The Combination of Dynamic and Static Typing from a Categorical Perspective

Harley Eades III and Michael Townsend
Augusta University
{heades,mitownsend}@augusta.edu

Categories and Subject Descriptors D.3.3 [*??*]: ??—??

General Terms TODO

Keywords TODO

Abstract

TODO

1. Introduction

(Scott 1980) showed how to model the untyped λ-calculus within a cartesian closed category, $\mathcal{C}$, with a distinguished object we will call ? – read as the type of untyped terms – such that the object[1] $? \to ?$ is a retract of ?. That is, there are morphisms $\mathsf{squash} : (? \to ?) \longrightarrow ?$ and $\mathsf{split} : ? \longrightarrow (? \to ?)$ where $\mathsf{squash}; \mathsf{split} = \mathsf{id} : (? \to ?) \longrightarrow (? \to ?)$[2]. For example, taking these morphisms as terms in the typed λ-calculus we can define the prototypical looping term $(\lambda x.x\,x)(\lambda x.x\,x)$ by $(\lambda x : ?.(\mathsf{split}\,x)\,x)\,(\mathsf{squash}\,(\lambda x : ?.(\mathsf{split}\,x)\,x))$.

In the same volume as Scott (Lambek 1980) showed that cartesian closed categories also model the typed λ-calculus. Suppose we want to model the typed λ-calculus with pairs and natural numbers. That is, given two types A_1 and A_2 there is a type $A_1 \times A_2$, and there is a type $\mathbf{Nat}$. Furthermore, we have first and second projections, and zero and successor functions. This situation can easily be modeled by a cartesian closed category $\mathcal{C}$ – see Section 2 for the details – but also add to $\mathcal{C}$ the type of untyped terms ?, squash, and split. At this point $\mathcal{C}$ is a model of both the typed and the untyped λ-calculus. However, the two theories are really just sitting side by side in $\mathcal{C}$ and cannot really interact much.

Suppose $\mathcal{T}$ is a discrete category with the objects $\mathbf{Nat}$ and 1 (the terminal object or empty product) and $\mathsf{T} : \mathcal{T} \longrightarrow \mathcal{C}$ is a full and faithful functor. This implies that $\mathcal{T}$ is a subcategory of $\mathcal{C}$, and that $\mathcal{T}$ is the category of atomic types. Then for any type A of $\mathcal{T}$ we add to $\mathcal{C}$ the morphisms $\mathsf{box} : \mathsf{T}A \longrightarrow ?$ and $\mathsf{unbox} : ? \longrightarrow \mathsf{T}A$ such that $\mathsf{box}; \mathsf{unbox} = \mathsf{id} : \mathsf{T}A \longrightarrow \mathsf{T}A$ making $\mathsf{T}A$ a retract of ?. This is the bridge allowing the typed world to interact with the untyped one. We can think of box as injecting typed data into the untyped world, and unbox as taking it back. Notice that the only time we can actually get the typed data back out is if it were injected into the untyped world initially. In the model this is enforced through composition, but in the language this will be enforced at runtime, and hence, requires the language to contain dynamic typing. Thus, what we have just built up is a categorical model that offers a new perspective of how to combine static and dynamic typing.

(Siek and Taha 2006) define gradual typing to be the combination of both static and dynamic typing that allows for the programmer to program in dynamic style, and thus, annotations should be suppressed. This means that a gradually typed program can utilize both static types which will be enforced during compile time, but may also utilize dynamic typing that will be enforced during runtime. Therefore, gradual typing is the best of both worlds.

Siek and Taha's gradually typed functional language is the typed λ-calculus with the type of untyped terms ? and the following rules:

$$\frac{\Gamma \vdash t_1 : ? \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\,t_1 : ?} \qquad \frac{\Gamma \vdash t_1 : A_1 \to B \quad \Gamma \vdash t_2 : A_2 \quad A_1 \sim A_2}{\Gamma \vdash t_1\,t_2 : B}$$

The premise $A \sim B$ is read, the type A is consistent with the type B, and is defined by the following rules:

$$\frac{}{A \sim A} \qquad \frac{}{A \sim ?} \qquad \frac{}{? \sim A}$$

$$\frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \to B_1 \sim A_2 \to B_2} \qquad \frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \times B_1 \sim A_2 \times B_2}$$

This is a reflexive and symmetric relation, but is a non-transitive relation. If we squint we can see split, squash, box, and unbox hiding in the definition of the previous rules, but they have been suppressed. We will show that when one uses either of the two typing rules above that really one is implicitly using a casting morphism built from split, squash, box, and unbox. In fact, the consistency relation $A \sim B$ can be interpreted as such a morphism. Then the typing above can be read as a saying if a casting morphism exists, then the programmer is allowed to act like it is not there, but we can always put it in if we need to.

The annotated language they present

$$\frac{\Gamma \vdash t : A \quad A \sim B}{\Gamma \vdash \langle B \rangle t : B}$$

Contributions. This paper offers the following contributions:

- A new categorical model for gradual typing for functional languages.
- A functional programming language called Grady that corresponds through the Curry-Howard-Lambek Correspondence to the categorical model.

[1] We will use the terms "object" and "type" interchangeably.

[2] We denote composition of morphisms by $f; g : A \longrightarrow C$ given morphisms $f : A \longrightarrow B$ and $g : B \longrightarrow C$.

- A proof that Grady is as expressive as (Siek and Taha 2006)'s annotated language and vice versa.
- Several fun examples leveraging

2. Categorical Model

Definition 1. *Suppose $\mathcal{C}$ is a category. Then an object A is a* ***retract*** *of an object B if there are morphisms $i : A \longrightarrow B$ and $r : B \longrightarrow A$ such that the following diagram commutes:*

$$\begin{tikzcd}
A \arrow[r, "i"] \arrow[dr, equal] & B \arrow[d, "r"] \\
& A
\end{tikzcd}$$

Definition 2. *An* ***untyped λ-model****, $(\mathcal{C}, ?, \mathsf{split}, \mathsf{squash})$, is a cartesian closed category $\mathcal{C}$ with a distinguished object $?$ and two morphisms $\mathsf{squash} : (? \to ?) \longrightarrow ?$ and $\mathsf{split} : ? \longrightarrow (? \to ?)$ making the object $? \to ?$ a retract of $?$.*

Theorem 3 (Scott (1980))**.** *An untyped λ-model is a sound and complete model of the untyped λ-calculus.*

Definition 4. *An object $\mathbb{N}$ of a category $\mathcal{C}$ with a terminal object 1 is a* ***natural number object (NNO)*** *if and only if there are morphisms $z : 1 \longrightarrow \mathbb{N}$ and $s : \mathbb{N} \longrightarrow \mathbb{N}$ such that for any other object A and morphisms $z' : 1 \longrightarrow A$ and $s' : A \longrightarrow A$ there is a unique morphism $u : \mathbb{N} \longrightarrow A$ making the following diagram commute:*

$$\begin{tikzcd}
1 \arrow[r, "z"] \arrow[dr, "z'"'] & \mathbb{N} \arrow[r, "s"] \arrow[d, "u"] & \mathbb{N} \arrow[d, "u"] \\
& A \arrow[r, "s'"] & A
\end{tikzcd}$$

Definition 5. *A* ***gradual λ-model****, $(\mathcal{T}, \mathcal{C}, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox})$, where $\mathcal{T}$ and $\mathcal{C}$ are cartesian closed categories with NNOS, $(\mathcal{C}, ?, \mathsf{split}, \mathsf{squash})$ is an untyped λ-model, $\mathsf{T} : \mathcal{T} \longrightarrow \mathcal{C}$ is a cartesian closed embedding – a full and faithful cartesian closed functor that is injective on objects and preserves the NNO – and for every object, A, of $\mathcal{T}$ there are morphisms $\mathsf{box}_A : TA \longrightarrow ?$ and $\mathsf{unbox}_A : ? \longrightarrow TA$ making TA a retract of $?$.*

3. Grady

References

Joachim Lambek. From lambda calculus to cartesian closed categories. *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 376–402, 1980.

Dana Scott. Relating theories of the lambda-calculus. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism (eds. Hindley and Seldin)*, pages 403–450. Academic Press, 1980.

Jeremy G Siek and Walid Taha. Gradual typing for functional languages. In *Scheme and Functional Programming Workshop*, volume 6, pages 81–92, 2006.

A. The Complete Spec of Grady

$termvar, x, z$
$index, k$

t	$::=$			term
	$\vert$	x		variable
	$\vert$	triv		unit
	$\vert$	squash_S		injection of the retract
	$\vert$	split_S		surjection of the retract
	$\vert$	box_T		generalize to the untyped universe
	$\vert$	unbox_T		specialize the untyped universe to a
	$\vert$	$\langle A \rangle t$		type cast
	$\vert$	$\lambda x : A.t$		λ-abstraction
	$\vert$	$t_1\ t_2$		function application
	$\vert$	(t_1, t_2)		pair constructor
	$\vert$	$\mathsf{fst}\ t$		first projection
	$\vert$	$\mathsf{snd}\ t$		second projection
	$\vert$	$\mathsf{succ}\ t$		successor function
	$\vert$	0		zero
	$\vert$	(t)	S	
h	$::=$			head-normal forms
	$\vert$	triv		
	$\vert$	split_S		
	$\vert$	squash_S		
	$\vert$	box_T		
	$\vert$	unbox_T		
	$\vert$	$\lambda x : A.t$		
	$\vert$	(t_1, t_2)		
	$\vert$	$\mathsf{fst}\ t$		
	$\vert$	$\mathsf{snd}\ t$		
	$\vert$	$\mathsf{succ}\ t$		
	$\vert$	0		
T	$::=$			terminating types
	$\vert$	1		unit type
	$\vert$	$\mathbf{Nat}$		natural number type
	$\vert$	$T_1 \to T_2$		function type
	$\vert$	$T_1 \times T_2$		cartesian product type
	$\vert$	(T)	S	
S	$::=$			
	$\vert$	$? \to ?$		
	$\vert$	$? \times ?$		
$A,\ B,\ C$	$::=$			type
	$\vert$	1		unit type
	$\vert$	$\mathbf{Nat}$		natural number type
	$\vert$	$?$		untyped universe
	$\vert$	$A_1 \to A_2$		function type
	$\vert$	$A_1 \times A_2$		cartesian product type
	$\vert$	(A)	S	
Γ	$::=$			typing context
	$\vert$	$.$		empty context
	$\vert$	$\Gamma, x : A$		cons
vd	$::=$			
	$\vert$	$\vdash$		
	$\vert$	$\not\vdash$		

$\boxed{A \sim B}$ A is consistent with B

$$\frac{}{A \sim A}$$

$$\frac{}{A \sim ?}$$

$$\frac{}{? \sim A}$$

$$\frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \to B_1 \sim A_2 \to B_2}$$

$$\frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \times B_1 \sim A_2 \times B_2}$$

$\boxed{\Gamma v d t : A}$ t has type A in context Γ

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \textsc{var}$$

$$\frac{}{\Gamma \vdash \mathsf{box}_T : T \to ?}\ \textsc{Box}$$

$$\frac{}{\Gamma \vdash \mathsf{unbox}_T : ? \to T}\ \textsc{Unbox}$$

$$\frac{}{\Gamma \vdash \mathsf{squash}_S : S \to ?}\ \textsc{squash}$$

$$\frac{}{\Gamma \vdash \mathsf{split}_S : ? \to S}\ \textsc{split}$$

$$\frac{}{\Gamma \vdash \mathsf{triv} : 1}\ \textsc{unit}$$

$$\frac{}{\Gamma \vdash 0 : \mathbf{Nat}}\ \textsc{zero}$$

$$\frac{\Gamma \vdash t : \mathbf{Nat}}{\Gamma \vdash \mathsf{succ}\, t : \mathbf{Nat}}\ \textsc{succ}$$

$$\frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash (t_1, t_2) : A_1 \times A_2}\ \textsc{pair}$$

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \mathsf{fst}\, t : A_1}\ \textsc{fst}$$

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \mathsf{snd}\, t : A_2}\ \textsc{snd}$$

$$\frac{\Gamma, x : A_1 \vdash t : A_2}{\Gamma \vdash \lambda x : A_1.t : A_1 \to A_2}\ \textsc{lam}$$

$$\frac{\Gamma \vdash t_1 : A_1 \to A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash t_1\, t_2 : A_2}\ \textsc{app}$$

$$\frac{\Gamma \vdash t_1 : ? \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\, t_1 : ?}$$

$$\frac{\begin{array}{c}\Gamma \vdash t_1 : A_1 \to B \\ \Gamma \vdash t_2 : A_2 \quad A_1 \sim A_2\end{array}}{\Gamma \vdash t_1\, t_2 : B}$$

$$\frac{\Gamma \vdash t : A \quad A \sim B}{\Gamma \vdash \langle B \rangle t : B}$$

$\boxed{\Gamma \vdash t_1 \rightsquigarrow t_2 : A}$ t_1 reduces to t_2 with type A in context Γ

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \mathsf{unbox}_T\, (\mathsf{box}_T\, t) \rightsquigarrow t : T}\ \textsc{rd\_retracT}$$

$$\frac{\Gamma \not\vdash t : T}{\Gamma \vdash \mathsf{unbox}_T\, (\mathsf{box}_{T'}\, t) \rightsquigarrow \mathsf{wrong} : \mathsf{TypeError}}\ \textsc{rd\_tWrong}$$

$$\frac{h \neq \mathsf{box}_T\, t}{\Gamma \vdash \mathsf{unbox}_T\, h \rightsquigarrow \mathsf{wrong} : \mathsf{TypeError}}\ \textsc{rd\_hWrong}$$

$$\frac{\Gamma \vdash t : S}{\Gamma \vdash \mathsf{split}_S\, (\mathsf{squash}_S\, t) \rightsquigarrow t : S}\ \textsc{rd\_retractU}$$

$$\frac{\Gamma, x : A_1 \vdash t_2 : A_2 \quad \Gamma \vdash t_1 : A_1}{\Gamma \vdash (\lambda x : A_1.t_2)\, t_1 \rightsquigarrow [t_1/x]t_2 : A_2}\ \textsc{rd\_beta}$$

$$\frac{\Gamma \vdash t : A_1 \to A_2 \quad x \notin \mathsf{FV}(t)}{\Gamma \vdash \lambda x : A_1.t\, x \rightsquigarrow t : A_1 \to A_2}\ \textsc{rd\_eta}$$

$$\frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash \mathsf{fst}\, (t_1, t_2) \rightsquigarrow t_1 : A_1}\ \textsc{rd\_proj1}$$

$$\frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash \mathsf{snd}\, (t_1, t_2) \rightsquigarrow t_2 : A_2}\ \textsc{rd\_proj2}$$

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash (\mathsf{fst}\, t, \mathsf{snd}\, t) \rightsquigarrow t : A_1 \times A_2}\ \textsc{rd\_etaP}$$

$$\frac{\Gamma, x : A_1 \vdash t \rightsquigarrow t' : A_2}{\Gamma \vdash \lambda x : A_1.t \rightsquigarrow \lambda x : A_1.t' : A_1 \to A_2}\ \textsc{rd\_lam}$$

$$\frac{\Gamma \vdash t_1 \rightsquigarrow t_1' : A_1 \to A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash t_1\, t_2 \rightsquigarrow t_1'\, t_2 : A_2}\ \textsc{rd\_app1}$$

$$\frac{\Gamma \vdash t_1 : A_1 \to A_2 \quad \Gamma \vdash t_2 \rightsquigarrow t_2' : A_1}{\Gamma \vdash t_1\, t_2 \rightsquigarrow t_1\, t_2' : A_2}\ \textsc{rd\_app2}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' : A_1 \times A_2}{\Gamma \vdash \mathsf{fst}\, t \rightsquigarrow \mathsf{fst}\, t' : A_1}\ \textsc{rd\_fst}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' : A_1 \times A_2}{\Gamma \vdash \mathsf{snd}\, t \rightsquigarrow \mathsf{snd}\, t' : A_2}\ \textsc{rd\_snd}$$

$$\frac{\Gamma \vdash t_1 \rightsquigarrow t_1' : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash (t_1, t_2) \rightsquigarrow (t_1', t_2) : A_1 \times A_2}\ \textsc{rd\_pair1}$$

$$\frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 \rightsquigarrow t_2' : A_2}{\Gamma \vdash (t_1, t_2) \rightsquigarrow (t_1, t_2') : A_1 \times A_2}\ \textsc{rd\_pair2}$$