The Combination of Dynamic and Static Typing from a Categorical Perspective

ANONYMOUS AUTHOR(S)

Gradual typing was first proposed by Siek and Taha in 2006 as a way for a programming language to combine the strengths of both static and dynamic typing. However one question we must ask is, what is gradual typing? This paper contributes to answering this question by providing the first categorical model of gradual typing using the seminal work of Scott and Lambek on the categorical models of the untyped and typed λ-calculus. We then extract a functional programming language, called Grady, from the categorical model using the Curry-Howard-Lambek correspondence that combines both static and dynamic typing, but Grady is an annotated language and not a gradual type system. Finally, we show that Siek and Taha's gradual type system can be translated into Grady, and that their original annotated language is equivalent in expressive power to Grady.

CCS Concepts: •**Theory of computation → Denotational semantics; Categorical semantics;** *Type theory; Functional constructs; Type structures;*

Additional Key Words and Phrases: static typing, dynamic typing, gradual typing, categorical semantics, retract,typed lambda-calculus, untyped lambda-calculus, gradual typing, static typing, dynamic typing, categorical model, functional programming

ACM Reference format:
Anonymous Author(s). 2017. The Combination of Dynamic and Static Typing from a Categorical Perspective. *PACM Progr. Lang.* 1, 1, Article 1 (January 2017), 35 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

(Scott 1980) showed how to model the untyped λ-calculus within a cartesian closed category, 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 Nat. Furthermore, we have first and second projections, and zero and successor functions. This situation can easily be modeled by a cartesian closed category C – see Section **??** for the details – but also add to C the type of untyped terms ?, squash, and split. At this point 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 C and cannot really interact much.

Suppose $\mathcal{T}$ is a discrete category with the objects Nat and Unit (the terminal object or empty product) and $\mathsf{T} : \mathcal{T} \longrightarrow C$ is a full and faithful functor. This implies that $\mathcal{T}$ is a subcategory of C, and that $\mathcal{T}$ is the category of atomic types. Then for any type A of $\mathcal{T}$ we add to C the morphisms $\mathsf{box} : \mathsf{T}A \longrightarrow ?$ and $\mathsf{unbox} : ? \longrightarrow \mathsf{T}A$ such that $\mathsf{box}; \mathsf{unbox}\ \ \mathsf{id} : \mathsf{T}A \longrightarrow \mathsf{T}A$

[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 $g : B \longrightarrow C$.

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:

$$TODO$$

The premise $A \sim B$ is read, the type A is consistent with the type B, and is defined in Figure 2. 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 then one is really 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 semantically as a saying if a casting morphism exists, then the type really can be converted into the necessary type.

Contributions. This paper offers the following contributions:

- A new categorical model for gradual typing for functional languages. We show how to interpret (Siek and Taha 2006)'s gradual type system in the categorical model outlined above. As far as the authors are aware this is the first categorical model for gradual typing.
- We then extract a functional programming language called Grady from the categorical model via the Curry-Howard-Lambek correspondence. This is not a gradual type system, but can be seen as an alternative annotated language in which Siek and Taha's gradual type system can be translated to.
- A proof that Grady is as expressive as (Siek and Taha 2006)'s annotated language and vice versa. We give a type directed translation of Siek and Taha's annotated language to Grady and vice versa, then we show that these translations preserve evaluation.
- Having the untyped λ-calculus along side the typed λ-calculus can be a lot of fun. We show how to Church encode typed data, utilize the Y-combinator, and even obtain terminating recursion on natural numbers by combining the Y-combinator with a natural number eliminator. Thus, obtaining the expressive power of Gödel's system T (Girard et al. 1989).

Related work. TODO

2 GRADUAL TYPING

We begin by introducing a slight variation of (Siek and Taha 2006)'s gradually typed functional language. It has been extended with product types and natural numbers, and instead of a big-step call-by-value operational semantics it uses a

$$\frac{x : A \in \Gamma}{\Gamma \vdash_S x : A}\,\text{var} \qquad \frac{}{\Gamma \vdash_S \mathsf{triv} : \mathsf{Unit}}\,\text{unit} \qquad \frac{}{\Gamma \vdash_S 0 : \mathsf{Nat}}\,\text{zero} \qquad \frac{\Gamma \vdash_S t : A \quad \mathsf{nat} A \;\; \mathsf{Nat}}{\Gamma \vdash_S \mathsf{succ}\, t : \mathsf{Nat}}\,\text{succ}$$

$$\frac{\Gamma \vdash_S t_1 : A_1 \quad \Gamma \vdash_S t_2 : A_2}{\Gamma \vdash_S t_1, t_2 : A_1 \times A_2}\,\times \qquad \frac{\Gamma \vdash_S t : B \quad \mathsf{prod} B \;\; A_1 \times A_2}{\Gamma \vdash_S \mathsf{fst}\, t : A_1}\,\times_{e_1} \qquad \frac{\Gamma \vdash_S t : B \quad \mathsf{prod} B \;\; A_1 \times A_2}{\Gamma \vdash_S \mathsf{snd}\, t : A_2}\,\times_{e_2}$$

$$\frac{\Gamma, x : A \vdash_S t : B}{\Gamma \vdash_S \lambda x : A_1.t : A \to B}\,\to \qquad \frac{\begin{array}{c}\Gamma \vdash_S t_1 : C \quad \mathsf{fun} C \;\; A_1 \to B_1 \\ \Gamma \vdash_S t_2 : A_2 \quad A_2 \sim A_1\end{array}}{\Gamma \vdash_S t_1\, t_2 : B_1}\,\to_e$$

Fig. 1. Typing rules for $\lambda^{?}_{\to}$

$$\frac{}{A \sim A}\,\text{refl} \qquad \frac{}{A \sim ?}\,\text{box} \qquad \frac{}{? \sim A}\,\text{unbox} \qquad \frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \to B_1 \sim A_2 \to B_2}\,\to \qquad \frac{A_1 \sim A_2 \quad B_1 \sim B_2}{A_1 \times B_1 \sim A_2 \times B_2}\,\times$$

Fig. 2. Type Consistency for $\lambda^{?}_{\to}$

single-step type directed full $\beta\eta$-evaluator. One thing we strive for in this paper is to keep everything as simple as possible so that the underlying structure of these languages shines through. In this vein, the change in evaluation makes it easier to interpret the language into the categorical model.

The syntax of the gradual type system $\lambda^{?}_{\to}$ is defined in the following definition.

Definition 2.1. Syntax for $\lambda^{?}_{\to}$:

$$\begin{array}{lrl}
\text{(types)} & A, B & :: \mathsf{Unit} \mid \mathsf{Nat} \mid\, ? \mid A \times B \mid A_1 \to A_2 \\
\text{(terms)} & t & :: x \mid \mathsf{triv} \mid 0 \mid \mathsf{succ}\, t \mid \lambda x : A.t \mid t_1\, t_2 \mid t_1, t_2 \mid \mathsf{fst}\, t \mid \mathsf{snd}\, t \\
\text{(contexts)} & \Gamma & :: \cdot \mid x : A \mid \Gamma_1, \Gamma_2
\end{array}$$

This definition is the base syntax for every language in this paper. The typing rules are defined in Figure 1 and the type consistency relation is defined in Figure 2. The main changes of the version of $\lambda^{?}_{\to}$ defined here from the original due to (Siek and Taha 2006) is that products and natural numbers have been added. The definition of products follows how casting is done for functions. So it allows casting projections of products, for example, it is reasonable for terms like $\lambda x : ? \times ?.\mathsf{succ}\,\mathsf{fst}\, x$ to type check.

We can view gradual typing as a surface language feature much like type inference, and we give it a semantics by translating it into an annotated core. (Siek and Taha 2006) do just that and give $\lambda^{?}_{\to}$ an operational semantics by translating it to a fully annotated core language called $\lambda^{\langle A \rangle}_{\to}$. Its syntax is an extension of the syntax of $\lambda^{?}_{\to}$ (Definition 2.1) where terms are the only syntactic class that differs, and so we do not repeat the syntax of types or contexts.

Definition 2.2. Syntax for $\lambda^{\langle A \rangle}_{\to}$:

$$\begin{array}{lrl}
\text{(values)} & v & :: TODO \\
\text{(terms)} & t & :: \ldots \mid << t : A > B >>
\end{array}$$

The typing rules for $\lambda^{\langle A \rangle}_{\to}$ can be found in Figure 3, and the reduction rules in Figure 4.

The major difference from the formalization of $\lambda^{\langle A \rangle}$ given here and Siek and Taha's is that it is single step and full β-reduction, but it is based on their original definition.

$$\frac{x : A \in \Gamma}{\Gamma \vdash_{\mathsf{C}} x : A}\,\text{var} \qquad \frac{}{\Gamma \vdash_{\mathsf{C}} \mathsf{triv} : \mathsf{Unit}}\,\text{unit} \qquad \frac{}{\Gamma \vdash_{\mathsf{C}} 0 : \mathsf{Nat}}\,\text{zero} \qquad \frac{\Gamma \vdash_{\mathsf{C}} t : \mathsf{Nat}}{\Gamma \vdash_{\mathsf{C}} \mathsf{succ}\, t : \mathsf{Nat}}\,\text{succ}$$

$$\frac{\Gamma \vdash_{\mathsf{C}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{C}} t_2 : A_2}{\Gamma \vdash_{\mathsf{C}} t_1, t_2 : A_1 \times A_2}\,\times \qquad \frac{\Gamma \vdash_{\mathsf{C}} t : A_1 \times A_2}{\Gamma \vdash_{\mathsf{C}} \mathsf{fst}\, t : A_1}\,\times_{e_1} \qquad \frac{\Gamma \vdash_{\mathsf{C}} t : A_1 \times A_2}{\Gamma \vdash_{\mathsf{C}} \mathsf{snd}\, t : A_2}\,\times_{e_2} \qquad \frac{\Gamma, x : A \vdash_{\mathsf{C}} t : B}{\Gamma \vdash_{\mathsf{C}} \lambda x : A_1.t : A \to B}\,\to$$

$$\frac{\Gamma \vdash_{\mathsf{C}} t_1 : A \to B \quad \Gamma \vdash_{\mathsf{C}} t_2 : A}{\Gamma \vdash_{\mathsf{C}} t_1\, t_2 : B}\,\to_e \qquad \frac{\Gamma \vdash_{\mathsf{C}} t : A \quad A \sim B}{\Gamma \vdash_{\mathsf{C}} t : A \Rightarrow B : B}\,\text{cast}$$

Fig. 3. Typing rules for $\lambda^{\langle A \rangle}_{\to}$

TODO

Fig. 4. Reduction rules for $\lambda^{\langle A \rangle}_{\to}$

TODO

Fig. 5. Cast Insertion

This function is used when casting values to their appropriate type.

Since the formalization of both $\lambda^{?}_{\to}$ and $\lambda^{\langle A \rangle}$ differ from their original definitions we give the definition of cast insertion in Figure 5, but this is only a slightly modified version from the one given by Siek and Taha.

3 THE CATEGORICAL PERSPECTIVE

The strength and main motivation for giving a categorical model to a programming language is that it can expose the fundamental structure of the language. This arises because a lot of the language features that often cloud the picture go away, for example, syntactic notions like variables disappear. This can often simplify things and expose the underlying structure. Reynolds (**?**) was a big advocate for the use of category theory in programming language research for these reasons. For example, when giving the simply typed λ-calculus a categorical model we see that it is a cartesian closed category, but we also know that intuitionistic logic has the same model due to (Lambek 1980); on the syntactic side these two theories are equivalent as well due to (Howard 1980). Thus, the fundamental structure of the simply typed λ-calculus is intuitionistic logic. This also shows a relationship between seemingly unrelated theories. It is quite surprising that these two theories are related. Another more recent example of this can be found in the connection between dependent type theory and homotopy theory (**?**).

Another major benefit of studying the categorical model of programming languages is that it gives us a powerful tool to study language extensions. For example, purely functional programming in Haskell would not be where it is without the seminal work of Moggi and Wadler (**?**) on using monads – a purely categorical notion – to add side effects to Haskell. Thus, we believe that developing these types of models for new language designs and features can be hugely beneficial.

Interpreting a programming language into a categorical model requires three steps. First, the types are interpreted as objects. Then programs are interpreted as morphisms in the category, but this is a simplification. Every morphism, f, in a category has a source object and a target object, we usually denote this by $f : A \longrightarrow B$. Thus, in order to interpret programs as morphisms the program must have a source and target. So instead of interpreting raw terms as morphisms we interpret terms in their typing context. That is, we must show how to interpret every $\Gamma \vdash t : A$ as a morphism $t : [\![\Gamma]\!] \longrightarrow [\![A]\!]$. The third step is to show that whenever one program reduces to another their interpretations are isomorphic in the model. This means that whenever $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$, then $[\![t_1]\!] \quad [\![t_2]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!]$. This is the reason why we defined our reduction in a typed fashion to aid us in understanding how it relates to the model. For a more thorough introduction see (Crole 1994).

3.1 The Categorical Model

We now give a categorical model for $\lambda^{?}_{\to}$ and $\lambda^{\langle A \rangle}_{\to}$. The model we develop here builds on the seminal work of (Lambek 1980) and (Scott 1980). (Lambek 1980) showed that the typed λ-calculus can be modeled by a cartesian closed category. In the same volume as Lambek, Scott essentially showed that the untyped λ-calculus is actually typed. That is, typed theories are more fundamental than untyped ones. He accomplished this by adding a single type, ?, and two functions $\mathsf{squash} : ? \to ? \to ?$ and $\mathsf{split} : ? \to ? \to ?$, such that, $\mathsf{squash}; \mathsf{split} \quad \mathsf{id} : ? \to ? \to ? \to ?$. At this point he was able to translate the untyped λ-calculus into this unityped one. Categorically, he modeled split and squash as the morphisms in a retract within a cartesian closed category – the same model as typed λ-calculus.

Definition 3.1. Suppose 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:

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

Thus, $? \to ?$ is a retract of ?, but we extend this slightly to include $? \times ?$ being a retract of ?. This is only a slight extension of Scott's model, because our languages will have products where he did not consider products, because he was considering the traditional definition of the untyped λ-calculus.

We can now define our categorical model of the untyped λ-calculus with products.

Definition 3.2. An **untyped λ-model**, $C, ?, \mathsf{split}, \mathsf{squash}$, is a cartesian closed category C with a distinguished object ? and morphisms $\mathsf{squash} : S \longrightarrow ?$ and $\mathsf{split} : ? \longrightarrow S$ making the object S a retract of ?, where S is either $? \to ?$ or $? \times ?$.

Theorem 3.3 (Scott (1980)). *An untyped λ-model is a sound and complete model of the untyped λ-calculus.*

Since all of the languages we are studying here contain the natural numbers we must be able to interpret them into our model. We give a novel approach to modeling the natural numbers with their (non-recursive) eliminator using what we call a Scott natural number object. Now the natural number eliminator is not part of $\lambda^{?}_{\to}$ or $\lambda^{\langle A \rangle}_{\to}$, but we want Grady to contain it, and Grady will directly correspond to the model.

Definition 3.4. Suppose C is a cartesian closed category. A **Scott natural number object (SNNO)** is an object Nat of C and morphisms $\mathsf{z} : 1 \longrightarrow \mathsf{Nat}$ and $\mathsf{succ} : \mathsf{Nat} \longrightarrow \mathsf{Nat}$ of C, such that, for any morphisms $f : Y \longrightarrow X$ and $g :<< YxNat >> \longrightarrow X$ of C there is a unique morphism $\mathsf{case}_X :<< YxNat >> \longrightarrow X$ making the following diagrams commute:

$$\begin{tikzcd}
<< YxNat >> \arrow[r, "\mathsf{id}_Y \times \mathsf{z}"] \arrow[dr, "\pi_1;f"'] & << YxNat >> \arrow[d, "\mathsf{case}_X"] & << YxNat >> \arrow[l, "\mathsf{id}_Y \times << succ >>"'] \arrow[dl, "g"] \\
& X &
\end{tikzcd}$$

Informally, the two diagrams essentially assert that we can define case_X as follows:

$$\begin{aligned} \mathsf{case}_X\, y\, 0 &\quad f\, y \\ \mathsf{case}_X\, y\, \mathsf{succ}\, x &\quad g\, y\, x \end{aligned}$$

This formalization of natural numbers is inspired by the definition of Scott Numerals (**?**) where the notion of a case distinction is built into the encoding. We can think of Y in the source object of case as the type of additional inputs that will be passed to both f and g, but we can think of Nat in the source object of case as the type of the scrutiny. Thus, since in the base case there is no predecessor, f, will not require the scrutiny, and so it is ignored.

One major difference between SNNOs and the more traditional natural number objects is that in the definition of the latter g is defined by well-founded recursion. However, SNNOs do not allow this, but in the presence of fixpoints we are able to regain this feature without having to bake it into the definition of natural number objects. However, to allow this we have found that when combining fixpoints and case analysis to define terminating functions on the natural numbers it is necessary to uniformly construct the input to both f and g due to the reduction rule of the Y combinator. Thus, we extend the type of f to $<< YxNat >>$, but then ignore the second projection when reaching the base case.

So far we can model the untyped and the typed λ-calculi within a cartesian closed category, but we do not have any way of moving typed data into the untyped part and vice versa. To accomplish this we add two new morphisms $\mathsf{box}_C : C \longrightarrow ?$ and $\mathsf{unbox}_C : ? \longrightarrow C$ such that $\mathsf{box}_C ; \mathsf{unbox}_C \;\; \mathsf{id} : C \longrightarrow C$ for every atomic type C. Thus, each atomic type is a retract of ?. This enforces that the only time we can really consider something as typed is if it were boxed up in the first place. We can look at this from another perspective as well. If the programmer tries to unbox something that is truly untyped, then their program may actually type check, but they will obtain a dynamic type error at runtime, because the unbox will never have been matched up with the correct boxed data. For example, we can cast 3 to type Bool by $\mathsf{unbox}_{\mathsf{Bool}}\mathsf{box}_{\mathsf{Nat}}\, 3$, but if this program is every run, then we will obtain a dynamic type error. Note that we can type the previous program in $\lambda^{\langle A \rangle}_{\rightarrow}$ as well, but if we run the program it will result in a dynamic type error too.

Now we combine everything we have discussed so far to obtain the categorical model.

Definition 3.5. A **gradual λ-model**, $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$, where $\mathcal{T}$ is a discrete category with at least two objects Nat and Unit, C is a cartesian closed category with a SNNO, $C, ?, \mathsf{split}, \mathsf{squash}$ is an untyped λ-model, $\mathsf{T} : \mathcal{T} \longrightarrow C$ is an embedding – a full and faithful functor that is injective on objects – 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 ?.

We call the category $\mathcal{T}$ the category of atomic types. We call an object, A, **atomic** iff there is some object A' in $\mathcal{T}$ such that $A \;\; \mathsf{T}A'$. Note that we do not consider ? an atomic type. The model really is the cartesian closed category C, but it is extended with the structure of both the typed and the untyped λ-calculus with the ability to cast data.

Interpreting the typing rules for $\lambda^{?}_{\rightarrow}$ will require the interpretation of type consistency. Thus, we must be able to cast any type A to ?, but as stated the model only allows atomic types to be casted. It turns out that this can be lifted to any type.

We call any morphism defined completely in terms of id, the functors $- \times -$ and $- \rightarrow -$, split and squash, and box and unbox a **casting morphism**. To cast any type A to ? we will build casting morphisms that first take the object A to its skeleton, and then takes the skeleton to ?.

Definition 3.6. Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. Then the **skeleton** of an object A of C is an object S that is constructed by replacing each atomic type in A with ?. Given an object A we denote its skeleton by **skeleton** A.

One should think of the skeleton of an object as the supporting type structure of the object, but we do not know what kind of data is actually in the structure. For example, the skeleton of the object Nat is ?, and the skeleton of $\mathsf{Nat} \times \mathsf{Unit} \to \mathsf{Nat} \to \mathsf{Nat}$ is $? \times ? \to ? \to ?$.

The next definition defines a means of constructing a casting morphism that casts a type A to its skeleton and vice versa. This definition is by mutual recursion on the input type.

Definition 3.7. Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. Then for any object A whose skeleton is S we define the morphisms $\widehat{\mathsf{box}}_A : A \longrightarrow S$ and $\widehat{\mathsf{unbox}}_A : S \longrightarrow A$ by mutual recursion on A as follows:

$$
\begin{array}{l|l}
\widehat{\mathsf{box}}_A \quad \mathsf{box}_A & \widehat{\mathsf{unbox}}_A \quad \mathsf{unbox}_A \\
\quad \text{when } A \text{ is atomic} & \quad \text{when } A \text{ is atomic} \\
& \\
\widehat{\mathsf{box}}_? \quad \mathsf{id}_? & \widehat{\mathsf{unbox}}_? \quad \mathsf{id}_? \\
& \\
\widehat{\mathsf{box}}_{A_1 \to A_2} \quad \widehat{\mathsf{unbox}}_{A_1} \to \widehat{\mathsf{box}}_{A_2} & \widehat{\mathsf{unbox}}_{A_1 \to A_2} \quad \widehat{\mathsf{box}}_{A_1} \to \widehat{\mathsf{unbox}}_{A_2} \\
& \\
\widehat{\mathsf{box}}_{A_1 \times A_2} \quad \widehat{\mathsf{box}}_{A_1} \times \widehat{\mathsf{box}}_{A_2} & \widehat{\mathsf{unbox}}_{A_1 \times A_2} \quad \widehat{\mathsf{unbox}}_{A_1} \times \widehat{\mathsf{unbox}}_{A_2}
\end{array}
$$

The definition of both $\widehat{\mathsf{box}}$ or $\widehat{\mathsf{unbox}}$ uses the functor $- \to - : C^{\mathsf{op}} \times C \longrightarrow C$ which is contravariant in its first argument, and thus, in that contravariant position we must make a recursive call to the opposite function, and hence, they must be mutually defined. Every call to either $\widehat{\mathsf{box}}$ or $\widehat{\mathsf{unbox}}$ in the previous definition is on a smaller object than the input object. Thus, their definitions are well founded. Furthermore, $\widehat{\mathsf{box}}$ and $\widehat{\mathsf{unbox}}$ form a retract between A and S.

Lemma 3.8 (Boxing and Unboxing Lifted Retract). *Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. Then for any object A,*

$$\widehat{\mathsf{box}}_A ; \widehat{\mathsf{unbox}}_A \quad \mathsf{id}_A : A \longrightarrow A.$$

Proof. This proof holds by induction on the form A. Please see Appendix B.1 for the complete proof. □

As an example, suppose we wanted to cast the type $\mathsf{Nat} \times ? \to \mathsf{Nat}$ to its skeleton $? \times ? \to ?$. Then we can obtain a casting morphisms that will do this as follows:

$$
\begin{array}{ll}
\widehat{\mathsf{box}}_{\mathsf{Nat} \times ? \to \mathsf{Nat}} & \widehat{\mathsf{unbox}}_{\mathsf{Nat} \times ?} \to \widehat{\mathsf{box}}_{\mathsf{Nat}} \\
& \widehat{\mathsf{unbox}}_{\mathsf{Nat}} \times \widehat{\mathsf{unbox}}_? \to \widehat{\mathsf{box}}_{\mathsf{Nat}} \\
& \mathsf{unbox}_{\mathsf{Nat}} \times \mathsf{id}_? \to \mathsf{box}_{\mathsf{Nat}}
\end{array}
$$

We can also cast a morphism $A \xrightarrow{f} B$ to a morphism

$$S_1 \xrightarrow{\widehat{\mathsf{unbox}}_A} A \xrightarrow{f} B \xrightarrow{\widehat{\mathsf{box}}_B} S_2$$

where $S_1 \quad \mathbf{skeleton}\, A$ and $S_2 \quad \mathbf{skeleton}\, B$. Now if we have a second

$$S_2 \xrightarrow{\widehat{\mathsf{unbox}}_B} B \xrightarrow{g} C \xrightarrow{\widehat{\mathsf{box}}_C} S_3$$

then their composition reduces to composition at the typed level:

The right most diagram commutes because B is a retract of S_2, and the left unannotated arrow is the composition $\widehat{\mathsf{unbox}}_A; f; g; \widehat{\mathsf{box}}_C$. This tells us that we have a functor $\mathsf{S} : C \longrightarrow \mathcal{S}$:

$$\begin{array}{ll} \mathsf{S}A & \textbf{skeleton}\,A \\ \mathsf{S}f : A \longrightarrow B & \widehat{\mathsf{unbox}}_A; f; \widehat{\mathsf{box}}_A \end{array}$$

where $\mathcal{S}$ is the full subcategory of C consisting of the skeletons and morphisms between them, that is, $\mathcal{S}$ is a cartesian closed category with one basic object ? such that $\mathcal{S}, ?, \mathsf{split}, \mathsf{squash}$ is an untyped λ-model. The following turns out to be true.

Lemma 3.9 (S is faithful). *Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model, and $\mathcal{S}, ?, \mathsf{split}, \mathsf{squash}$ is the category of skeletons. Then the functor $\mathsf{S} : C \longrightarrow \mathcal{S}$ is faithful.*

Proof. This proof follows from the definition S and Lemma 3.8. For the full proof see Appendix B.2. □

Thus, we can think of the functor S as an injection of the typed world into the untyped one.

Now that we can cast any type into its skeleton we must show that every skeleton can be cast to ?. We do this similarly to the above and lift split and squash to arbitrary skeletons.

Definition 3.10. Suppose $\mathcal{S}, ?, \mathsf{split}, \mathsf{squash}$ is the category of skeletons. Then for any skeleton S we define the morphisms $\widehat{\mathsf{squash}}_S : S \longrightarrow ?$ and $\widehat{\mathsf{split}}_S : ? \longrightarrow S$ by mutual recursion on S as follows:

$$\begin{array}{ll|ll} \widehat{\mathsf{squash}}_? & \mathsf{id}_? & \widehat{\mathsf{split}}_? & \mathsf{id}_? \\ \\ \widehat{\mathsf{squash}}_{S_1 \to S_2} & \widehat{\mathsf{split}}_{S_1} \to \widehat{\mathsf{squash}}_{S_2}; \mathsf{squash}_{? \to ?} & \widehat{\mathsf{split}}_{S_1 \to S_2} & \mathsf{split}_{? \to ?}; \widehat{\mathsf{squash}}_{S_1} \to \widehat{\mathsf{split}}_{S_2} \\ \\ \widehat{\mathsf{squash}}_{S_1 \times S_2} & \widehat{\mathsf{squash}}_{S_1} \times \widehat{\mathsf{squash}}_{S_2}; \mathsf{squash}_{? \times ?} & \widehat{\mathsf{split}}_{S_1 \times S_2} & \mathsf{split}_{? \times ?}; \widehat{\mathsf{split}}_{S_1} \times \widehat{\mathsf{split}}_{S_2} \end{array}$$

As an example we will construct the casting morphism that casts the skeleton $? \times ? \to ?$ to ?:

$$\begin{array}{ll} \widehat{\mathsf{squash}}_{? \times ? \to ?} & \widehat{\mathsf{split}}_{? \times ?} \to \widehat{\mathsf{squash}}_?; \mathsf{squash}_{? \to ?} \\ & \mathsf{split}_{? \times ?}; \widehat{\mathsf{split}}_? \times \widehat{\mathsf{split}}_? \to \widehat{\mathsf{squash}}_?; \mathsf{squash}_{? \to ?} \\ & \mathsf{split}_{? \times ?}; \mathsf{id}_? \times \mathsf{id}_? \to \mathsf{id}_?; \mathsf{squash}_{? \to ?} \\ & \mathsf{split}_{? \times ?} \to \mathsf{id}_?; \mathsf{squash}_{? \to ?} \end{array}$$

The morphisms $\widehat{\mathsf{split}}_S$ and $\widehat{\mathsf{squash}}_S$ form a retract between S and ?.

Lemma 3.11 (Splitting and Squashing Lifted Retract). *Suppose $\mathcal{S}, ?, \mathsf{split}, \mathsf{squash}$ is the category of skeletons. Then for any skeleton S,*

$$\widehat{\mathsf{squash}}_S; \widehat{\mathsf{split}}_S \quad \mathsf{id}_S : S \longrightarrow S$$

Proof. The proof is similar to the proof of the boxing and unboxing lifted retract (Lemma 3.8). □

There is also a faithful functor from $\mathcal{S}$ to $\mathcal{U}$ where $\mathcal{U}$ is the full subcategory of $\mathcal{S}$ that consists of the single object ? and all its morphisms between it:

$$\begin{array}{ll} \mathsf{U}S & ? \\ \mathsf{U}f : S_1 \longrightarrow S_2 & \widehat{\mathsf{split}}_{S_1}; f; \widehat{\mathsf{squash}}_{S_2} \end{array}$$

This finally implies that there is a functor $\mathsf{C} : C \longrightarrow \mathcal{U}$ that injects all of C into the object ?.

Lemma 3.12 (Casting to ?). *Suppose $\mathcal{T}, \mathcal{C}, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model, $\mathcal{S}, ?, \mathsf{split}, \mathsf{squash}$ is the full subcategory of skeletons, and $\mathcal{U}$, ? is the full subcategory containing only ? and its morphisms. Then there is a faithful functor* C $\mathcal{C} \xrightarrow{\mathsf{S}} \mathcal{S} \xrightarrow{\mathsf{U}} \mathcal{U}$.

In a way we can think of $\mathsf{C} : \mathcal{C} \longrightarrow \mathcal{U}$ as a forgetful functor. It forgets the type information.

Getting back the typed information is harder. There is no nice functor from $\mathcal{U}$ to $\mathcal{C}$, because we need more information. However, given a type A we can always obtain a casting morphism from ? to A by $\widehat{\mathsf{split}}_{\mathbf{skeleton}\,A}; \widehat{\mathsf{unbox}}_A : ? \longrightarrow A$. Finally, we have the following result.

Lemma 3.13 (Casting Morphisms to ?). *Suppose $\mathcal{T}, \mathcal{C}, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model, and A is an object of $\mathcal{C}$. Then there exists casting morphisms from A to ? and vice versa that make A a retract of ?.*

Proof. The two morphisms are as follows:

$$\mathsf{Box}_A : \widehat{\mathsf{box}}_A; \widehat{\mathsf{squash}}_{\mathbf{skeleton}\,A} : A \longrightarrow ?$$

$$\mathsf{Unbox}_A : \widehat{\mathsf{split}}_{\mathbf{skeleton}\,A}; \widehat{\mathsf{unbox}}_A : ? \longrightarrow A$$

The fact the these form a retract between A and ? holds by Lemma 3.8 and Lemma 3.11. □

3.2 The Interpretation

In this section we show how to interpret $\lambda^{?}_{\to}$ and $\lambda^{\langle A \rangle}_{\to}$ into the categorical model given in the previous section. We complete the three steps summarized above. We will show how to interpret the typing of the former into the model, and then show how to do the same for the latter, furthermore, we show that reduction can be interpreted into the model as well, thus concluding soundness for $\lambda^{\langle A \rangle}_{\to}$ with respect to our model.

First, we must give the interpretation of types and contexts, but this interpretation is obvious, because we have been making sure to match the names of types and objects throughout this paper.

Definition 3.14. Suppose $\mathcal{T}, \mathcal{C}, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. Then we define the interpretation of types into $\mathcal{C}$ as follows:

$$\begin{array}{ll}
[\![\mathsf{Unit}]\!] & 1 \\
[\![\mathsf{Nat}]\!] & \mathsf{Nat} \\
[\![?]\!] & ? \\
[\![A_1 \to A_2]\!] & [\![A_1]\!] \to [\![A_2]\!] \\
[\![A_1 \times A_2]\!] & [\![A_1]\!] \times [\![A_2]\!]
\end{array}$$

We extend this interpretation to typing contexts as follows:

$$\begin{array}{ll}
[\![\cdot]\!] & 1 \\
[\![\Gamma, x : A]\!] & [\![\Gamma]\!] \times [\![A]\!]
\end{array}$$

Throughout the remainder of this paper we will drop the interpretation symbols around types.

Before we can interpret the typing rules of $\lambda^{?}_{\to}$ and $\lambda^{\langle A \rangle}_{\to}$ we must show how to interpret the consistency relation from Figure 2. These will correspond to casting morphisms.

Lemma 3.15 (Type Consistency in the Model). *Suppose $\mathcal{T}, \mathcal{C}, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model, and << A B >> for some types A and B. Then there are two casting morphisms $c_1 : A \longrightarrow B$ and $c_2 : B \longrightarrow A$.*

Proof. This proof holds by induction on the form $<< A\ B >>$ using the morphisms $\mathsf{Box}_A : A \longrightarrow ?$ and $\mathsf{Unbox}_A : ? \longrightarrow A$. Please see Appendix B.3 for the complete proof. $\square$

Corollary 3.16. *Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. Then we know the following:*

i. If $<< A1- > B1\ A2- > B2 >>$, then there are casting morphisms:

$$\begin{array}{ll} c & c_1 \to c_2 : A_1 \to B_1 \longrightarrow A_2 \to B_2 \\ c' & c_3 \to c_4 : A_2 \to B_2 \longrightarrow A_1 \to B_1 \end{array}$$

where $c_1 : A_2 \longrightarrow A_1$ and $c_2 : B_1 \longrightarrow B_2$, and $c_3 : A_1 \longrightarrow A_2$ and $c_4 : B_2 \longrightarrow B_1$.

ii. If $<< A1xB1\ A2xB2 >>$, then there are casting morphisms:

$$\begin{array}{ll} c & c_1 \times c_2 : A_1 \times B_1 \longrightarrow A_2 \times B_2 \\ c' & c_3 \times c_4 : A_2 \times B_2 \longrightarrow A_1 \times B_1 \end{array}$$

where $c_1 : A_1 \longrightarrow A_2$ and $c_2 : B_1 \longrightarrow B_2$, and $c_3 : A_2 \longrightarrow A_1$ and $c_4 : B_2 \longrightarrow B_1$.

Proof. This proof holds by the construction of the casting morphisms from the proof of the previous result, and the fact that the type consistency rules are unique for each type. $\square$

Showing that both c_1 and c_2 exist corresponds to the fact that $<< A\ B >>$ is symmetric. But, this interpretation is an over approximation of type consistency, because type consistency is not transitive, but function composition is. Leaving type consistency implicit in the model just does not make good sense categorically, because it would break composition. For example, if we have morphisms $f : A \longrightarrow ?$ and $g : B \longrightarrow C$, then if we implicitly allowed ? to be cast to B, then we could compose these two morphisms, but this does not fit the definition of a category, because it requires the target of f to match the source of g, but this just is not the case. Thus, the explicit cast must be used to obtain $f; \widehat{\mathsf{unbox}_B}; g$.

At this point we have everything we need to show our main result which is that typing in both $\lambda^?_\to$ and $\lambda^{\langle A \rangle}_\to$, and evaluation in $\lambda^{\langle A \rangle}$ can be interpreted into the categorical model.

Theorem 3.17 (Interpretation of Typing). *Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. If $\Gamma \vdash t : A$ in either $\lambda^?_\to$ or $\lambda^{\langle A \rangle}_\to$, then there is a morphism $[\![t]\!] : [\![\Gamma]\!] \longrightarrow A$ in C.*

Proof. The proof holds by induction on the form of $\Gamma \vdash t : A$ and uses most of the results we have developed up to this point. Please see Appendix B.4 for the complete proof. $\square$

Theorem 3.18 (Interpretation of Evaluation). *Suppose $\mathcal{T}, C, ?, \mathsf{T}, \mathsf{split}, \mathsf{squash}, \mathsf{box}, \mathsf{unbox}$ is a gradual λ-model. If $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$, then $[\![t_1]\!] \quad [\![t_2]\!] : [\![\Gamma]\!] \longrightarrow A$.*

Proof. This proof holds by induction on the form of $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$, and uses Theorem 3.17, Lemma 3.15, and Corollary 3.16. Please see Appendix B.5 for the complete proof. $\square$

4 SIMPLY TYPED CORE GRADY

Just as the simply typed λ-calculus corresponds to cartesian closed categories our categorical model has a corresponding type theory we call Grady. It consists of all of the structure found in the model. Its syntax is an extension of the syntax for $\lambda^?_\to$.

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\,var \qquad \frac{}{\Gamma \vdash \mathsf{box}_T : T \to ?}\,\mathrm{box} \qquad \frac{}{\Gamma \vdash \mathsf{unbox}_T : ? \to T}\,\mathrm{unbox} \qquad \frac{}{\Gamma \vdash \mathsf{squash}_U : U \to ?}\,\mathrm{squash}$$

$$\frac{}{\Gamma \vdash \mathsf{split}_U : ? \to U}\,\mathrm{split} \qquad \frac{}{\Gamma \vdash \mathsf{triv} : \mathsf{Unit}}\,unit \qquad \frac{}{\Gamma \vdash 0 : \mathsf{Nat}}\,zero \qquad \frac{\Gamma \vdash t : \mathsf{Nat}}{\Gamma \vdash \mathsf{succ}\,t : \mathsf{Nat}}\,succ$$

$$\frac{\begin{array}{l}\Gamma \vdash t : \mathsf{Nat}\\ \Gamma \vdash t_1 : A \quad \Gamma, x : \mathsf{Nat} \vdash t_2 : A\end{array}}{\Gamma \vdash \mathsf{case}\,t\,\mathsf{of}\,0 \to t_1, \mathsf{succ}\,x \to t_2 : A}\,\mathsf{Nat}_e \qquad \frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash t_1, t_2 : A_1 \times A_2}\,pair \qquad \frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \mathsf{fst}\,t : A_1}\,fst$$

$$\frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \mathsf{snd}\,t : A_2}\,snd \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A_1.t : A \to B}\,lam \qquad \frac{\Gamma \vdash t_1 : A \to B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\,t_2 : B}\,app$$

Fig. 6. Typing rules for Grady

Definition 4.1. Syntax for Grady:

$$\begin{array}{lll}
\text{(basic skeletons)} & U & :: ? \to ? \mid ? \times ? \\
\text{(skeletons)} & S & :: ? \mid S_1 \times S_2 \mid S_1 \to S_2 \\
\text{(atomic types)} & C & :: \mathsf{Unit} \mid \mathsf{Nat} \\
\text{(terms)} & t & :: \ldots \mid \mathsf{split}_U \mid \mathsf{squash}_U \mid \mathsf{box}_C \mid \mathsf{unbox}_C \mid \mathsf{case}\,t\,\mathsf{of}\,0 \to t_1, \mathsf{succ}\,x \to t_2 \\
\text{(natural numbers)} & n & :: 0 \mid \mathsf{succ}\,n \\
\text{(simple values)} & s & :: x \mid \mathsf{triv} \mid n \mid \mathsf{squash}_U \mid \mathsf{split}_U \mid \mathsf{box}_C \mid \mathsf{unbox}_C
\end{array}$$

The types of Grady are the same as the types of $\lambda^?_\to$ (Definition 2.1), in addition, it encompasses all the terms of $\lambda^?_\to$, and so we do not repeat either of them here. The typing rules for Grady can be found in Figure 6 and its reduction rules can be found in Figure 7.

Just as we did for the categorical model (Lemma 3.13) we can lift box_C and unbox_C to arbitrary type.

Lemma 4.2 (Syntactic Box_A and Unbox_A). *Given any type A there are functions Box_A and Unbox_A such that the following typing rules are admissible:*

$$\frac{}{\Gamma \vdash \mathsf{Box}_A : A \to ?}\,Box \qquad \frac{}{\Gamma \vdash \mathsf{Unbox}_A : ? \to A}\,Unbox$$

Furthermore, the following reduction rule is admissible:

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{Unbox}_A\,\mathsf{Box}_A\,t \rightsquigarrow t : A}\,retract_3$$

Proof. The functions Box_A and Unbox_A can be defined using the construction from the categorical model, e.g. Definition 3.7, Definition 3.10, and Lemma 3.13. However, the categorical notions of composition, identity, and the functors $- \to -$ and $- \times -$ must be defined as meta-functions first, but after they are, then the same constructions apply. Please see Appendix B.6 for the constructions. □

Perhaps unsurprisingly, due to our results with respect to the categorical model, we can use the previous result to construct a type-directed translation of both $\lambda^?_\to$ and $\lambda^{\langle A\rangle}_\to$ into Grady.

$$\frac{\Gamma \vdash s : A}{\Gamma \vdash s \rightsquigarrow s : A}\ \text{values} \qquad \frac{\Gamma \vdash t : T}{\Gamma \vdash \mathsf{unbox}_T\ \mathsf{box}_T\ t \rightsquigarrow t : T}\ \text{retract}_1 \qquad \frac{\Gamma \vdash t : U}{\Gamma \vdash \mathsf{split}_U\ \mathsf{squash}_U\ t \rightsquigarrow t : U}\ \text{retract}_2$$

$$\frac{\Gamma, x : A_1 \vdash t_1 : A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash \lambda x : A_1.t_1\ v \rightsquigarrow [t_2/x]t_2 : A_2}\ \beta \qquad \frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash \mathsf{fst}\ t_1, t_2 \rightsquigarrow t_1 : A_1}\ \times_{e_1} \qquad \frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash \mathsf{snd}\ t_1, t_2 \rightsquigarrow t_2 : A_2}\ \times_{e_2}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' : \mathsf{Nat}}{\Gamma \vdash \mathsf{succ}\ t \rightsquigarrow \mathsf{succ}\ t' : \mathsf{Nat}}\ \text{succ} \qquad \frac{\Gamma \vdash t_1 : A \quad \Gamma, x : \mathsf{Nat} \vdash t_2 : A}{\Gamma \vdash \mathsf{case}\ 0\ \mathsf{of}\ 0 \to t_1, \mathsf{succ}\ x \to t_2 \rightsquigarrow t_1 : A}\ \mathsf{Nat}_{e_0}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : \mathsf{Nat}\\ \Gamma \vdash t_1 : A \quad \Gamma, x : \mathsf{Nat} \vdash t_2 : A\end{array}}{\Gamma \vdash \mathsf{case}\ \mathsf{succ}\ t\ \mathsf{of}\ 0 \to t_1, \mathsf{succ}\ x \to t_2 \rightsquigarrow [t/x]t_2 : A}\ \mathsf{Nat}_{e_1}$$

$$\frac{\begin{array}{l}\Gamma \vdash t \rightsquigarrow t' : \mathsf{Nat}\\ \Gamma \vdash t_1 : A \quad \Gamma, x : \mathsf{Nat} \vdash t_2 : A\end{array}}{\Gamma \vdash \mathsf{case}\ t\ \mathsf{of}\ 0 \to t_1, \mathsf{succ}\ x \to t_2 \rightsquigarrow \mathsf{case}\ t'\ \mathsf{of}\ 0 \to t_1, \mathsf{succ}\ x \to t_2 : A}\ \text{case}_1$$

$$\frac{\Gamma \vdash t_1 \rightsquigarrow t_1' : A_1 \to A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash t_1\ t_2 \rightsquigarrow t_1'\ t_2 : A_2}\ \to_{e_1} \qquad \frac{\Gamma \vdash v : A_1 \to A_2 \quad \Gamma \vdash t \rightsquigarrow t' : A_1}{\Gamma \vdash v\ t \rightsquigarrow v\ t' : A_2}\ \to_{e_2}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' : A_1 \times A_2}{\Gamma \vdash \mathsf{fst}\ t \rightsquigarrow \mathsf{fst}\ t' : A_1}\ \mathsf{fst} \qquad \frac{\Gamma \vdash t \rightsquigarrow t' : A_1 \times A_2}{\Gamma \vdash \mathsf{snd}\ t \rightsquigarrow \mathsf{snd}\ t' : A_2}\ \mathsf{snd} \qquad \frac{\Gamma \vdash t_1 \rightsquigarrow t_1' : A_1 \quad \Gamma \vdash t_2 : A_2}{\Gamma \vdash t_1, t_2 \rightsquigarrow t_1', t_2 : A_1 \times A_2}\ \times_1$$

$$\frac{\Gamma \vdash t_1 : A_1 \quad \Gamma \vdash t_2 \rightsquigarrow t_2' : A_2}{\Gamma \vdash t_1, t_2 \rightsquigarrow t_1, t_2' : A_1 \times A_2}\ \times_2$$

Fig. 7. Reduction rules for Grady

Lemma 4.3 (Translations).

i. *If* $\Gamma \vdash t : A$ *hold in either* $\lambda^{?}_{\to}$ *or* $\lambda^{\langle A \rangle}_{\to}$*, then there exists a term* t' *such that* $\Gamma \vdash t' : A$ *holds in Grady.*

ii. *If* $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$ *holds in* $\lambda^{\langle A \rangle}_{\to}$*, then* $\Gamma \vdash t_1' \rightsquigarrow t_2' : A$ *holds in Grady, where* $\Gamma \vdash t_1' : A$ *and* $\Gamma \vdash t_2' : A$ *are both the corresponding Grady terms.*

Proof. The proof of this result is similar to the proof that both $\lambda^{?}_{\to}$ and $\lambda^{\langle A \rangle}_{\to}$ can be interpreted into the categorical model, Theorem 3.17 and Theorem 3.18, and thus, we do not give the full proof. The proof of part one holds by induction on $\Gamma \vdash t : A$, and using the realization that if $A \sim B$ for some types A and B then there are casting terms $\cdot \vdash c_1 : A \to B$ and $\cdot \vdash c_2 : B \to A$ following the proof of Lemma 3.15. The proof of part two holds by induction on $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$ making use of part one; it is similar to the proof of Theorem 3.18. □

4.1 Exploiting the Untyped λ-Calculus

Having the untyped λ-calculus along side the typed λ-calculus can be a lot of fun. This section can be seen from two perspectives: it gives a number of examples in Grady, and shows several ways the typed and untyped fragments can be mixed.

Michael is writing this section.

$$\frac{\Gamma \vdash A : \star}{\Gamma \vdash A \lesssim A}\,\text{refl} \qquad \frac{\Gamma \vdash A : \star}{\Gamma \vdash A \lesssim \top}\,S\_Top \qquad \frac{X <: A' \in \Gamma \quad \Gamma \vdash A' \sim A}{\Gamma \vdash X \lesssim A}\,\text{var} \qquad \frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash A \lesssim ?}\,\mathsf{box}$$

$$\frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash ? \lesssim A}\,\mathsf{unbox} \qquad \frac{\Gamma\,\mathrm{Ok}}{\Gamma \vdash ? \lesssim \mathbb{S}}\,S\_USL \qquad \frac{\Gamma\,\mathrm{Ok}}{\Gamma \vdash \mathsf{Nat} \lesssim \mathbb{S}}\,S\_NatSL \qquad \frac{\Gamma\,\mathrm{Ok}}{\Gamma \vdash \mathsf{Unit} \lesssim \mathbb{S}}\,S\_UnitSL$$

$$\frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash \mathsf{List}\,A \lesssim \mathbb{S}}\,S\_ListSL \qquad \frac{\Gamma \vdash A \lesssim \mathbb{S} \quad \Gamma \vdash B \lesssim \mathbb{S}}{\Gamma \vdash A \times B \lesssim \mathbb{S}}\,S\_ProdSL \qquad \frac{\Gamma \vdash A \lesssim \mathbb{S} \quad \Gamma \vdash B \lesssim \mathbb{S}}{\Gamma \vdash A \to B \lesssim \mathbb{S}}\,S\_ArrowSL$$

$$\frac{\Gamma \vdash A \lesssim B}{\Gamma \vdash \mathsf{List}\,A \lesssim \mathsf{List}\,B}\,\mathsf{List} \qquad \frac{\Gamma \vdash A_1 \lesssim A_2 \quad \Gamma \vdash B_1 \lesssim B_2}{\Gamma \vdash A_1 \times B_1 \lesssim A_2 \times B_2}\,\times \qquad \frac{\Gamma \vdash A_2 \lesssim A_1 \quad \Gamma \vdash B_1 \lesssim B_2}{\Gamma \vdash A_1 \to B_1 \lesssim A_2 \to B_2}\,\to$$

$$\frac{\Gamma, X <: A \vdash B_1 \lesssim B_2}{\Gamma \vdash \forall X <: A.B_1 \lesssim \forall X <: A.B_2}\,\forall$$

Fig. 8. Subtyping for Surface Grady

$$\frac{\Gamma \vdash A : \star}{\Gamma \vdash A \sim A}\,\text{refl} \qquad \frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash A \sim ?}\,\mathsf{box} \qquad \frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash ? \sim A}\,\mathsf{unbox} \qquad \frac{\Gamma \vdash A \sim B}{\Gamma \vdash \mathsf{List}\,A \sim \mathsf{List}\,B}\,\mathsf{List}$$

$$\frac{\Gamma \vdash A_2 \sim A_1 \quad \Gamma \vdash B_1 \sim B_2}{\Gamma \vdash A_1 \to B_1 \sim A_2 \to B_2}\,\to \qquad \frac{\Gamma \vdash A_1 \sim A_2 \quad \Gamma \vdash B_1 \sim B_2}{\Gamma \vdash A_1 \times B_1 \sim A_2 \times B_2}\,\times \qquad \frac{\Gamma, X <: A \vdash B_1 \sim B_2}{\Gamma \vdash \forall X <: A.B_1 \sim \forall X <: A.B_2}\,\forall$$

Fig. 9. Type consistency for Surface Grady

- Church Encoded Data
- Y-combinator and the natural number eliminator, e.g. terminating recursion on natural numbers
- Scott Encoded data, this is not available in terminating type theories
- Parigot Encoded Data, better efficiency

5 GRADY: A CATEGORICALLY INSPIRED GRADUAL TYPE SYSTEM

5.1 Surface Grady: A Gradual Type System

5.2 Core Grady: The Casting Calculus

5.3 Analyzing Grady

Lemma 5.1 (Inclusion of Bounded System F). *Suppose t is fully annotated and does not contain any applications of* box *or* unbox*, and A is static. Then*

i. $\Gamma \vdash_F t : A$ *if and only if* $\Gamma \vdash_{\mathsf{SG}} t : A$*, and*
ii. $t \rightsquigarrow_F^* t'$ *if and only if* $t \rightsquigarrow^* t'$.

Proof. We give proof sketches for both parts. The interesting cases are the right-to-left directions of each part. If we simply remove all rules mentioning the unknown type ? and the type consistency relation, and then remove box, unbox,

$$\frac{x : A \in \Gamma \quad \Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} x : A}\,\mathsf{var} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{box} : \forall X <: \mathbb{S}.X \to ?}\,\mathsf{box} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{unbox} : \forall X <: \mathbb{S}.? \to X}\,\mathsf{unbox}$$

$$\frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{triv} : \mathsf{Unit}}\,\mathsf{Unit} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} 0 : \mathsf{Nat}}\,\mathsf{zero} \qquad \frac{\Gamma \vdash_{\mathsf{SG}} t : A \quad \mathsf{nat}A\ \ \mathsf{Nat}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{succ}\,t : \mathsf{Nat}}\,\mathsf{succ}$$

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathsf{SG}} t : C \quad \mathsf{nat}C\ \ \mathsf{Nat} \quad \Gamma \vdash A_1 \sim A \\ \Gamma \vdash_{\mathsf{SG}} t_1 : A_1 \quad \Gamma, x : \mathsf{Nat} \vdash_{\mathsf{SG}} t_2 : A_2 \quad \Gamma \vdash A_2 \sim A\end{array}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{case}\,t\,\mathsf{of}\,0 \to t_1, \mathsf{succ}\,x \to t_2 : A}\,\mathsf{Nat}_e \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash_{\mathsf{SG}} [\,] : \forall X <: \top.\mathsf{List}\,X}\,\mathsf{empty}$$

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{SG}} t_2 : A_2 \quad \mathsf{list}A_2\ \ \mathsf{List}\,A_3 \quad \Gamma \vdash A_1 \sim A_3}{\Gamma \vdash_{\mathsf{SG}} t_1 :: t_2 : \mathsf{List}\,A_3}\,\mathsf{List}_i$$

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathsf{SG}} t : C \quad \mathsf{list}C\ \ \mathsf{List}\,A \\ \Gamma \vdash_{\mathsf{SG}} t_1 : B_1 \quad \Gamma, x : A, y : \mathsf{List}\,A \vdash_{\mathsf{SG}} t_2 : B_2 \quad \Gamma \vdash B_1 \sim B \quad \Gamma \vdash B_2 \sim B\end{array}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{case}\,t\,\mathsf{of}\,[\,] \to t_1, x :: y \to t_2 : B}\,\mathsf{List}_e$$

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{SG}} t_2 : A_2}{\Gamma \vdash_{\mathsf{SG}} t_1, t_2 : A_1 \times A_2}\,\times_i \qquad \frac{\Gamma \vdash_{\mathsf{SG}} t : B \quad \mathsf{prod}B\ \ A_1 \times A_2}{\Gamma \vdash_{\mathsf{SG}} \mathsf{fst}\,t : A_1}\,\times_{e_1}$$

$$\frac{\Gamma \vdash_{\mathsf{SG}} t : B \quad \mathsf{prod}B\ \ A_1 \times A_2}{\Gamma \vdash_{\mathsf{SG}} \mathsf{snd}\,t : A_2}\,\times_{e_2} \qquad \frac{\Gamma, x : A \vdash_{\mathsf{SG}} t : B}{\Gamma \vdash_{\mathsf{SG}} \lambda x : A.t : A \to B}\,\to_i \qquad \frac{\begin{array}{l}\Gamma \vdash_{\mathsf{SG}} t_1 : C \quad \mathsf{fun}C\ \ A_1 \to B_1 \\ \Gamma \vdash_{\mathsf{SG}} t_2 : A_2 \quad \Gamma \vdash A_2 \sim A_1\end{array}}{\Gamma \vdash_{\mathsf{SG}} t_1\,t_2 : B_1}\,\to_e$$

$$\frac{\Gamma, X <: A \vdash_{\mathsf{SG}} t : B}{\Gamma \vdash_{\mathsf{SG}} \Lambda X <: A.t : \forall X <: A.B}\,\forall_i \qquad \frac{\Gamma \vdash_{\mathsf{SG}} t : \forall X <: B.C \quad \Gamma \vdash A \lesssim B}{\Gamma \vdash_{\mathsf{SG}} [A]t : [A/X]C}\,\forall_e \qquad \frac{\Gamma \vdash_{\mathsf{SG}} t : A \quad \Gamma \vdash A \lesssim B}{\Gamma \vdash_{\mathsf{SG}} t : B}\,\mathsf{sub}$$

Fig. 10. Typing rules for Surface Grady

and ? from the syntax of Surface Grady, then what we are left with is bounded system F. Since t is fully annotated and A is static, then $\Gamma \vdash_{\mathsf{SG}} t : A$ will hold within this fragment.

Moving on to part two, first, we know that t does not contain any occurrence of box or unbox and is fully annotated. This implies that t lives within the bounded system F fragment of Surface Grady. Thus, before evaluation of t Surface Grady will apply the cast insertion algorithm which will at most insert applications of the identity function into t producing a term $\widehat{t}$, but then after potentially more than one step of evaluation within Core Grady, those applications of the identity function will be β-reduced away resulting in $\widehat{t} \rightsquigarrow^* t \rightsquigarrow^* t'$. In addition, since t in Surface Grady is the exact same program as t in bounded system F, then we know $t \rightsquigarrow^*_F t'$ will hold. □

Lemma 5.2 (Inclusion of DTLC). *Suppose t is a closed term of DTLC. Then*

i. $\cdot \vdash_{\mathsf{SG}} \lceil t \rceil : ?$*, and*

ii. $t \rightsquigarrow^*_{DTLC} t'$ *if and only if* $\lceil t \rceil \rightsquigarrow^* \lceil t' \rceil$.

Proof. In this case DTLC is embedded into the simply typed fragment of Grady, and hence, this proof is the same result proven by (Siek and Taha 2006), and (Siek et al. 2015). □

$$\frac{x : A \in \Gamma \quad \Gamma\,\mathsf{Ok}}{\Gamma \vdash x \Rightarrow x : A} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash \mathsf{box} \Rightarrow \mathsf{box} : \forall X <: \mathbb{S}.X \to ?} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash \mathsf{unbox} \Rightarrow \mathsf{unbox} : \forall X <: \mathbb{S}.? \to X}$$

$$\frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash 0 \Rightarrow 0 : \mathsf{Nat}} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash \mathsf{triv} \Rightarrow \mathsf{triv} : \mathsf{Unit}} \qquad \frac{\Gamma \vdash t_1 \Rightarrow t_2 : ?}{\Gamma \vdash \mathsf{succ}\,t_1 \Rightarrow \mathsf{succ}\,\mathsf{unbox}_{\mathsf{Nat}}\,t_2 : \mathsf{Nat}}$$

$$\frac{\Gamma \vdash t_1 \Rightarrow t_2 : \mathsf{Nat}}{\Gamma \vdash \mathsf{succ}\,t_1 \Rightarrow \mathsf{succ}\,t_2 : \mathsf{Nat}}$$

$$\frac{\begin{array}{c}\Gamma \vdash t \Rightarrow t' : ? \quad \Gamma \vdash A_1 \sim A \quad \mathsf{caster}\,A_1, A\ \ c_1 \\ \Gamma \vdash t_1 \Rightarrow t_1' : A_1 \quad \Gamma, x : \mathsf{Nat} \vdash t_2 \Rightarrow t_2' : A_2 \quad \Gamma \vdash A_2 \sim A \quad \mathsf{caster}\,A_2, A\ \ c_2\end{array}}{\Gamma \vdash \mathsf{case}\,t\,\mathsf{of}\,0 \to t_1, \mathsf{succ}\,x \to t_2 \Rightarrow \mathsf{case}\,\mathsf{unbox}_{\mathsf{Nat}}\,t'\,\mathsf{of}\,0 \to c_1\,t_1', \mathsf{succ}\,x \to c_2\,t_2' : A}$$

$$\frac{\begin{array}{c}\Gamma \vdash t \Rightarrow t' : \mathsf{Nat} \quad \Gamma \vdash A_1 \sim A \quad \mathsf{caster}\,A_1, A\ \ c_1 \\ \Gamma \vdash t_1 \Rightarrow t_1' : A_1 \quad \Gamma, x : \mathsf{Nat} \vdash t_2 \Rightarrow t_2' : A_2 \quad \Gamma \vdash A_2 \sim A \quad \mathsf{caster}\,A_2, A\ \ c_2\end{array}}{\Gamma \vdash \mathsf{case}\,t\,\mathsf{of}\,0 \to t_1, \mathsf{succ}\,x \to t_2 \Rightarrow \mathsf{case}\,t'\,\mathsf{of}\,0 \to t_1', \mathsf{succ}\,x \to t_2' : A}$$

$$\frac{\Gamma \vdash t_1 \Rightarrow t_3 : A_1 \quad \Gamma \vdash t_2 \Rightarrow t_4 : A_2}{\Gamma \vdash t_1, t_2 \Rightarrow t_3, t_4 : A_1 \times A_2} \qquad \frac{\Gamma \vdash t_1 \Rightarrow t_2 : ?}{\Gamma \vdash \mathsf{fst}\,t_1 \Rightarrow \mathsf{fst}\,\mathsf{split}_{? \times ?}\,t_2 : ?} \qquad \frac{\Gamma \vdash t_1 \Rightarrow t_2 : A_1 \times A_2}{\Gamma \vdash \mathsf{fst}\,t_1 \Rightarrow \mathsf{fst}\,t_2 : A_1}$$

$$\frac{\Gamma \vdash t_1 \Rightarrow t_2 : ?}{\Gamma \vdash \mathsf{snd}\,t_1 \Rightarrow \mathsf{snd}\,\mathsf{split}_{? \times ?}\,t_2 : ?} \qquad \frac{\Gamma \vdash t_1 \Rightarrow t_2 : A \times B}{\Gamma \vdash \mathsf{snd}\,t_1 \Rightarrow \mathsf{snd}\,t_2 : B} \qquad \frac{\Gamma\,\mathsf{Ok}}{\Gamma \vdash [] \Rightarrow [] : \forall X <: \top.\mathsf{List}\,X}$$

$$\frac{\Gamma \vdash t_1 \Rightarrow t_1' : A_1 \quad \Gamma \vdash t_2 \Rightarrow t_2' : \mathsf{List}\,A_2 \quad \Gamma \vdash A_1 \lesssim A_2 \quad \mathsf{caster}\,A_1, A_2\ \ c}{\Gamma \vdash t_1 :: t_2 \Rightarrow c\,t_1' :: t_2' : \mathsf{List}\,A_2}$$

$$\frac{\begin{array}{c}\Gamma \vdash t \Rightarrow t' : ? \quad \mathsf{caster}\,B_1, B\ \ c_1 \quad \mathsf{caster}\,B_2, B\ \ c_2 \\ \Gamma \vdash t_1 \Rightarrow t_1' : B_1 \quad \Gamma, x : ?, y : \mathsf{List}\,? \vdash t_2 \Rightarrow t_2' : B_2 \quad \Gamma \vdash B_1 \sim B \quad \Gamma \vdash B_2 \sim B\end{array}}{\Gamma \vdash \mathsf{case}\,t\,\mathsf{of}\,[] \to t_1, x :: y \to t_2 \Rightarrow \mathsf{case}\,\mathsf{split}_{\mathsf{List}\,?}\,t'\,\mathsf{of}\,[] \to c_1\,t_1', x :: y \to c_2\,t_2' : B}$$

$$\frac{\begin{array}{c}\Gamma \vdash t \Rightarrow t : \mathsf{List}\,A \quad \mathsf{caster}\,B_1, B\ \ c_1 \quad \mathsf{caster}\,B_2, B\ \ c_2 \\ \Gamma \vdash t_1 \Rightarrow t_1' : B_1 \quad \Gamma, x : A, y : \mathsf{List}\,A \vdash t_2 \Rightarrow t_2' : B_2 \quad \Gamma \vdash B_1 \sim B \quad \Gamma \vdash B_2 \sim B\end{array}}{\Gamma \vdash \mathsf{case}\,t\,\mathsf{of}\,[] \to t_1, x :: y \to t_2 \Rightarrow \mathsf{case}\,t'\,\mathsf{of}\,[] \to c_1\,t_1', x :: y \to c_2\,t_2' : B}$$

$$\frac{\Gamma, x : A_1 \vdash t_1 \Rightarrow t_2 : A_2}{\Gamma \vdash \lambda x : A_1.t_1 \Rightarrow \lambda x : A_1.t_2 : A_1 \to A_2} \qquad \frac{\begin{array}{l}\Gamma \vdash t_1 \Rightarrow t_1' : ? \\ \Gamma \vdash t_2 \Rightarrow t_2' : A_2 \quad \mathsf{caster}\,A_2, ?\ \ c\end{array}}{\Gamma \vdash t_1\,t_2 \Rightarrow \mathsf{split}_{? \to ?}\,t_1'\,c\,t_2' : ?}$$

$$\frac{\begin{array}{l}\Gamma \vdash t_2 \Rightarrow t_2' : A_2 \\ \Gamma \vdash t_1 \Rightarrow t_1' : A_1 \to B \quad \Gamma \vdash A_2 \sim A_1 \quad \mathsf{caster}\,A_2, A_1\ \ c\end{array}}{\Gamma \vdash t_1\,t_2 \Rightarrow t_1'\,c\,t_2' : B} \qquad \frac{\Gamma, X <: A \vdash t_1 \Rightarrow t_2 : B}{\Gamma \vdash \Lambda X <: A.t_1 \Rightarrow \Lambda X <: A.t_2 : \forall X <: A.B}$$

$$\frac{\Gamma \vdash t_1 \Rightarrow t_2 : \forall X <: B.C \quad \Gamma \vdash A \sim A' \quad \Gamma \vdash A' <: B}{\Gamma \vdash [A]t_1 \Rightarrow [A']t_2 : [A'/X]C}$$

Fig. 11. Cast insertion for Surface Grady

TODO

Fig. 12. Subtyping for Core Grady

$$\frac{x : A \in \Gamma \quad \Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} x : A}\ \text{var} \qquad \frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{box} : \forall X <: \mathbb{S}.X \to ?}\ \mathsf{box} \qquad \frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{unbox} : \forall X <: \mathbb{S}.? \to X}\ \mathsf{unbox}$$

$$\frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{squash}_K : K \to ?}\ \mathsf{squash} \qquad \frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{split}_K : ? \to K}\ \mathsf{split} \qquad \frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{triv} : \mathsf{Unit}}\ \mathsf{Unit}$$

$$\frac{\Gamma\ \mathrm{Ok}}{\Gamma \vdash_{\mathsf{CG}} 0 : \mathsf{Nat}}\ \text{zero} \qquad \frac{\Gamma \vdash_{\mathsf{CG}} t : \mathsf{Nat}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{succ}\, t : \mathsf{Nat}}\ \mathsf{succ} \qquad \frac{\begin{array}{c}\Gamma \vdash_{\mathsf{CG}} t : \mathsf{Nat} \\ \Gamma \vdash_{\mathsf{CG}} t_1 : A \quad \Gamma, x : \mathsf{Nat} \vdash_{\mathsf{CG}} t_2 : A\end{array}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{case}\, t : \mathsf{Nat}\, \mathsf{of}\, 0 \to t_1, \mathsf{succ}\, x \to t_2 : A}\ \mathsf{Nat}_e$$

$$\frac{\Gamma\ \mathrm{Ok} \quad \Gamma \vdash A : \star}{\Gamma \vdash_{\mathsf{CG}} [] : \forall X <: ?.\mathsf{List}\, X}\ \text{empty} \qquad \frac{\Gamma \vdash_{\mathsf{CG}} t_1 : A \quad \Gamma \vdash_{\mathsf{CG}} t_2 : \mathsf{List}\, A}{\Gamma \vdash_{\mathsf{CG}} t_1 :: t_2 : \mathsf{List}\, A}\ \mathsf{List}_i$$

$$\frac{\begin{array}{l}\Gamma \vdash_{\mathsf{CG}} t : \mathsf{List}\, A \\ \Gamma \vdash_{\mathsf{CG}} t_1 : B \quad \Gamma, x : A, y : \mathsf{List}\, A \vdash_{\mathsf{CG}} t_2 : B\end{array}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{case}\, t : \mathsf{List}\, A\, \mathsf{of}\, [] \to t_1, x :: y \to t_2 : B}\ \mathsf{List}_e \qquad \frac{\Gamma \vdash_{\mathsf{CG}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{CG}} t_2 : A_2}{\Gamma \vdash_{\mathsf{CG}} t_1, t_2 : A_1 \times A_2}\ \times_i$$

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : A_1 \times A_2}{\Gamma \vdash_{\mathsf{CG}} \mathsf{fst}\, t : A_1}\ \times_{e_1} \qquad \frac{\Gamma \vdash_{\mathsf{CG}} t : A_1 \times A_2}{\Gamma \vdash_{\mathsf{CG}} \mathsf{snd}\, t : A_2}\ \times_{e_2} \qquad \frac{\Gamma, x : A \vdash_{\mathsf{CG}} t : B}{\Gamma \vdash_{\mathsf{CG}} \lambda x : A.t : A \to B}\ \to_i$$

$$\frac{\Gamma \vdash_{\mathsf{CG}} t_1 : A \to B \quad \Gamma \vdash_{\mathsf{CG}} t_2 : A}{\Gamma \vdash_{\mathsf{CG}} t_1\, t_2 : B}\ \to_e \qquad \frac{\Gamma, X <: A \vdash_{\mathsf{CG}} t : B}{\Gamma \vdash_{\mathsf{CG}} \Lambda X <: A.t : \forall X <: A.B}\ \forall_i$$

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : \forall X <: B.C \quad \Gamma \vdash A <: B}{\Gamma \vdash_{\mathsf{CG}} [A]t : [A/X]C}\ \forall_e \qquad \frac{\Gamma \vdash_{\mathsf{CG}} t : A \quad \Gamma \vdash A <: B}{\Gamma \vdash_{\mathsf{CG}} t : B}\ \text{sub} \qquad \frac{}{\Gamma \vdash_{\mathsf{CG}} \mathsf{error}_A : A}\ \text{error}$$

Fig. 13. Typing rules for Core Grady

TODO

Fig. 14. Reduction rules for Core Grady

$$\frac{\Gamma \vdash A \lesssim \mathbb{S}}{A \sqsubseteq ?}\ ? \qquad \frac{}{A \sqsubseteq A}\ \text{refl} \qquad \frac{A \sqsubseteq C \quad B \sqsubseteq D}{A \to B \sqsubseteq C \to D}\ \to \qquad \frac{A \sqsubseteq C \quad B \sqsubseteq D}{A \times B \sqsubseteq C \times D}\ \times \qquad \frac{A \sqsubseteq B}{\mathsf{List}\, A \sqsubseteq \mathsf{List}\, B}\ \mathsf{List}$$

$$\frac{B_1 \sqsubseteq B_2}{\forall X <: A.B_1 \sqsubseteq \forall X <: A.B_2}\ \forall$$

Fig. 15. Type Precision

Lemma 5.3 (Left-to-Right Consistent Subtyping). *Suppose* $\Gamma \vdash A \lesssim B$.

i. $\Gamma \vdash A \sim A'$ *and* $\Gamma \vdash A' <: B$ *for some* A'.
ii. $\Gamma \vdash B' \sim B$ *and* $\Gamma \vdash A <: B'$ *for some* B'.

Proof. This is a proof by induction on $\Gamma \vdash A \lesssim B$. See Appendix B.7 for the complete proof. □

Corollary 5.4 (Consistent Subtyping).
i. $\Gamma \vdash A \lesssim B$ *if and only if* $\Gamma \vdash A \sim A'$ *and* $\Gamma \vdash A' <: B$ *for some* A'.
ii. $\Gamma \vdash A \lesssim B$ *if and only if* $\Gamma \vdash B' \sim B$ *and* $\Gamma \vdash A <: B'$ *for some* B'.

Proof. The left-to-right direction of both cases easily follows from Lemma 5.3, and the right-to-left direction of both cases follows from induction on the subtyping derivation and Lemma A.20. □

Lemma 5.5 (Gradual Guarantee Part One). *If* $\Gamma \vdash_{\mathsf{SG}} t : A$, $t \sqsubseteq t'$, *and* $\Gamma \sqsubseteq \Gamma'$ *then* $\Gamma' \vdash_{\mathsf{SG}} t' : B$ *and* $A \sqsubseteq B$.

Proof. This is a proof by induction on $\Gamma \vdash_{\mathsf{SG}} t : A$; see Appendix B.10 for the complete proof. □

Lemma 5.6 (Type Preservation for Cast Insertion). *If* $\Gamma \vdash_{\mathsf{SG}} t_1 : A$ *and* $\Gamma \vdash t_1 \Rightarrow t_2 : B$, *then* $\Gamma \vdash_{\mathsf{CG}} t_2 : B$ *and* $\Gamma \vdash A \sim B$.

Proof. The cast insertion algorithm is type directed and with respect to every term t_1 it will produce a term t_2 of the core language with the type A – this is straightforward to show by induction on the form of $\Gamma \vdash_{\mathsf{SG}} t_1 : A$ making use of typing for casting morphisms Lemma A.26 – except in the case of type application. Please see Appendix B.11 for the complete proof. □

Lemma 5.7 (Type Preservation). *If* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$ *and* $t_1 \rightsquigarrow t_2$, *then* $\Gamma \vdash_{\mathsf{CG}} t_2 : A$.

Proof. This proof holds by induction on $\Gamma \vdash_{\mathsf{CG}} t_1 : A$ with further case analysis on the structure the derivation $t_1 \rightsquigarrow t_2$. □

Lemma 5.8 (Simulation of More Precise Programs). *Suppose* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$, $\Gamma \vdash t_1 \sqsubseteq t_1'$, $\Gamma \vdash_{\mathsf{CG}} t_1' : A'$, *and* $t_1 \rightsquigarrow t_2$. *Then* $t_1' \rightsquigarrow^* t_2'$ *and* $\Gamma \vdash t_2 \sqsubseteq t_2'$ *for some* t_2'.

Proof. This proof holds by induction on $\Gamma \vdash_{\mathsf{CG}} t_1 : A_1$. See Appendix B.12 for the complete proof. □

Theorem 5.9 (Gradual Guarantee).
i. *If* $\cdot \vdash_{\mathsf{SG}} t : A$ *and* $t \sqsubseteq t'$, *then* $\cdot \vdash_{\mathsf{SG}} t' : B$ *and* $A \sqsubseteq B$.
ii. *Suppose* $\cdot \vdash_{\mathsf{CG}} t : A$ *and* $\cdot \vdash t \sqsubseteq t'$. *Then*
a. *if* $t \rightsquigarrow^* v$, *then* $t' \rightsquigarrow^* v'$ *and* $\cdot \vdash v \sqsubseteq v'$,
b. *if* $t \Uparrow$, *then* $t' \Uparrow$,
c. *if* $t' \rightsquigarrow^* v'$, *then* $t \rightsquigarrow^* v$ *where* $\cdot \vdash v \sqsubseteq v'$, *or* $t \rightsquigarrow^* \mathsf{error}_A$, *and*
d. *if* $t' \Uparrow$, *then* $t \Uparrow$ *or* $t \rightsquigarrow^* \mathsf{error}_A$.

Proof. This result follows from the same proof as (Siek et al. 2015), and so, we only give a brief summary. Part i. holds by Lemma 5.5, and Part ii. follows from simulation of more precise programs (Lemma 5.8). □

REFERENCES

Roy L. Crole. 1994. *Categories for Types*. Cambridge University Press. DOI:http://dx.doi.org/10.1017/CBO9781139172707

Jean-Yves Girard, Yves Lafont, and Paul Taylor. 1989. *Proofs and Types (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press.

W. A. Howard. 1980. The Formulae-as-Types Notion of Construction. *To H. B. Curry: Essays on Combznatory Logic, Lambda-Calculus, and Formalism* (1980), 479–490.

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

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

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

Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, and John Tang Boyland. 2015. Refined Criteria for Gradual Typing. In *1st Summit on Advances in Programming Languages (SNAPL 2015) (Leibniz International Proceedings in Informatics (LIPIcs))*, Thomas Ball, Rastislav Bodik, Shriram Krishnamurthi, Benjamin S. Lerner, and Greg Morrisett (Eds.), Vol. 32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 274–293.

A AUXILIARY RESULTS WITH PROOFS

Lemma A.1 (Kinding).

i. *If* $\Gamma \vdash A \sim B$, *then* $\Gamma \vdash A : \star$ *and* $\Gamma \vdash B : \star$.

ii. *If* $\Gamma \vdash A \lesssim B$, *then* $\Gamma \vdash A : \star$ *and* $\Gamma \vdash B : \star$.

iii. *If* $\Gamma \vdash_{\mathsf{SG}} t : A$, *then* $\Gamma \vdash A : \star$.

Proof. This proof holds by straightforward induction the form of each assumed judgment. □

Lemma A.2 (Strengthening for Kinding). *If* $\Gamma, x : A \vdash B : \star$, *then* $\Gamma \vdash B : \star$.

Proof. This proof holds by straightforward induction on the form of $\Gamma, x : A \vdash B : \star$. □

Lemma A.3 (Inversion for Type Precision). *Suppose* $\Gamma \vdash A : \star$, $\Gamma \vdash B : \star$, *and* $A \sqsubseteq B$. *Then:*

i. *if* $A \;\; ?$, *then* $\Gamma \vdash B \lesssim \mathbb{S}$.

ii. *if* $A \;\; A_1 \to B_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; A_2 \to B_2$, $A_1 \sqsubseteq A_2$, *and* $B_1 \sqsubseteq B_2$.

iii. *if* $A \;\; A_1 \times B_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; A_2 \times B_2$, $A_1 \sqsubseteq A_2$, *and* $B_1 \sqsubseteq B_2$.

iv. *if* $A \;\; \mathsf{List}\, A_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; \mathsf{List}\, A_2$ *and* $A_1 \sqsubseteq A_2$.

v. *if* $A \;\; \forall X <: A_1.B_1$, *then* $B \;\; \forall X <: A_1.B_1$ *and* $B_1 \sqsubseteq B_2$.

Proof. This proof holds by straightforward induction on the form of $A \sqsubseteq B$. □

Lemma A.4 (Surface Grady Inversion for Term Precision). *Suppose* $t \sqsubseteq t'$. *Then:*

i. *if* $t \;\; \mathsf{succ}\, t_1$, *then* $t' \;\; \mathsf{succ}\, t_2$ *and* $t_1 \sqsubseteq t_2$.

ii. *if* $t \;\; \mathsf{case}\, t_1\, \mathsf{of}\, 0 \to t_2, \mathsf{succ}\, x \to t_3$, *then* $t' \;\; \mathsf{case}\, t'_1\, \mathsf{of}\, 0 \to t'_2, \mathsf{succ}\, x \to t'_3$, $t_1 \sqsubseteq t'_1$, $t_2 \sqsubseteq t'_2$, *and* $t_3 \sqsubseteq t'_3$.

iii. *if* $t \;\; t_1, t_2$, *then* $t' \;\; t'_1, t'_2$, $t_1 \sqsubseteq t'_1$, *and* $t_2 \sqsubseteq t'_2$.

iv. *if* $t \;\; \mathsf{fst}\, t_1$, *then* $t' \;\; \mathsf{fst}\, t'_1$ *and* $t_1 \sqsubseteq t'_1$.

v. *if* $t \;\; \mathsf{snd}\, t_1$, *then* $t' \;\; \mathsf{snd}\, t'_1$ *and* $t_1 \sqsubseteq t'_1$.

vi. *if* $t \;\; t_1 :: t_2$, *then* $t' \;\; t'_1 :: t'_2$, $t_1 \sqsubseteq t'_1$, *and* $t_2 \sqsubseteq t'_2$.

vii. *if* $t \;\; \mathsf{case}\, t_1\, \mathsf{of}\, [] \to t_2, x :: y \to t_3$, *then* $t' \;\; \mathsf{case}\, t'_1\, \mathsf{of}\, [] \to t'_2, x :: y \to t'_3$, $t_1 \sqsubseteq t'_1$, $t_2 \sqsubseteq t'_2$, *and* $t_3 \sqsubseteq t'_3$.

viii. *if* $t \;\; \lambda x : A_1.t_1$, *then* $t' \;\; \lambda x : A_1.t'_1$ *and* $t_1 \sqsubseteq t'_1$.

ix. *if* $t \;\; t_1\, t_2$, *then* $t' \;\; t'_1\, t'_2$, $t_1 \sqsubseteq t'_1$, *and* $t_2 \sqsubseteq t'_2$.

x. if $t \;\; \Lambda X <: A_1.t_1$, *then* $t' \;\; \Lambda X <: A_1.t'_1$ *and* $t_1 \sqsubseteq t'_1$.

xi. if $t \;\; [A]t_1$, *then* $t' \;\; [A]t'_1$ *and* $t_1 \sqsubseteq t'_1$.

Proof. This proof holds by straightforward induction on the form of $t \sqsubseteq t'$. □

Lemma A.5 (Inversion for Type Consistency). *Suppose* $\Gamma \vdash A \sim B$. *Then:*

i. if $A \;\; ?$, *then* $\Gamma \vdash B \lesssim \mathbb{S}$.

ii. if $A \;\; \mathsf{List}\, A'$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; \mathsf{List}\, B'$ *and* $\Gamma \vdash A' \sim B'$.

iii. if $A \;\; A_1 \to B_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; A_2 \to B_2$, $\Gamma \vdash A_2 \sim A_1$, *and* $\Gamma \vdash B_1 \sim B_2$.

iv. if $A \;\; A_1 \to B_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; A_2 \to B_2$, $\Gamma \vdash A_2 \sim A_1$, *and* $\Gamma \vdash B_1 \sim B_2$.

v. if $A \;\; A_1 \times B_1$, *then* $B \;\; ?$ *and* $\Gamma \vdash A \lesssim \mathbb{S}$, *or* $B \;\; A_2 \times B_2$, $\Gamma \vdash A_1 \sim A_2$, *and* $\Gamma \vdash B_1 \sim B_2$.

vi. if $A \;\; \forall X <: A_1.B_1$, *then* $B \;\; \forall X <: A_1.B_2$ *and* $\Gamma, X <: A_1 \vdash B_1 \sim B_2$.

Proof. This proof holds by straightforward induction on the the form of $\Gamma \vdash A \sim B$. □

Lemma A.6 (Inversion for Consistent Subtyping). *Suppose* $\Gamma \vdash A \lesssim B$. *Then:*

i. if $A \;\; ?$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *or* $\Gamma \vdash B \lesssim \mathbb{S}$.

ii. if $A \;\; X$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, *or* $X <: B' \in \Gamma$ *and* $\Gamma \vdash B' \sim B$.

iii. if $A \;\; \mathsf{Nat}$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, *or* $B \;\; \mathbb{S}$.

iv. if $A \;\; \mathsf{Unit}$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, *or* $B \;\; \mathbb{S}$.

v. if $A \;\; \mathsf{List}\, A_1$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, $B \;\; \mathbb{S}$ *and* $\Gamma \vdash A_1 \lesssim \mathbb{S}$, *or* $B \;\; \mathsf{List}\, A'_1$ *and* $\Gamma \vdash A_1 \lesssim A'_1$.

vi. if $A \;\; A_1 \to B_1$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, $B \;\; \mathbb{S}$, $\Gamma \vdash A_1 \lesssim \mathbb{S}$ *and* $\Gamma \vdash B_1 \lesssim \mathbb{S}$, *or* $B \;\; A'_1 \to B'_1$, $\Gamma \vdash A'_1 \lesssim A_1$, *and* $\Gamma \vdash B_1 \lesssim B'_1$.

vii. if $A \;\; A_1 \times B_1$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, $B \;\; \mathbb{S}$, $\Gamma \vdash A_1 \lesssim \mathbb{S}$ *and* $\Gamma \vdash B_1 \lesssim \mathbb{S}$, *or* $B \;\; A'_1 \times B'_1$, $\Gamma \vdash A_1 \lesssim A'_1$, *and* $\Gamma \vdash B_1 \lesssim B'_1$.

viii. if $A \;\; \forall X <: A_1.B_1$, *then* $B \;\; A$ *and* $\Gamma \vdash A : \star$, $B \;\; \top$ *and* $\Gamma \vdash A : \star$, *or* $B \;\; \forall X <: A_1.B'_1$ *and* $\Gamma, X <: A_1 \vdash B_1 \lesssim B'_1$.

Proof. This proof holds by straightforward induction on the the form of $\Gamma \vdash A \lesssim B$. □

Lemma A.7 (Symmetry for Type Consistency). *If* $\Gamma \vdash A \sim B$, *then* $\Gamma \vdash B \sim A$.

Proof. This holds by straightforward induction on the form of $\Gamma \vdash A \sim B$. □

Lemma A.8. *If* $\Gamma \vdash A <: B$, *then* $\Gamma \vdash A \lesssim B$.

Proof. This proof holds by straightforward induction on $\Gamma \vdash A <: B$. □

Lemma A.9. *if* $\Gamma \vdash A \sim B$, *then* $\Gamma \vdash A \lesssim B$.

Proof. By straightforward induction on $\Gamma \vdash A \sim B$. □

Lemma A.10 (Type Precision and Consistency). *Suppose* $\Gamma \vdash A : \star$ *and* $\Gamma \vdash B : \star$. *Then if* $A \sqsubseteq B$, *then* $\Gamma \vdash A \sim B$.

Proof. This proof holds by straightforward induction on $A \sqsubseteq B$. □

Corollary A.11 (Type Precision and Subtyping). *Suppose* $\Gamma \vdash A : \star$ *and* $\Gamma \vdash B : \star$. *Then if* $A \sqsubseteq B$, *then* $\Gamma \vdash A \lesssim B$.

Proof. This easily follows from the previous two lemmas. □

Lemma A.12. *Suppose* $\Gamma \vdash A : \star$, $\Gamma \vdash B : \star$, *and* $\Gamma \vdash C : \star$. *If* $A \sqsubseteq B$ *and* $A \sqsubseteq C$, *then* $\Gamma \vdash B \sim C$.

Proof. It must be the case that either $B \sqsubseteq C$ or $C \sqsubseteq B$, but in both cases we know $\Gamma \vdash B \sim C$ by Lemma A.10. □

Lemma A.13 (Transitivity for Type Precision). *If* $A \sqsubseteq B$ *and* $B \sqsubseteq C$, *then* $A \sqsubseteq C$.

Proof. This proof holds by straightforward induction on $A \sqsubseteq B$ with a case analysis over $B \sqsubseteq C$. □

Lemma A.14. *If* $\Gamma \vdash A \sim B$, *then* $A \sqsubseteq B$ *or* $B \sqsubseteq A$.

Proof. This proof holds by straightforward induction over $\Gamma \vdash A \sim B$. □

Lemma A.15. *If* $\Gamma \vdash A \lesssim B$ *and* $A \sqsubseteq A'$, *then* $B \sqsubseteq A'$ *or* $A' \sqsubseteq B$.

Proof. Suppose $\Gamma \vdash A \lesssim B$ and $A \sqsubseteq A'$. The former implies that $A \sqsubseteq B$ or $B \sqsubseteq A$ by Lemma 5.3 and Lemma A.14. At this point the result easily follows. □

Lemma A.16. *Suppose* $A \sqsubseteq B$. *Then*

i. If $\mathsf{nat}A$ Nat, *then* $\mathsf{nat}B$ Nat.
ii. If $\mathsf{list}A$ $\mathsf{List}\,C$, *then* $\mathsf{list}B$ $\mathsf{List}\,C'$ *and* $C \sqsubseteq C'$.
iii. If $\mathsf{fun}A$ $A_1 \to A_2$, *then* $\mathsf{fun}B$ $A'_1 \to A'_2$, $A_1 \sqsubseteq A'_1$, *and* $A_2 \sqsubseteq A'_2$.

Proof. This proof holds by straightforward induction on $A \sqsubseteq B$. □

Lemma A.17. *If* $\Gamma \vdash A \sim B$, $\Gamma \vdash C : \star$, *and* $A \sqsubseteq C$, *then* $\Gamma \vdash C \sim B$.

Proof. Suppose $\Gamma \vdash A \sim B$ and $A \sqsubseteq C$. Then we know that $A \sqsubseteq B$ or $B \sqsubseteq A$. If the former, then we know that $\Gamma \vdash C \sim B$. If the latter, then we obtain $B \sqsubseteq C$ by transitivity, and $\Gamma \vdash B \sim C$ which implies that $\Gamma \vdash C \sim B$ by symmetry. □

Lemma A.18. *If* Γ' *Ok*, $\Gamma \sqsubseteq \Gamma'$ *and* $\Gamma \vdash A \sim B$, *then* $\Gamma' \vdash A \sim B$.

Proof. This proof holds by straightforward induction on $\Gamma \vdash A \sim B$. □

Lemma A.19 (Subtyping Context Precision). *If* $\Gamma \vdash A \lesssim B$ *and* $\Gamma \sqsubseteq \Gamma'$, *then* $\Gamma' \vdash A \lesssim B$.

Proof. Context precision does not manipulate the bounds on type variables, and thus, with respect to subtyping Γ and Γ' are essentially equivalent. □

Lemma A.20 (Simply Typed Consistent Types are Subtypes of $\mathbb{S}$). *If* $\Gamma \vdash A \lesssim \mathbb{S}$ *and* $\Gamma \vdash A \sim B$, *then* $\Gamma \vdash B \lesssim \mathbb{S}$.

Proof. This holds by straightforward induction on the form of $\Gamma \vdash A \lesssim \mathbb{S}$. □

Lemma A.21 (Type Precision Preserves $\mathbb{S}$).

i. If $\Gamma \vdash B : \star$, $\Gamma \vdash A \lesssim \mathbb{S}$ *and* $A \sqsubseteq B$, *then* $\Gamma \vdash B \lesssim \mathbb{S}$.
ii. If $\Gamma \vdash A : \star$, $\Gamma \vdash B \lesssim \mathbb{S}$ *and* $A \sqsubseteq B$, *then* $\Gamma \vdash A \lesssim \mathbb{S}$.

Proof. Both cases follow by induction on the assumed consistent subtyping derivation. □

Lemma A.22 (Congruence of Type Consistency Along Type Precision).

i. If $A_1 \sqsubseteq A'_1$ *and* $\Gamma \vdash A_1 \sim A_2$ *then* $\Gamma \vdash A'_1 \sim A_2$.
ii. If $A_2 \sqsubseteq A'_2$ *and* $\Gamma \vdash A_1 \sim A_2$ *then* $\Gamma \vdash A_1 \sim A'_2$.

Proof. Both parts hold by induction on the assumed type consistency judgment. See Appendix B.8 for the complete proof. □

Corollary A.23 (Congruence of Type Consistency Along Type Precision Condensed). *If* $A_1 \sqsubseteq A'_1$, $A_2 \sqsubseteq A'_2$, *and* $\Gamma \vdash A_1 \sim A_2$ *then* $\Gamma \vdash A'_1 \sim A'_2$.

Lemma A.24 (Congruence of Subtyping Along Type Precision). *Suppose* $\Gamma \vdash B : \star$ *and* $A \sqsubseteq B$.

i. If $\Gamma \vdash A \lesssim C$ *then* $\Gamma \vdash B \lesssim C$.
ii. If $\Gamma \vdash C \lesssim A$ *then* $\Gamma \vdash C \lesssim B$.

Proof. This is a proof by induction on the form of $A \sqsubseteq B$; see Appendix B.9 for the complete proof. □

Corollary A.25 (Congruence of Subtyping Along Type Precision). *If* $A_1 \sqsubseteq A_2$, $B_1 \sqsubseteq B_2$, *and* $\Gamma \vdash A_1 \lesssim B_1$, *then* $\Gamma \vdash A_2 \lesssim B_2$.

Lemma A.26 (Typing Casting Morphisms). *If* $\Gamma \vdash A \sim B$ *and* $\mathsf{caster} A, B\ \ c$, *then* $\Gamma \vdash_{\mathsf{CG}} c : A \to B$.

Proof. This proof holds similarly to how we constructed casting morphisms in the categorical model. See Lemma 3.13. □

Lemma A.27 (Substitution for Consistent Subtyping). *If* $\Gamma, X <: B_1 \vdash B_2 \lesssim B_3$ *and* $\Gamma \vdash A_1 \lesssim B_1$, *then* $\Gamma \vdash [A_1/X]B_2 \lesssim [A_1/X]B_3$.

Proof. This holds by straightforward induction on the form of $\Gamma, X <: B_1 \vdash B_2 \lesssim B_3$. □

Lemma A.28 (Substitution for Reflexive Type Consistency). *If* $\Gamma, X <: B_1 \vdash B \sim B$, $\Gamma \vdash A_1 \sim A_2$, *and* $\Gamma \vdash A_2 <: B_1$, *then* $\Gamma \vdash [A_1/X]B \sim [A_2/X]B$.

Proof. This holds by straightforward induction on the form of B. □

Lemma A.29 (Substitution for Type Consistency). *If* $\Gamma, X <: B_1 \vdash B_2 \sim B_3$, $\Gamma \vdash A_1 \sim A_2$, *and* $\Gamma \vdash A_1 <: B_1$, *then* $\Gamma \vdash [A_1/X]B_2 \sim [A_2/X]B_3$.

Proof. This holds by straightforward induction on $\Gamma, X <: B_1 \vdash B_2 \sim B_3$ using both substitution for consistent subtyping (Lemma A.27) and substitution for reflexive type consistent (Lemma A.28). □

Lemma A.30 (Typing for Type Precision). *If* $\Gamma \vdash_{\mathsf{SG}} t_1 : A$, $t_1 \sqsubseteq t_2$, *and* $\Gamma \sqsubseteq \Gamma'$, *then* $\Gamma' \vdash_{\mathsf{SG}} t_2 : B$ *and* $A \sqsubseteq B$.

Proof. This proof holds by induction on $\Gamma \vdash_{\mathsf{SG}} t_1 : A$ with a case analysis over $t_1 \sqsubseteq t_2$. □

Lemma A.31 (Substitution for Term Precision).

i. If $\Gamma, x : A \vdash t_1 \sqsubseteq t_2$ *and* $\Gamma \vdash t'_1 \sqsubseteq t'_2$, *then* $\Gamma \vdash [t'_1/x]t_1 \sqsubseteq [t'_2/x]t_2$.
ii. If $\Gamma, X <: A_2 \vdash t_1 \sqsubseteq t_2$ *and* $A_1 \sqsubseteq A'_1$, *then* $\Gamma \vdash [A_1/X]t_1 \sqsubseteq [A'_1/X]t_2$.

Proof. This proof of part one holds by straightforward induction on $\Gamma, x : A \vdash t_1 \sqsubseteq t_2$, and the proof of part two holds by straightforward induction on $\Gamma, X <: A_2 \vdash t_1 \sqsubseteq t_2$. □

Lemma A.32 (Typeability Inversion).

i. *If* $\Gamma \vdash_{\mathsf{CG}} \mathsf{succ}\, t : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A'$ *for some* A'*.*
ii. *If* $\Gamma \vdash_{\mathsf{CG}} \mathsf{case}\, t \colon \mathsf{Nat}\, \mathsf{of}\, 0 \to t_1, \mathsf{succ}\, x \to t_2 : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A_1$*,* $\Gamma \vdash_{\mathsf{CG}} t_1 : A_2$*, and* $\Gamma, x : \mathsf{Nat} \vdash_{\mathsf{CG}} t_2 : A_3$ *for types* A_1*,* A_2*,* A_3*.*
iii. *If* $\Gamma \vdash_{\mathsf{CG}} t_1, t_2 : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t_1 : A_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_2 : A_2$ *for types* A_1 *and* A_2*.*
iv. *If* $\Gamma \vdash_{\mathsf{CG}} \Lambda X <: B.t : A$*, then* $\Gamma, X <: B \vdash_{\mathsf{CG}} t : A_1$ *for some type* A_1*.*
v. *If* $\Gamma \vdash_{\mathsf{CG}} [B]t : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A_1$ *for some type* A_1*.*
vi. *If* $\Gamma \vdash_{\mathsf{CG}} \lambda x : B.t : A$*, then* $\Gamma, x : B \vdash_{\mathsf{CG}} t : A_1$ *for some type* A_1*.*
vii. *If* $\Gamma \vdash_{\mathsf{CG}} t_1\, t_2 : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t_1 : A_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_2 : A_2$ *for types* A_1 *and* A_2*.*
viii. *If* $\Gamma \vdash_{\mathsf{CG}} \mathsf{fst}\, t : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A_1$ *for some type* A_1*.*
ix. *If* $\Gamma \vdash_{\mathsf{CG}} \mathsf{snd}\, t : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A_1$ *for some type* A_1*.*
x. *If* $\Gamma \vdash_{\mathsf{CG}} t_1 :: t_2 : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t_1 : A_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_2 : A_2$ *for some types* A_1 *and* A_2*.*
xi. *If* $\Gamma \vdash_{\mathsf{CG}} \mathsf{case}\, t \colon \mathsf{List}\, B\, \mathsf{of}\, [] \to t_1, x :: y \to t_2 : A$*, then* $\Gamma \vdash_{\mathsf{CG}} t : A_1$*,* $\Gamma \vdash_{\mathsf{CG}} t_1 : A_2$*, and* $\Gamma, x : A, y : \mathsf{List}\, A \vdash_{\mathsf{CG}} t_2 : A_3$ *for types* A_1*,* A_2*,* A_3*.*

Lemma A.33 (Inversion for Term Precision for Core Grady). *Suppose* $\Gamma \vdash t_1 \sqsubseteq t_2$*.*

i. *If* $t_1 \;\; x$*, then one of the following is true:*
   a. $t_2 \;\; x$*,* $x : A \in \Gamma$*, and* Γ *Ok*
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
ii. *If* $t_1 \;\; \mathsf{split}_{K_1}$*, then one of the following is true:*
   a. $t_2 \;\; \mathsf{split}_{K_2}$ *and* $K_1 \sqsubseteq K_2$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
iii. *If* $t_1 \;\; \mathsf{squash}_{K_1}$*, then one of the following is true:*
   a. $t_2 \;\; \mathsf{squash}_{K_2}$ *and* $K_1 \sqsubseteq K_2$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
iv. *If* $t_1 \;\; \mathsf{box}$*, then one of the following is true:*
   a. $t_2 \;\; \mathsf{box}$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
v. *If* $t_1 \;\; \mathsf{unbox}$*, then one of the following is true:*
   a. $t_2 \;\; \mathsf{unbox}$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
vi. *If* $t_1 \;\; 0$*, then one of the following is true:*
   a. $t_2 \;\; 0$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$
vii. *If* $t_1 \;\; \mathsf{triv}$*, then one of the following is true:*

a. t_2 triv
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

viii. *If* t_1 $[]$*, then one of the following is true:*
a. t_2 $[]$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

ix. *If* t_1 $\mathsf{succ}\, t'_1$*, then one of the following is true:*
a. t_2 $\mathsf{succ}\, t'_2$ *and* $\Gamma \vdash t'_1 \sqsubseteq t'_2$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

x. *If* t_1 $\mathsf{case}\, t'_1 : \mathsf{Nat}\,\mathsf{of}\, 0 \to t'_2, \mathsf{succ}\, x \to t'_3$*, then one of the following is true:*
a. t_2 $\mathsf{case}\, t'_4 : \mathsf{Nat}\,\mathsf{of}\, 0 \to t'_5, \mathsf{succ}\, x \to t'_6$, $\Gamma \vdash t'_1 \sqsubseteq t'_4$, $\Gamma \vdash t'_2 \sqsubseteq t'_5$*, and* $\Gamma, x : \mathsf{Nat} \vdash t'_3 \sqsubseteq t'_6$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xi. *If* t_1 t'_1, t'_2*, then one of the following is true:*
a. t_2 t'_3, t'_4, $\Gamma \vdash t'_1 \sqsubseteq t'_3$*, and* $\Gamma \vdash t'_2 \sqsubseteq t'_4$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xii. *If* t_1 $\mathsf{fst}\, t'_1$*, then one of the following is true:*
a. t_2 $\mathsf{fst}\, t'_2$ *and* $\Gamma \vdash t'_1 \sqsubseteq t'_2$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xiii. *If* t_1 $\mathsf{snd}\, t'_1$*, then one of the following is true:*
a. t_2 $\mathsf{snd}\, t'_2$ *and* $\Gamma \vdash t'_1 \sqsubseteq t'_2$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xiv. *If* t_1 $t'_1 :: t'_2$*, then one of the following is true:*
a. t_2 $t'_3 :: t'_4$, $\Gamma \vdash t'_1 \sqsubseteq t'_3$*, and* $\Gamma \vdash t'_2 \sqsubseteq t'_4$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xv. *If* t_1 $\mathsf{case}\, t'_1 : \mathsf{List}\, A_1\, \mathsf{of}\, [] \to t'_2, x :: y \to t'_3$*, then one of the following is true:*
a. t_2 $\mathsf{case}\, t'_4 : \mathsf{List}\, A_2\, \mathsf{of}\, [] \to t'_5, x :: y \to t'_6$, $\Gamma \vdash t'_1 \sqsubseteq t'_4$, $\Gamma \vdash t'_2 \sqsubseteq t'_5$*, and* $\Gamma, x : A_2, y : \mathsf{List}\, A_2 \vdash t'_3 \sqsubseteq t'_6$*, and* $A_1 \sqsubseteq A_2$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xvi. *If* t_1 $\lambda x : A_1.t_1$*, then one of the following is true:*
a. t_2 $\lambda x : A_2.t_2$ *and* $\Gamma, x : A_2 \vdash t_1 \sqsubseteq t_2$ *and* $A_1 \sqsubseteq A_2$
b. t_2 $\mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
c. t_2 $\mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xvii. *If* $t_1 \;\; t'_1\, t'_2$*, then one of the following is true:*
   a. $t_2 \;\; t'_3\, t'_4$*,* $\Gamma \vdash t_3 \sqsubseteq t'_3$*, and* $\Gamma \vdash t_4 \sqsubseteq t'_4$
   b. $t'_1 \;\; \mathsf{unbox}_A$ *and* $t_2 \;\; t'_2$
   c. $t'_1 \;\; \mathsf{split}_K$ *and* $t_2 \;\; t'_2$
   d. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   e. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xviii. *If* $t_1 \;\; \mathsf{unbox}_A\, t'_1$*, then one of the following is true:*
   a. $t_2 \;\; t'_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t'_1 : ?$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xix. *If* $t_1 \;\; \mathsf{split}_K\, t'_1$*, then one of the following is true:*
   a. $t_2 \;\; t'_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t'_1 : K$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xx. *If* $t_1 \;\; \Lambda X <: A.t'_1$*, then one of the following is true:*
   a. $t_2 \;\; \Lambda X <: A.t'_2$ *and* $\Gamma, X <: A_2 \vdash t'_1 \sqsubseteq t'_2$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xxi. *If* $t_1 \;\; [A_1]t'_1$*, then one of the following is true:*
   a. $t_2 \;\; [A_2]t'_2$*,* $\Gamma \vdash t'_1 \sqsubseteq t'_2$*, and* $A_1 \sqsubseteq A_2$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

xxii. *If* $t_1 \;\; \mathsf{error}_{A_1}$*, then one of the following is true:*
   a. $\Gamma \vdash_{\mathsf{CG}} t_2 : A_2$ *and* $A_1 \sqsubseteq A_2$
   b. $t_2 \;\; \mathsf{box}_A\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
   c. $t_2 \;\; \mathsf{squash}_K\, t_1$ *and* $\Gamma \vdash_{\mathsf{CG}} t_1 : K$

Proof. The proof of this result holds by straightforward induction on $\Gamma \vdash t_1 \sqsubseteq t_2$. □

B PROOFS

B.1 Proof of Lifted Retract (Lemma 3.8)

This is a proof by induction on the form of A.

Case. Suppose A is atomic. Then:

$$\widehat{\mathsf{box}_A}; \widehat{\mathsf{unbox}_A} \;\; \mathsf{box}_A; \mathsf{unbox}_A \;\; \mathsf{id}_A$$

Case. Suppose A is ?. Then:

$$\begin{array}{ll} \widehat{\mathsf{box}_A}; \widehat{\mathsf{unbox}_A} & \widehat{\mathsf{box}_?}; \widehat{\mathsf{unbox}_?} \\ & \mathsf{id}_?; \mathsf{id}_? \\ & \mathsf{id}_? \\ & \mathsf{id}_A \end{array}$$

Case. Suppose $A \quad A_1 \to A_2$. Then:

$$\begin{aligned}
\widehat{\mathsf{box}}_A ; \widehat{\mathsf{unbox}}_A \quad & \widehat{\mathsf{box}}_{A_1 \to A_2} ; \widehat{\mathsf{unbox}}_{A_1 \to A_2} \\
& \widehat{\mathsf{unbox}}_{A_1} \to \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{box}}_{A_1} \to \widehat{\mathsf{box}}_{A_2} \\
& \widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \to \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2}
\end{aligned}$$

By two applications of the induction hypothesis we know the following:

$$\widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \quad \mathsf{id}_{A_1} \qquad \text{and} \qquad \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2} \quad \mathsf{id}_{A_2}$$

Thus, we know the following:

$$\begin{aligned}
\widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \to \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2} \quad & \mathsf{id}_{A_1} \to \mathsf{id}_{A_2} \\
& \mathsf{id}_{A_1 \to A_2} \\
& \mathsf{id}_A
\end{aligned}$$

Case. Suppose $A \quad A_1 \times A_2$. Then:

$$\begin{aligned}
\widehat{\mathsf{box}}_A ; \widehat{\mathsf{unbox}}_A \quad & \widehat{\mathsf{box}}_{A_1 \times A_2} ; \widehat{\mathsf{unbox}}_{A_1 \times A_2} \\
& \widehat{\mathsf{box}}_{A_1} \times \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_1} \times \widehat{\mathsf{box}}_{A_2} \\
& \widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \times \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2}
\end{aligned}$$

By two applications of the induction hypothesis we know the following:

$$\widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \quad \mathsf{id}_{A_1} \qquad \text{and} \qquad \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2} \quad \mathsf{id}_{A_2}$$

Thus, we know the following:

$$\begin{aligned}
\widehat{\mathsf{box}}_{A_1} ; \widehat{\mathsf{unbox}}_{A_1} \times \widehat{\mathsf{box}}_{A_2} ; \widehat{\mathsf{unbox}}_{A_2} \quad & \mathsf{id}_{A_1} \times \mathsf{id}_{A_2} \\
& \mathsf{id}_{A_1 \times A_2} \\
& \mathsf{id}_A
\end{aligned}$$

B.2 Proof of Lemma 3.9

We must show that the function

$$\mathsf{S}_{A,B} : \mathsf{Hom}_{\mathcal{C}} A, B \longrightarrow \mathsf{Hom}_{\mathcal{S}} \mathsf{S}A, \mathsf{S}B$$

is injective.

So suppose $f \in \mathsf{Hom}_{\mathcal{C}} A, B$ and $g \in \mathsf{Hom}_{\mathcal{C}} A, B$ such that $\mathsf{S}f \quad \mathsf{S}g : \mathsf{S}A \longrightarrow \mathsf{S}B$. Then we can easily see that:

$$\begin{aligned}
\mathsf{S}f \quad & \widehat{\mathsf{unbox}}_A ; f ; \widehat{\mathsf{box}}_B \\
& \widehat{\mathsf{unbox}}_A ; g ; \widehat{\mathsf{box}}_B \\
& \mathsf{S}g
\end{aligned}$$

But, we have the following equalities:

$$\begin{aligned}
\widehat{\mathsf{unbox}}_A ; f ; \widehat{\mathsf{box}}_B \quad & \widehat{\mathsf{unbox}}_A ; g ; \widehat{\mathsf{box}}_B \\
\widehat{\mathsf{box}}_A ; \widehat{\mathsf{unbox}}_A ; f ; \widehat{\mathsf{box}}_B ; \widehat{\mathsf{unbox}}_B \quad & \widehat{\mathsf{box}}_A ; \widehat{\mathsf{unbox}}_A ; g ; \widehat{\mathsf{box}}_B ; \widehat{\mathsf{unbox}}_B \\
\mathsf{id}_A ; f ; \widehat{\mathsf{box}}_B ; \widehat{\mathsf{unbox}}_B \quad & \mathsf{id}_A ; g ; \widehat{\mathsf{box}}_B ; \widehat{\mathsf{unbox}}_B \\
\mathsf{id}_A ; f ; \mathsf{id}_B \quad & \mathsf{id}_A ; g ; \mathsf{id}_B \\
f \quad & g
\end{aligned}$$

The previous equalities hold due to Lemma 3.8.

B.3 Proof of Type Consistency in the Model (Lemma 3.15)

This is a proof by induction on the form of $A \sim B$.

Case.

$$\frac{}{A \sim A}\,refl$$

Choose c_1 c_2 $\mathsf{id}_A : A \rightarrowtail A$.

Case.

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

Choose c_1 $\mathsf{Box}_A : A \rightarrowtail ?$ and c_2 $\mathsf{Unbox}_A : ? \to A$.

Case.

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

Choose c_1 $\mathsf{Unbox}_A : ? \rightarrowtail A$ and c_2 $\mathsf{Box}_A : A \to ?$.

Case.

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

By the induction hypothesis there exists four casting morphisms $c'_1 : A_1 \rightarrowtail A_2$, $c'_2 : A_2 \rightarrowtail A_1$, $c'_3 : B_1 \rightarrowtail B_2$, and $c'_4 : B_2 \rightarrowtail B_1$. Choose c_1 $c'_2 \to c'_3 : A_1 \to B_1 \rightarrowtail A_2 \to B_2$ and c_2 $c'_1 \to c'_4 : A_2 \to B_2 \rightarrowtail A_1 \to B_1$.

Case.

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

By the induction hypothesis there exists four casting morphisms $c'_1 : A_1 \rightarrowtail A_2$, $c'_2 : A_2 \rightarrowtail A_1$, $c'_3 : B_1 \rightarrowtail B_2$, and $c'_4 : B_2 \rightarrowtail B_1$. Choose c_1 $c'_1 \times c'_3 : A_1 \times B_1 \rightarrowtail A_2 \times B_2$ and c_2 $c'_2 \times c'_4 : A_2 \times B_2 \rightarrowtail A_1 \times B_1$.

B.4 Proof of Interpretation of Types Theorem 3.17

B.5 Proof of Interpretation of Evaluation (Theorem 3.18)

This proof holds by induction on the form of $\Gamma \vdash t_1 \rightsquigarrow t_2 : A$. We only show the cases for the casting rules, because the others are well-known to hold within any cartesian closed category; see (Lambek 1980) or (Crole 1994). We will routinely use Theorem 3.17 throughout this proof without mention.

B.6 Proof of Lemma 4.2

First, we define the identify meta-function:

$$\mathsf{id}_A : \lambda x : A.x$$

Then composition. Suppose $\Gamma \vdash t_1 : A \to B$ and $\Gamma \vdash t_2 : B \to D$ are two terms, then we define:

$$t_1; t_2 : \lambda x : A.t_2\, t_1\, x$$

It is easy to see that the following rule is admissible:

$$\frac{\Gamma \vdash t_1 : A \to B \qquad \Gamma \vdash t_2 : B \to D}{\Gamma \vdash t_1; t_2 : A \to D}\,\text{comp}$$

The functor $- \times -$ requires two morphisms $\Gamma \vdash t_1 : A \to D$ and $\Gamma \vdash t_2 : B \to E$, and is defined as follows:

$$t_1 \times t_2 : \lambda x : A \times B.t_1\, \mathsf{fst}\, x, t_2\, \mathsf{snd}\, x$$

The following rule is admissible:

$$\frac{\Gamma \vdash t_1 : A \to D \qquad \Gamma \vdash t_2 : B \to E}{\Gamma \vdash t_1 \times t_2 : A \times B \to D \times E}\ \text{prod}$$

The functor $- \to -$ requires two morphisms $\Gamma \vdash t_1 : D \to A$ and $\Gamma \vdash t_2 : B \to E$, and is defined as follows:

$$t_1 \to t_2 : \lambda f : A \to B.\lambda y : D.t_2\, f\, t_1\, y$$

The following rule is admissible:

$$\frac{\Gamma \vdash t_1 : D \to A \qquad \Gamma \vdash t_2 : B \to E}{\Gamma \vdash t_1 \to t_2 : A \to B \to D \to E}\ \text{prod}$$

At this point it is straightforward to carry out the definition of Box_A and Unbox_A using the definitions from the model. Showing the admissibility of the typing and reduction rules follows by induction on A.

B.7 Proof of Left-to-Right Consistent Subtyping (Lemma 5.3)

This is a proof by induction on $\Gamma \vdash A \lesssim B$. We only show a few of the most interesting cases.

Case.

$$\frac{\Gamma \vdash A \lesssim \mathbb{S}}{\Gamma \vdash A \lesssim ?}\ \mathsf{box}$$

In this case $B\ \ ?$.

Part i. Choose $A'\ \ ?$.

Part ii. Choose $B'\ \ A$.

Case.

$$\frac{\Gamma \vdash B \lesssim \mathbb{S}}{\Gamma \vdash ? \lesssim B}\ \mathsf{unbox}$$

In this case $A\ \ ?$.

Part i. Choose $A'\ \ B$.

Part ii. Choose $B'\ \ ?$.

Case.

$$\frac{\Gamma \vdash A_2 \lesssim A_1 \quad \Gamma \vdash B_1 \lesssim B_2}{\Gamma \vdash A_1 \to B_1 \lesssim A_2 \to B_2}\ \to$$

In this case $A\ \ A_1 \to B_1$ and $B\ \ A_2 \to B_2$.

Part i. By part two of the induction hypothesis we know that $\Gamma \vdash A'_1 \sim A_1$ and $\Gamma \vdash A_2 <: A'_1$, and by part one of the induction hypothesis $\Gamma \vdash B_1 \sim B'_1$ and $\Gamma \vdash B'_1 <: B_2$. By symmetry of type consistency we may conclude that $\Gamma \vdash A_1 \sim A'_1$ which along with $\Gamma \vdash B_1 \sim B'_1$ implies that $\Gamma \vdash A_1 \to B_1 \sim A'_1 \to B'_1$, and by reapplying the rule we may conclude that $\Gamma \vdash A'_1 \to B'_1 <: A_2 \to B_2$.

Part ii. Similar to part one, except that we first applying part one of the induction hypothesis to the first premise, and then the second part to the second premise.

B.8 Proof of Congruence of Type Consistency Along Type Precision (Lemma A.22)

The proofs of both parts are similar, and so we only show a few cases of the first part, but the omitted cases follow similarly.

Proof of part one. This is a proof by induction on the form of $A_1 \sqsubseteq A'_1$.

Case.

$$\frac{\Gamma \vdash A_1 \lesssim \mathbb{S}}{A_1 \sqsubseteq ?}\ ?$$

In this case $A'_1 \quad ?$. Suppose $\Gamma \vdash A_1 \sim A_2$. Then it suffices to show that $\Gamma \vdash ? \sim A_2$, and hence, we must show that $\Gamma \vdash A_2 \lesssim \mathbb{S}$, but this follows by Lemma A.20.

Case.

$$\frac{A \sqsubseteq C \quad B \sqsubseteq D}{A \to B \sqsubseteq C \to D}\ \to$$

In this case $A_1 \quad A \to B$ and $A'_1 \quad C \to D$. Suppose $\Gamma \vdash A_1 \sim A_2$. Then by inversion for type consistency it must be the case that either $A_2 \quad ?$ and $\Gamma \vdash A_1 \lesssim \mathbb{S}$, or $A_2 \quad A' \to B'$, $\Gamma \vdash A \sim A'$, and $\Gamma \vdash B \sim B'$.

Consider the former. Then it suffices to show that $\Gamma \vdash A'_1 \sim ?$, and hence we must show that $\Gamma \vdash A'_1 \lesssim \mathbb{S}$, but this follows from Lemma A.21.

Consider the case when $A_2 \quad A' \to B'$, $\Gamma \vdash A \sim A'$, and $\Gamma \vdash B \sim B'$. It suffices to show that $\Gamma \vdash C \to D \sim A' \to B'$ which follows from $\Gamma \vdash A' \sim C$ and $\Gamma \vdash D \sim B'$. Thus, it suffices to show that latter. By assumption we know the following:

$$A \sqsubseteq C \text{ and } \Gamma \vdash A \sim A'$$
$$B \sqsubseteq D \text{ and } \Gamma \vdash B \sim B'$$

Now by two applications of the induction hypothesis we obtain $\Gamma \vdash C \sim A'$ and $\Gamma \vdash D \sim B'$. By symmetry the former implies $\Gamma \vdash A \sim C$ and we obtain our result.

B.9 Proof of Congruence of Subtyping Along Type Precision (Lemma A.24)

This is a proof by induction on the form of $A \sqsubseteq B$. The proof of part two follows similarly to part one. We only give the most interesting cases. All others follow similarly.

Proof of part one. We only show the most interesting case, because all others are similar.

Case.

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

In this case $A \;\; A_1 \to B_1$ and $B \;\; A_2 \to B_2$. Suppose $\Gamma \vdash A \lesssim C$. Thus, by inversion for consistency subtyping it must be the case that $C \;\; \top$ and $\Gamma \vdash A : \star$, $C \;\; ?$ and $\Gamma \vdash A \lesssim \mathbb{S}$, or $C \;\; A'_1 \to B'_1$, $\Gamma \vdash A'_1 \lesssim A_1$, and $\Gamma \vdash B_1 \lesssim B'_1$. The case when $C \;\; \top$ is trivial, and the case when $C \;\; ?$ is similarly to the proof of Lemma A.22.

Consider the case when $C \;\; A'_1 \to B'_1$, $\Gamma \vdash A'_1 \lesssim A_1$, and $\Gamma \vdash B_1 \lesssim B'_1$. By assumption we know the following:

$$A_1 \sqsubseteq A_2 \text{ and } \Gamma \vdash A'_1 \lesssim A_1$$
$$B_1 \sqsubseteq B_2 \text{ and } \Gamma \vdash B_1 \lesssim B'_1$$

So by part two and one, respectively, of the induction hypothesis we know that $\Gamma \vdash A'_1 \lesssim A_2$ and $\Gamma \vdash B_2 \lesssim B'_1$. Thus, by reapplying the rule above we may now conclude that $\Gamma \vdash A_2 \to B_2 \lesssim A'_1 \to B'_2$ to obtain our result.

B.10 Proof of Gradual Guarantee Part One (Lemma 5.5)

This is a proof by induction on $\Gamma \vdash_{\mathsf{SG}} t : A$. We only show the most interesting cases, because the others follow similarly.

Case.

$$\frac{x : A \in \Gamma \quad \Gamma\,\mathrm{Ok}}{\Gamma \vdash_{\mathsf{SG}} x : A}\ \mathrm{var}$$

In this case $t \;\; x$. Suppose $t \sqsubseteq t'$. Then it must be the case that $t' \;\; x$. If $x : A \in \Gamma$, then there is a type A' such that $x : A' \in \Gamma'$ and $A \sqsubseteq A'$. Thus, choose $B \;\; A'$ and the result follows.

Case.

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : A' \quad \mathsf{nat}A' \;\; \mathsf{Nat}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{succ}\, t_1 : \mathsf{Nat}}\ \mathsf{succ}$$

In this case $A \;\; \mathsf{Nat}$ and $t \;\; \mathsf{succ}\, t_1$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. Then by definition it must be the case that $t' \;\; \mathsf{succ}\, t_2$ where $t_1 \sqsubseteq t_2$. By the induction hypothesis $\Gamma' \vdash_{\mathsf{SG}} t_2 : B'$ where $A' \sqsubseteq B'$. Since $\mathsf{nat}A' \;\; \mathsf{Nat}$ and $A' \sqsubseteq B'$, then it must be the case that $\mathsf{nat}B' \;\; \mathsf{Nat}$ by Lemma A.16. At this point we obtain our result by choosing $B \;\; \mathsf{Nat}$, and reapplying the rule above.

Case.

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathsf{SG}} t_1 : C \quad \mathsf{nat}C \;\; \mathsf{Nat} \quad \Gamma \vdash A_1 \sim A \\ \Gamma \vdash_{\mathsf{SG}} t_2 : A_1 \quad \Gamma, x : \mathsf{Nat} \vdash_{\mathsf{SG}} t_3 : A_2 \quad \Gamma \vdash A_2 \sim A\end{array}}{\Gamma \vdash_{\mathsf{SG}} \mathsf{case}\, t_1\, \mathsf{of}\, 0 \to t_2, \mathsf{succ}\, x \to t_3 : A}\ \mathsf{Nat}_e$$

In this case $t \;\; \mathsf{case}\, t_1\, \mathsf{of}\, 0 \to t_2, \mathsf{succ}\, x \to t_3$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. This implies that $t' \;\; \mathsf{case}\, t'_1\, \mathsf{of}\, 0 \to t'_2, \mathsf{succ}\, x \to t'_3$ such that $t_1 \sqsubseteq t'_1$, $t_2 \sqsubseteq t'_2$, and $t_3 \sqsubseteq t'_3$. Since $\Gamma \sqsubseteq \Gamma'$ then $\Gamma, x : \mathsf{Nat} \sqsubseteq \Gamma', x : \mathsf{Nat}$. By the induction hypothesis we know the following:

$$\Gamma' \vdash_{\mathsf{SG}} t'_1 : C' \text{ for } C \sqsubseteq C'$$
$$\Gamma' \vdash_{\mathsf{SG}} t_2 : A'_1 \text{ for } A_1 \sqsubseteq A'_1$$
$$\Gamma', x : \mathsf{Nat} \vdash_{\mathsf{SG}} t_3 : A'_2 \text{ for } A_2 \sqsubseteq A'_2$$

By assumption we know that $\Gamma \vdash A_1 \sim A$, $\Gamma \vdash A_2 \sim A$, and $\Gamma \sqsubseteq \Gamma'$, hence, by Lemma A.18 we know $\Gamma' \vdash A_1 \sim A$ and $\Gamma' \vdash A_2 \sim A$. By the induction hypothesis we know that $A_1 \sqsubseteq A'_1$ and $A_2 \sqsubseteq A'_2$, so by using Lemma A.17 we may obtain that $\Gamma' \vdash A'_1 \sim A$ and $\Gamma' \vdash A'_2 \sim A$. At this point choose $B \;\; A$ and we obtain our result by reapplying the rule.

Case.

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{SG}} t_2 : A_2 \quad \mathsf{list} A_2 \;\; \mathsf{List}\, A_3 \quad \Gamma \vdash A_1 \sim A_3}{\Gamma \vdash_{\mathsf{SG}} t_1 :: t_2 : \mathsf{List}\, A_3}\ \mathsf{List}_i$$

In this case $A \;\; \mathsf{List}\, A_3$ and $t \;\; t_1 :: t_2$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. Then it must be the case that $t' \;\; t'_1 :: t'_2$ where $t_1 \sqsubseteq t'_1$ and $t_2 \sqsubseteq t'_2$. Then by the induction hypothesis we know the following:

$$\Gamma' \vdash_{\mathsf{SG}} t'_1 : A'_1 \text{ where } A_1 \sqsubseteq A'_1$$
$$\Gamma' \vdash_{\mathsf{SG}} t'_2 : A'_2 \text{ where } A_2 \sqsubseteq A'_2$$

By Lemma A.16 $\mathsf{list} A'_2 \;\; \mathsf{List}\, A'_3$ where $A_3 \sqsubseteq A'_3$. Now by Lemma A.18 and Lemma A.17 we know that $\Gamma' \vdash A'_1 \sim A_3$, and by using the same lemma again, $\Gamma' \vdash A'_1 \sim A'_3$ because $\Gamma' \vdash A_3 \sim A'_1$ holds by symmetry. Choose $B \;\; \mathsf{List}\, A'_3$ and the result follows.

Case.

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : A_1 \quad \Gamma \vdash_{\mathsf{SG}} t_2 : A_2}{\Gamma \vdash_{\mathsf{SG}} t_1, t_2 : A_1 \times A_2}\ \times_i$$

In this case $A \;\; A_1 \times A_2$ and $t \;\; t_1, t_2$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. This implies that $t' \;\; t'_1, t'_2$ where $t_1 \sqsubseteq t'_1$ and $t_2 \sqsubseteq t'_2$.

By the induction hypothesis we know:

$$\Gamma' \vdash_{\mathsf{SG}} t'_1 : A'_1 \text{ and } A_1 \sqsubseteq A'_1$$
$$\Gamma' \vdash_{\mathsf{SG}} t'_2 : A'_2 \text{ and } A_2 \sqsubseteq A'_2$$

Then choose $B \;\; A'_1 \times A'_2$ and the result follows by reapplying the rule above and the fact that $A_1 \times A_2 \sqsubseteq A'_1 \times A'_2$.

Case.

$$\frac{\Gamma, x : A_1 \vdash_{\mathsf{SG}} t_1 : B_1}{\Gamma \vdash_{\mathsf{SG}} \lambda x : A_1.t_1 : A_1 \to B_1}\ \to_i$$

In this case $A_1 \to B_2$ and $t \;\; \lambda x : A_1.t_1$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. Then it must be the case that $t' \;\; \lambda x : A_2.t_2$, $t_1 \sqsubseteq t_2$, and $A_1 \sqsubseteq A_2$. Since $\Gamma \sqsubseteq \Gamma'$ and $A_1 \sqsubseteq A_2$, then $\Gamma, x : A_1 \sqsubseteq \Gamma', x : A_2$ by definition. Thus, by the induction hypothesis we know the following:

$$\Gamma', x : A_2 \vdash_{\mathsf{SG}} t'_1 : B_2 \text{ and } B_1 \sqsubseteq B_2$$

Choose $B \;\; A_2 \to B_2$ and the result follows by reapplying the rule above and the fact that $A_1 \to B_1 \sqsubseteq A_2 \to B_2$.

Case.

$$\frac{\Gamma \vdash_{\mathsf{SG}} t_1 : \forall X <: C_0.C_2 \quad \Gamma \vdash C_1 \lesssim C_0}{\Gamma \vdash_{\mathsf{SG}} [C_1]t_1 : [C_1/X]C_2}\ \forall_e$$

In this case $t \;\; [C_1]t_1$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. Then it must be the case that $t' \;\; [C'_1]t_2$ such that $t_1 \sqsubseteq t_2$ and $C_1 \sqsubseteq C'_1$. By the induction hypothesis:

$$\Gamma' \vdash_{\mathsf{SG}} t_2 : C \text{ where } \forall X <: C_0.C_2 \sqsubseteq C$$

Thus, it must be the case that $C \;\; \forall X <: C_0.C'_2$ such that $C_2 \sqsubseteq C'_2$. By assumption we know that $\Gamma \vdash C_1 \lesssim C_0$ and $C_1 \sqsubseteq C'_1$, and thus, by Corollary A.25 and Lemma A.19 we know $\Gamma' \vdash C'_1 \lesssim C_0$. Thus, choose $B \;\; C$, and the result follows by reapplying the rule above, and the fact that $A \sqsubseteq C$, because $C_2 \sqsubseteq C'_2$.

Case.

$$\frac{\Gamma \vdash_{\mathsf{SG}} t : A' \quad \Gamma \vdash A' \lesssim A}{\Gamma \vdash_{\mathsf{SG}} t : A}\ \text{sub}$$

Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. By the induction hypothesis we know that $\Gamma' \vdash_{\mathsf{SG}} t' : A''$ for $A' \sqsubseteq A''$. We know $A'' \sqsubseteq A$ or $A \sqsubseteq A''$, because we know that $\Gamma \vdash A' \lesssim A$ and $A' \sqsubseteq A''$. Suppose $A'' \sqsubseteq A$, then by Corollary A.11 $\Gamma' \vdash A'' \lesssim A$, and then by subsumption $\Gamma' \vdash_{\mathsf{SG}} t' : A$, hence, choose $B \;\; A$ and the result follows. If $A \sqsubseteq A''$, then choose $B \;\; A''$ and the result follows.

Case.

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathsf{SG}} t_1 : C \quad \mathsf{fun}C \;\; A_1 \to B_1 \\ \Gamma \vdash_{\mathsf{SG}} t_2 : A_2 \quad \Gamma \vdash A_2 \sim A_1\end{array}}{\Gamma \vdash_{\mathsf{SG}} t_1\, t_2 : B_1}\ \to_e$$

In this case $A \;\; B_1$ and $t \;\; t_1\, t_2$. Suppose $t \sqsubseteq t'$ and $\Gamma \sqsubseteq \Gamma'$. The former implies that $t' \;\; t'_1\, t'_2$ such that $t_1 \sqsubseteq t'_1$ and $t_2 \sqsubseteq t'_2$. By the induction hypothesis we know the following:

$$\Gamma' \vdash_{\mathsf{SG}} t'_1 : C' \text{ for } C \sqsubseteq C'$$
$$\Gamma' \vdash_{\mathsf{SG}} t'_2 : A'_2 \text{ for } A_2 \sqsubseteq A'_2$$

We know by assumption that $\Gamma \vdash A_2 \sim A_1$ and hence $\Gamma' \vdash A_2 \sim A_1$ because bounds on type variables are left unchanged by context precision. Since $C \sqsubseteq C'$ and $\mathsf{fun}C \;\; A_1 \to B_1$, then $\mathsf{fun}C' \;\; A'_1 \to B'_1$ where $A_1 \sqsubseteq A'_1$ and $B_1 \sqsubseteq B'_1$ by Lemma A.16. Furthermore, we know $\Gamma' \vdash A_2 \sim A_1$ and $A_2 \sqsubseteq A'_2$ and $A_1 \sqsubseteq A'_1$, then we know $\Gamma' \vdash A'_2 \sim A'_1$ by Corollary A.23. So choose $B \;\; B'_1$. Then reapply the rule above and the result follows, because $B_1 \sqsubseteq B'_1$.

B.11 Proof of Type Preservation for Cast Insertion (Lemma 5.6)

The cast insertion algorithm is type directed and with respect to every term t_1 it will produce a term t_2 of the core language with the type A – this is straightforward to show by induction on the form of $\Gamma \vdash_{\mathsf{SG}} t_1 : A$ making use of typing for casting morphisms Lemma A.26 – except in the case of type application. We only consider this case here.

This is a proof by induction on the form of $\Gamma \vdash_{\mathsf{SG}} t_1 : A$. Suppose the form of $\Gamma \vdash_{\mathsf{SG}} t_1 : A$ is as follows:

$$\frac{\Gamma \vdash_{\mathsf{SG}} t'_1 : \forall X <: B_1.B_2 \quad \Gamma \vdash A_1 \lesssim B_1}{\Gamma \vdash_{\mathsf{SG}} [A_1]t'_1 : [A_1/X]B_2}\ \forall_e$$

In this case $t_1 \;\; [A_1]t'_1$ and $A \;\; [A_1/X]B_2$. Cast insertion is syntax directed, and hence, inversion for it holds trivially. Thus, it must be the case that the form of $\Gamma \vdash t_1 \Rightarrow t_2 : B$ is as follows:

$$\frac{\Gamma \vdash t'_1 \Rightarrow t'_2 : \forall X <: B_1.B'_2 \quad \Gamma \vdash A_1 \sim A_2 \quad \Gamma \vdash A_2 <: B_1}{\Gamma \vdash [A_1]t'_1 \Rightarrow [A_2]t'_2 : [A_2/X]B'_2}$$

So $t_2 \quad [A_2]t'_2$ and $B \quad [A_2/X]B'_2$. Since we know $\Gamma \vdash_{\mathsf{SG}} t'_1 : \forall X <: B_1.B_2$ and $\Gamma \vdash t'_1 \Rightarrow t'_2 : \forall X <: B_1.B'_2$ we can apply the induction hypothesis to obtain $\Gamma \vdash_{\mathsf{CG}} t'_2 : \forall X <: B_1.B'_2$ and $\Gamma \vdash \forall X <: B_1.B_2 \sim \forall X <: B_1.B'_2$, and thus, $\Gamma, X <: B_1 \vdash B_2 \sim B'_2$ by inversion for type consistency. If $\Gamma, X <: B_1 \vdash B_2 \sim B'_2$ holds, then $\Gamma \vdash [A_1/X]B_2 \sim [A_2/X]B'_2$ when $\Gamma \vdash A_1 \sim A_2$ by substitution for type consistency (Lemma A.29). Since we know $\Gamma \vdash_{\mathsf{CG}} t'_2 : \forall X <: B_1.B'_2$ by the induction hypothesis and $\Gamma \vdash A_2 <: B_1$ by assumption, then we know $\Gamma \vdash_{\mathsf{CG}} [A_2]t'_2 : [A_2/X]B'_2$ by applying the Core Grady typing rule $\forall_e$.

B.12 Proof of Simulation of More Precise Programs (Lemma 5.8)

This is a proof by induction on $\Gamma \vdash_{\mathsf{CG}} t_1 : A_1$. We only give the most interesting cases. All others follow similarly. Throughout the proof we implicitly make use of typability inversion (Lemma A.32) when applying the induction hypothesis.

Case.

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : \mathsf{Nat}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{succ}\, t : \mathsf{Nat}}\ \mathsf{succ}$$

In this case $t_1 \quad \mathsf{succ}\, t$ and $A \quad \mathsf{Nat}$. Suppose $\Gamma \vdash_{\mathsf{CG}} t'_1 : A'$. By inversion for term precision we must consider the following cases:

i. $t'_1 \quad \mathsf{succ}\, t'$ and $\Gamma \vdash t \sqsubseteq t'$

ii. $t'_1 \quad \mathsf{box}_{\mathsf{Nat}}\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : \mathsf{Nat}$

Proof of part i. Suppose $t'_1 \quad \mathsf{succ}\, t'$, $\Gamma \vdash t \sqsubseteq t'$, and $t_1 \rightsquigarrow t_2$. Then $t_2 \quad \mathsf{succ}\, t''$ and $t \rightsquigarrow t''$. Then by the induction hypothesis we know that there is some t''' such that $t' \rightsquigarrow^* t'''$ and $\Gamma \vdash t'' \sqsubseteq t'''$. Choose $t'_2 \quad \mathsf{succ}\, t'''$ and the result follows.

Proof of part ii. Suppose $t'_1 \quad \mathsf{box}_{\mathsf{Nat}}\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : \mathsf{Nat}$, and $t_1 \rightsquigarrow t_2$. Then choose $t'_2 \quad \mathsf{box}_{\mathsf{Nat}}\, t_2$, and the result follows, because we know by type preservation that $\Gamma \vdash_{\mathsf{CG}} t_2 : \mathsf{Nat}$, and hence, $\Gamma \vdash t_2 \sqsubseteq t'_2$.

Case.

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathsf{CG}} t : \mathsf{Nat} \\ \Gamma \vdash_{\mathsf{CG}} t_3 : A \quad \Gamma, x : \mathsf{Nat} \vdash_{\mathsf{CG}} t_4 : A\end{array}}{\Gamma \vdash_{\mathsf{CG}} \mathsf{case}\, t : \mathsf{Nat}\, \mathsf{of}\, 0 \to t_3, \mathsf{succ}\, x \to t_4 : A}\ \mathsf{Nat}_e$$

In this case $t_1 \quad \mathsf{case}\, t : \mathsf{Nat}\, \mathsf{of}\, 0 \to t_3, \mathsf{succ}\, x \to t_4$. Suppose $\Gamma \vdash_{\mathsf{CG}} t'_1 : A'$. Then inversion of term precision implies that one of the following must hold:

- $t'_1 \quad \mathsf{case}\, t' : \mathsf{Nat}\, \mathsf{of}\, 0 \to t'_3, \mathsf{succ}\, x \to t'_4$, $\Gamma \vdash t \sqsubseteq t'$, $\Gamma \vdash t_3 \sqsubseteq t'_3$, and $\Gamma, x : \mathsf{Nat} \vdash t_4 \sqsubseteq t'_4$
- $t'_1 \quad \mathsf{box}_A\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
- $t'_1 \quad \mathsf{squash}_K\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : K$, and $A \quad K$

Proof of part i. Suppose $t'_1 \quad \mathsf{case}\, t' : \mathsf{Nat}\, \mathsf{of}\, 0 \to t'_3, \mathsf{succ}\, x \to t'_4$, $\Gamma \vdash t \sqsubseteq t'$, $\Gamma \vdash t_3 \sqsubseteq t'_3$, and $\Gamma, x : \mathsf{Nat} \vdash t_4 \sqsubseteq t'_4$.

We case split over $t_1 \rightsquigarrow t_2$.

Case. Suppose $t \;\; 0$ and $t_2 \;\; t_3$. Since $\Gamma \vdash t_1 \sqsubseteq t'_1$ we know that it must be the case that $t' \;\; 0$ and $t'_1 \rightsquigarrow t'_3$ by inversion for term precision or t'_1 would not be typable which is a contradiction. Thus, choose $t'_2 \;\; t'_3$ and the result follows.

Case. Suppose $t \;\; \mathsf{succ}\, t''$ and $t_2 \;\; [t''/x]t_4$. Since $\Gamma \vdash t_1 \sqsubseteq t'_1$ we know that $t' \;\; \mathsf{succ}\, t'''$, or t'_1 would not be typable, and $\Gamma \vdash t'' \sqsubseteq t'''$ by inversion for term precision. In addition, $t'_1 \rightsquigarrow [t'''/x]t'_4$. Choose $t_2 \;\; [t'''/x]t'_4$. Then it suffices to show that $\Gamma \vdash [t''/x]t_4 \sqsubseteq [t'''/x]t'_4$ by substitution for term precision (Lemma A.31).

Case. Suppose a congruence rule was used. Then $t_2 \;\; \mathsf{case}\, t'' : \mathsf{Nat}\, \mathsf{of}\, 0 \to t''_3, \mathsf{succ}\, x \to t''_4$. This case will follow straightforwardly by induction and a case split over which congruence rule was used.

Proof of part ii. Suppose $t'_1 \;\; \mathsf{box}_A\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : A$, and $t_1 \rightsquigarrow t_2$. Then choose $t'_2 \;\; \mathsf{box}_A\, t_2$, and the result follows, because we know by type preservation that $\Gamma \vdash_{\mathsf{CG}} t_2 : A$, and hence, $\Gamma \vdash t_2 \sqsubseteq t'_2$.

Proof of part iii. Similar to the previous case.

Case.

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : A \times B}{\Gamma \vdash_{\mathsf{CG}} \mathsf{fst}\, t : A} \times_{e_1}$$

In this case $t_1 \;\; \mathsf{fst}\, t$. Suppose $\Gamma \vdash t_1 \sqsubseteq t'_1$ and $\Gamma \vdash_{\mathsf{CG}} t'_1 : A'$. Then inversion for term precision implies that one of the following must hold:

- $t'_1 \;\; \mathsf{fst}\, t'$ and $\Gamma \vdash t \sqsubseteq t'$
- $t'_1 \;\; \mathsf{box}_A\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
- $t'_1 \;\; \mathsf{squash}_K\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : K$, and $A \;\; K$

We only consider the proof of part i, because the others follow similarly to the previous case. Case split over $t_1 \rightsquigarrow t_2$.

Case. Suppose $t \;\; t'_3, t''_3$ and $t_2 \;\; t'_3$. By inversion for term precision it must be the case that $t' \;\; t'_4, t''_4$ because $\Gamma \vdash t_1 \sqsubseteq t'_1$ or else t'_1 would not be typable. In addition, this implies that $\Gamma \vdash t'_3 \sqsubseteq t'_4$ and $\Gamma \vdash t''_3 \sqsubseteq t''_4$. Thus, $t'_1 \rightsquigarrow t'_4$. Thus, choose $t'_2 \;\; t'_4$ and the result follows.

Case. Suppose a congruence rule was used. Then $t_2 \;\; \mathsf{fst}\, t''$. This case will follow straightforwardly by induction and a case split over which congruence rule was used.

Case.

$$\frac{\Gamma, x : A_1 \vdash_{\mathsf{CG}} t : A_2}{\Gamma \vdash_{\mathsf{CG}} \lambda x : A_1.t : A_1 \to A_2} \to_i$$

In this case $t_1 \;\; \lambda x : A_1.t$ and $A \;\; A_1 \to A_2$. Suppose $\Gamma \vdash t_1 \sqsubseteq t'_1$ and $\Gamma \vdash_{\mathsf{CG}} t'_1 : A'$. Then inversion of term precision implies that one of the following must hold:

- $t'_1 \;\; \lambda x : A'_1.t'$
- $t'_1 \;\; \mathsf{box}_A\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
- $t'_1 \;\; \mathsf{squash}_K\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : K$, and $A \;\; K$

We only consider the proof of part i. The reduction relation does not reduce under λ-expressions. Hence, $t_2 \;\; t_1$, and thus, choose $t_2' \;\; t_1'$, and the case trivially follows.

Case.

$$\frac{\Gamma \vdash_{\mathsf{CG}} t_3 : A_1 \to A_2 \quad \Gamma \vdash_{\mathsf{CG}} t_4 : A_1}{\Gamma \vdash_{\mathsf{CG}} t_3\, t_4 : A_2} \to_e$$

In this case $t_1 \;\; t_3\, t_4$. Suppose $\Gamma \vdash t_1 \sqsubseteq t_1'$ and $\Gamma \vdash_{\mathsf{CG}} t_1' : A'$. Then by inversion for term prevision we know one of the following is true:

i. $t_1' \;\; t_3'\, t_4'$, $\Gamma \vdash t_3 \sqsubseteq t_3'$, and $\Gamma \vdash t_4 \sqsubseteq t_4'$

ii. $t_1' \;\; \mathsf{box}_{A_2}\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$

iii. $t_3 \;\; \mathsf{unbox}_{A_2}$, $t_1' \;\; t_4$, and $\Gamma \vdash_{\mathsf{CG}} t_4 : ?$

iv. $t_3 \;\; \mathsf{split}_{K_2}$, $t_1' \;\; t_4$, and $\Gamma \vdash_{\mathsf{CG}} t_4 : ?$

v. $t_1' \;\; \mathsf{squash}_{K_2}\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : K_2$

Proof of part i. Suppose $t_1' \;\; t_3'\, t_4'$, $\Gamma \vdash t_3 \sqsubseteq t_3'$, and $\Gamma \vdash t_4 \sqsubseteq t_4'$.

We case split on the from of $t_1 \rightsquigarrow t_2$.

Case. Suppose $t_3 \;\; \lambda x : A_1.t_5$ and $t_2 \;\; [t_4/x]t_5$. Then by inversion for term precision we know that $t_3' \;\; \lambda x : A_1'.t_5'$ and $\Gamma, x : A_2' \vdash t_5 \sqsubseteq t_5'$, because $\Gamma \vdash t_3 \sqsubseteq t_3'$ and the requirement that t_1' is typable. Choose $t_2' \;\; [t_4'/x]t_5'$ and it is easy to see that $t_1' \rightsquigarrow [t_4'/x]t_4'$. We know that $\Gamma, x : A_2' \vdash t_5 \sqsubseteq t_5'$ and $\Gamma \vdash t_4 \sqsubseteq t_4'$, and hence, by Lemma A.31 we know that $\Gamma \vdash [t_4/x]t_5 \sqsubseteq [t_4'/x]t_5'$, and we obtain our result.

Case. Suppose $t_3 \;\; \mathsf{unbox}_A$, $t_4 \;\; \mathsf{box}_A\, t_5$, and $t_2 \;\; t_5$. Then by inversion for term prevision $t_3' \;\; \mathsf{unbox}_A$, $t_4' \;\; \mathsf{box}_A\, t_5'$, and $\Gamma \vdash t_5 \sqsubseteq t_5'$. Note that $t_4' \;\; \mathsf{box}_A\, t_5'$ and $\Gamma \vdash t_5 \sqsubseteq t_5'$ hold even though there are two potential rules that could have been used to construct $\Gamma \vdash t_4 \sqsubseteq t_4'$. Choose $t_2' \;\; t_5'$ and it is easy to see that $t_1' \rightsquigarrow t_5'$. Thus, we obtain our result.

Case. Suppose $t_3 \;\; \mathsf{unbox}_A$, $t_4 \;\; \mathsf{box}_B\, t_5$, $A \neq B$, and $t_2 \;\; \mathsf{error}_B$. Then $t_3' \;\; \mathsf{unbox}_A$ and $t_4' \;\; \mathsf{box}_B\, t_5'$. Choose $t_2' \;\; \mathsf{error}_B$ and it is easy to see that $t_1' \rightsquigarrow t_5'$. Finally, we can see that $\Gamma \vdash t_2 \sqsubseteq t_2'$ by reflexivity.

Case. Suppose $t_3 \;\; \mathsf{split}_U$, $t_4 \;\; \mathsf{squash}_U\, t_5$, and $t_2 \;\; t_5$. Similar to the case for boxing and unboxing.

Case. Suppose $t_3 \;\; \mathsf{split}_{U_1}$, $t_4 \;\; \mathsf{squash}_{U_2}\, t_5$, $U_1 \neq U_2$, and $t_2 \;\; t_5$. Similar to the case for boxing and unboxing.

Case. Suppose a congruence rule was used. Then $t_2 \;\; t_5'\, t_6'$. This case will follow straightforwardly by induction and a case split over which congruence rule was used.

Proof of part ii. We know that $t_1 \;\; t_3\, t_4$. Suppose $t_1' \;\; \mathsf{box}_{A_2}\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$. If $t_1 \rightsquigarrow t_2$, then $t_1' \;\; \mathsf{box}_{A_2}\, t_1 \rightsquigarrow \mathsf{box}_{A_2}\, t_2$. Thus, choose $t_2' \;\; \mathsf{box}_{A_2}\, t_2$.

Proof of part iii. We know that $t_1 \;\; t_3\, t_4$. Suppose $t_3 \;\; \mathsf{unbox}_{A_2}$, $t_1' \;\; t_4$, and $\Gamma \vdash_{\mathsf{CG}} t_4 : ?$. Then $t_1 \;\; \mathsf{unbox}_{A_2}\, t_4$. We case split over $t_1 \rightsquigarrow t_2$. We have three cases to consider.

Suppose $t_4 \;\; \mathsf{box}_{A_2}\, t_5$ and $t_2 \;\; t_5$. Then choose $t_2' \;\; t_4 \;\; t_1'$, and we obtain our result.

Suppose $t_4 \quad \mathsf{box}_{A_3}\, t_5$, $A_2 \neq A_3$, and $t_2 \quad \mathsf{error}_{A_2}$. Then choose $t_2' \quad t_4 \quad t_1'$, and we obtain our result.

Suppose a congruence rule was used. Then $t_2 \quad t_3\, t_4'$. This case will follow straightforwardly by induction.

Proof of part iv. Similar to part iii.

Proof of part v. Similar to part ii.

Case.

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : \forall X <: A_2.A_3 \quad \Gamma \vdash A_1 <: A_2}{\Gamma \vdash_{\mathsf{CG}} [A_1]t : [A_1/X]A_3}\ \forall_e$$

In this case $t_1 \quad [A_1]t$ and $A \quad [A_1/X]A_3$. Suppose $\Gamma \vdash t_1 \sqsubseteq t_1'$ and $\Gamma \vdash_{\mathsf{CG}} t_1' : A'$.

- $t_1' \quad [A_1']t'$, $\Gamma \vdash t \sqsubseteq t'$, and $A_1 \sqsubseteq A_1'$
- $t_1' \quad \mathsf{box}_A\, t_1$ and $\Gamma \vdash_{\mathsf{CG}} t_1 : A$
- $t_1' \quad \mathsf{squash}_K\, t_1$, $\Gamma \vdash_{\mathsf{CG}} t_1 : K$, and $A \quad K$

We only consider the proof of part i. We case split over the form of $t_1 \rightsquigarrow t_2$.

Case. Suppose $t \quad \Lambda X <: A_2.t_3$ and $t_2 \quad [A_1/X]t_3$. Then inversion for term precision on $\Gamma \vdash t \sqsubseteq t'$ and the fact that $\Gamma \vdash_{\mathsf{CG}} t : \forall X <: A_2.A_3$ and $t_1' \quad [A_1']t'$ then it can only be the case that $t' \quad \Lambda X <: A_2.t_3'$ and $\Gamma, X <: A_2 \vdash t_3 \sqsubseteq t_3'$, or t_1' would not be typable which is a contradiction. Then by substitution for term precision we know that $\Gamma \vdash [A_1/X]t_3 \sqsubseteq [A_1'/X]t_3'$ by substitution for term precision (Lemma A.31), because we know that $A_1 \sqsubseteq A_1'$. Choose $t_2' \quad [A_1'/X]t_3'$ and the result follows, because $t_1' \rightsquigarrow t_2'$.

Case. Suppose a congruence rule was used. Then $t_2 \quad [A_1]t''$. This case will follow straightforwardly by induction and a case split over which congruence rule was used.

Case.

$$\frac{\Gamma \vdash_{\mathsf{CG}} t : A_1 \quad \Gamma \vdash A_1 <: A_2}{\Gamma \vdash_{\mathsf{CG}} t : A_2}\ \text{sub}$$

In this case $t_1 \quad t$ and $A \quad A_2$. Suppose $\Gamma \vdash t_1 \sqsubseteq t_1'$ and $\Gamma \vdash_{\mathsf{CG}} t_1' : A'$. Assume $t_1 \rightsquigarrow t_2$. Then by the induction hypothesis there is a t_2' such that $t_1' \rightsquigarrow^* t_2'$ and $\Gamma \vdash t_2 \sqsubseteq t_2'$, thus, we obtain our result.