'}$**:** the fact that $\mathrm{Bij}_{(-)}$ is a functor gives us directly such a morphism which can be verified to be a target-identity.
- **Case** $\mathrm{Bij}_n \times \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n \times n'}$**:** The morphism is given by the λ-term

$$\lambda(p : \mathrm{Fin}(n) \times \mathrm{Fin}(n')). \lambda(x : \mathrm{o}^{n \times n'}). p_1 \ \langle \mathrm{Fin}(1 \times \mathrm{Id}) \ p_2 \ x, \ldots, \mathrm{Fin}(n \times \mathrm{Id}) \ p_2 \ x \rangle$$

where $i \times \mathrm{Id}$ is the function $n' \to n \times n'$ sending j on (i, j), from which we get the λ-term $\mathrm{Fin}(i \times \mathrm{Id})$ of simple type $\mathrm{Fin}(n') \Rightarrow \mathrm{Fin}(n \times n')$.
This λ-term lifts to a target-identity

$$\mathrm{Bij}_n \times \mathrm{Bij}_{n'} \ \longrightarrow \ \mathrm{Bij}_{n \times n'} \ .$$

- **Case** $\mathrm{Bij}_{n \Rightarrow n'} \to \mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'}$**:** We have the target-identity

$$\mathrm{Bij}_{n \Rightarrow n'} \times \mathrm{Bij}_n \ \longrightarrow \ \mathrm{Bij}_{(n \Rightarrow n') \times n}$$

which, when composed with the morphism $\mathrm{Bij}_{\mathrm{ev}_{n,n'}}$ which has the evaluation morphism $\mathrm{ev}_{n,n'}$ as target-component, yields a morphism

$$\mathrm{Bij}_{n \Rightarrow n'} \times \mathrm{Bij}_n \ \longrightarrow \ \mathrm{Bij}_{n'}$$

which, after curryfication, gets us a target-identity $\mathrm{Bij}_{n \Rightarrow n'} \to \mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'}$.
- **Case** $\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n \Rightarrow n'}$**:** Notice that the equality

$$n \Rightarrow n' \ = \ \underbrace{n' \times \cdots \times n'}_{n \text{ times}}$$

shows that the λ-term of type $\mathrm{Fin}(n) \Rightarrow \mathrm{Fin}(n') \to (\mathrm{Fin}(n'))^n$

$$\lambda(F : \mathrm{Fin}(n) \Rightarrow \mathrm{Fin}(n')). \langle F \ \pi_1, \ldots, F \ \pi_n \rangle$$

lifts to a target-identity $\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n'}{}^n$. By postcomposing this target-identity with an iteration of the target-identities $\mathrm{Bij}_m \times \mathrm{Bij}_{m'} \to \mathrm{Bij}_{m \times m'}$, we obtain the target-identity

$$\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \ \longrightarrow \ \mathrm{Bij}_{n \Rightarrow n'} \ .$$

This finishes the proof that there is a squeezing structure as described in the statement of the proposition.