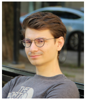
Initial Semantics for Polymorphic Type Systems

Benedikt Ahrens

jww Ambroise Lafont  and Thomas Lamiaux 

EuroProofNet WG6 Meeting 2024

2024-04-04

Summary

Goals

1. Develop a notion of signature, and of model of a signature, for **polymorphic** type systems.
   - $\rightsquigarrow$ category of models
2. Find sufficient conditions for a signature to have initial model
   - $\rightsquigarrow$ the **syntax** generated by the signature

Motivation

- Obtain recursion principle from initiality
- Specify well-behaved translations between languages

Outline

Overview of Initial Semantics

Signatures

Language constructors are specified by a notion of signature

Models of Signatures

- Substitution is modelled by **monoid** structure in suitable monoidal category
- Constructors are modelled by some notion of **algebra**
- Interplay between constructors and substitution governed by some mathematical structure

Construction of Syntax

- Syntax is constructed via a suitable colimit construction
- Substitution is constructed via (categorical) recursion scheme

Functorial syntax — syntax with explicit contexts

```
Inductive LC (X : Set) : Set
| Var : X -> LC X
| App : LC X -> LC X -> LC X
| Abs : LC (X + 1) -> LC X
```

- Well-scoped lambda terms as a functor

$$\mathsf{LC} : \mathsf{Set} \to \mathsf{Set}$$
$$\mathsf{LC}(\Gamma) := \{\text{set of lambda terms in context } \Gamma\}$$

- Constructors are natural transformations

$$\mathsf{App} : \mathsf{LC} \times \mathsf{LC} \to \mathsf{LC}$$
$$\mathsf{Abs} : \mathsf{LC}^* \to \mathsf{LC}$$

with $\mathsf{LC}^*(X) := \mathsf{LC}(X + 1)$

Simply-Typed Syntax

- Fix a set T of types, e.g., for STLC:

$$T ::= \; B \; \mid \; T_1 \Rightarrow T_2$$

- Simply-typed syntax with set T of types:

$$\mathsf{STLC} : \mathsf{Set}^T \to \mathsf{Set}^T$$

with constructors

$$\mathsf{App}_{s,t} : \mathsf{STLC}_{s \Rightarrow t} \to \mathsf{STLC}_s \to \mathsf{STLC}_t$$
$$\mathsf{Abs}_{s,t} : \mathsf{STLC}^s_t \to \mathsf{STLC}_{s \Rightarrow t}$$

Monads/Monoids for Substitution

- Variables embed into terms

$$\mathsf{Var} : \Gamma \to T(\Gamma)$$

- Substitution

$$\mathsf{subst} : (\Delta \to T(\Gamma)) \to (T(\Delta) \to T(\Gamma))$$

  or

$$\mu : (T \circ T)(\Gamma) \to T(\Gamma)$$

  gives structure of monad to T

- Besides $([\mathsf{Set}, \mathsf{Set}], \circ)$, can consider different monoidal categories and monoids therein, e.g., $([\mathbb{F}, \mathsf{Set}], \circ)$, $([\mathsf{Set}^T, \mathsf{Set}^T], \circ)$

In short

Syntax with substitution is monoid in a suitable monoidal category

Constructors and Interplay with Substitution

- Language constructors commute, in a suitable sense, with substitution, e.g.,

$$\mathsf{subst}(f)(\mathsf{App}(M,N)) = \mathsf{App}(\mathsf{subst}(f)(M), \mathsf{subst}(f)(N))$$
$$\mathsf{subst}(f)(\mathsf{Abs}(M)) = \mathsf{Abs}(\mathsf{subst}(\uparrow f)(M))$$

- Expressed by saying that

$$\mathsf{App} : \mathsf{LC} \times \mathsf{LC} \to \mathsf{LC}$$
$$\mathsf{Abs} : \mathsf{LC}^* \to \mathsf{LC}$$

are **morphisms of modules**

In short

Module morphisms = natural transformations + commutativity with substitution

Signatures and Models

Definition (Signature Σ and Model of Σ)

$$\begin{array}{c} \int_{T:\mathsf{Monad}} \mathsf{Module}(T) \\ \downarrow \; \uparrow \Sigma \\ \mathsf{Monad} \end{array}$$

A model M of Σ is a monad T and a T-module morphism

$$\Sigma(T) \xrightarrow{M} T$$

Example (Lambda calculus)

$$\Sigma_{\mathsf{LC}} : M \mapsto M \times M + M^*$$

A model of Σ_{LC} is a monad M together with two module morphisms

$$\mathsf{App} : M \times M \to M$$
$$\mathsf{Abs} : M^* \to M$$

Initial Semantics and Translations

Definition (Syntax generated by a signature)

The **syntax** generated by Σ is the initial model, if it exists.

- Not all signatures admit a syntax
- Suitable subcategories of signatures that do admit syntax can be identified

Well-behaved translations via initiality

- Translation from a language S to another T can be specified by equipping T with the structure of model for S
- Resulting translation commutes with substitution by construction

- Extensions to include equations between terms and reductions (operational semantics)

Outline

System F

Types of System F

$$\tau ::= \; x \mid B \mid \tau_1 \Rightarrow \tau_2 \mid \forall x.\tau$$

or, in terms of an untyped signature as before,

$$M \mapsto B + M \times M + M^*$$

Terms of System F

$$t ::= \; x \mid \lambda x.t \mid t_1.t_2 \mid \Lambda \alpha.t \mid t.\sigma$$

Syntax as a Functor, à la Hamana

Types

Have a(n untyped) language $T : [\mathbb{F}, \mathsf{Set}] \to [\mathbb{F}, \mathsf{Set}]$ for types

Define

$$G : \mathbb{F} \to \mathsf{Cat}$$
$$G(n) := (\mathbb{F} \downarrow T(n)) \times T(n)$$

The category $\int G$ has

- objects $n \,|\, \Gamma \vdash \tau$, with $\Gamma \in \mathbb{F} \downarrow T(n)$ and $\tau \in T(n)$
- arrows (ρ, π) with
  - $\rho : m \to n$ such that $T(\rho)(\tau) = \sigma$ and
  - $\pi : (\mathbb{F} \downarrow T(\rho))(\Gamma) \to \Delta$ in $\mathbb{F} \downarrow T(n)$.

Terms

Terms are given by a functor $\int G \to \mathsf{Set}$

A More Intuitive(?) View on Syntax

- Replace $\mathbb{F}$ by Set
-

$$
\begin{aligned}
&[\int^{n} \mathsf{Set} \downarrow T(n) \times T(n), \mathsf{Set}] \\
&\simeq \int_{n} [\mathsf{Set} \downarrow T(n) \times T(n), \mathsf{Set}] \\
&\simeq \int_{n} [\mathsf{Set}^{T(n)}, \mathsf{Set}^{T(n)}]
\end{aligned}
$$

- This category has a simple, “point-wise” monoidal product

The signature of System F

Models of System F

- Take $T := T_F : \mathsf{Set} \to \mathsf{Set}$ to be initial monad generated by the signature for types.
- Models of System F are monoids in $\oint_n [\mathsf{Set}^{T(n)}, \mathsf{Set}^{T(n)}]$ + some module morphisms for constructors

Signature of System F is, as before, a section to a forgetful functor

$$\int_{T:\mathsf{Monoid}} \mathsf{Module}(T)$$

$$\downarrow \; \uparrow \Sigma$$

$$\mathsf{Monoid}(\oint_n [\mathsf{Set}^{T(n)}, \mathsf{Set}^{T(n)}])$$

obtained as the sum of several signatures.

Signatures for Constructors of System F

$$\frac{n+1 \,|\, \mathsf{wk}(\Gamma) \vdash t : A}{n \,|\, \Gamma \vdash \Lambda t : \forall A}$$

is specified by the module

$$\Lambda(M) := \quad \mathsf{Set}^{T(n)} \xrightarrow{\mathsf{Lan(wk)}} \mathsf{Set}^{T(n+1)} \xrightarrow{M_{n+1}} \mathsf{Set}^{T(n+1)} \xrightarrow{\mathsf{Lan}(\Lambda)} \mathsf{Set}^{T(n)}$$

$$\frac{n \,|\, \Gamma \vdash t : \forall \tau \qquad n \,|\, \Gamma \vdash A}{n \,|\, \Gamma \vdash t.A : \tau[A]}$$

is specified by the module

$$\mathsf{App}(M) := \quad \mathsf{Set}^{T(n)} \xrightarrow{M_n} \mathsf{Set}^{T(n)} \xrightarrow{\mathsf{Res}_p} \mathsf{Set}^{T(n+1)\times T(n)} \xrightarrow{\mathsf{Lan(subst)}} \mathsf{Set}^{T(n)}$$

Signature and Models for System F

Definition

$$\int_{T:\mathsf{Monoid}} \mathsf{Module}(T)$$

$$\Big\downarrow \quad \Big\uparrow \; \Lambda + \mathsf{App} + \ldots$$

$$\mathsf{Monoid}\left(\int_n [\mathsf{Set}^{T(n)}, \mathsf{Set}^{T(n)}]\right)$$

Definition

A model of System F in a monoid M is a module morphism

$$\Lambda(M) + \mathsf{App}(M) + \ldots \to M$$

Conclusion

- Work in progress on a novel approach towards initial semantics for polymorphic type systems
- Advantages (?) compared to Hamana's approach:
  - simpler monoidal structure
  - easier to formalize (cf. Dima's talk this morning)
- Next steps:
  - Study general signatures; in particular, identify sufficient criteria for a signature to yield a syntax
  - More complicated systems such as System Fω.

Conclusion

- Work in progress on a novel approach towards initial semantics for polymorphic type systems
- Advantages (?) compared to Hamana's approach:
  - simpler monoidal structure
  - easier to formalize (cf. Dima's talk this morning)
- Next steps:
  - Study general signatures; in particular, identify sufficient criteria for a signature to yield a syntax
  - More complicated systems such as System Fω.

Thanks for your attention!

Some References

- Marcelo Fiore and Makoto Hamana, Multiversal Polymorphic Algebraic Theories, LICS 2013, pp. 520-529, `https://www.cl.cam.ac.uk/~mpf23/papers/Algebra/mpat.pdf`
- Makoto Hamana, Polymorphic Abstract Syntax via Grothendieck Construction, FoSSaCS 2011, pp. 381-395, `https://link.springer.com/content/pdf/10.1007/978-3-642-19805-2_26.pdf`
- Thomas Lamiaux and Benedikt Ahrens, An Introduction to Different Approaches to Initial Semantics, `https://arxiv.org/abs/2401.09366`