Initiality for Typed Syntax

Benedikt Ahrens

Université Nice Sophia Antipolis
IAS, Princeton

WoLLIC 2012

A Translation from PCF to LC

PCF

- Typed Language
  $T ::= Nat \mid Bool \mid T \Rightarrow T$
- Abstraction
  Application
  Fixpoint Operator
  Arithmetic Consts.
  Logic Consts.

Lambda Calculus

- Untyped
- Abstraction
  Application

Translation i : PCF $\rightarrow$ LC

$$i(\lambda x.M) = \lambda x.i(M)$$

$$i(M@N)) = i(M)@i(N)$$

$$i(Fix(f)) = \Theta@i(f) \qquad \text{or} \qquad i(Fix(f)) = \mathbf{Y}@i(f)$$

$$\ldots$$

Translation, mathematically

Mathematical structure of

$$i : \mathrm{PCF} \to \mathrm{LC} \qquad ?$$

Challenges:

- varying types
- capture compatibility with substitution + reduction

More precisely

- $i : \mathrm{PCF} \to \mathrm{LC}$ morphism in some category ?

Translation, mathematically

Mathematical structure of

$$i : \mathrm{PCF} \to \mathrm{LC} \qquad ?$$

Challenges:

- varying types
- capture compatibility with substitution + reduction

More precisely

- $i : \mathrm{PCF} \to \mathrm{LC}$ initial morphism in some category ?

Translation, mathematically

Mathematical structure of

$$i : \mathrm{PCF} \to \mathrm{LC} \qquad ?$$

Challenges:

- varying types
- capture compatibility with substitution + reduction

More precisely

- $i : \mathrm{PCF} \to \mathrm{LC}$ initial morphism in some category ?
- Extra structure on LC specifies different such translations

Initial Semantics

Ingredients:

- Signature Σ

$\rightsquigarrow$ specifies a language $\hat{\Sigma}$

$\rightsquigarrow$ Category of Semantics of Σ with Initial Semantics $\hat{\Sigma}$

Why Initial Semantics?

- Characterization of language $\hat{\Sigma}$ via universal property
- Categorical iteration operator

Language Features

Starting Point:

Universal Algebra, Birkhoff ’35

Features:

- Variable Binding
- Typing
- Reductions

Adding a Feature

- Defining a new notion of Signature
- Adapting the notion of Semantics

Initial Semantics for Untyped Syntax

Recall the Goals:

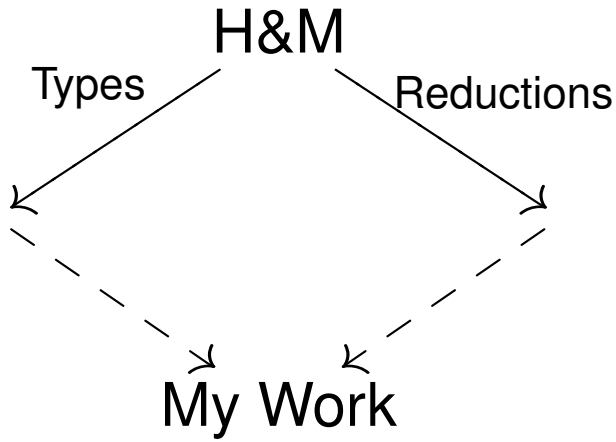
- Signatures for specifying untyped binding syntax
- $\rightsquigarrow$ Category of Semantics of a Signature
- $\rightsquigarrow$ exhibit Syntax as initial object

Has been done by

- Fiore, Plotkin, Turi ’99
- Hofmann ’99
- Gabbay, Pitts ’99
- Hirschowitz, Maggesi ’07

My Work

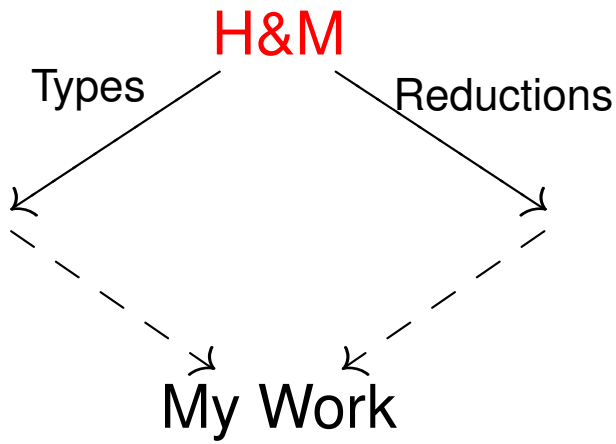
Extending H&M to types and reduction rules



In this presentation

- review H&M
- Types

Reviewing H&M



Ex.: Categorical Structure of Lambda Calculus

Syntax:

```
Inductive LC (V : Set) : Set :=
      | Var : V -> LC(V)
      | Abs : LC(V+1) -> LC(V)
      | App : LC(V) x LC(V) -> LC(V)
```

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
      | Var : V -> LC(V)
      | Abs : LC(V+1) -> LC(V)
      | App : LC(V) x LC(V) -> LC(V)
```

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
      | Var : V -> LC(V)
      | Abs : LC(V+1) -> LC(V)
      | App : LC(V) x LC(V) -> LC(V)
```

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
      | Var : V -> LC(V)
      | Abs : LC(V+1) -> LC(V)
      | App : LC(V) x LC(V) -> LC(V)
```

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
      | Var : V -> LC(V)
      | Abs : LC(V+1) -> LC(V)
      | App : LC(V) x LC(V) -> LC(V)
```

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
        | Var : V -> LC(V)
        | Abs : LC(V+1) -> LC(V)
        | App : LC(V) x LC(V) -> LC(V)
```

Semantics/Representations of Σ_{LC} ?

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
        | Var : V -> LC(V)
        | Abs : LC(V+1) -> LC(V)
        | App : LC(V) x LC(V) -> LC(V)
```

Categorical Structure of LC

- $\mathrm{LC} : \mathit{Set} \to \mathit{Set}$
- $\mathit{Var} : \mathit{Id} \Rightarrow \mathrm{LC}$, $\mathit{Abs} : \mathrm{LC}' \Rightarrow \mathrm{LC}$, $\mathit{App} : \mathrm{LC} \times \mathrm{LC} \Rightarrow \mathrm{LC}$

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
        | Var : V -> LC(V)
        | Abs : LC(V+1) -> LC(V)
        | App : LC(V) x LC(V) -> LC(V)
```

Abstracting from LC, a Representation of Σ_{LC} is...

- $F : Set \to Set$
- $Var : Id \Rightarrow F \ , \ Abs : F' \Rightarrow F \ , \ App : F \times F \Rightarrow F$

Ex.: Categorical Structure of Lambda Calculus

Signature: $\Sigma_{LC} = \{abs : [1] \ , \quad app : [0, 0]\}$

Syntax:

```
Inductive LC (V : Set) : Set :=
        | Var : V -> LC(V)
        | Abs : LC(V+1) -> LC(V)
        | App : LC(V) x LC(V) -> LC(V)
```

Abstracting from LC, a Representation of Σ_{LC} is...

- $F : Set \to Set$
- $Var : Id \Rightarrow F \ , \ Abs : F' \Rightarrow F \ , \ App : F \times F \Rightarrow F$

Goal: Refine “Representation” by integrating Substitution...

LC as a Monad

Adding Substitution & its Properties

Representations: Functor $\rightsquigarrow$ Monad

Constructors: Nat. Transformation $\rightsquigarrow$ Morphism of Modules

LC as a Monad

Adding Substitution & its Properties

Representations: Functor $\rightsquigarrow$ Monad

Constructors: Nat. Transformation $\rightsquigarrow$ Morphism of Modules

"Variables–as–terms" and Substitution for LC

- $\mathrm{LC} : V \mapsto \mathrm{LC}(V)$
- $\mathit{Var} : \mathit{Id} \Rightarrow \mathrm{LC}$
- $(\gg\!=)_{V,W} : \mathrm{LC}(V) \times (V \to \mathrm{LC}(W)) \to \mathrm{LC}(W)$ + properties

Equips LC with structure of Monad (Altenkirch & Reus '99)

Substitution and Constructors of LC

Substitution distributes over *App*

- $M, N \in \mathrm{LC}(V)$
- $f : V \to \mathrm{LC}(W)$
- $App(M)(N) \gg\!= f \quad == \quad App(M \gg\!= f)(N \gg\!= f)$

Going under a Binder:

- $M \in \mathrm{LC}(V + 1)$
- $f : V \to \mathrm{LC}(W)$
- $f' : V + 1 \to \mathrm{LC}(W + 1)$
- $Abs(M) \gg\!= f \quad == \quad Abs(M \gg\!= f')$

Substitution distributes – Module Morphisms

Substitution distributes over *App*

- $App(M)(N) \gg\!= f \quad == \quad App(M \gg\!= f)(N \gg\!= f)$

Going under a Binder:

- $Abs(M) \gg\!= f \quad == \quad Abs(M \gg\!= f')$

Equations encoded in notion of Module Morphisms

$$App : \mathrm{LC} \times \mathrm{LC} \to \mathrm{LC}$$
$$Abs : \mathrm{LC}' \to \mathrm{LC}$$

Morphisms of Modules for LC

Modules over LC

$$\mathrm{LC}' : V \mapsto \mathrm{LC}(V + 1) \qquad \mathrm{LC} : V \mapsto \mathrm{LC}(V)$$

$$\mathrm{LC} \times \mathrm{LC} : V \mapsto \mathrm{LC}(V) \times \mathrm{LC}(V)$$

Morphisms of Modules over LC

$$\mathit{Abs} : \mathrm{LC}' \to \mathrm{LC}$$

$$\mathit{App} : \mathrm{LC} \times \mathrm{LC} \to \mathrm{LC}$$

Morphisms of Modules for LC

Modules over LC

$$\mathrm{LC}' : V \mapsto \mathrm{LC}(V+1) \qquad \mathrm{LC} : V \mapsto \mathrm{LC}(V)$$

$$\mathrm{LC} \times \mathrm{LC} : V \mapsto \mathrm{LC}(V) \times \mathrm{LC}(V)$$

Morphisms of Modules over LC

$$Abs : \mathrm{LC}' \to \mathrm{LC}$$

$$App : \mathrm{LC} \times \mathrm{LC} \to \mathrm{LC}$$

Abstracting from LC, a Representation of Σ_{LC} is...

- Monad $\qquad P : Set \to Set$
- Module Morphisms $\qquad Abs : P' \to P \;\;,\;\; App : P \times P \to P$

Signatures & their Representations

Definition (Signature)

- Arity : List of natural numbers
- Signature: Family of arities

Arity $[s_1, \ldots, s_n]$ associates module $P^{(s_1)} \times \ldots \times P^{(s_n)}$ to any monad P

Signatures & their Representations

Definition (Signature)

- Arity : List of natural numbers
- Signature: Family of arities

Arity $[s_1, \ldots, s_n]$ associates module $P^{(s_1)} \times \ldots \times P^{(s_n)}$ to any monad P

Definition (Representation of Signature Σ, H&M)

- Monad $P : Set \to Set$
- Morphism of Modules over P for each $[s_1, \ldots, s_n] \in \Sigma$,

$$P^{(s_1)} \times \ldots \times P^{(s_n)} \to P$$

Initial Semantics, untyped

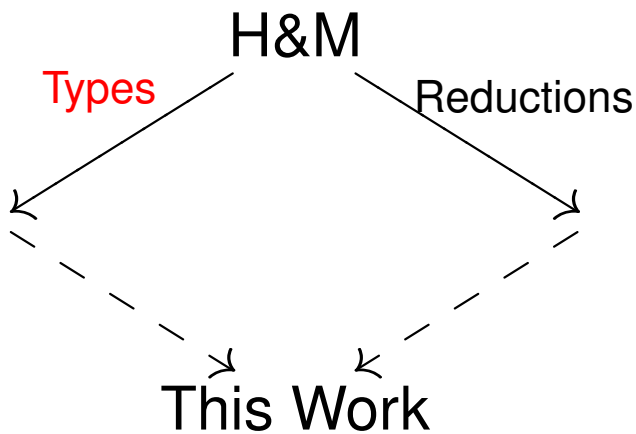
Theorem (Hirschowitz & Maggesi)

For any signature Σ, the category of representations of Σ has an initial object.

Example

(LC, *App*, *Abs*) is the initial representation of Σ_{LC}.

Extending H&M to Types



Goals

- “Signature” to specify simply–typed language w. binding

⇝ Category of Representations of a Signature

- Monads ? Modules over Monads ?

⇝ exhibit Initial Object: Syntax

Goals

- “Signature” to specify simply–typed language w. binding

⇝ Category of Representations of a Signature

- Monads ? Modules over Monads ?

⇝ exhibit Initial Object: Syntax

Generalizing Zsidó’s Theorem

- Existing solution w. fixed types

⇝ Generalize to varying sets of types

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Σ_{TLC} :

$$\left\{ \left(abs_{s,t} : \left[([s], t)\right] \to s \Rightarrow t\right)_{s,t \in T},\right.$$
$$\left.\left(app_{s,t} : \left[([], s \Rightarrow t), ([], s)\right] \to t\right)_{s,t \in T} \right\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Example: Simply–Typed LC

Types: $$T \;::=\; * \;\mid\; T \Rightarrow T$$

Σ_{TLC} :

$$\Big\{ \big(abs_{s,t} : \big[([s], t)\big] \to s \Rightarrow t\big)_{s,t \in T} \,, \quad \big(app_{s,t} : \big[([], s \Rightarrow t), ([], s)\big] \to t\big)_{s,t \in T} \Big\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Σ_{TLC} :

$$\left\{ \left(abs_{s,t} : \left[([s], t)\right] \to s \Rightarrow t\right)_{s,t \in T},\right.$$
$$\left.\left(app_{s,t} : \left[([], s \Rightarrow t), ([], s)\right] \to t\right)_{s,t \in T} \right\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Σ_{TLC} :

$$\left\{ \left(abs_{s,t} : \left[([s], t) \right] \to s \Rightarrow t \right)_{s,t \in T} , \right.$$
$$\left. \left(app_{s,t} : \left[([], s \Rightarrow t), ([], s) \right] \to t \right)_{s,t \in T} \right\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Σ_{TLC} :

$$\left\{ \left(abs_{s,t} : \left[([s], t) \right] \to s \Rightarrow t \right)_{s,t \in T} , \right.$$
$$\left. \left(app_{s,t} : \left[([], s \Rightarrow t), ([], s) \right] \to t \right)_{s,t \in T} \right\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Representations:

Example: Simply–Typed LC

Types: $T ::= * \mid T \Rightarrow T$

Σ_{TLC} :

$$\left\{ \left(abs_{s,t} : \left[([s], t)\right] \to s \Rightarrow t\right)_{s,t \in T},\right.$$
$$\left. \left(app_{s,t} : \left[([], s \Rightarrow t), ([], s)\right] \to t\right)_{s,t \in T} \right\}$$

Syntax:

```
Inductive TLC (V: T -> Set): T -> Set :=
| Var: ∀t, V(t) -> TLC(V)(t)
| Abs: ∀s t, TLC(V+s)(t) -> TLC(V)(s ⇒ t)
| App: ∀s t, TLC(V)(s⇒t) x TLC(V)(s) -> TLC(V)(t)
```

Representations:

Monads? Modules?

TLC as a Monad

The Monad TLC

- $\mathrm{TLC} : \mathit{Set}^T \to \mathit{Set}^T$
- $\mathit{Var} : \mathit{Id} \Rightarrow \mathrm{TLC}$
- $(\gg\!=)_{V,W} : \mathrm{TLC}(V) \times (V \to \mathrm{TLC}(W)) \to \mathrm{TLC}(W)$

TLC as a Monad

The Monad TLC

- $\mathrm{TLC} : \mathit{Set}^T \to \mathit{Set}^T$
- $\mathit{Var} : \mathit{Id} \Rightarrow \mathrm{TLC}$
- $(\gg\!=)_{V,W} : \mathrm{TLC}(V) \times (V \to \mathrm{TLC}(W)) \to \mathrm{TLC}(W)$

Fibre Module over TLC

- $t \in T$
- $\mathrm{TLC}[t] : \mathit{Set}^T \to \mathit{Set} \quad , \quad V \mapsto \mathrm{TLC}(V)(t)$

Modules over TLC

Domain Modules for Abstraction and Application

- Context Extension:

$$\mathrm{TLC}^{(s)}[t] : V \mapsto \mathrm{TLC}(V + s)(t)$$

- Pointwise Product:

$$\mathrm{TLC}[s \Rightarrow t] \times \mathrm{TLC}[s] : \\ V \mapsto \mathrm{TLC}(V)(s \Rightarrow t) \times \mathrm{TLC}(V)(s)$$

Simply–Typed LC and its Representations

The Lambda Calculus

- Monad $\quad \mathrm{TLC} : \mathit{Set}^T \to \mathit{Set}^T$
- $\mathit{Abs}_{s,t} : \mathrm{TLC}^{(s)}[t] \to \mathrm{TLC}[s \Rightarrow t] \qquad \forall\ s\ t \in T$
- $\mathit{App}_{s,t} : \mathrm{TLC}[s \Rightarrow t] \times \mathrm{TLC}[s] \to \mathrm{TLC}[t] \qquad \forall\ s\ t \in T$

Simply–Typed LC and its Representations

The Lambda Calculus

- Monad $\quad \mathrm{TLC} : Set^T \to Set^T$
- $Abs_{s,t} : \mathrm{TLC}^{(s)}[t] \to \mathrm{TLC}[s \Rightarrow t] \qquad \forall\ s\ t \in T$
- $App_{s,t} : \mathrm{TLC}[s \Rightarrow t] \times \mathrm{TLC}[s] \to \mathrm{TLC}[t] \qquad \forall\ s\ t \in T$

Abstracting from TLC, a Representation of Σ_{TLC} is...

- Monad $\quad P : Set^T \to Set^T$
- $Abs_{s,t} : P^{(s)}[t] \to P[s \Rightarrow t] \qquad \forall\ s\ t \in T$
- $App_{s,t} : P[s \Rightarrow t] \times P[s] \to P[t] \qquad \forall\ s\ t \in T$

Initiality, typed

Definition (Representation of signature Σ over types T)

- Monad P on Set^T
- Morphism of modules over P for each arity $s \in \Sigma$

Theorem (Zsidó '10)

The category of representations of Σ has an initial object.

Initiality, typed

Definition (Representation of signature Σ over types T)

- Monad P on Set^T
- Morphism of modules over P for each arity $s \in \Sigma$

Problem: The set T of types is hard–coded.

$\rightsquigarrow$ does not account for $i : \mathrm{PCF} \to \mathrm{LC}$

Initiality, typed

Definition (Representation of signature Σ over types T)

- Monad P on Set^T
- Morphism of modules over P for each arity $s \in \Sigma$

Problem: The set T of types is hard–coded.

$\rightsquigarrow$ does not account for $i : \mathrm{PCF} \to \mathrm{LC}$

Definition II (Representation)

+ Set U
+ $g : T \to U$
- Monad P on Set^U
- Representation of "Σ transported along g" in P

Example: A Representation of PCF is given by...

... a representation of the types of PCF ...

... and one of the terms of PCF:

Example: A Representation of PCF is given by...

... a representation of the types of PCF ...

- Set U
- $Nat : U$ $\quad$ $Bool : U$ $\quad$ $(\Rightarrow) : U \times U \to U$

... and one of the terms of PCF:

Example: A Representation of PCF is given by...

... a representation of the types of PCF ...

- Set U
- $Nat : U \qquad Bool : U \qquad (\Rightarrow) : U \times U \to U$

... and one of the terms of PCF:

- Monad $\quad P : Set^U \to Set^U$
- $Abs_{u,v} : P^{(u)}[v] \to P[u \Rightarrow v] \qquad \forall\, u\ v \in U$
- $App_{u,v} : P[u \Rightarrow v] \times P[u] \to P[v] \qquad \forall\, u\ v \in U$
- $Fix_u : P[u \Rightarrow u] \to P[u] \qquad \forall\, u \in U$
- ...

Ex.: A Representation of PCF in LC...

... a representation of the types of PCF

- $U := \{*\}$
- $Nat := *$ $\quad$ $Bool := *$ $\quad$ $x \Rightarrow y := *$

... and one of the terms of PCF:

- Monad $\quad$ LC : $Set^{\{*\}} \to Set^{\{*\}}$
- $Abs_{u,v} : \mathrm{LC}^{u}[v] \to \mathrm{LC}[u \Rightarrow v]$ $\quad$ $\forall\, u\ v \in \{*\}$
- $App_{u,v} : \mathrm{LC}[u \Rightarrow v] \times \mathrm{LC}[u] \to \mathrm{LC}[v]$ $\quad$ $\forall\, u\ v \in \{*\}$
- $Fix_u : \mathrm{LC}[u \Rightarrow u] \to \mathrm{LC}[u]$ $\quad$ $\forall\, u \in \{*\}$
- ...

Example: A Representation of PCF in LC...

... a representation of the types of PCF

- $U := \{*\}$
- $Nat := *$ $\quad$ $Bool := *$ $\quad$ $x \Rightarrow y := *$

... and one of the terms of PCF:

- Monad $\quad$ LC : $Set \to Set$
- $Abs : \mathrm{LC}' \to \mathrm{LC}$
- $App : \mathrm{LC} \times \mathrm{LC} \to \mathrm{LC}$
- $Fix : \mathrm{LC} \to \mathrm{LC}$
- ...

Initiality with Varying Types

Definition (Signature)

- a signature S for types
- a signature Σ for terms over those types

Definition (Representation of (S, Σ))

- a representation of S in a set U
- a representation of Σ in a monad P on U

Theorem

The category of representations of (S, Σ) *has an initial object.*

Example: Translation from PCF to LC by Initiality

A representation of PCF in LC

specifies an initial morphism of representations

Representing *Fix* in LC

- $M \mapsto App(\Theta, M) : \mathrm{LC} \to \mathrm{LC}$ or
- $M \mapsto App(\mathbf{Y}, M) : \mathrm{LC} \to \mathrm{LC}$

yields the two aforementioned translations

Another Example: Logic Translations via Initiality

Propositional Logic as Simple Type System

via Curry–Howard

Double Negation Translation $g : CL \to IL$ via Initiality:

- Translation of Propositions (Types) $g : P \to P$
  - $(A \wedge B)^g = A^g \wedge B^g$,
  - $(A \vee B)^g = \neg(\neg A^g \wedge \neg B^g)$,
  - ...
- Translation of Proofs (Terms)
  - Type Safety of Translation:

$$\Gamma \vdash_C A \quad \text{implies} \quad \Gamma^g \vdash_I A^g$$

Thanks for your attention