Introduction and Comparison of Different Approaches to Initial Semantics

Thomas Lamiaux, Benedikt Ahrens

DutchCat, Utrecht, March 2023

Table of contents

Introduction to Initial Semantics

Introduction to Initial Semantics

Motivations

The Objectives of Initial Semantics

Issues with high-order languages

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables: $\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + \mathbf{2})$

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\twoheadrightarrow \lambda y.\, (y + y + y)$

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\twoheadrightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \nrightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

Objectives of Initial Semantics

Provide a mathematical framework to:

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables: $\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\rightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

Objectives of Initial Semantics

Provide a mathematical framework to:

- Deal with variable binding and α-equivalence

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\rightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

Objectives of Initial Semantics

Provide a mathematical framework to:

- Deal with variable binding and α-equivalence
- Have a recursion principle for the language

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\rightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

Objectives of Initial Semantics

Provide a mathematical framework to:

- Deal with variable binding and α-equivalence
- Have a recursion principle for the language
- Characterize the language with its substitution

The Objectives of Initial Semantics

Issues with high-order languages

- Semantic is partially independent of names: $\lambda x.\, x$ vs $\lambda y.\, y$
- Shadowing of variables:$\lambda \mathbf{x}.\, \mathbf{x}(\lambda \mathbf{x}.\, \mathbf{x} + 2)$
- Capture of variable: $(\lambda x.\, \lambda y.\, (x + y))(y + y) \not\rightarrow \lambda y.\, (y + y + y)$

It implies working up to α-equivalence. Not that easy !

Objectives of Initial Semantics

Provide a mathematical framework to:

- Deal with variable binding and α-equivalence
- Have a recursion principle for the language
- Characterize the language with its substitution
- Characterize the commutation of constructors and substitution

Introduction to Initial Semantics

Principles of Initial Semantics

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

- $X : \mathrm{Set}$ are variable names, $\Lambda(X) : \mathrm{Set}$ the lambda terms over it.

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

- $X : \mathrm{Set}$ are variable names, $\Lambda(X) : \mathrm{Set}$ the lambda terms over it.
- Given $f : X \to Y$, $\Lambda(f) : \Lambda(X) \to \Lambda(Y)$ is variable renaming.

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

- $X : \mathrm{Set}$ are variable names, $\Lambda(X) : \mathrm{Set}$ the lambda terms over it.
- Given $f : X \to Y$, $\Lambda(f) : \Lambda(X) \to \Lambda(Y)$ is variable renaming.

Constructors are natural transformations

Constructors, including variable binding, are natural transformation:

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

- $X : \mathrm{Set}$ are variable names, $\Lambda(X) : \mathrm{Set}$ the lambda terms over it.
- Given $f : X \to Y$, $\Lambda(f) : \Lambda(X) \to \Lambda(Y)$ is variable renaming.

Constructors are natural transformations

Constructors, including variable binding, are natural transformation:

- Variable binding is $\mathrm{abs}_X : \Lambda(X + 1) \to \Lambda(X)$

Variable Binding and Renaming

The lambda calculus is presheaf

The lambda calculus is a functor $\Lambda : \mathrm{Set} \to \mathrm{Set}$:

- $X : \mathrm{Set}$ are variable names, $\Lambda(X) : \mathrm{Set}$ the lambda terms over it.
- Given $f : X \to Y$, $\Lambda(f) : \Lambda(X) \to \Lambda(Y)$ is variable renaming.

Constructors are natural transformations

Constructors, including variable binding, are natural transformation:

- Variable binding is $\mathrm{abs}_X : \Lambda(X+1) \to \Lambda(X)$
- Naturality specify commutation with renaming:

$$
\begin{tikzcd}
\Lambda(X+1) \arrow[r, "\mathrm{abs}_X"] \arrow[d, "\Lambda(f+1)"'] & \Lambda(X) \arrow[d, "\Lambda(f)"] \\
\Lambda(Y+1) \arrow[r, "\mathrm{abs}_Y"'] & \Lambda(X)
\end{tikzcd}
$$

A Recursion Principle

The lambda calculus is an initial algebra

- The lambda calculus is an **initial** algebra on $\mathrm{Set} \to \mathrm{Set}$ for the functor $\mathcal{H} : F \mapsto \mathrm{Id} + F \times F + F \circ (X \mapsto X + 1)$:

$$\begin{tikzcd}
X + \Lambda(X) \times \Lambda(X) + \Lambda(X+1) \arrow[r, "\mathrm{var}+\mathrm{app}+\mathrm{abs}"] \arrow[d, "h+\Lambda(h)\times\Lambda(h)+\Lambda(h+1)"'] & \Lambda(X) \arrow[d, "\exists! h"] \\
X + B(X) \times B(X) + B(X+1) \arrow[r, "b+b'+b''"'] & B
\end{tikzcd}$$

A Recursion Principle

The lambda calculus is an initial algebra

- The lambda calculus is an **initial** algebra on $\mathrm{Set} \to \mathrm{Set}$ for the functor $\mathcal{H} : F \mapsto \mathrm{Id} + F \times F + F \circ (X \mapsto X + 1)$:

$$\begin{tikzcd}
X + \Lambda(X) \times \Lambda(X) + \Lambda(X+1) \arrow[r, "\mathrm{var}+\mathrm{app}+\mathrm{abs}"] \arrow[d, "h+\Lambda(h)\times\Lambda(h)+\Lambda(h+1)"'] & \Lambda(X) \arrow[d, "\exists! h"] \\
X + B(X) \times B(X) + B(X+1) \arrow[r, "b+b'+b''"'] & B
\end{tikzcd}$$

A recursion principle

- Building an algebra is exactly building a map to B by recursion

The Subtitution Structure

The lamda calculus is a Monad

- The lambda calculus forms a monad on Set for η the variable constructor and σ the simultaneous substitution.

The Subtitution Structure

The lamda calculus is a Monad

- The lambda calculus forms a monad on Set for η the variable constructor and σ the simultaneous substitution.

Monad on a category $\mathcal{C}$

A monad is a triple (T, η, σ) where:

- $T : \mathcal{C} \to \mathcal{C}$ is a functor
- $\eta_X : X \to T(X)$ and $\sigma_{X,Y} : (X \to T(Y)) \to (T(X) \to T(Y))$ are natural transformations verifying:

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & T(X) \arrow[d, "\sigma(f)"] \\
& T(Y)
\end{tikzcd}
\qquad
\begin{tikzcd}
T(X) \arrow[dr, bend left, "\sigma(\eta_X)"] \arrow[dr, bend right, "id"'] & \\
& T(X)
\end{tikzcd}
\qquad
\begin{tikzcd}
T(X) \arrow[r, "\sigma(f)"] \arrow[dr, "\sigma(\sigma(g)\circ f)"'] & T(Y) \arrow[d, "\sigma(g)"] \\
& T(Z)
\end{tikzcd}
$$

Model Substitution of Constructors

Subtitution of Constructors

Given $f : X \to \Lambda(Y)$ how to model commutation of constructors with substitution $\sigma(f) : T(X) \to T(Y)$ for

$$\begin{tikzcd}
\Lambda(X+1) \arrow[r, "\mathrm{abs}_X"] \arrow[d, "?"'] & \Lambda(X) \arrow[d, "\sigma(f)"] \\
\Lambda(Y+1) \arrow[r, "\mathrm{abs}_Y"'] & \Lambda(Y)
\end{tikzcd}$$

Model Substitution of Constructors

Subtitution of Constructors

Given $f : X \to \Lambda(Y)$ how to model commutation of constructors with substitution $\sigma(f) : T(X) \to T(Y)$ for

$$\begin{tikzcd}
\Lambda(X+1) \arrow[r, "\mathrm{abs}_X"] \arrow[d, "?"'] & \Lambda(X) \arrow[d, "\sigma(f)"] \\
\Lambda(Y+1) \arrow[r, "\mathrm{abs}_Y"'] & \Lambda(Y)
\end{tikzcd}$$

Some solution

- Using modules
- Using a notion of strength

The principle of initial semantics

The General Method

1. Define a notion of signatures
2. Build a category of models for signatures:
   - An algebra structure on a presheaf category
   - With a mathematical structure for substitution
   - And a structure expressing commutation of constructors with substitution
3. Identity a class of signatures that always have an initial model

Unifying the Different Traditions

Unifying the Different Traditions

Different Traditions to Initial Semantics

Traditions of Untyped Initial Semantics

Some basic papers on the subject:

1999 Altenkirch and Reus, "Monadic Presentations of Lambda Terms Using Generalized Inductive Types"

1999 Fiore, Plotkin, and Turi, "Abstract Syntax and Variable Binding"

2003 Matthes and Uustalu, "Substitution in non-wellfounded syntax with variable binding"

2007 Hirschowitz and Maggesi, "Modules over Monads and Linearity"

2010 Zsido, "Typed Abstract Syntax"

2010 Hirschowitz and Maggesi, "Modules over monads and initial semantics"

2012 Hirschowitz and Maggesi, "Initial Semantics for Strengthened Signatures"

2015 Ahrens and Matthes, "Heterogeneous Substitution Systems Revisited"

2018 Ahrens et al., "High-Level Signatures and Initial Semantics"

Traditions of Untyped Initial Semantics

Year	Signatures	Category	Model	Initiality	Proofs
1999	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads	No	Yes
1999	Algebraic	$\mathbb{F} \to \mathrm{Set}$	Monoids + Strength	Yes	No
2004	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	No / No	Yes
2007	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2010	Comparing models of $\mathbb{F} \to \mathrm{Set}$ vs $\mathrm{Set} \to \mathrm{Set}$				
2010	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes
2012	Strength	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2015	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	Yes / No	Yes
2018	Presentable	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes

Traditions of Untyped Initial Semantics

Year	Signatures	Category	Model	Initiality	Proofs
1999	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads	No	Yes
1999	Algebraic	$\mathbb{F} \to \mathrm{Set}$	Monoids + Strength	Yes	No
2004	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	No / No	Yes
2007	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2010	Comparing models of $\mathbb{F} \to \mathrm{Set}$ vs $\mathrm{Set} \to \mathrm{Set}$				
2010	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes
2012	Strength	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2015	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	Yes / No	Yes
2018	Presentable	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes

Challenges

- Different notions of Signatures, Categories and Models !

Traditions of Untyped Initial Semantics

Year	Signatures	Category	Model	Initiality	Proofs
1999	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads	No	Yes
1999	Algebraic	$\mathbb{F} \to \mathrm{Set}$	Monoids + Strength	Yes	No
2004	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	No / No	Yes
2007	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2010	Comparing models of $\mathbb{F} \to \mathrm{Set}$ vs $\mathrm{Set} \to \mathrm{Set}$				
2010	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes
2012	Strength	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2015	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	Yes / No	Yes
2018	Presentable	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes

Challenges

- Different notions of Signatures, Categories and Models !
- Often no initiality or no proofs !

Unifying the Different Traditions

Different Signatures

Algebraic Signatures

Algebraic Signatures

- Elementary signatures are lists of $\mathbb{N}$, $[n_1, ..., n_k]$ also written as $\Theta^{(n_1)} \times ... \times \Theta^{(n_k)}$
- Algebraic Signatures are list / coproduct of elementary signatures.

Algebraic Signatures

Algebraic Signatures

- Elementary signatures are lists of $\mathbb{N}$, $[n_1, ..., n_k]$ also written as $\Theta^{(n_1)} \times ... \times \Theta^{(n_k)}$
- Algebraic Signatures are list / coproduct of elementary signatures.

Examples

- app can be seen as $[0, 0]$ or $\Theta \times \Theta$
- abs can be seen as $[1]$ or $\Theta^{(1)}$
- The lambda calculus is $[[0, 0], [1]]$ or $\Theta \times \Theta + \Theta^{(1)}$

Algebraic Signatures

Algebraic Signatures

- Elementary signatures are lists of $\mathbb{N}$, $[n_1, ..., n_k]$ also written as $\Theta^{(n_1)} \times ... \times \Theta^{(n_k)}$
- Algebraic Signatures are list / coproduct of elementary signatures.

Examples

- app can be seen as $[0, 0]$ or $\Theta \times \Theta$
- abs can be seen as $[1]$ or $\Theta^{(1)}$
- The lambda calculus is $[[0, 0], [1]]$ or $\Theta \times \Theta + \Theta^{(1)}$

Typed ?

- Those signatures do not raise as such to the typed case

Signatures with strength

Signatures with Strength

Signatures with strength are tuples (H, θ) such that :

- $H : [\mathrm{Set}, \mathrm{Set}] \to [\mathrm{Set}, \mathrm{Set}]$
- Θ is natural transformation, such that for all $A, B : \mathrm{Set} \to \mathrm{Set}$ and $b : \mathrm{Id} \Rightarrow B$:

$$\theta_{A,b} : H(A) \circ B \to H(A \circ B)$$

Signatures with strength

Signatures with Strength

Signatures with strength are tuples (H, θ) such that :

- $H : [\mathrm{Set}, \mathrm{Set}] \to [\mathrm{Set}, \mathrm{Set}]$
- Θ is natural transformation, such that for all $A, B : \mathrm{Set} \to \mathrm{Set}$ and $b : \mathrm{Id} \Rightarrow B$:

$$\theta_{A,b} : H(A) \circ B \to H(A \circ B)$$

Examples

- Algebraic signatures "are" signatures with strength

Signatures with strength

Signatures with Strength

Signatures with strength are tuples (H, θ) such that :

- $H : [\mathrm{Set}, \mathrm{Set}] \to [\mathrm{Set}, \mathrm{Set}]$
- Θ is natural transformation, such that for all $A, B : \mathrm{Set} \to \mathrm{Set}$ and $b : \mathrm{Id} \Rightarrow B$:

$$\theta_{A,b} : H(A) \circ B \to H(A \circ B)$$

Examples

- Algebraic signatures "are" signatures with strength
- $T \mapsto T \circ T$ is a signatures with strength

Presentable Signatures

Presentable Signatures

- Presentable signatures are signatures Σ such that there is an algebraic signatures Υ and an epimorphism of signatures:

$$\Upsilon \twoheadrightarrow \Sigma$$

Presentable Signatures

Presentable Signatures

- Presentable signatures are signatures Σ such that there is an algebraic signatures Υ and an epimorphism of signatures:

$$\Upsilon \twoheadrightarrow \Sigma$$

Examples

- Adding a commutative binary operator
- Adding a coherent fixpoint operator

Unifying the Different Traditions

Our Proposition

Building Models

Monoidal Category

Work directly on Monoidal Category:

- $\mathbb{F} \to \textit{Set}$, $\mathrm{Set} \to \mathrm{Set}$, $\mathcal{C} \to \mathcal{C}$

Building Models

Monoidal Category

Work directly on Monoidal Category:

- $\mathbb{F} \to \mathit{Set}$, $\mathrm{Set} \to \mathrm{Set}$, $\mathcal{C} \to \mathcal{C}$

Connecting Signatures

- Define general notion of signatures
- Define substitution with strength
- Show that signatures with strength are signatures
- Show that algebraic signatures are a subclass of signatures with strength.

Connecting the Frameworks

Building Models

- Define monoids
- Define modules over monoids
- Define models as monoid + module morphism

Connecting the Frameworks

Building Models

- Define monoids
- Define modules over monoids
- Define models as monoid + module morphism

Connecting Models

- Show that Monoid + Stength = Monads + Module
- Show that there is an initial hss
- We can use the hss to build a model structure
- We can prove this model is initial

Connecting $\mathbb{F} \to \mathrm{Set}$ and $\mathrm{Set} \to \mathrm{Set}$

Zsido phd

- **2010**, Zsido, "Typed Abstract Syntax"

Connecting $\mathbb{F} \to \mathrm{Set}$ and $\mathrm{Set} \to \mathrm{Set}$

Zsido phd

- **2010**, Zsido, "Typed Abstract Syntax"

Zsido Theorem

- For algebraic signatures, we can build the initial model of $\mathbb{F} \to \mathrm{Set}$ and $\mathrm{Set} \to \mathrm{Set}$ and vice versa

Traditions of Untyped Initial Semantics

Year	Signatures	Category	Model	Initiality	Proofs
1999	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads	No	Yes
1999	Algebraic	$\mathbb{F} \to \mathrm{Set}$	Monoids + Strength	Yes	No
2004	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	No / No	Yes
2007	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2010	Comparing models of $\mathbb{F} \to \mathrm{Set}$ vs $\mathrm{Set} \to \mathrm{Set}$				
2010	Algebraic	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes
2012	Strength	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	No
2015	Strength	$\mathcal{C} \to \mathcal{C}$	Hss / Monads	Yes / No	Yes
2018	Presentable	$\mathrm{Set} \to \mathrm{Set}$	Monads + Module	Yes	Yes

What is left to do ?

Some general goals

- Write it down, explain it and making it accessible
- Extend Zsido's phd for signatures with strength
- Look at presentable signatures
- Look at signatures with equations
- Adding reduction rules ?
- Simply typed languages ?

Let's eat !