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

Lê Thành Dũng (Tito) Nguyễn — `nltd@nguyentito.eu` – École normale supérieure de Lyon
joint work with Vincent Moreau (IRIF, Université Paris Cité)

22 March 2024, séminaire Gallinette, LS2N / Inria Nantes

Defining languages in the simply typed λ-calculus (assuming you know the latter)

Church encodings of binary strings [Böhm & Berarducci 1985]

$\simeq$ *fold_right* on a list of characters (generalizable to any alphabet; $\mathsf{Nat} = \mathsf{Str}_{\{1\}}$):

$$\overline{\mathtt{011}} = \lambda f_0.\ \lambda f_1.\ \lambda x.\ f_0\ (f_1\ (f_1\ x)) : \mathsf{Str}_{\{0,1\}} = (o \to o) \to (o \to o) \to o \to o$$

can also be “type-cast” to $\overline{\mathtt{011}}[A] : \mathsf{Str}_{\{0,1\}}[A] = \mathsf{Str}_{\{0,1\}}\{o := A\}$ for any simple type A

Defining languages in the simply typed λ-calculus (assuming you know the latter)

Church encodings of binary strings [Böhm & Berarducci 1985]

$\simeq$ *fold_right* on a list of characters (generalizable to any alphabet; $\mathsf{Nat} = \mathsf{Str}_{\{1\}}$):

$$\overline{011} = \lambda f_0.\ \lambda f_1.\ \lambda x.\ f_0\ (f_1\ (f_1\ x)) : \mathsf{Str}_{\{0,1\}} = (o \to o) \to (o \to o) \to o \to o$$

can also be “type-cast” to $\overline{011}[A] : \mathsf{Str}_{\{0,1\}}[A] = \mathsf{Str}_{\{0,1\}}\{o := A\}$ for any simple type A

Simply typed λ-terms $t : \mathsf{Str}_{\{0,1\}}[A] \to \mathsf{Bool}$ define **languages** $L \subseteq \{0, 1\}^*$

Defining languages in the simply typed λ-calculus (assuming you know the latter)

Church encodings of binary strings [Böhm & Berarducci 1985]

$\simeq$ `fold_right` on a list of characters (generalizable to any alphabet; $\mathsf{Nat} = \mathsf{Str}_{\{1\}}$):

$$\overline{\texttt{011}} = \lambda f_0.\ \lambda f_1.\ \lambda x.\ f_0\ (f_1\ (f_1\ x)) : \mathsf{Str}_{\{\texttt{0},\texttt{1}\}} = (o \to o) \to (o \to o) \to o \to o$$

can also be "type-cast" to $\overline{\texttt{011}}[A] : \mathsf{Str}_{\{\texttt{0},\texttt{1}\}}[A] = \mathsf{Str}_{\{\texttt{0},\texttt{1}\}}\{o := A\}$ for any simple type A

Simply typed λ-terms $t : \mathsf{Str}_{\{\texttt{0},\texttt{1}\}}[A] \to \mathsf{Bool}$ define **languages** $L \subseteq \{\texttt{0}, \texttt{1}\}^*$

Example: $t = \lambda s.\ s\ \mathit{id}\ \mathit{not}\ \mathit{true} : \mathsf{Str}_{\{\texttt{0},\texttt{1}\}}[\mathsf{Bool}] \to \mathsf{Bool}$ (even number of $\texttt{1}$s)

$$t\ \overline{\texttt{011}}[\mathsf{Bool}] \longrightarrow_\beta \overline{\texttt{011}}[\mathsf{Bool}]\ \mathit{id}\ \mathit{not}\ \mathit{true} \longrightarrow_\beta \mathit{id}\ (\mathit{not}\ (\mathit{not}\ \mathit{true})) \longrightarrow_\beta \mathit{true}$$

Defining languages in the simply typed λ-calculus (assuming you know the latter)

Church encodings of binary strings [Böhm & Berarducci 1985]

$\simeq$ `fold_right` on a list of characters (generalizable to any alphabet; $\mathsf{Nat} = \mathsf{Str}_{\{1\}}$):

$$\overline{\mathtt{011}} = \lambda f_0.\ \lambda f_1.\ \lambda x.\ f_0\ (f_1\ (f_1\ x)) : \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}} = (o \to o) \to (o \to o) \to o \to o$$

can also be "type-cast" to $\overline{\mathtt{011}}[A] : \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}}[A] = \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}}\{o := A\}$ for any simple type A

Simply typed λ-terms $t : \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}}[A] \to \mathsf{Bool}$ define **languages** $L \subseteq \{\mathtt{0}, \mathtt{1}\}^*$

Example: $t = \lambda s.\ s\ \mathtt{id}\ \mathtt{not}\ \mathtt{true} : \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}}[\mathsf{Bool}] \to \mathsf{Bool}$ (even number of $\mathtt{1}$s)

$$t\ \overline{\mathtt{011}}[\mathsf{Bool}] \longrightarrow_\beta \overline{\mathtt{011}}[\mathsf{Bool}]\ \mathtt{id}\ \mathtt{not}\ \mathtt{true} \longrightarrow_\beta \mathtt{id}\ (\mathtt{not}\ (\mathtt{not}\ \mathtt{true})) \longrightarrow_\beta \mathtt{true}$$

Theorem (Hillebrand & Kanellakis 1996)

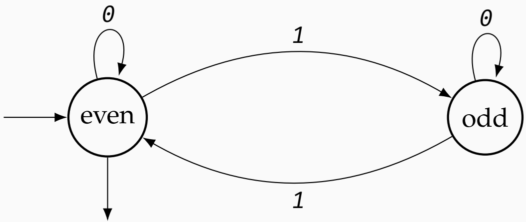
All regular languages, *and only those, can be defined this way.*

i.e. "**syntactically regular**" lang. $\subseteq \{u \mid u : \mathsf{Str}_{\{\mathtt{0},\mathtt{1}\}}\}/(=_{\beta\eta}) \iff$ regular lang. $\subseteq \{\mathtt{0}, \mathtt{1}\}^*$

Regular languages

Many classical equivalent definitions (+ STLC with Church encodings!):

- *regular expressions*: `0*(10*10*)*` = "only `0`s and `1`s & even number of `1`s"
- *finite automata* (DFA/NFA): e.g. drawing below



Regular languages

Many classical equivalent definitions (+ STLC with Church encodings!):

- *regular expressions*: `0*(10*10*)*` = "only `0`s and `1`s & even number of `1`s"
- *finite automata* (DFA/NFA)
- *algebraic* definition below (very close to DFA), e.g. $M = \mathbb{Z}/(2)$

Theorem (classical – attributed to Myhill by [Rabin & Scott 1958])

A language $L \subseteq \Sigma^*$ is regular $\iff$ the corresponding decision problem factors as

$$\Sigma^* \xrightarrow{\textit{some morphism}} \textit{some}\ \text{finite monoid}\ M \to \{\textit{yes}, \textit{no}\}$$

$\rightsquigarrow$ *compositional* (as in denotational semantics!) and *finitary* interpretation of strings

Recognizing languages of simply typed λ-terms via semantics

Naive set-theoretic interpretation of simply typed λ-terms

$$[\![o]\!]_Q = Q \text{ (an arbitrary set)}$$

$$[\![A \to B]\!]_Q = [\![A]\!]_Q \to [\![B]\!]_Q = [\![B]\!]_Q^{[\![A]\!]_Q}$$

$$t : A \implies [\![t]\!]_Q \in [\![A]\!]_Q$$

- Always compositional by def., e.g. $[\![t\, u]\!]_Q = [\![t]\!]_Q\,([\![u]\!]_Q)$ + invariant mod $=_{\beta\eta}$
- Q finite $\implies$ every $[\![A]\!]_Q$ finite

Recognizing languages of simply typed λ-terms via semantics

Naive set-theoretic interpretation of simply typed λ-terms

$$[\![o]\!]_Q = Q \text{ (an arbitrary set)}$$

$$[\![A \to B]\!]_Q = [\![A]\!]_Q \to [\![B]\!]_Q = [\![B]\!]_Q^{[\![A]\!]_Q}$$

$$t : A \implies [\![t]\!]_Q \in [\![A]\!]_Q$$

- Always compositional by def., e.g. $[\![t\,u]\!]_Q = [\![t]\!]_Q\,([\![u]\!]_Q)$ + invariant mod $=_{\beta\eta}$
- Q finite $\implies$ every $[\![A]\!]_Q$ finite

Definition (Regular languages of λ-terms of type A [Salvati 2009])

$$\{t \mid t : A\}/(=_{\beta\eta}) \xrightarrow{[\![-]\!]_Q} [\![A]\!]_Q \text{ where } Q \text{ is a chosen } \textit{finite} \text{ set} \to \{\text{yes}, \text{no}\}$$

$[\![\overline{w}]\!]_Q \in [\![\mathsf{Str}_\Sigma]\!]_Q \cong$ results of all runs of DFAs with states Q on $\mathtt{rev}(w)$ (via `fold_right`):
"**semantically regular**" (à la Salvati) lang. at type $\mathsf{Str}_\Sigma \underbrace{\implies}_{\text{converse also holds (easy)}}$ regular lang. over Σ^*

Notions of regular languages of simply typed λ-terms

Definition (Regularity of a "language" $\{t \mid t : A\}/(=_{\beta\eta}) \to \{\mathbf{yes}, \mathbf{no}\}$)

Semantically regular: factors through naive set semantics $[\![-]\!]_Q$ for Q finite [Salvati 2009]

Syntactically regular: defined by some term of type $A[B] \to \mathtt{Bool}$

inspired by [Hillebrand & Kanellakis 1996]

For $A = \mathsf{Str}_\Sigma$, both equivalent to regular languages over Σ^* $\rightsquigarrow$ *robust/canonical notion!*

many equivalent defs: regexp, automata variants, monoids, monadic second-order logic, ...

Notions of regular languages of simply typed λ-terms

Definition (Regularity of a "language" $\{t \mid t : A\}/(=_{\beta\eta}) \to \{\mathbf{yes}, \mathbf{no}\}$)

Semantically regular: factors through naive set semantics $[\![-]\!]_Q$ for Q finite [Salvati 2009]

Syntactically regular: defined by some term of type $A[B] \to \mathsf{Bool}$

inspired by [Hillebrand & Kanellakis 1996]

For $A = \mathsf{Str}_\Sigma$, both equivalent to regular languages over Σ^* $\rightsquigarrow$ *robust/canonical notion!*

many equivalent defs: regexp, automata variants, monoids, monadic second-order logic, …

Are regular languages of simply typed λ-terms a robust notion?

- syntactically reg. $\iff$ semantically reg. $\forall A$?
- other definitions?

Notions of regular languages of simply typed λ-terms

Definition (Regularity of a "language" $\{t \mid t : A\}/(=_{\beta\eta}) \to \{\mathbf{yes}, \mathbf{no}\}$)

Semantically regular: factors through naive set semantics $[\![-]\!]_Q$ for Q finite [Salvati 2009]

Syntactically regular: defined by some term of type $A[B] \to \mathsf{Bool}$

inspired by [Hillebrand & Kanellakis 1996]

For $A = \mathsf{Str}_\Sigma$, both equivalent to regular languages over $\Sigma^* \rightsquigarrow$ *robust/canonical notion!*
many equivalent defs: regexp, automata variants, monoids, monadic second-order logic, …

Are regular languages of simply typed λ-terms a robust notion?

- syntactically reg. $\iff$ semantically reg. $\forall A$? **Yes!** [Moreau & N., CSL'24]
- other definitions?

Notions of regular languages of simply typed λ-terms

Definition (Regularity of a "language" $\{t \mid t : A\}/(=_{\beta\eta}) \to \{\mathbf{yes}, \mathbf{no}\}$)

Semantically regular: factors through naive set semantics $[\![-]\!]_Q$ for Q finite [Salvati 2009]

Syntactically regular: defined by some term of type $A[B] \to \mathsf{Bool}$

inspired by [Hillebrand & Kanellakis 1996]

For $A = \mathsf{Str}_\Sigma$, both equivalent to regular languages over Σ^* $\rightsquigarrow$ *robust/canonical notion!*
many equivalent defs: regexp, automata variants, monoids, monadic second-order logic, ...

Are regular languages of simply typed λ-terms a robust notion?

- syntactically reg. $\iff$ semantically reg. $\forall A$? **Yes!** [Moreau & N., CSL'24]
- other definitions? some other finite semantics, e.g. finite Scott domains [Salvati]

Notions of regular languages of simply typed λ-terms

Definition (Regularity of a "language" $\{t \mid t : A\}/(=_{\beta\eta}) \to \{\mathbf{yes}, \mathbf{no}\}$)

Semantically regular: factors through naive set semantics $[\![-]\!]_Q$ for Q finite [Salvati 2009]

Syntactically regular: defined by some term of type $A[B] \to \mathsf{Bool}$

inspired by [Hillebrand & Kanellakis 1996]

For $A = \mathsf{Str}_\Sigma$, both equivalent to regular languages over Σ^* $\rightsquigarrow$ *robust/canonical notion!*
many equivalent defs: regexp, automata variants, monoids, monadic second-order logic, ...

Are regular languages of simply typed λ-terms a robust notion?

- syntactically reg. $\iff$ semantically reg. $\forall A$? **Yes!** [Moreau & N., CSL'24]
- other definitions? some other finite semantics, e.g. finite Scott domains [Salvati]

- Statman's finite completeness theorem = regularity of singleton languages
- applications to categorial grammars, higher-order matching, ... cf. Salvati's HDR

Syntactically implies semantically regular

Proof.

Fix $t : A[B] \to \mathsf{Bool}$. Choose $Q = \{0, 1\}$ so that $[\![\mathtt{true}]\!]_Q \neq [\![\mathtt{false}]\!]_Q$.

$$\forall u : A,\ t\, u[B] \to_\beta^* \mathtt{true} \iff [\![t\, u[B]]\!]_Q = [\![t]\!]_Q([\![u[B]]\!]_Q) = [\![\mathtt{true}]\!]_Q$$

Since $[\![u[B]]\!]_Q = [\![u]\!]_{[\![B]\!]_Q}$, the language defined by t factors as

$$\{u \mid u : A\}/(=_{\beta\eta}) \xrightarrow{[\![-]\!]_{[\![B]\!]_Q}} [\![A]\!]_{[\![B]\!]_Q} = [\![A[B]]\!]_Q \xrightarrow{[\![t]\!]_Q(-)=[\![\mathtt{true}]\!]_Q\ ?} \{\text{yes}, \text{no}\}$$

$\square$

Syntactically implies semantically regular

Proof.

Fix $t : A[B] \to \mathsf{Bool}$. Choose $Q = \{0,1\}$ so that $[\![\mathtt{true}]\!]_Q \neq [\![\mathtt{false}]\!]_Q$.

$$\forall u : A,\ t\,u[B] \to_\beta^* \mathtt{true} \iff [\![t\,u[B]]\!]_Q = [\![t]\!]_Q([\![u[B]]\!]_Q) = [\![\mathtt{true}]\!]_Q$$

Since $[\![u[B]]\!]_Q = [\![u]\!]_{[\![B]\!]_Q}$, the language defined by t factors as

$$\{u \mid u : A\}/(=_{\beta\eta}) \xrightarrow{[\![-]\!]_{[\![B]\!]_Q}} [\![A]\!]_{[\![B]\!]_Q} = [\![A[B]]\!]_Q \xrightarrow{[\![t]\!]_Q(-)=[\![\mathtt{true}]\!]_Q\ ?} \{\text{yes}, \text{no}\}$$

□

- this is the "hard" direction of "syntactically reg. at $\mathsf{Str}_\Sigma \iff$ reg. over Σ^*" [HK96]
  becomes easy once you know you should go through finite semantics
- works for *any* "non-trivial" model of STλC = non-posetal cartesian closed category $\mathcal{C}$
  $\rightsquigarrow$ inducing $[\![-]\!]' :$ types $\to$ objects of $\mathcal{C}$ + $[\![-]\!]' : (A \in \text{types}) \to (t : A) \to \mathcal{C}(1, [\![A]\!]')$

The other equivalences

1. Syntactically regular $\implies$ recognized by any non-trivial model: done
2. Semantically reg. i.e. recognized by **FinSet** $\implies$ syntactically reg.: slightly tricky, later
3. Recognized by a finite *extensional* model $\implies$ by **FinSet**: claimed in Salvati's HDR

The other equivalences

1. Syntactically regular $\implies$ recognized by any non-trivial model: done
2. Semantically reg. i.e. recognized by **FinSet** $\implies$ syntactically reg.: slightly tricky, later
3. Recognized by a finite *extensional* model $\implies$ by **FinSet**: claimed in Salvati's HDR

"[...] using logical relations one *easily* establishes that recognizability with standard models is equivalent to recognizability with any extensional model" (finiteness implicit)

The other equivalences

1. Syntactically regular $\implies$ recognized by any non-trivial model: done
2. Semantically reg. i.e. recognized by **FinSet** $\implies$ syntactically reg.: slightly tricky, later
3. Recognized by a finite *extensional* model $\implies$ by **FinSet**: claimed in Salvati's HDR

"[...] using **logical relations** one easily establishes that recognizability with standard models is equivalent to recognizability with any extensional model" (finiteness implicit)

The other equivalences

1. Syntactically regular $\implies$ recognized by any non-trivial model: done
2. Semantically reg. i.e. recognized by **FinSet** $\implies$ syntactically reg.: slightly tricky, later
3. Recognized by a finite *extensional* model $\implies$ by **FinSet**: claimed in Salvati's HDR

"[...] using **logical relations** one easily establishes that recognizability with standard models is equivalent to recognizability with any extensional model" (finiteness implicit)

Logical relations also prove (2)! As a warm-up, we'll start with (3)

The other equivalences

1. Syntactically regular $\implies$ recognized by any non-trivial model: done
2. Semantically reg. i.e. recognized by **FinSet** $\implies$ syntactically reg.: slightly tricky, later
3. Recognized by a finite *extensional* model $\implies$ by **FinSet**: claimed in Salvati's HDR

"[...] using **logical relations** one easily establishes that recognizability with standard models is equivalent to recognizability with any extensional model" (finiteness implicit)

Logical relations also prove (2)! As a warm-up, we'll start with (3)

Extensional models (finitary when $[\![o]\!]'$ finite)

$[\![o]\!]'$ = an arbitrary set $\qquad t : A \implies [\![t]\!]' \in [\![A]\!]'$

$[\![A \to B]\!]' \subseteq [\![A]\!]' \to [\![B]\!]'$ e.g. monotone functions between posets (finite Scott domains)

Equivalently: *well-pointed* cartesian closed categories i.e. $\mathcal{C}(X, Y) \hookrightarrow (\mathcal{C}(1, X) \to \mathcal{C}(1, Y))$

A logical relation between two models

$\Vdash_A \subseteq \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}} \times \overbrace{[\![A]\!]'}^{\text{some other model}}$ defined inductively: choose $\Vdash_o$ and take

$$f \Vdash_{A \to B} g \iff \forall (x, y) \in [\![A]\!]_Q \times [\![A]\!]',\ x \Vdash_A y \Rightarrow f(x) \Vdash_B g(y)$$

A logical relation between two models

$\Vdash_A \subseteq \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}} \times \overbrace{[\![A]\!]'}^{\text{some other model}}$ defined inductively: choose $\Vdash_o$ and take

$$f \Vdash_{A \to B} g \iff \forall (x, y) \in [\![A]\!]_Q \times [\![A]\!]',\ x \Vdash_A y \Rightarrow f(x) \Vdash_B g(y)$$

Fundamental lemma of logical relations

For any $t : A$, we have $[\![t]\!]_Q \Vdash_A [\![t]\!]'$.

Proof by induction on the syntax – amounts to proving "$A \mapsto ([\![A]\!]_Q, [\![A]\!]', \Vdash_A)$ is a model";
the interpretation of t in that model witnesses that $[\![t]\!]_Q \Vdash_A [\![t]\!]'$
(categorically: *gluing* of two CCCs – later if time allows)

A logical relation between two models

$\Vdash_A \subseteq \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}} \times \overbrace{[\![A]\!]'}^{\text{some other model}}$ defined inductively: choose $\Vdash_o$ and take

$$f \Vdash_{A \to B} g \iff \forall (x, y) \in [\![A]\!]_Q \times [\![A]\!]',\ x \Vdash_A y \Rightarrow f(x) \Vdash_B g(y)$$

Fundamental lemma of logical relations

For any $t : A$, we have $[\![t]\!]_Q \Vdash_A [\![t]\!]'$.

Proof by induction on the syntax – amounts to proving "$A \mapsto ([\![A]\!]_Q, [\![A]\!]', \Vdash_A)$ is a model";
the interpretation of t in that model witnesses that $[\![t]\!]_Q \Vdash_A [\![t]\!]'$
(categorically: *gluing* of two CCCs – later if time allows)

Purpose here: relate recognition by $[\![-]\!]_Q$ and by $[\![-]\!]'$

Partial surjections

$\Vdash_A \subseteq \underbrace{[\![A]\!]_Q}_{\textbf{FinSet}} \times \underbrace{[\![A]\!]'}_{\text{some other model}}$ defined inductively from $\Vdash_o$ ...

partial *function*: $\forall x,\ |\{y \mid x \Vdash_A y\}| \leqslant 1$

surjective relation: $\forall y,\ |\{x \mid x \Vdash_A y\}| \geqslant 1$

Classical fact

Suppose that $[\![-]\!]'$ is an *extensional* model. If $\Vdash_o$ is a *partial surjection*, so is $\Vdash_A$ for all A.

Partial surjections

$\Vdash_A \subseteq \underbrace{[\![A]\!]_Q}_{\textbf{FinSet}} \times \underbrace{[\![A]\!]'}_{\text{some other model}}$ defined inductively from $\Vdash_o$ ...

partial *function*: $\forall x,\ |\{y \mid x \Vdash_A y\}| \leqslant 1$

surjective relation: $\forall y,\ |\{x \mid x \Vdash_A y\}| \geqslant 1$

Classical fact

Suppose that $[\![-]\!]'$ is an *extensional* model. If $\Vdash_o$ is a *partial surjection*, so is $\Vdash_A$ for all A.

Proof by induction: suppose $\Vdash_A$ and $\Vdash_B$ are partial surjections.

Proof that $\Vdash_{A\to B}$ is a partial function.

Suppose $f \Vdash_{A\to B} g$. Let $y \in [\![A]\!]'$.

$\Vdash_A$ is surjective so $\exists x.\ x \Vdash_A y$. By definition, $f(x) \Vdash_B g(y)$.

$\Vdash_B$ is a partial function so $f(x)$ determines $g(y)$. When y varies, f determines g. $\square$

Note: the final argument uses *extensionality*!

Partial surjections

$\Vdash_A \subseteq \underbrace{[\![A]\!]_Q}_{\textbf{FinSet}} \times \underbrace{[\![A]\!]'}_{\text{some other model}}$ defined inductively from $\Vdash_o$ ...

partial *function*: $\forall x,\ |\{y \mid x \Vdash_A y\}| \leqslant 1$

surjective relation: $\forall y,\ |\{x \mid x \Vdash_A y\}| \geqslant 1$

Classical fact

Suppose that $[\![-]\!]'$ is an *extensional* model. If $\Vdash_o$ is a *partial surjection*, so is $\Vdash_A$ for all A.

Proof by induction: suppose $\Vdash_A$ and $\Vdash_B$ are partial surjections.

Proof that $\Vdash_{A \to B}$ is a surjection.

Let $g \in [\![A \to B]\!]'$. Let $x \in [\![A]\!]_Q$.

- If $x \Vdash_A y$ then y unique ($\Vdash_A$ partial function): choose $f(x) \Vdash_B g(y)$ ($\Vdash_B$ surjective)
- Otherwise, if no such y, choose $f(x)$ arbitrary $\square$

Note: this part of the argument requires having **FinSet** on the left of $\Vdash$

(any collection of choices of $f(x)$ can be glued into a function f)

Recognizable by finite extensional model $\Longrightarrow$ FinSet-recognizable

$$\Vdash_A \subseteq \underbrace{[\![A]\!]_Q}_{\textbf{FinSet}} \times \underbrace{[\![A]\!]'}_{\text{some other model}} \text{ defined inductively from } \Vdash_o = \underbrace{\text{equality}}_{\text{a bijective relation}} \text{ on the set } [\![o]\!]_Q = \underbrace{Q = [\![o]\!]'}_{\text{assumption: } [\![o]\!]' \text{ finite}}$$

Classical fact (previous slide)

In this case, $\Vdash_A$ is a partial surjection for all A.

Recognizable by finite extensional model $\implies$ FinSet-recognizable

$\Vdash_A \subseteq \underbrace{[\![A]\!]_Q}_{\textbf{FinSet}} \times \underbrace{[\![A]\!]'}_{\text{some other model}}$ defined inductively from $\Vdash_o = \underbrace{\text{equality}}_{\text{a bijective relation}}$ on the set $[\![o]\!]_Q = \underbrace{Q = [\![o]\!]'}_{\text{assumption: } [\![o]\!]' \text{ finite}}$

Classical fact (previous slide)

In this case, $\Vdash_A$ is a partial surjection for all A.

Let L be a $[\![-]\!]'$-recognizable language of λ-terms of type A, i.e. $t \in L \iff [\![t]\!]' \in P \subseteq [\![A]\!]'$.

Fundamental lemma: $[\![t]\!]_Q \Vdash_A [\![t]\!]'$.

Since $\Vdash_A$ is a partial function, $t \in L \iff \exists y.\ ([\![t]\!]_Q \Vdash_A y) \wedge (y \in P)$

condition purely on $[\![t]\!]_Q$: **FinSet**-recognizable!

A logical relation between a syntactic model and FinSet (1)

Key observation

"Type-casting" $t : A \rightsquigarrow t[B] : A[B]$ is the interpretation in a syntactic model with $o \mapsto B$.

(non-extensional and non-finitary model!)

Let $\mathsf{Fin}(n) = \overbrace{o \to \cdots \to o}^{n \text{ times}} \to o$ and $\Lambda(A) = \{t \mid t : A\}/(=_{\beta\eta})$.

$\Vdash^n_A \subseteq \overbrace{\Lambda(A[\mathsf{Fin}(n)])}^{\text{syntactic model}} \times \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}}$ defined inductively for $Q = \{1, \ldots, n\}$:

$$t \Vdash^n_o q \iff t =_{\beta\eta} \lambda x_1. \ \ldots \ \lambda x_n. x_q \quad \text{(bijective encoding)}$$

$$t \Vdash^n_{A \to B} f \iff \forall (u, x),\ u \Vdash^n_A x \Rightarrow t\, u \Vdash^n_B f(x)$$

A logical relation between a syntactic model and FinSet (1)

Key observation

"Type-casting" $t : A \rightsquigarrow t[B] : A[B]$ is the interpretation in a syntactic model with $o \mapsto B$.

(non-extensional and non-finitary model!)

Let $\mathsf{Fin}(n) = \overbrace{o \to \cdots \to o}^{n \text{ times}} \to o$ and $\Lambda(A) = \{t \mid t : A\}/(=_{\beta\eta})$.

$\Vdash^n_A \subseteq \overbrace{\Lambda(A[\mathsf{Fin}(n)])}^{\text{syntactic model}} \times \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}}$ defined inductively for $Q = \{1, \ldots, n\}$:

$$t \Vdash^n_o q \iff t =_{\beta\eta} \lambda x_1. \ldots \lambda x_n. x_q \quad \text{(bijective encoding)}$$

$$t \Vdash^n_{A \to B} f \iff \forall (u, x),\ u \Vdash^n_A x \Rightarrow t\,u \Vdash^n_B f(x)$$

Rough idea

Establish that $\Vdash^n_A$ is (a bit more than) a partial surjection for all A

$\Longrightarrow$ (as before) the syntactic model recognizes $[\![-]\!]_Q$-recognizable languages

$\Longrightarrow$ (with some work) such languages syntactically regular: definable at $A[\mathsf{Fin}(n)] \to \mathsf{Bool}$

A logical relation between a syntactic model and FinSet (2)

Let $\mathsf{Fin}(n) = o \to \cdots \to o \to o$ and $\Lambda(A) = \{t \mid t : A\}/(=_{\beta\eta})$.

$\Vdash^n_A \subseteq \overbrace{\Lambda(A[\mathsf{Fin}(n)])}^{\text{syntactic model}} \times \overbrace{[\![A]\!]_Q}^{\textbf{FinSet}}$ def. inductively for $Q = \{1, \dots, n\}$: case $A \to B$ as expected and

$$t \Vdash^n_o q \iff t =_{\beta\eta} \lambda x_1. \ \dots \ \lambda x_n. x_q$$

Key lemma

For all A, there are $u_A : \mathsf{Fin}(|\,[\![A]\!]_Q\,|) \to A[\mathsf{Fin}(n)]$ & $v_A : A[\mathsf{Fin}(n)] \to \mathsf{Fin}(|\,[\![A]\!]_Q\,|)$ such that

$$s \Vdash_o^{|[\![A]\!]_Q|} i \implies u_A\, s \Vdash^n_A i \qquad\qquad t \Vdash^n_A j \implies v_A\, t \vdash_o^{|[\![A]\!]_Q|} j$$

where we identify $[\![A]\!]_Q$ with $\{1, \dots, |\,[\![A]\!]_Q\,|\}$

- u_A = "λ-definable surjectivity" of $\Vdash^n_A$
- v_A = "λ-definable partial functionality" of $\Vdash^n_A$

A logical relation between a syntactic model and FinSet (3)

Key lemma

For all A, there are $u_A : \mathsf{Fin}(|\llbracket A \rrbracket_Q|) \to A[\mathsf{Fin}(n)]$ & $v_A : A[\mathsf{Fin}(n)] \to \mathsf{Fin}(|\llbracket A \rrbracket_Q|)$ such that

$$s \Vdash_o^{|\llbracket A \rrbracket_Q|} i \implies u_A\, s \Vdash_A^n i \qquad\qquad t \Vdash_A^n j \implies v_A\, t \vdash_o^{|\llbracket A \rrbracket_Q|} j$$

where we identify $i \in \{1, \ldots, |\llbracket A \rrbracket_Q|\}$ with $i \in \llbracket A \rrbracket_Q$ (recall $|Q| = n$)

Let $\mathsf{app}_{A,B} : \mathsf{Fin}(|\llbracket A \to B \rrbracket_Q|) \to \mathsf{Fin}(|\llbracket A \rrbracket_Q|) \to \mathsf{Fin}(|\llbracket B \rrbracket_Q|)$ (definable by case analysis) correspond to the application map $\llbracket A \to B \rrbracket_Q \times \llbracket A \rrbracket_Q \to \llbracket B \rrbracket_Q$

$$u_{A \to B} = \lambda(x : \mathsf{Fin}(|\llbracket A \to B \rrbracket_Q|)).\, \lambda(y : A[\mathsf{Fin}(n)]).\, u_B\, (\mathsf{app}_{A,B}\, x\, (v_A\, y))$$

A logical relation between a syntactic model and FinSet (3)

Key lemma

For all A, there are $u_A : \mathsf{Fin}(|\,[\![A]\!]_Q\,|) \to A[\mathsf{Fin}(n)]$ & $v_A : A[\mathsf{Fin}(n)] \to \mathsf{Fin}(|\,[\![A]\!]_Q\,|)$ such that

$$s \Vdash_o^{|[\![A]\!]_Q|} i \implies u_A\, s \Vdash_A^n i \qquad\qquad t \Vdash_A^n j \implies v_A\, t \vdash_o^{|[\![A]\!]_Q|} j$$

where we identify $i \in \{1, \dots, |\,[\![A]\!]_Q\,|\}$ with $i \in [\![A]\!]_Q$ (recall $|Q| = n$)

Let $\mathsf{app}_{A,B} : \mathsf{Fin}(|\,[\![A \to B]\!]_Q\,|) \to \mathsf{Fin}(|\,[\![A]\!]_Q\,|) \to \mathsf{Fin}(|\,[\![B]\!]_Q\,|)$ (definable by case analysis) correspond to the application map $[\![A \to B]\!]_Q \times [\![A]\!]_Q \to [\![B]\!]_Q$

$$u_{A\to B} = \lambda(x : \mathsf{Fin}(|\,[\![A \to B]\!]_Q\,|)).\, \lambda(y : A[\mathsf{Fin}(n)]).\, u_B\,(\mathsf{app}_{A,B}\, x\,(v_A\, y))$$

Similarly, define $v_{A\to B}$ from $(\mathsf{Fin}(|\,[\![A]\!]_Q\,|) \to \mathsf{Fin}(|\,[\![B]\!]_Q\,|)) \to \mathsf{Fin}(|\,[\![A \to B]\!]_Q\,|)$
(evaluate argument on all inhabitants of $\mathsf{Fin}(|\,[\![A]\!]_Q\,|$ ...)

Towards squeezing

Key structure here: λ-terms of respective types

$$\mathsf{app}_{n,m} : \mathsf{Fin}(m^n) \to (\mathsf{Fin}(n) \to \mathsf{Fin}(m)) \qquad\qquad (\mathsf{Fin}(n) \to \mathsf{Fin}(m)) \to \mathsf{Fin}(m^n)$$

compatible with the logical relation: $s \Vdash_o^{m^n} i \implies (\mathsf{app}_{n,m}\, s)\ [\Vdash_o^n \to \Vdash_o^m]\ i$

Similarly for a finite extensional model $[\![-]\!]'$ before: implicitly uses set-theoretic maps

$$[\![A \to B]\!]' \to ([\![A]\!]' \to [\![B]\!]') \qquad\qquad \underbrace{([\![A]\!]' \to [\![B]\!]') \to [\![A \to B]\!]'}_{\textit{extensionality} + \text{dummy values}}$$

$\to$ generalize common pattern: *squeezing*

$\to$ Vincent Moreau's slides