Graduality and Parametricity: Together Again for the First Time

MAX S. NEW, Northeastern University, USA
DUSTIN JAMNER, Northeastern University, USA
AMAL AHMED, Northeastern University, USA

Parametric polymorphism and gradual typing have proven to be a difficult combination, with no language yet produced that satisfies the fundamental theorems of each: parametricity and graduality. Notably, in their POPL'19 paper, Toro, Labrada, and Tanter claim to prove that graduality (also known as the dynamic gradual guarantee) is "simply incompatible with parametricity." In this paper, we show that graduality and parametricity can, in fact, coexist. Toro et al.'s proof that the two are incompatible merely applies to *existing* syntax of polymorphic gradual languages, all of which dynamically generate a fresh type name upon type instantiation and subsequently using the type name to seal values of abstract type. We also show that their semantics has seemingly non-parametric behavior.

We break with previous work by rejecting the *coupling* of creation of type names with sealing. This guides the design of a new gradual language, PolyG$^\nu$, that supports an exotic form of existential and universal types where the creation of fresh types is exposed in a controlled way and sealing is explicit by the programmer. The semantics of PolyG$^\nu$ is similar to previous gradual parametric languages, but the typing is crucially different. We elaborate PolyG$^\nu$ into a cast calculus and give a translation from the cast calculus into a typed language with errors. Our translation illuminates the semantics of the dynamic type in gradual parametric languages, showing it must be interpreted as an extensible sum type that can have new cases dynamically added to it at runtime. We validate the design of PolyG$^\nu$ by proving graduality and parametricity theorems for the language. Far from being in opposition to each other, the parametricity theorem turns out to be a simple *corollary* of a relational graduality theorem.

1 INTRODUCTION

Gradually typed languages seek to support the smooth transition from dynamically typed to statically typed programming styles [20–22]. They allow freely composing dynamic and static components without sacrificing the strong reasoning principles the static language provides by inserting type casts at the boundaries. Recent research has aimed to extend gradual typing with reasoning principles enabled by a variety of advanced static features—such as refinement types [12], union and intersection types [5], typestates [25], effect tracking [4], subtyping [9], ownership [18], session types [10], and secure information flow [6, 7, 23].

Data abstraction is one of the most useful reasoning principles that programming languages can provide from a very simple syntactic discipline. It allows the programmer to decouple the implementation details from client code, making the code more modular and reusable. For instance, a programmer might implement the interface for an unordered set using ordered lists as the carrier datatype, but as long as the client only uses operations valid for sets, such as insertion, deletion and membership testing, the ordering inherent in the implementation does not leak out. Data abstraction can be achieved through many different means: the use of closures, dynamic generation of opaque data structures, ML-style module systems, and the use of existential and universal types.

Authors' addresses: Max S. New, Khoury College of Computer Sciences, Northeastern University, USA, maxnew@ccs.neu.edu; Dustin Jamner, Khoury College of Computer Sciences, Northeastern University, USA, jamner.d@husky.neu.edu; Amal Ahmed, Khoury College of Computer Sciences, Northeastern University, USA, amal@ccs.neu.edu.

2018. 2475-1421/2018/1-ART1 $15.00
https://doi.org/

A long-standing open problem in gradual typing has been to design a gradual language that provides strong data abstraction principles. To see the issue, consider an application of a function $f : \forall X.X \to \mathbb{B} \to X$ (where $\mathbb{B}$ is the boolean type):

$$f\ [\mathbb{B}]\ \texttt{true}\ \texttt{false}$$

In a language satisfying a relational parametricity theorem, the only value that the function can possibly return is the first argument passed into it, i.e. `true`. However, the natural combination of gradual typing and polymorphism would allow us to cast the second argument to X, which would then reduce to `false`, violating parametric reasoning.

$$(\Lambda X.\lambda x : X.\lambda y : \mathbb{B}.(y :: ?) :: X)\ [\mathbb{B}]\ \texttt{true}\ \texttt{false} \mapsto^*\ (\texttt{false} :: ?) :: \mathbb{B} \mapsto^*\ \texttt{false}$$

There have been several recent designs for gradual languages that depart from this semantics in order to satisfy a relational parametricity theorem. The first major attempt is the polymorphic blame calculus [2]. A later variant, the cast calculus λB, is the first one to actually come with a proof of parametricity [3]. Other recent efforts include: System F_G [11], a gradual surface language (without explicit casts) that is compiled to System F_C, a cast calculus similar to λB; a gradual language with implicit polymorphism by Xie et al. [26] (henceforth CSA) that is compiled to λB; and GSF [24], which is designed using the Abstracting Gradual Typing methodology [9] and proved to satisfy parametricity.

To guarantee parametricity, all of the above polymorphic gradual languages use variants of the same strategy: they eliminate these parametricity violations using runtime type generation (dynamic sealing). When a type is instantiated, a fresh type name (seal) is generated at type instantiation and added to the type-name store; it is used to seal/unseal all values of abstract type within the body of the type abstraction. In λB, the example from above would reduce as follows:

$$\begin{aligned}
&\cdot \rhd (\Lambda X.\lambda x : X.\lambda y : \mathbb{B}.(y :: ?) :: X)\ [\mathbb{B}]\ \texttt{true}\ \texttt{false}\\
&\mapsto\ \alpha := \mathbb{B} \rhd ((\lambda x : \alpha.\lambda y : \mathbb{B}.(y :: ?) :: \alpha) \overset{+\alpha}{\Longrightarrow} \mathbb{B} \to \mathbb{B} \to \mathbb{B})\ \texttt{true}\ \texttt{false}\\
&\mapsto^*\ \alpha := \mathbb{B} \rhd ((\texttt{false} : \mathbb{B}) :: ?) :: \alpha\ \mapsto\ \text{error}
\end{aligned}$$

The intuition is that the type X in the body of the Λ is *different* from the type $\mathbb{B}$ with which it is instantiated. Instantiation generates a fresh type name α, records a correspondence $\alpha := \mathbb{B}$ in the type-name store, and substitutes the "seal" α for X in the body of the type abstraction, rather than $\mathbb{B}$. However since α and $\mathbb{B}$ *are* different, there is a type mismatch: above the expression must be converted from type $\alpha \to \mathbb{B} \to \alpha$ to $\mathbb{B} \to \mathbb{B} \to \mathbb{B}$: the *conversion* labeled $+\alpha$ "reveals" what α actually maps to. Now, when we apply this $\mathbb{B} \to \mathbb{B} \to \mathbb{B}$ function to `true` and `false`, we end up passing `false` to a function that expects $\mathbb{B}$, which then passes it to a function that expects ?, so we end up casting $\texttt{false} : \mathbb{B}$ to ?. Now when we try to "unseal" this value using α, we get an error because $\alpha \neq \mathbb{B}$, i.e., because the value embedded in ? is not an α-sealed value but rather just a value of type $\mathbb{B}$. Thus, this strategy catches the parametricity violation at runtime. Moreover, if we change the example so it casts x instead of y through ? to X, we simply get `true`:

$$\begin{aligned}
&\cdot \rhd (\Lambda X.\lambda x : X.\lambda y : \mathbb{B}.(x :: ?) :: X)\ [\mathbb{B}]\ \texttt{true}\ \texttt{false}\\
&\mapsto\ \alpha := \mathbb{B} \rhd ((\lambda x : \alpha.\lambda y : \mathbb{B}.(x :: ?) :: \alpha) \overset{+\alpha}{\Longrightarrow} \mathbb{B} \to \mathbb{B} \to \mathbb{B})\ \texttt{true}\ \texttt{false}\\
&\mapsto^*\ \alpha := \mathbb{B} \rhd ((\texttt{true} : \alpha) :: ?) :: \alpha\ \mapsto\ \texttt{true}
\end{aligned}$$

However, parametricity is not the only design goal for these languages; gradually typed languages are expected to satisfy certain criteria [19]. Besides being type safe and a superset of the static and the dynamic language, they should also ensure a smooth transition between dynamically typed and statically typed styles, a property known as the gradual guarantee. The static gradual guarantee says that if a program is well typed, then making its types less precise, as defined by the type precision relation $A \sqsubseteq B$, should yield a well typed program. Additionally, the *dynamic gradual*

guarantee, henceforth *graduality* [15], says that making a program's types more precise/less precise, as defined by the term precision relation $M \sqsubseteq N$ (read: M is more precise than N), should only change the program's behavior in predictable ways. In particular, making types more precise should either result in the same behavior as the original program or cause a dynamic type error. And if the program does not error, then making its types less precise should result in the same behavior.

Designing a polymorphic gradual language that satisfies *both parametricity and graduality* remains an open problem. In fact, it's worse than that: Toro et al. [24] recently showed that graduality is "simply incompatible with parametricity," which would mean that we must choose one or the other. Their polymorphic gradual language GSF satisfies parametricity but not graduality. But while parametricity and graduality are indeed incompatible in GSF—and in λB and, modulo some technicalities, all polymorphic gradual languages designed to date—we disagree with Toro et al. [24]'s analysis that they are incompatible *in general*. To see why, let us consider their counterexample to graduality ([24], §9). Let $\mathtt{id}_X = \Lambda X.\lambda x : X.x$ (the polymorphic identity function) and let $\mathtt{id}_? = \Lambda X.\lambda x : ?.x$. By the standard definition, $\mathtt{id}_X \sqsubseteq \mathtt{id}_?$ and for any $v \sqsubseteq v'$, we have $\mathtt{id}_X[A]v \sqsubseteq \mathtt{id}_?[A]v'$. Since the identity function always terminates, for graduality to hold, $\mathtt{id}_?[A]v'$ must also terminate producing v'. However, they show that in their gradually typed language, for any $v' : ?$ and type A, $\mathtt{id}_?[A]v' \Downarrow \mathfrak{U}$, where we write $\mathfrak{U}$ as the runtime type error, violating graduality. The latter reduction can be proven using their parametricity theorem, showing that in some sense that parametricity theorem is incompatible with graduality.

However, there's something fishy about this reduction $\mathtt{id}_?[A]v' \Downarrow \mathfrak{U}$. While it is proven using the parametricity theorem, it is not in a sense a "free theorem" about the all values of type $\forall X.? \to X$, and they show that there are terms of this type that do not always error: specifically, casting $\mathtt{id}_X$ to $\forall X.? \to X$ results in a term that does not always error. We argue that the problem is not with the term $\mathtt{id}_?$, but with a *failure of expressivity* of their language, where the only way to achieve certain behaviors is through casts.

Let's consider why the values of type $\forall X.? \to X$ should not always error by a semantic analysis. Here our type contains ? so we must consider how to interpret that.

In previous work on gradual languages [15, 16], the type ? has been interpreted as a recursive sum of all of the basic types in the language, e.g., for a language with booleans $\mathbb{B}$, integers $\mathbb{I}$, pairs, and functions: $? \equiv \mu Z^?.\, \mathbb{B} + \mathbb{I} + (Z^? \times Z^?) + (Z^? \to Z^?)$, but in our present setting, it leads to a question: should the type X in $\forall X.? \to X$ be included as one of the cases of the sum? Since X is in scope, it seems the answer should be yes—that is, intuitively we should view the type $\forall X.? \to X$ as follows:

$$\forall X.? \to X \equiv \forall X.?_X \to X \qquad\qquad ?_X \equiv \mu Z^?.\, \mathbb{B} + \mathbb{I} + (Z^? \times Z^?) + (Z^? \to Z^?) + X$$

With the above interpretation of $\forall X.? \to X$, based on the type alone, a term of this type does not always have to produce an error when applied: it could successfully project the value of type X from the sum $?_X$ and return it.

There is, however, an additional subtlety at play due to type-name generation: note that X is actually replaced by a type name α generated at type instantiation. Hence, we should model ? not as a statically fixed sum of a few cases, but instead as a *dynamically extensible* sum type that can have new cases added to it at runtime, one case for each fresh type name generated. So the only scenario in which $\mathtt{id}_?[A]v'$ can succeed (not error) is when v' is a value $v_A : A$ sealed with α and then embedded into ?, i.e., v' of the form $v_A : A :: \alpha :: ?$. But in all prior work on gradual parametricity, such a value is impossible to directly construct given the scoping rules for normal type-application syntax since X is not in scope for v', and temporally, the α we wish to replace X with has not been generated yet! The only way that such a value is constructed in GSF is indirectly by casting a term like $\mathtt{id}_X$.

Our analysis indicates that previous approaches have failed to satisfy both parametricity and graduality because they have insisted on using the syntax of polymorphic lambda calculus while introducing exotic type-name generation mechanisms at runtime to enable parametricity, while hiding the presence of these mechanisms from the surface language programmer. Our solution to this problem is to change the *syntax* of the gradual language, giving a new language that takes name generation into account. Our language design follows the methodology laid out in New and Ahmed [15] that shows that graduality holds if all casts are given by a coherent system of *embedding-projection pairs*. From this point of view, the flaw in previous approaches is that in addition to inserting dynamic type casts, the semantics also introduces *conversions* between fresh types like α and instantiating types like $\mathbb{B}$, which do not come in embedding-projection pairs. Our solution then is to expose the conversion between these types to the programmer herself which not only side-steps the graduality violations of previous semantics, but also increases the expressive power of the language, since the programmer can directly specify which values should be sealed, and which should not.

We summarize the contributions of this work as follows

- We identify a problem with Toro et al. [24]'s analysis of why parametricity and graduality are incompatible and identify some non-parametric behavior in their semantics(§2).
- We present a new surface language PolyG$^\nu$ with explicit polymorphism that supports a novel form of universal and existential types where the creation of fresh types is exposed in a controlled way. The semantics of PolyG$^\nu$ is similar to previous gradual parametric languages, but the typing is very different.
- We elaborate PolyG$^\nu$ into an explicit cast calculus PolyC$^\nu$. We then give a translation from PolyC$^\nu$ into a typed target language, CBPV$_{\text{OSum}}$, essentially call-by-push-value with polymorphism and an extensible sum type.
- We develop a novel logical relation that proves both graduality and parametricity for PolyG$^\nu$. Thus, we show that parametricity and graduality are compatible, and we strengthen the connection alluded to by New and Ahmed [15] that graduality and parametricity are analogous properties.

Complete typing rules, definitions, and proofs are in the included technical appendix, submitted as anonymous supplementary material.

2 THE LANDSCAPE OF GRADUAL PARAMETRIC SEMANTICS

Example	λB	F_G	GSF	PolyG$^\nu$ w/o seal	PolyG$^\nu$ w/ seal, unseal
① $(((\Lambda X.\lambda x : X.x) :: \forall X.X \to ?)\ [\mathbb{I}]\ 1) + 3$	$\mathfrak{U}$	ill-typed	4	ill-typed	4
② $(((\Lambda X.\lambda x : X.x) :: ?)\ [\mathbb{I}]\ \mathtt{true})$	true	true	$\mathfrak{U}$	$\mathfrak{U}$	ill-typed
③ $(((\Lambda X.\lambda x : X.x) :: \forall X.? \to X)\ [\mathbb{I}]\ 5)$	$\mathfrak{U}$	ill-typed	5	$\mathfrak{U}$	5
④ $((\Lambda X.\lambda x : X.\mathtt{true}) :: \forall X.? \to \mathbb{B})\ [\mathbb{I}]\ 5$	true	ill-typed	true	$\mathfrak{U}$	true
⑤ $((\Lambda X.\lambda x : X.\mathtt{true}) :: \forall X.? \to \mathbb{B})\ [\mathbb{B}]\ 5$	true	ill-typed	$\mathfrak{U}$	$\mathfrak{U}$	ill-typed
⑥ $((\Lambda X.\lambda x : ?.\mathtt{true}) :: \forall X.? \to \mathbb{B})\ [\mathbb{B}]\ 5$	true	true	true	true	true

Table 1. Examples Illustrating Differences in Semantics (read colors as black for all but last column)

Table 1 presents examples to highlight some of the surprising behavior of prior gradual polymorphic designs, λB, F_G, and GSF, including examples that show GSF violates parametricity. We discuss these examples next and informally explain how they're handled in PolyG$^\nu$.

The first two examples are taken from Toro et al. [24] (henceforth, TLT).[1] Example ① casts the polymorphic identity function to have return type ?. Instantiating this with the integer type and passing 1, we would expect 1, so adding 3 to that should yield 4. TLT point out that λB returns a sealed 1 which is unusable directly: attempting to add the sealed value to 3 leads to an error, while in F_G, X is not consistent with ?, so the ascription is not well-typed. GSF fixes the problem: in effect, their semantics *unseals* the sealed 1 when the scope of the polymorphic function ends, so the addition succeeds.

In PolyG$^\nu$, we rely on the programmer to make her intent to seal and unseal explicit, instead of trying to "guess" what she intends. As written (column: w/o seal), the program is ill-typed: the function has as input the type X, but the programmer has supplied an integer 1. On the other hand, if the programmer replaces the red 1 with $\mathtt{seal}_X 1$, the function will return a value $\mathtt{seal}_X 1$, and if the programmer unseals, replacing the (. . .), then we unseal with $\mathtt{unseal}_X(\ldots)$ then we unseal the result $\mathtt{seal}_X 1$ producing 1, so the subsequent addition succeeds, reproducing GSF's behavior. However if the programmer seals, but does not unseal, then we will reduce to adding $\mathtt{seal}_X 1$ to 3 which will error, reproducing λB's behavior.

TLT use example ② to point out that λB and F_G fail to catch certain errors. If the programmer instantiates $\mathtt{id}_X :: ?$ with $\mathbb{I}$ then that should behave like an $\mathbb{I} \to \mathbb{I}$ function wrapped in ?, which means passing in a boolean `true` should produce an error. λB and F_G simply return `true` and GSF fixes this, producing an error. In PolyG$^\nu$, if the programmer passes in `true` without sealing, then reducing the elaborated term we have `true` cast to ? and then to X, which produces an error since $\mathbb{B}$ is not X. However, if the programmer instantiates X with $\mathbb{I}$ then replaces `true` with $\mathtt{seal}_X\mathtt{true}$, the latter fails to type check since only integers can be sealed at type X.

Example ③ revisits the type $\forall X.? \to X$ discussed in §1, casting $\mathtt{id}_X$ into that type. Again, in F_G the ascription is statically rejected. Applying this function always results in error in λB. In fact, in λB, we can prove the following free theorem: for any term $M : \forall X.? \to X$ and any type A and $v : A$, $M\ [A]\ v$ diverges or reduces to $\mathfrak{V}$. However, in GSF there is no such free theorem and example ③ successfully reduces to 5. GSF seems to take the view that here this type should be viewed as $\forall X.?_X \to X$ as we sketched in §1. They *optimistically assume that the programmer must have intended to seal* 5 *with* X which is then embedded in ?; that reasoning means if we cast this to X, we can extract the underlying 5-sealed-with-X, which is unsealed when the scope of the polymorphic function ends, so we get 5. In PolyG$^\nu$, we do not make assumptions about the programmer intent. If the programmer passes in an unsealed value, the program errors since 5 cast to ? and then to X fails because 5 has type $\mathbb{I}$ not X. But if the programmer replaces red 5 with $\mathtt{seal}_X 5$ then casting it to ? then X yields $\mathtt{seal}_X 5$, which is unsealed when the scope of (. . .) ends, producing 5. So again, we are able to reproduce both behaviors rather than choosing one arbitrarily.

Examples ④ and ⑤ shows what is intuitively a parametricity violation of the GSF semantics. Here, a $\forall X.X \to \mathbb{B}$ constant function is cast to $\forall X.? \to \mathbb{B}$. By parametricity, applying this to $\mathbb{I}$ and 5 or $\mathbb{B}$ and 5 should produce the same result. And in λB they do. But in GSF, even though we have the same program being applied to the *same* input 5, the programs produce *unrelated* outputs! The parametricity violation is surprising since GSF comes with a proof of parametricity. The example illustrates that GSF is actually allowing a form of intensional type analysis, affecting the behavior of the program. We believe this behavior arises due to the desire for a semantics that (a) errors on ② and (b) produces 5 on ③.

In PolyG$^\nu$, ④ and ⑤ produce an error if the programmer passes in an unsealed 5 (due to casting 5 to ? to X during reduction as in earlier examples). With sealing for ④, passing in 5 sealed with X

[1] λB is a cast calculus so we assume the obvious translation of these programs into λB. F_G does not have ascription but it can be simulated with λ.

produces `true`, which matches GSF's behavior where they assume that the programmer intended to seal 5. However, with sealing for ⑤, replacing 5 with $\texttt{seal}_X 5$ results in an ill-typed program for the reasons discussed for ②. We argue that this is the right way to handle the combined desire for (a) and (b).

Finally, example ⑥ sheds light on the limits of GSF's optimistic assumption that the programmer meant to seal. ⑥ is identical to ⑤ but this time, neither the type of the inner function, nor the type that it's being ascribed contain an X in argument position. This time, unlike for ⑤, GSF assumes that the programmer did *not* mean to seal 5 with X, so we get `true` instead of $\mathfrak{U}$.

We should comment on why so many of these examples are ill-typed in F_G. In an effort to achieve graduality, Igarashi et al. [11] tweak their precision relation—and hence consistency, which is used in type checking—so that ? and X are unrelated. But this tweaked precision goes against the programmer's natural intuition about when one program is more precise than another as argued in the Refined Criteria [19]. From a semantic perspective, if ? is a sum that contains all the fresh names generated, including the one for X, then $X \sqsubseteq ?$ makes sense even under $\forall X$ since X would be a subset of ?. F_G's precision relation is consistent with the interpretation that ? is essentially a non-extensible sum, so the fresh name generated for X is not included in it.

3 PolyG$^\nu$: A GRADUAL LANGUAGE WITH POLYMORPHISM

Next, we present our gradual language that supports a version of existential and universal quantification while maintaining the crucial graduality property. The language has some unusual features, so we start with an extended example to illustrate what programs look like, and then in § 3.2 introduce the formal syntax and typing rules.

3.1 PolyG$^\nu$ Informally

Let's consider an example of existential types, since they are simpler than universal types in PolyG$^\nu$. In a typed, non-gradual language, we can define an abstract "flipper" type, $\texttt{FLIP} = \exists X.X \times (X \to X) \times (X \to \mathbb{B})$. The first element is the initial state, the second is a "toggle" function and the last element reads out the value as a concrete boolean.

Then we could create an instance of this abstract flipper using booleans as the carrier type X and negation as the toggle function $\texttt{pack}(\mathbb{B}, (\texttt{true}, (\texttt{NOT}, \texttt{id})))\ \texttt{as}\ \texttt{FLIP}$. Note that we must explicitly mark the existential package with a type annotation, because otherwise we wouldn't be able to tell which occurrences of $\mathbb{B}$ should be hidden and which should be exposed. With different type annotations, the same package could be given types $\exists X.\mathbb{B} \times (\mathbb{B} \to \mathbb{B}) \times (\mathbb{B} \to \mathbb{B})$ or $\exists X.X \times (X \to X) \times (X \to X)$.

The PolyG$^\nu$ language existential type works differently in a few ways. Firstly, we write $\exists^\nu$ rather than $\exists$ to emphasize that we are only quantifying over fresh types, and not arbitrary types. The equivalent of the above existential package would be written as

$$\texttt{pack}^\nu(X \cong \mathbb{B}, (\texttt{seal}_X \texttt{true}, ((\lambda x : X.\texttt{seal}_X(\texttt{NOT}(\texttt{unseal}_X x))), (\lambda x : X.\texttt{unseal}_X x))) : \texttt{FLIP}$$

The first thing to notice is that rather than just providing a type $\mathbb{B}$ to instantiate the existential, we write a declaration $X \cong \mathbb{B}$. The X here is a *binding position* and the body of the package is typed under the assumption that $X \cong \mathbb{B}$. Then, rather than *substituting* $\mathbb{B}$ for X when typing the body of the package, the type checker checks that the body has type $X \times ((X \to X) \times (X \to \mathbb{B}))$ under the assumption that $X \cong \mathbb{B}$:

$$X \cong \mathbb{B} \vdash (\texttt{seal}_X \texttt{true}, ((\lambda x : X.\texttt{seal}_X(\texttt{NOT}(\texttt{unseal}_X x))), (\lambda x : X.\texttt{unseal}_X x))) : X \times ((X \to X) \times (X \to \mathbb{B}))$$

Crucially, $X \cong \mathbb{B}$ is a *weaker* assumption than $X = \mathbb{B}$. In particular, there are no *implicit* casts from X to $\mathbb{B}$ or vice-versa, but the programmer can *explicitly* "seal" $\mathbb{B}$ values to be X using the form $\texttt{seal}_X M$, which is only well-typed under the assumption that $X \cong A$ for some A consistent with $\mathbb{B}$.

We also get a corresponding unseal form $\texttt{unseal}_X M$, and the runtime semantics in § 4.4 defines these to be a bijection. An interesting side-effect of making the difference between X and $\mathbb{B}$ explicit in the term is that existential packages do not require type annotations to resolve any ambiguities. For instance, unlike in the typed case, the gradual package above could *not* be ascribed the type $\exists^\nu X.\mathbb{B} \times ((\mathbb{B} \to \mathbb{B}) \times (\mathbb{B} \to \mathbb{B}))$ because the functions explicitly take X values, and not $\mathbb{B}$ values.

The corresponding elimination form for $\exists^\nu$ is a standard unpack: $\texttt{unpack}\ (X, x) = M; N$, where the continuation for the unpack is typed with just X and x added to the context, it doesn't know that $X \cong A$ for any particular A. We call this ordinary type variable assumption an *abstract type variable*, whereas the new assumption $X \cong A$ is a *known type variable* which acts more like a *type definition* than an abstract type. At runtime, when an existential is unpacked, a fresh type X is created that is isomorphic to A but whose behavior with respect to casts is different.

While explicit sealing and unsealing might seem burdensome to the programmer, note that this is directly analogous to a common pattern in Haskell, where modules are used in combination with `newtype` to create a datatype that at runtime is represented in the same way as an existing type, but for type-checking purposes is considered distinct. We give an analogous Haskell module as follows:

```
module Flipper(State, start, toggle, toBool) where
newtype State = Seal { unseal :: Bool }
start :: State
start = Seal True
inc :: State -> State
inc s = Seal (not (unseal s)
toBool :: State -> Bool
toBool = unseal
```

Then a different module that imports `Flipper` is analogous to an unpack, as its only interface to the `State` type is through the functions provided.

We also add universal quantification to the language, using the duality between universals and existentials as a guide. Again we write the type differently, as $\forall^\nu X.A$. In an ordinary polymorphic language, we would write the type of the identity function as $\forall X.X \to X$ and implement it using a Λ form: $\Lambda X.\lambda x : X.x$. The elimination form passes in a type for X. For instance applying the identity function to a boolean would be written as $\texttt{id}\ \mathbb{B}\texttt{true}$. And a free theorem tells us that this term must either diverge, error, or return `true`.

The introduction form Λ is dual to the unpack form, and correspondingly looks the same as the ordinary Λ: $\Lambda^\nu X.\lambda x : X.x : \forall^\nu X.X \to X$. We will refer to this term as $\texttt{id}^\nu$. The body of the Λ^ν is typed with an abstract type variable X in scope. The elimination form of type application is dual to the *pack* form, and so similarly introduce a *known* type variable assumption. Instantiating the identity function as above would be written as $\texttt{unseal}_X(\texttt{id}^\nu\{X \cong \mathbb{B}\}(\texttt{seal}_X\texttt{true})) : \mathbb{B}$. which introduces a known type variable $X \cong \mathbb{B}$ into the context. Rather than the resulting type being $\mathbb{B} \to \mathbb{B}$, it is $X \to X$ with the assumption $X \cong \mathbb{B}$. Then the argument to the function must be explicitly sealed as an X to be passed to the function. The output of the function is also of type X and so must be explicitly *unsealed* to get a boolean out. However, there is something quite unusual about this term: the $X \cong \mathbb{B}$ binding site is not binding X in a *subterm* of the application, but rather into the *context*: the argument is sealed, and the continuation is performing an unseal! These bindings in $\forall^\nu$ instantiations follow this "inside-out" structure and complicate the typing rules: every term in the language "exports" known type variable bindings that go outwards in addition to the other typing assumptions coming inwards from the context. While unusual, they are intuitively justified by the duality with existentials: we can think of the *continuation* for an instantiation of a $\forall^\nu$ as being analogous to the *body* of the existential package.

$$
\begin{array}{lrcl}
\text{types} & A & ::= & ? \mid X \mid \mathbb{B} \mid A \times A \mid A \to A \mid \exists^{\nu} X.A \mid \forall^{\nu} X.A \\
\text{Ground types} & G & ::= & X \mid \mathbb{B} \mid ? \times ? \mid ? \to ? \mid \exists^{\nu} X.? \mid \forall^{\nu} X.? \\
\text{terms} & M & ::= & x \mid M :: A \mid \mathtt{seal}_X M \mid \mathtt{unseal}_X M \mid \mathtt{is}(G)?\ M \mid \mathtt{true} \mid \mathtt{false} \\
 & & \mid & \mathtt{if}\ M\ \mathtt{then}\ M\ \mathtt{else}\ M \mid (M, M) \mid \mathtt{let}\ (x, x) = M; M \\
 & & \mid & M\ M \mid \lambda x : A.M \mid \mathsf{pack}^{\nu}(X \cong A, M) \mid \mathsf{unpack}\ (X, x) = M; N \\
 & & \mid & \Lambda^{\nu} X.M \mid M\{X \cong A\} \mid \mathtt{let}\ x = M; M \\
\text{environment} & \Gamma & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, X \mid \Gamma, X \cong A
\end{array}
$$

Fig. 1. PolyG$^{\nu}$ Syntax

3.2 PolyG$^{\nu}$ Formal Syntax and Semantics

Figure 1 presents the syntax of the surface language types, terms and environments. Most of the language is a typical gradual functional language, using ? as the dynamic type, and including type ascription $M :: A$. The unusual aspects of the language are the $\mathtt{seal}_X M$ and $\mathtt{unseal}_X M$ forms and the "fresh" existential $\exists^{\nu} X.A$ and universal $\forall^{\nu} X.A$. Note also the non-standard environments Γ, which include ordinary typing assumptions $x : A$, abstract type variable assumptions X and known type variable assumptions $X \cong A$.

The typing rules are presented in Figure 2. On a first pass, we suggest ignoring all shaded parts of the rules, which only concern the inside-out scoping needed for the $\forall^{\nu} X.A$ forms and would not be necessary if this type was removed. We follow the usual formulation of gradual surface languages in the style of [20]: type checking is strict when checking compatibility of different connectives, but lax when the dynamic type is involved. The first $M :: B$ form is type-ascription, which is well formed when the types are *consistent* with each other, written $A \sim B$. We define this in the standard way in as being the least congruence relation including equality and rules making ? consistent with every type.

We include variable and let-binding rules, which are standard other than the shaded parts. Next, we include sealing $\mathtt{seal}_X M$ and unsealing $\mathtt{unseal}_X M$ forms. The sealing and unsealing forms are valid when the assumption $X \cong A$ is in the environment and give the programmer access to an explicit bijection between the types X and A. It is crucial for graduality to hold that this bijection is explicit and not implicit, because the behavior of casts involving X and A are very different. To show that it has no adverse effect on the calculus, we also include a form $\mathtt{is}(G)?\ M$ that checks at runtime whether M returns a value that is compatible with the ground type G. M can have any type in this case because it is always a safe operation, but the result is either trivially `true` or `false` unless M has type ?.

Next, we have booleans, whose values are `true` and `false`, and whose elimination form is an if-statement. The if-statement checks that the scrutinee has a type compatible with $\mathbb{B}$, and as in previous work [] uses gradual meet $B_t \sqcap B_f$ for the output type. Gradual meet is only partially defined, since this ensures that if the two sides have different (non-?) head connectives then type checking errors, in keeping with the philosophy of strict checking when precise types are used.

Next, we have pairs and functions, which are fairly standard. We use pattern-matching as the elimination form for pairs. To reduce the number of rules, we present the elimination forms in the style of Garcia and Cimini [8], using partial functions π_i, dom, cod and later un$\forall^{\nu}$, un$\exists^{\nu}$ to extract the subformula from a type "up to ?". For the correct type this extracts the actual subformula, but for ? is defined to be ? and for other connectives is undefined. We define these at the bottom of the figure, where uncovered cases are undefined. Next, we have existentials, which are as described in §3.1.

Finally, we consider the shaded components of the judgment. The full form of the judgment is $\Gamma \vdash M : A; \Gamma_o$ where Γ_o is the list of bindings that are *generated by* M and exported outward. Note that the type A of M can use variables in Γ_o as well as variables in Γ. Also, while we write these as Γ_o, Γ_M,

$$\frac{\Gamma \vdash M : A; \Gamma_o \qquad A \sim B}{\Gamma \vdash (M :: B) : B; \Gamma_o} \qquad \frac{x : A \in \Gamma}{\Gamma \vdash x : A; \cdot} \qquad \frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M, x : A \vdash N : B; \Gamma_N}{\Gamma \vdash \mathtt{let}\ x = M; N : B; \Gamma_M, \Gamma_N}$$

$$\frac{\Gamma \vdash M : B; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o \qquad B \sim A}{\Gamma \vdash \mathtt{seal}_X M : X; \Gamma_o} \qquad \frac{\Gamma \vdash M : B; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o \qquad B \sim X}{\Gamma \vdash \mathtt{unseal}_X M : A; \Gamma_o}$$

$$\frac{\Gamma \vdash M : A; \Gamma_o \qquad \Gamma, \Gamma_o \vdash G}{\Gamma \vdash \mathtt{is}(G)?\ M : \mathbb{B}; \Gamma_o} \qquad \Gamma \vdash \mathtt{true} : \mathbb{B}; \cdot \qquad \Gamma \vdash \mathtt{false} : \mathbb{B}; \cdot$$

$$\frac{\begin{array}{c}\Gamma \vdash M : A; \Gamma_M \qquad A \sim \mathbb{B} \\ \Gamma, \Gamma_M \vdash N_t : B_t; \Gamma_t \qquad \Gamma, \Gamma_M \vdash N_f : B_f; \Gamma_f\end{array}}{\Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ N_t\ \mathtt{else}\ N_f : B_t \sqcap B_f; \Gamma_M, \Gamma_t \cap \Gamma_f}$$

$$\frac{\Gamma \vdash M_1 : A_1; \Gamma_1 \qquad \Gamma, \Gamma_1 \vdash M_2 : A_2; \Gamma_2}{\Gamma \vdash (M_1, M_2) : A_1 \times A_2; \Gamma_1, \Gamma_2} \qquad \frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M, x : \pi_1(A), y : \pi_2(A) \vdash N : B; \Gamma_N}{\Gamma \vdash \mathtt{let}\ (x, y)\ =\ M; N : B; \Gamma_M, \Gamma_N}$$

$$\frac{\Gamma, x : A \vdash M : B; \Gamma_o}{\Gamma \vdash \lambda x : A.M : A \to B; \cdot} \qquad \frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M \vdash N : B; \Gamma_N \qquad \mathrm{dom}(A) \sim B}{\Gamma \vdash M\ N : \mathrm{cod}(A); \Gamma_M, \Gamma_N}$$

$$\frac{\Gamma, X \cong A \vdash M : B; \Gamma_o}{\Gamma \vdash \mathsf{pack}^\nu(X \cong A, M) : \exists^\nu X B; \cdot} \qquad \frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M, X, x : \mathrm{un}\exists^\nu(A) \vdash N : \Gamma_N \qquad \Gamma, \Gamma_M, \Gamma_N|_X \vdash B}{\Gamma \vdash \mathtt{unpack}\ (X, x) = M; N : B; \Gamma_M, \Gamma_N|_X}$$

$$\frac{\Gamma, X \vdash M : A; \Gamma_o}{\Gamma \vdash \Lambda^\nu X.M : \forall^\nu X.A; \cdot} \qquad \frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M \vdash B}{\Gamma \vdash M\{X \cong B\} : \mathrm{un}\forall^\nu(A); \Gamma_M, X \cong B}$$

$$? \sim A \qquad A \sim ? \qquad \mathbb{B} \sim \mathbb{B} \qquad X \sim X$$

$$\frac{A_i \sim B_i \qquad A_o \sim B_o}{A_i \to A_o \sim B_i \to B_o} \qquad \frac{A_1 \sim B_1 \qquad A_2 \sim B_2}{A_1 \times A_2 \sim B_1 \times B_2} \qquad \frac{A \sim B}{\exists^\nu X.A \sim \exists^\nu X.B} \qquad \frac{A \sim B}{\forall^\nu X.A \sim \forall^\nu X.B}$$

$$\begin{array}{rcl} \mathrm{dom}(A \to B) & = & A \\ \mathrm{dom}(?) & = & ? \\ \mathrm{cod}(A \to B) & = & B \\ \mathrm{cod}(?) & = & ? \\ \pi_i(A_1 \times A_2) & = & A_i \\ \pi_i(?) & = & ? \\ \mathrm{un}\forall^\nu(\forall^\nu X.A) & = & A \\ \mathrm{un}\forall^\nu(?) & = & ? \end{array} \qquad \begin{array}{rcl} \mathrm{un}\exists^\nu(\exists^\nu X.A) & = & A \\ \mathrm{un}\exists^\nu(?) & = & ? \end{array}$$

$$\begin{array}{lll} \cdot|_{\Gamma'} = \cdot & & \\ (X \cong A, \Gamma)|_{\Gamma'} = X \cong A, (\Gamma|_{\Gamma'}) & & (\mathrm{FV}(A) \cap \Gamma' = \emptyset) \\ (X \cong A, \Gamma)|_{\Gamma'} = \Gamma|_{\Gamma', X} & & (\mathrm{FV}(A) \cap \Gamma' \neq \emptyset) \end{array}$$

Fig. 2. PolyG$^\nu$ Type System

etc they only contain sequences of known type variables, and never any abstract type variables or typing assumptions $x : A$. These bindings are generated in the $\forall^\nu$ elimination rule, where the instantiation $M\{X \cong B\}$ adds $X \cong B$ to the output context. Rules that produce delayed thunks—the function, existential and universal introduction rules—have bodies that generate bindings, but these are not exported because these bindings will only be generated at the point where the thunk is

type names	α	::=	$\sigma \mid X$
types	A, B	+ ::=	σ
ground types	G	::=	$\alpha \mid \mathbb{B} \mid ? \times ? \mid ? \to ? \mid \exists^\nu X.? \mid \forall^\nu X.?$
precision derivations	$A^\sqsubseteq, B^\sqsubseteq$	::=	$? \mid \mathrm{tag}_G A^\sqsubseteq \mid \alpha \mid \mathbb{B} \mid A^\sqsubseteq \times A^\sqsubseteq \mid A^\sqsubseteq \to B^\sqsubseteq$
		|	$\exists^\nu X.A^\sqsubseteq \mid \forall^\nu X.A^\sqsubseteq$
values	V	::=	$\mathtt{seal}_\alpha V \mid \mathtt{true} \mid \mathtt{false} \mid x \mid (V, V) \mid \lambda(x : A).M$
		|	$\Lambda^\nu X.M \mid \mathtt{inj}_G\ V \mid \langle A_1^\sqsubseteq \to A_2^\sqsubseteq \rangle\!\uparrow M \mid \langle A_1^\sqsubseteq \to A_2^\sqsubseteq \rangle\!\downarrow M$
		|	$\langle \forall^\nu X.A^\sqsubseteq \rangle\!\uparrow M \mid \langle \forall^\nu X.A^\sqsubseteq \rangle\!\downarrow M \mid \mathtt{pack}^\nu (X \cong A', [A^\sqsubseteq \updownarrow, \ldots], M)$
expressions	M, N	− ::=	$(M :: A)$
		+ ::=	$\mho \mid \langle A^\sqsubseteq \rangle\!\uparrow M \mid \langle A^\sqsubseteq \rangle\!\downarrow M \mid \mathtt{hide}\ X \cong A; M$
		|	$\mathtt{pack}^\nu (X \cong A', [A^\sqsubseteq \updownarrow, \ldots], M) \mid \mathtt{seal}_\sigma M \mid \mathtt{unseal}_\sigma M$
		|	$\mathtt{inj}_G\ M$
Evaluation Context	E	::=	$[] \mid (E, M) \mid (V, E) \mid E[A] \mid E\ M \mid V\ E \mid \mathtt{inj}_G\ E$
		|	$\mathtt{if}\ E\ \mathtt{then}\ M\ \mathtt{else}\ M \mid \mathtt{let}\ (x, y) = E; M \mid \langle A^\sqsubseteq \rangle\!\uparrow E$
		|	$\mathtt{unpack}\ (X, x) = E; M \mid \mathtt{seal}_\alpha E \mid \mathtt{unseal}_\alpha E \mid \langle A^\sqsubseteq \rangle\!\downarrow E$

Fig. 3. PolyC$^\nu$ Syntax

forced to evaluate. The rest of the rules work similarly to an effect system: for instance in the function application rule $M\ N$ the bindings generated in M are bound in N, and the application produces all of the bindings they generate, and similarly for product introduction. In the unpack form, care must be taken to make sure that the X from the unpack is not leaked in the output Γ_N, in addition to making sure the output type B does not mention X. Any known type variables that mention X are removed from the output context, using the *restriction* form $\Gamma_{\Gamma'}$ defined at the bottom of Figure 2. Finally, in the if form, each branch might export different known type variables, so the if statement as a whole only exports the intersection of the two branches, since these are the only ones guaranteed to be generated.

4 PolyC$^\nu$: CAST CALCULUS

As is standard in gradual languages, rather than giving the surface language an operational semantics directly, we define a *cast calculus* that makes explicit the casts that perform the dynamic type checking in gradual programs. We present the cast calculus syntax in Figure 3. The cast calculus syntax is almost the same as the surface syntax, though the typing is quite different. First, the type ascription form is removed, and several forms are added to replace it. , two cast forms are added, and some new values are added Based on the analysis in [15], we add two cast forms: an upcast $\langle A^\sqsubseteq \rangle\!\uparrow M$ and a downcast $\langle A^\sqsubseteq \rangle\!\downarrow M$, whereas most prior work includes a single cast form $\langle A \Leftarrow B \rangle$. The $A^\sqsubseteq$ used in the upcast and downcast forms here is a proof that $A_l \sqsubseteq A_r$ for some types A_l, A_r, i.e., that A_l is a more precise (less dynamic) type than A_r. This type precision definition is key to formalizing the graduality property, but previous work has shown that it is useful for formalizing the semantics of casts as well. We emphasize the structure of these proofs because the central semantic constructions of this work: the operational semantics of casts, the translation of casts into functions and finally our graduality logical relation are all naturally defined by recursion on these derivations.

4.1 PolyC$^\nu$ Type Precision

We present the definition of type precision in Figure 4. Ignoring for the moment our notation for precision derivations, our definition of type precision is a simple extension of the usual notion: type variables are only related to the dynamic type and themselves, and similarly for $\forall$ and $\exists$. Since we have quantifiers and type variables, we include a context Γ of known and abstract type variables.

$$\frac{\Gamma \vdash A^{\sqsubseteq} : A \sqsubseteq G}{\Gamma \vdash \mathrm{tag}(G, A^{\sqsubseteq}) : A \sqsubseteq ?} \qquad \Gamma \vdash ? : ? \sqsubseteq ? \qquad \Gamma \vdash \mathbb{B} : \mathbb{B} \sqsubseteq \mathbb{B} \qquad \frac{X \in \Gamma}{\Gamma \vdash X : X \sqsubseteq X}$$

$$\frac{\Gamma \vdash A_1^{\sqsubseteq} : A_{l1} \sqsubseteq A_{r1} \qquad \Gamma \vdash A_2^{\sqsubseteq} : A_{l2} \sqsubseteq A_{r2}}{\Gamma \vdash A_1^{\sqsubseteq} \times A_2^{\sqsubseteq} : A_{l1} \times A_{l2} \sqsubseteq A_{r1} \times A_{r2}} \qquad \frac{\Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r \qquad \Gamma \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma \vdash A^{\sqsubseteq} \to B^{\sqsubseteq} : A_l \to B_l \sqsubseteq A_r \to B_r}$$

$$\frac{\Gamma, X \vdash A_l \sqsubseteq A_r}{\Gamma \vdash \exists^{\nu} X.A^{\sqsubseteq} : \exists^{\nu} X.A_l \sqsubseteq \exists^{\nu} X.A_r} \qquad \frac{\Gamma, X \vdash A_l \sqsubseteq A_r}{\Gamma \vdash \forall^{\nu} X.A^{\sqsubseteq} : \forall^{\nu} X.A_l \sqsubseteq \forall^{\nu} X.A_r}$$

Fig. 4. PolyC$^{\nu}$ Type Precision

Crucially, even under the assumption that $X \cong A$, X and A are *unrelated* precision-wise unless A is ?. As before, $X \in \Gamma$ ranges over both known and abstract type variables. It is easy to see that precision reflexive and transitive, and that ? is the greatest element. It is important to note that while we give a syntax for derivations, there is at most *one* derivation that $A \sqsubseteq B$ for any two types. Finally, ? is the least precise type, meaning for any type A there is a derivation that $A \sqsubseteq ?$. We prove these and several more lemmas about type precision in the appendix.

4.2 PolyC$^{\nu}$ Type System

The static type system for the cast calculus is given Figure 5. The cast calculus type system differs from the surface language in that all type checking is *strict* and precise. This manifests in two ways. First, the dynamic type is not considered implicitly compatible with other types. Instead, in the translation from PolyG$^{\nu}$ to PolyC$^{\nu}$, we must insert casts consistency is used in the judgment. Second, in the if rule, the branches must have the *same* type, and an upcast is inserted in the translation to make the two align. Finally, the outward scoping of known type variables is handled more explicitly. We add a new form $\texttt{hide}\ X \cong A; M$ that delimits the scope of $X \cong A$ from going further outward. Then in rules that include delayed computations, i.e., values of function, existential and universal type, whereas in the surface language the delayed term could produce any names, now in PolyC$^{\nu}$, they must all be manually hidden. Similarly in the branches of an if statement, the two sides must have the same generated names, and hides must be used in the elaboration to make them align.

4.3 Elaboration from PolyG$^{\nu}$ to PolyC$^{\nu}$

We define the elaboration of PolyG$^{\nu}$ into the cast calculus PolyC$^{\nu}$ in Figure 6. Following [15], an ascription is interpreted as a cast up to ? followed by a cast down to the ascribed type. Most of the elaboration is standard, with elimination forms being directly translated to the corresponding PolyC$^{\nu}$ form if the head connective is correct, and inserting a downcast if the elimination position has type ?. We formalize this using the metafunction $G \wr M$ defined towards the bottom of the figure. For the if case, in PolyC$^{\nu}$ the two branches of the if have to have the same output type and export the same names, so we downcast each branch, but also we hide any names not generated by both sides, using the metafunction $\texttt{hide}\ \Gamma \subseteq \Gamma'; M$, defined at the bottom of the figure, which hides all names present in Γ' that are not in Γ. Finally, in the values that are thunks (pack, λ and Λ), the bodies of the thunks must not generate names in PolyC$^{\nu}$, so we hide names there as well.

4.4 PolyC$^{\nu}$ Dynamic Semantics

The dynamic semantics of PolyC$^{\nu}$ extends traditional cast semantics with appropriate rules for our name-generating universals and existentials. The runtime state is a pair of a term M and a *case store* Σ. A case store Σ represents the set of cases allocated so far in the program. We use a very concrete representation: a store Σ is just a pair of a number $\Sigma.n$ and a function $\Sigma.f : [n] \to \mathrm{Ty}$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A; \cdot} \qquad \frac{\Gamma \vdash M : A_l; \Gamma_M \qquad \Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma \vdash \langle A^{\sqsubseteq} \rangle\!\uparrow M : A_r; \Gamma_M} \qquad \frac{\Gamma \vdash M : A_r; \Gamma_M \qquad \Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma \vdash \langle A^{\sqsubseteq} \rangle\!\downarrow M : A_l; \Gamma_M}$$

$$\frac{\Gamma \vdash M : A; \Gamma_M \qquad \Gamma, \Gamma_M, x : A \vdash N : B; \Gamma_N}{\Gamma \vdash \mathtt{let}\ x = M; N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Gamma \vdash M : A; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o}{\Gamma \vdash \mathtt{seal}_X M : X; \Gamma_o}$$

$$\frac{\Gamma \vdash M : X; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o}{\Gamma \vdash \mathtt{unseal}_X M : A; \Gamma_o} \qquad \frac{\Gamma \vdash M : ?; \Gamma_o \qquad \Gamma \vdash G}{\Gamma \vdash \mathtt{is}(G)?\ M : \mathbb{B}; \Gamma_o} \qquad \Gamma \vdash \mathtt{true} : \mathbb{B}; \cdot \qquad \Gamma \vdash \mathtt{false} : \mathbb{B}; \cdot$$

$$\frac{\Gamma \vdash M : \mathbb{B}; \Gamma_M \qquad \Gamma, \Gamma_M \vdash N_t : B; \Gamma_N \qquad \Gamma, \Gamma_M \vdash N_f : B; \Gamma_N}{\Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ N_t\ \mathtt{else}\ N_f : B; \Gamma_M, \Gamma_N} \qquad \frac{\Gamma \vdash M_1 : A_1; \Gamma_1 \qquad \Gamma, \Gamma_1 \vdash M_2 : A_2; \Gamma_2}{\Gamma \vdash (M_1, M_2) : A_1 \times A_2; \Gamma_1, \Gamma_2}$$

$$\frac{\Gamma \vdash M : A_1 \times A_2; \Gamma_M \qquad \Gamma, \Gamma_M, x : A_1, y : A_2 \vdash N : B; \Gamma_N}{\Gamma \vdash \mathtt{let}\ (x, y)\ =\ M; N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Gamma, x : A \vdash M : B; \cdot}{\Gamma \vdash \lambda x : A.M : A \to B; \cdot}$$

$$\frac{\Gamma \vdash M : A \to B; \Gamma_M \qquad \Gamma, \Gamma_M \vdash N : A; \Gamma_N}{\Gamma \vdash M\ N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Gamma, X \cong A \vdash M : B; \cdot}{\Gamma \vdash \mathsf{pack}^{\nu}(X \cong A, M) : \exists^{\nu} X.B; \cdot}$$

$$\frac{\Gamma \vdash M : \exists^{\nu} X.A; \Gamma_M \qquad \Gamma, \Gamma_M, X, x : A \vdash N : B; \Gamma_N \qquad \Gamma, \Gamma_M \vdash \Gamma_N \qquad \Gamma, \Gamma_M, \Gamma_N \vdash B}{\Gamma \vdash \mathtt{unpack}\ (X, x) = M; N : B; \Gamma_M, \Gamma_N}$$

$$\frac{\Gamma, X \vdash M : A; \cdot}{\Gamma \vdash \Lambda^{\nu} X.M : \forall^{\nu} X.A; \cdot} \qquad \frac{\Gamma \vdash M : \forall^{\nu} X.A; \Gamma_M \qquad \Gamma, \Gamma_M \vdash B}{\Gamma \vdash M\{X \cong B\} : A; \Gamma_M, X \cong B} \qquad \frac{\Gamma \vdash M : \Gamma_M, X \cong A, \Gamma'_M \qquad \Gamma, \Gamma_M \vdash \Gamma'_M}{\Gamma \vdash \mathtt{hide}\ X \cong A; M; \Gamma_M, \Gamma'_M}$$

Fig. 5. PolyC$^{\nu}$ Typing

where Ty is the set of all types and $[n] = \{m \in \mathbb{N} \mid m < n\}$ is from some prefix of natural numbers to types. All rules take configurations $\Sigma \triangleright M$ to configurations $\Sigma' \triangleright M'$. When the step does not change the store, we write $M \mapsto M'$ for brevity.

The first rule states that all non-trivial evaluation contexts propagate errors. Next, unsealing a seal gets out the underlying value, and $\mathtt{is}(G)?\ V$ literally checks if the tag of V is G. Next, there are two rules for unpacking an existential: one for a regular pack, and the other for unpacking an existential that has been cast. Here we use $\updownarrow$ to indicate one of $\uparrow$ and $\downarrow$. When unpacking a pack that has been cast, the cast is performed on the body of the package with the new type σ substituted for X. Next, rather than generate the fresh types in the instantiation reduction, we generate them in the *hide* reduction and then the instantiation reduction is a simple β reduction. As is typical for a cast calculus, the remaining types have ordinary call-by-value β reduction, and the complexity is in the semantics of casts.

Other than the use of type precision derivations, the behavior of our casts is mostly standard: identity casts for $\mathbb{B}$, σ and ? are just the identity, and the product cast proceeds structurally. Function casts are values, and when applied to a value, the cast is performed on the output and the oppositely oriented case on the input. We use $\updownarrow^{-}$ to indicate the opposite arrow, so $\uparrow^{-} = \downarrow$ and $\downarrow^{-} = \uparrow$ to cut down the number of rules. Next, the $\forall^{\nu}$ casts are also values that reduce when the instantiating type is supplied. As with existentials, the freshly generated type σ is substituted for X in the precision derivation guiding the cast. Finally, the upcast case for $\mathrm{tag}_G A^{\sqsubseteq}$ simply injects the result of upcasting with $A^{\sqsubseteq}$ into the dynamic type using the tag G. For the downcast case, the opposite is done if the input has the right tag, and otherwise a dynamic type error is raised.

$$
\begin{aligned}
(M :: B)^+ &= \langle B?^{\sqsubseteq}\rangle\!\downarrow \langle A?^{\sqsubseteq}\rangle\!\uparrow M^+ && \text{(where } M : A, A?^{\sqsubseteq} : A \sqsubseteq ?, B?^{\sqsubseteq} : B \sqsubseteq ?\text{)}\\
x^+ &= x\\
(\texttt{let } x = M; N)^+ &= \texttt{let } x = M^+; N^+\\
(\texttt{seal}_X M)^+ &= \texttt{seal}_X (M :: A)^+ && \text{(where } X \cong A\text{)}\\
(\texttt{unseal}_X M)^+ &= \texttt{unseal}_X (X \wr M)\\
(\texttt{is}(G)?\ M)^+ &= \texttt{is}(G)?\ (\langle A?^{\sqsubseteq}\rangle\!\uparrow M) && \text{(where } M : A, A?^{\sqsubseteq} : A \sqsubseteq ?\text{)}\\
\texttt{true}^+ &= \texttt{true}\\
\texttt{false}^+ &= \texttt{false}\\
(\texttt{if } M \texttt{ then } N_t \texttt{ else } N_f)^+ &= \texttt{if } \mathbb{B} \wr M\\
&\quad \texttt{then } (\langle B_t^{\sqsubseteq}\rangle\!\downarrow \texttt{hide } \Gamma_t \subseteq \Gamma_t \cap \Gamma_f; N_t^+)\\
&\quad \texttt{else } (\langle B_f^{\sqsubseteq}\rangle\!\downarrow \texttt{hide } \Gamma_f \subseteq \Gamma_t \cap \Gamma_f; N_f^+)\\
&\qquad \text{(and } \Gamma \vdash \texttt{if } M \texttt{ then } N_t \texttt{ else } N_f : B_t \sqcap B_f; \Gamma_M, \Gamma_t \cap \Gamma_f\text{)}\\
&\qquad \text{(and } B_t^{\sqsubseteq} : B_t \sqcap B_f \sqsubseteq B_t, B_f^{\sqsubseteq} : B_t \sqcap B_f \sqsubseteq B_f\text{)}\\
(M_1, M_2)^+ &= (M_1^+, M_2^+)\\
(\texttt{let } (x, y) = M; N)^+ &= \texttt{let } (x, y) = ? \times ? \wr M; N^+\\
(\lambda x : A.M)^+ &= \lambda x : A.\texttt{hide } \Gamma_o \subseteq \Gamma_o; M^+ && \text{(where } \Gamma, x : A \vdash M : B; \Gamma_o\text{)}\\
(M\ N)^+ &= (? \to ? \wr M)\ (N :: \mathrm{dom}(A))^+ && \text{(where } M : A\text{)}\\
(\texttt{pack}^\nu (X \cong A, M))^+ &= \texttt{pack}^\nu (X \cong A, \texttt{hide } \Gamma_o \subseteq \Gamma_o; M^+) && \text{(where } M : B; \Gamma_o\text{)}\\
(\texttt{unpack } (X, x) = M; N)^+ &= \texttt{unpack } (X, x) = \exists^\nu X.? \wr M; \texttt{hide } \Gamma_N|_X \subseteq \Gamma_N; N^+\\
\Lambda^\nu X.M^+ &= \Lambda^\nu X.\texttt{hide } \Gamma_o \subseteq \Gamma_o; M^+ && \text{(where } M : A; \Gamma_o\text{)}\\
M\{X \cong B\}^+ &= (\forall^\nu X.? \wr M)\{X \cong B\}\\
G \wr M &= \langle \mathrm{tag}(G, G)\rangle\!\downarrow M^+ && \text{(when } M : ?\text{)}\\
G \wr M &= M^+ && \text{(otherwise)}\\
\texttt{hide } \Gamma_s \subseteq (\Gamma_b, X \cong A); M &= \texttt{hide } \Gamma_s \subseteq \Gamma_b; \texttt{hide } X \cong A; M && (X \notin \Gamma_s)\\
\texttt{hide } (\Gamma_s, X \cong A) \subseteq (\Gamma_b, X \cong A); M &= \texttt{hide } \Gamma_s \subseteq \Gamma_s; M\\
\texttt{hide } \cdot \subseteq \cdot; M &= M
\end{aligned}
$$

Fig. 6. Elaborating PolyG$^\nu$ to PolyC$^\nu$

In the appendix, we extend the typing to runtime terms which are typed with respect to a case-store and prove a standard type safety theorem for the language that terms either take a step or are values or errors.

5 TYPED INTERPRETATION OF THE CAST CALCULUS

In the previous section we developed a cast calculus with an operational semantics defining the behavior of the name generation and gradual type casts. However, this ad hoc design addition of new type connectives and inside-out scoping of $\forall^\nu$-instantiations make the cast calculus less than ideal for proving meta-theoretic properties of the system.

Instead of directly proving metatheoretic properties of the cast calculus, we give a *contract translation* of the cast calculus into a statically typed core language, translating the gradual type casts to ordinary terms in the typed language that raise errors. The key benefit of the typed language is that it does not have built-in notions of fresh existential and universal quantification. Instead, the type translation decomposes these features into the combination of *ordinary* existential

$$
\begin{array}{lcl}
E[\mho] & \mapsto & \mho \text{ where } E \neq [] \\
E[\mathtt{unseal}_\sigma(\mathtt{seal}_\sigma V)] & \mapsto & E[V] \\
E[\mathtt{is}(G)?\ (\mathtt{inj}_G\ V)] & \mapsto & E[\mathtt{true}] \\
E[\mathtt{is}(G)?\ (\mathtt{inj}_H\ V)] & \mapsto & E[\mathtt{false}] \text{ where } G \neq H \\
\Sigma \triangleright E[\mathtt{hide}\ X \cong A; M] & \mapsto & \Sigma, \sigma : A \triangleright E[M[\sigma/X]] \\
\Sigma \triangleright E\left[\begin{array}{l}\mathtt{unpack}\ (X,x) = \mathsf{pack}^\nu(X \cong A', [A^{\sqsubseteq}\updownarrow, \ldots], M); \\ N\end{array}\right] & \mapsto & \Sigma, \sigma : A' \triangleright E\left[\begin{array}{l}\mathtt{let}\ x = \langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow \ldots M[\sigma/X]; \\ N[\sigma/X]\end{array}\right] \\
E\,[\mathsf{pack}^\nu(X \cong A, M)] & \mapsto & E[\mathsf{pack}^\nu(X \cong A', [], M)] \\
E[(\Lambda^\nu X.M)\{\sigma \cong A\}] & \mapsto & E[M[\sigma/X]] \\
E[(\lambda(x:A).M)\ V] & \mapsto & E[M[V/x]] \\
E[\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2] & \mapsto & E[M_1] \\
E[\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2] & \mapsto & E[M_2] \\
E[\mathtt{let}\ (x,y) = (V_1, V_2); M] & \mapsto & E[M[V_1/x][V_2/y]] \\
E[\mathtt{let}\ x = V; M] & \mapsto & E[M[V/x]] \\
E[\langle\mathbb{B}\rangle\updownarrow\ V] & \mapsto & E[V] \\
E[\langle\sigma\rangle\updownarrow\ V] & \mapsto & E[V] \\
E[\langle ?\rangle\updownarrow\ V] & \mapsto & E[V] \\
E[\langle A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}\rangle\updownarrow\ (V_1, V_2)] & \mapsto & E[(\langle A_1^{\sqsubseteq}\rangle\updownarrow\ V_1, \langle A_2^{\sqsubseteq}\rangle\updownarrow\ V_2)] \\
E[(\langle A_1^{\sqsubseteq} \to A_2^{\sqsubseteq}\rangle\updownarrow\ V_1)\ V_2] & \mapsto & E[\langle A_2^{\sqsubseteq}\rangle\updownarrow\ (V_1\ \langle A_1^{\sqsubseteq}\rangle\updownarrow^-\ V_2)] \\
E[\langle\exists^\nu X.A^{\sqsubseteq}\rangle\updownarrow\ \mathsf{pack}^\nu(X \cong A', [A'^{\sqsubseteq}\updownarrow', \ldots], M)] & \mapsto & E[\mathsf{pack}^\nu(X \cong A', [A^{\sqsubseteq}\updownarrow, A'^{\sqsubseteq}\updownarrow', \ldots], M)] \\
E[(\langle\forall^\nu X.A^{\sqsubseteq}\rangle\updownarrow\ V)\{\sigma \cong A\}] & \mapsto & E[\langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow\ (V\{\sigma \cong A\})] \\
E[\langle \mathrm{tag}_G A^{\sqsubseteq}\rangle\updownarrow\ V] & \mapsto & E[\mathtt{inj}_G\ \langle A^{\sqsubseteq}\rangle\updownarrow\ V] \\
E[\langle \mathrm{tag}_G A^{\sqsubseteq}\rangle\downarrow\ \mathtt{inj}_G\ V] & \mapsto & E[\langle A^{\sqsubseteq}\rangle\downarrow\ V] \\
E[\langle \mathrm{tag}_G A^{\sqsubseteq}\rangle\downarrow\ \mathtt{inj}_H\ V] & \mapsto & E[\mho] \text{ where } H \neq G
\end{array}
$$

Fig. 7. PolyC$^\nu$ Operational Semantics

and universal quantification combined with a somewhat well-studied programming feature: a dynamically extensible "open" sum type we call OSum. Finally, it gives a static type interpretation of the dynamic type: rather than being a finitary sum of a few statically fixed cases, the dynamic type is implemented as the open sum type which includes those types allocated at runtime.

5.1 Typed Metalanguage

We present the syntax of our typed language CBPV$_{\mathrm{OSum}}$ in Figure 8, an extension of Levy's Call-by-push-value calculus [13], which we use as a convenient metalanguage to extend with features of interest. Call-by-push-value (CBPV) is a typed calculus with highly explicit evaluation order, providing similar benefits to continuation-passing style and A-normal form [17]. The main distinguishing features of CBPV are that values V and effectful computations M are distinct syntactic categories, with distinct types: value types A and computation types B. The two "shift" types U and F mediate between the two worlds. A value of type UB is a first-class "thUnk" of a computation of type B that can be forced, behaving as a B. A computation of type FA is a computation that can perform effects and return a value of type A, and whose elimination form is a monad-like *bind*. Notably while sums and (strict) tuples are value types, function types $A \to B$ are computations since a function interacts with its environment by receiving an argument. We include existentials as value type and universals as computation types, that in each case quantify over *value* types because we are using it as the target of a translation from a call-by-value language.

We furthermore extend CBPV with two new value types: OSum and Case A, which add an open sum type similar to the extensible exception types in ML and Haskell, but with an expression-oriented interface more suitable to a core calculus. The open sum type OSum is initially empty, but

$$
\begin{array}{rrcl}
\text{value types} & A & ::= & X \mid \mathsf{Case}\ A \mid \mathsf{OSum} \mid A \times A \mid \mathbb{B} \mid \exists X.A \mid UB \\
\text{computation types} & B & ::= & A \to B \mid \forall X.B \mid FA \\
\text{values} & V & ::= & \sigma \mid \mathtt{inj}_V\ V \mid \mathtt{pack}(A, V)\ \mathtt{as}\ \exists X.A \mid x \mid (V, V) \\
 & & \mid & \mathtt{true} \mid \mathtt{false} \mid \mathtt{thunk}\ M \\
\text{computations} & M & ::= & \mathfrak{V} \mid \mathtt{force}\ V \mid \mathtt{ret}\ V \mid x \leftarrow M; N \mid M\ V \mid \lambda x : A.M \\
 & & \mid & \mathtt{newcase}_A\ x; M \mid \mathtt{match}\ V\ \mathtt{with}\ V\{\mathtt{inj}\ x.M \mid N\} \\
 & & \mid & \mathtt{unpack}\ (X, x) = V; M \mid \mathtt{let}\ (x, x) = V; M \mid \Lambda X.M \mid M[A] \\
 & & \mid & \mathtt{if}\ V\ \mathtt{then}\ M\ \mathtt{else}\ M \\
\text{stacks} & S & ::= & \bullet \mid S\,V \mid S\,[A] \mid x \leftarrow S; M \\
\text{value typing context} & \Gamma & ::= & \cdot \mid \Gamma, x : A \\
\text{type variable context} & \Delta & ::= & \cdot \mid \Delta, X
\end{array}
$$

Fig. 8. $\text{CBPV}_{\text{OSum}}$ Syntax

can have new cases allocated at runtime. A value of Case A is a first class representative of a case of OSum. The introduction form $\mathtt{inj}_{V_c}\ V$ for OSum uses a case $V_c : \mathsf{Case}\ A$ to inject a value $V : A$ into OSum. The elimination form $\mathtt{match}\ V_o\ \mathtt{with}\ V_c\{\mathtt{inj}\ x.M \mid N\}$ for OSum is to use a $V_c : \mathsf{Case}\ A$ to do a pattern match on a value $V_o : \mathsf{OSum}$. Since OSum is an *open* sum type, it is unknown what cases V_o might use, so the pattern-match has two branches: the one $\mathtt{inj}\ x.M$ binds the underlying value to $x : A$ and proceeds as M and the other is a catch-all case N in case V_o was not constructed using V_c. Finally, there is a form $\mathtt{newcase}_A\ x; M$ that allocates a fresh Case A, binds it to x and proceeds as M.

5.2 Static and Dynamic Semantics

We show a fragment of the typing rules for $\text{CBPV}_{\text{OSum}}$ in Figure 9. There are two judgments corresponding to the two syntactic categories of terms: $\Delta; \Gamma \vdash V : A$ for typing a value and $\Delta; \Gamma \vdash M : A$ for typing a computation. Δ is the environment of type variables and Γ is the environment for term variables. Unlike in PolyG^ν and PolyC^ν, these are completely standard, and there is no concept of a known type variable.

First, an error $\mathfrak{V}$ is a computation and can be given any type. Variables are standard and the OSum/Case forms are as described above. Existentials are a value form and are standard as in CBPV using ordinary substitution in the pack form. In all of the value type elimination rules, the discriminee is restricted to be a *value*. A computation $M : B$ can be thunked to form a value $\mathtt{thunk}\ M : UB$, which can be forced to run as a computation. Like the existentials, the universal quantification type is standard, using substitution in the elimination form. Finally, the introduction form for FA returns a value $V : A$, and the elimination form is a bind, similar to a monadic semantics of effects, except that the continuation can have any computation type B, rather than restricted to FA.

A representative fragment of the operational semantics is given in Figure 10, the full semantics are in the appendix. S represents a *stack*, the CBPV analogue of an evaluation context, defined in Figure 8. Here Σ is like the Σ in PolyC^ν, but maps to *value types*. The semantics is standard, other than the fact that we assign a count to each step of either 0 or 1. The only steps that count for 1 are those that introduce non-termination to the language, which is used later as a technical device in our logical relation in Section 6.

5.3 Translation

Next, we present the "contract translation" of PolyC^ν into $\text{CBPV}_{\text{OSum}}$. This translation can be thought of as an alternate semantics to operational semantics for PolyC^ν, but with a tight correspondence given in §5.4. Since $\text{CBPV}_{\text{OSum}}$ is a typed language that uses ordinary features like

$$\frac{}{\Delta;\Gamma \vdash \mho : B} \qquad \frac{}{\Delta;\Gamma, x : A, \Gamma' \vdash x : A} \qquad \frac{\Delta;\Gamma \vdash V_c : \mathrm{Case}\ A \qquad \Delta;\Gamma \vdash V : A}{\Delta;\Gamma \vdash \mathtt{inj}_{V_c}\ V : \mathrm{OSum}}$$

$$\frac{\Delta;\Gamma \vdash V : \mathrm{OSum} \qquad \Delta;\Gamma \vdash V_c : \mathrm{Case}\ A \qquad \Delta;\Gamma, x : A \vdash M : B \qquad \Delta;\Gamma \vdash N : B}{\Delta;\Gamma \vdash \mathtt{match}\ V\ \mathtt{with}\ V_c\{\mathtt{inj}\ x.M \mid N\} : B}$$

$$\frac{\Delta \vdash A \qquad \Delta;\Gamma, x : \mathrm{Case}\ A \vdash M : B}{\Delta;\Gamma \vdash \mathtt{newcase}_A\ x; M : B} \qquad \frac{\Delta;\Gamma \vdash V : A[A'/X]}{\Delta;\Gamma \vdash \mathtt{pack}(A', V)\ \mathtt{as}\ \exists X.A : \exists X.A}$$

$$\frac{\Delta;\Gamma \vdash V : \exists X.A \qquad \Delta \vdash B \qquad \Delta, X;\Gamma, x : A \vdash M : B}{\Delta;\Gamma \vdash \mathtt{unpack}\ (X, x) = V; M : B} \qquad \frac{\Delta;\Gamma \vdash M : B}{\Delta;\Gamma \vdash \mathtt{thunk}\ M : UB} \qquad \frac{\Delta;\Gamma \vdash V : UB}{\Delta;\Gamma \vdash \mathtt{force}\ V : B}$$

$$\frac{\Delta, X;\Gamma \vdash M : B}{\Delta;\Gamma \vdash \Lambda X.M : \forall X.B} \qquad \frac{\Delta;\Gamma \vdash M : \forall X.B \qquad \Delta \vdash A}{\Delta;\Gamma \vdash M[A] : B[A/X]} \qquad \frac{\Delta;\Gamma \vdash V : A}{\Delta;\Gamma \vdash \mathtt{ret}\ V : FA} \qquad \frac{\Delta;\Gamma \vdash M : FA \qquad \Delta;\Gamma \vdash N : B}{\Delta;\Gamma \vdash x \leftarrow M; N : B}$$

Fig. 9. $\mathrm{CBPV}_{\mathrm{OSum}}$ Type System (fragment)

$$\begin{aligned}
S[\mho] &\mapsto^0 \mho \\
\Sigma \triangleright S[\mathtt{newcase}_A\ x; M] &\mapsto^0 \Sigma, \sigma : A \triangleright S[M[\sigma/x]] \\
S[\mathtt{match}\ \mathtt{inj}_\sigma\ V\ \mathtt{with}\ \sigma\{\mathtt{inj}\ x.M \mid N\}] &\mapsto^1 S[M[V/x]] \\
S[\mathtt{match}\ \mathtt{inj}_{\sigma_1}\ V\ \mathtt{with}\ \sigma_2\{\mathtt{inj}\ x.M \mid N\}] &\mapsto^1 S[N] \qquad (\text{where } \sigma_1 \neq \sigma_2) \\
S[\mathtt{force}\ (\mathtt{thunk}\ M)] &\mapsto^0 S[M] \\
S[\mathtt{unpack}\ (X, x) = \mathtt{pack}(A, V); M] &\mapsto^0 S[M[A/X, V/x]] \\
S[(\Lambda X.M)[A]] &\mapsto^0 S[M[A/X]] \\
S[x \leftarrow \mathtt{ret}\ V; N] &\mapsto^0 S[N[V/x]]
\end{aligned}$$

Fig. 10. $\mathrm{CBPV}_{\mathrm{OSum}}$ Operational Semantics (fragment)

functions, quantification and an open sum type, this gives a simple explanation of the semantics of PolyC$^\nu$ in terms of fairly standard language features.

In the left side of Figure 11, we present the type translation from PolyC$^\nu$ to $\mathrm{CBPV}_{\mathrm{OSum}}$. Since PolyC$^\nu$ is a call-by-value language, types are translated to $\mathrm{CBPV}_{\mathrm{OSum}}$ *value* types. Booleans and pairs are translated directly, and the function type is given the standard CBPV translation for call-by-value functions $U(\llbracket A \rrbracket \to F\llbracket B \rrbracket)$: a call-by-value function is a thunked computation of a function that takes an $\llbracket A \rrbracket$ as input and possibly returns a $\llbracket B \rrbracket$ as output. The dynamic type ? is interpreted as the open sum type. The meaning of a type variable depends on the context: if it is an abstract type variable, it is translated to a type variable, but if it is a known type variable $X \cong A$, it is translated to $\llbracket A \rrbracket$! That is, at runtime, values of a known type variable are just values of the type that X is isomorphic to, and as we will see later, sealing and unsealing are no-ops. Similarly, a runtime type tag σ is translated to the type that the corresponding case maps to. These are inductively well-defined because Σ stays constant in the type translation and Γ only adds abstract type variables.

The final two cases are the most revealing. First the fresh universal quantifier, $\forall^\nu X.A$ is translated to a thunk of *not just* a universally quantified computation, but also takes in a Case X as input. The reason is that the body of a Λ will use that Case X in order to interpret casts involving X. This is precisely why parametricity is more complex for our source language: if it were translated to

$$
\begin{aligned}
[\![\Sigma;\Gamma \vdash ?]\!] &= \mathrm{OSum} \\
[\![\Sigma;\Gamma \vdash X]\!] &= X \quad (\text{where } X \in \Gamma) \\
[\![\Sigma;\Gamma \vdash X]\!] &= [\![A]\!] \quad (\text{where } X \cong A \in \Gamma) \\
[\![\Sigma;\Gamma \vdash \sigma]\!] &= [\![A]\!] \quad (\text{where } \sigma : A \in \Sigma) \\
[\![\Sigma;\Gamma \vdash \mathbb{B}]\!] &= \mathbb{B} \\
[\![\Sigma;\Gamma \vdash A \to B]\!] &= U([\![\Sigma;\Gamma \vdash A]\!] \to F[\![\Sigma;\Gamma \vdash B]\!]) \\
[\![\Sigma;\Gamma \vdash A_1 \times A_2]\!] &= [\![\Sigma;\Gamma \vdash A_1]\!] \times [\![\Sigma;\Gamma \vdash A_2]\!] \\
[\![\Sigma;\Gamma \vdash \exists^{\nu} X.A]\!] &= \exists X.U(\mathrm{Case}\ X \to F[\![\Sigma;\Gamma, X \vdash A]\!]) \\
[\![\Sigma;\Gamma \vdash \forall^{\nu} X.A]\!] &= U(\forall X.\mathrm{Case}\ X \to F[\![\Sigma;\Gamma, X \vdash A]\!])
\end{aligned}
$$

$$
\begin{aligned}
[\![\Sigma \vdash \cdot]\!] &= \cdot;\Gamma_p \\
[\![\Sigma \vdash \Gamma, x : A]\!] &= \Delta';\Gamma', x : [\![\Sigma;\Gamma \vdash A]\!] \\
&\quad (\text{where } [\![\Sigma \vdash \Gamma]\!] = \Delta';\Gamma') \\
[\![\Sigma \vdash \Gamma, X]\!] &= \Delta', X;\Gamma', c_X : \mathrm{Case}\ X \\
&\quad (\text{where } [\![\Sigma \vdash \Gamma]\!] = \Delta';\Gamma') \\
[\![\Sigma \vdash \Gamma, X \cong A]\!] &= \Delta';\Gamma', c_X : \mathrm{Case}\ [\![\Sigma;\Gamma \vdash A]\!] \\
&\quad (\text{where } [\![\Sigma \vdash \Gamma]\!] = \Delta';\Gamma')
\end{aligned}
$$

Fig. 11. PolyC$^{\nu}$ type and environment translation

just $U(\forall X.F[\![A]\!])$, then parametricity would follow directly by parametricity for CBPV$_{\mathrm{OSum}}$, but the Case X represents additional information that the function is being passed that potentially provides information about the type X. It is only because code translated from PolyC$^{\nu}$ always generates a fresh case that this extra input is benign. The fresh existential $\exists^{\nu} X.A$ is translated to a real existential of a thunk that expects a Case X and returns a $[\![A]\!]$. Note that while the quantification is the dual of the $\forall^{\nu}$ case, both of them receive a Case X from the environment, which is freshly generated.

Next, while PolyG$^{\nu}$ and PolyC$^{\nu}$ have a single environment Γ that includes type variables and term variables, in CBPV$_{\mathrm{OSum}}$, these are separated into a type variable environment Δ and a term variable environment Γ. For this reason in the right side of Figure11 we define the translation of an environment Γ to be a pair of environments in CBPV$_{\mathrm{OSum}}$. The type variable $x : A$ is just translated to a variable $x : [\![A]\!]$, but the type variables are more interesting. An abstract type variable X is translated to a pair of a type variable X but also an associated term variable $c_X : \mathrm{Case}\ X$, which represents the case of the dynamic type that will be instantiated with a freshly generated case. On the other hand, since known type variables $X \cong A$ are translated to $[\![A]\!]$, we do not extend Δ with a new variable, but still produce a variable $c_X : [\![A]\!]$ as with an unknown type variable. Finally, the empty context $\cdot$ is translated to an empty Δ, but a fixed "preamble" Γ_p, which is defined in Figure 12. The preamble contains cases for each of the ground types G, which we think of as globally allocated cases of OSum that are known to all programs. Note that the semantics of these cases is essentially the same as having a known type variable for each ground type.

To translate a whole program, written $[\![\Sigma;\Gamma \vdash e]\!]_p$, we insert a preamble that generates the cases of the open sum type for each ground type that our bound in Γ_p. This allows us to assume the existence of these cases in the rest of the translation. In Figure 12, we show our whole-program translation which inserts a preamble to generate a case of the OSum type for each ground type. We also define a function case$(\cdot)$ from types to their corresponding case value, which is a case variable for all types except those generated at runtime σ.

Next, we consider the term translation, which is defined with the below type preservation Theorem 5.1 in mind. First, all PolyC$^{\nu}$ terms of type A are translated to CBPV$_{\mathrm{OSum}}$ *computations*, with type $F[\![A]\!]$, which is standard for translating CBV to CBPV. Also, note that the *output* environment of fresh type names in a term is just translated as an extension to the input environment, the difference is irrelevant in the translated code, because the names themselves are actually generated in the translation of the *hide* form.

Theorem 5.1. *If* $\Gamma_1 \vdash M : A, \Gamma_2$ *then* $\Delta;\Gamma \vdash [\![M]\!] : F[\![\Sigma;\Gamma_1, \Gamma_2 \vdash A]\!]$ *where* $[\![\Gamma_1, \Gamma_2]\!] = \Delta;\Gamma$.

We define the term translation in Figure 13. To reduce the use of context clutter in the translations, we elide the contexts Σ, Γ in the definition of the semantics, though they are technically needed

$$
\begin{array}{rcl}
[\![M]\!]_p & = & \mathtt{newcase}_{\mathbb{B}}\ c_{\mathbb{B}}; \\
& & \mathtt{newcase}_{\mathrm{OSum}\to\mathrm{OSum}}\ c_{\to}; \\
& & \mathtt{newcase}_{\mathrm{OSum}\times\mathrm{OSum}}\ c_{\times}; \\
& & \mathtt{newcase}_{\exists^{\nu}X.\mathrm{OSum}}\ c_{\exists^{\nu}}; \\
& & \mathtt{newcase}_{\forall^{\nu}X.\mathrm{OSum}}\ c_{\forall^{\nu}}; \\
& & [\![M]\!]
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{case}(\mathbb{B}) & = & c_{\mathbb{B}} \\
\mathrm{case}(A\to B) & = & c_{\to} \\
\mathrm{case}(A\times B) & = & c_{\times} \\
\mathrm{case}(\exists^{\nu}X.A) & = & c_{\exists^{\nu}} \\
\mathrm{case}(\forall^{\nu}X.A) & = & c_{\forall^{\nu}} \\
\mathrm{case}(X) & = & c_X \\
\mathrm{case}(\sigma) & = & \sigma
\end{array}
$$

$$
\Gamma_p = c_{\mathbb{B}} : \mathrm{Case}\ \mathbb{B}, c_{\to} : \mathrm{Case}\ (\mathrm{OSum}\to\mathrm{OSum}), c_{\times} : \mathrm{Case}\ (\mathrm{OSum}\times\mathrm{OSum}), c_{\exists^{\nu}} : \mathrm{Case}\ (\exists^{\nu}X.\mathrm{OSum}), c_{\forall^{\nu}} : \mathrm{Case}\ (\forall^{\nu}X.\mathrm{OSum})
$$

Fig. 12. Ground type tag management

$$
\begin{array}{rcl}
[\![x]\!] & = & \mathtt{ret}\ x \\
[\![\mathtt{let}\ x = M; N]\!] & = & x \leftarrow [\![M]\!]; [\![N]\!] \\
[\![\mathcal{U}_A]\!] & = & \mathcal{U} \\
[\![\mathtt{seal}_X M]\!] & = & [\![M]\!] \\
[\![\mathtt{unseal}_X M]\!] & = & [\![M]\!] \\
[\![\mathtt{inj}_G\ M]\!] & = & r \leftarrow [\![M]\!]; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ r \\
[\![\mathtt{is}(G)?\ M]\!] & = & r \leftarrow [\![M]\!]; \mathtt{match}\ r\ \mathtt{with}\ \mathrm{case}(G)\{\mathtt{inj}\ y.\mathtt{ret}\ \mathtt{true} \mid \mathtt{ret}\ \mathtt{false}\} \\
[\![\mathtt{hide}\ X\cong A; M]\!] & = & \mathtt{newcase}_{[\![A]\!]}\ c_X; [\![M]\!] \\
[\![\langle A^{\sqsubseteq}\rangle\!\updownarrow\ M]\!] & = & [\![A^{\sqsubseteq}]\!]\!\updownarrow[\![M]\!] \\
[\![\mathtt{pack}^{\nu}(X\cong A, M)]\!] & = & \mathtt{ret}\ \mathtt{pack}(A, \mathtt{thunk}\ (\lambda c_X : \mathrm{Case}\ A.[\![M]\!])) \\
[\![\mathtt{unpack}\ (X,x) = M; N]\!] & = & r \leftarrow [\![M]\!]; \mathtt{unpack}\ (X,f) = r; \mathtt{newcase}_X\ c_X; x \leftarrow (\mathtt{force}\ f)\ c_X; [\![N]\!] \\
[\![\Lambda^{\nu}X.M]\!] & = & \mathtt{ret}\ (\mathtt{thunk}\ (\Lambda X.\lambda(c_X : \mathrm{Case}\ X).[\![M]\!])) \\
[\![M\{X\cong A\}]\!] & = & f \leftarrow [\![M]\!]; (\mathtt{force}\ f)[A]\ c_X \\
[\![\lambda(x:A).M]\!] & = & \mathtt{ret}\ \mathtt{thunk}\ \lambda(x : [\![A]\!]).[\![M]\!] \\
[\![M\ N]\!] & = & f \leftarrow [\![M]\!]; a \leftarrow [\![N]\!]; (\mathtt{force}\ f)\ a
\end{array}
$$

$$
\begin{array}{rcl}
[\![G]\!]\!\updownarrow & = & \bullet \quad (\text{when } G = ?,\ \alpha,\ \text{or } \mathbb{B}) \\
[\![\mathrm{tag}_G(A^{\sqsubseteq})]\!]\!\updownarrow & = & r \leftarrow [\![A^{\sqsubseteq}]\!]\!\updownarrow[\bullet]; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ r \\
[\![\mathrm{tag}_G(A^{\sqsubseteq})]\!]\!\downarrow & = & x \leftarrow \bullet; \mathtt{match}\ x\ \mathtt{with}\ \mathrm{case}(G)\{\mathtt{inj}\ y.[\![A^{\sqsubseteq}]\!]\!\downarrow[\mathtt{ret}\ y] \mid \mathcal{U}\} \\
[\![A_1^{\sqsubseteq}\times A_2^{\sqsubseteq}]\!]\!\updownarrow & = & x \leftarrow \bullet; \mathtt{let}\ (x_1,x_2) = x; x_1' \leftarrow [\![A_1^{\sqsubseteq}]\!]\!\updownarrow[\mathtt{ret}\ x_1]; x_2' \leftarrow [\![A_2^{\sqsubseteq}]\!]\!\updownarrow[\mathtt{ret}\ x_2]; \mathtt{ret}\ (x_1', x_2') \\
[\![A_1^{\sqsubseteq}\to A_2^{\sqsubseteq}]\!]\!\updownarrow & = & x \leftarrow \bullet; \mathtt{ret}\ \mathtt{thunk}\ (\lambda y : [\![r(A_1^{\sqsubseteq})]\!].a \leftarrow [\![A_1^{\sqsubseteq}]\!]\!\updownarrow^{-}[\mathtt{ret}\ y]; [\![A_2^{\sqsubseteq}]\!]\!\updownarrow[\mathtt{force}\ x\ a]) \\
[\![\forall^{\nu}X.A^{\sqsubseteq}]\!]\!\updownarrow & = & x \leftarrow \bullet; \mathtt{ret}\ \mathtt{thunk}\ (\Lambda X.\lambda c_X : \mathrm{Case}\ X.[\![A^{\sqsubseteq}]\!]\!\updownarrow[\mathtt{force}\ x\ [X]\ c_X]) \\
[\![\exists^{\nu}X.A^{\sqsubseteq}]\!]\!\updownarrow & = & x \leftarrow \bullet; \mathtt{unpack}\ (Y,f) = x; \mathtt{ret}\ \mathtt{pack}(Y, \mathtt{thunk}\ (\lambda c_X : \mathrm{Case}\ Y.[\![A^{\sqsubseteq}]\!]\!\updownarrow[(\mathtt{force}\ f)\ c_X]))
\end{array}
$$

Fig. 13. PolyC$^{\nu}$ term translation

for the translation of type annotations, they do not affect the operational semantics and so can be safely ignored. We put the bool, pair, and our pack-cast intermediate form cases in the appendix.

Variables translate to a return of the variable, let is translated bind, and errors are translated to errors. Since the type translation maps known type variables to their bound types, the target language seal and unseal disappear in the translation. Injection into the dynamic type translates to injection into the open sum type and ground type checks in PolyC$^{\nu}$ are implemented using pattern matching on OSum in CBPV$_{\mathrm{OSum}}$. Next, the hide form is translated to a newcase form.

We translate existential packages in the cast calculus to CBPV$_{\mathrm{OSum}}$ packages containing functions from a case of the open sum type to the body of the package. In PolyC$^{\nu}$ we delay execution of pack bodies, so the translation inserts a thunk to make the order of execution explicit. Since pack bodies translate to functions, the translation of an unpack must provide a case of the open sum type to the package it unwraps. Type abstractions (Λ^{ν}), like packs, wrap their bodies in functions that, on instantiation, expect a case of the open sum type matching the instantiating type. Since hide generates the requisite type name, it translates to a newcase. A type application then simply

plugs its given type and the tag associated with its type variable into the supplied type abstraction. Functions follow the usual CBV translation into CBPV: a λ is translated to a thunk of a λ, and the application translation makes the evaluation order explicit and forces the function with an input.

Next, we define the implementation of casts as "contracts", i.e., ordinary functions in the $\text{CBPV}_{\text{OSum}}$. Reflexive casts at atomic types, ?, α, and $\mathbb{B}$, translate away. Structural casts at composite types, pair types, function types, universals, and existentials, push casts for their sub-parts into terms of each type. Function and product casts are entirely standard. Universal casts delay until type application and then cast the output. Existential casts push their subcasts into whatever package they are given.

5.4 Simulation

In §6, we establish graduality and parametricity theorems for PolyG^{ν}/PolyC^{ν} by analysis of the semantics of terms translated into $\text{CBPV}_{\text{OSum}}$. But since we take the operational semantics of PolyC^{ν} as definitional, we need to bridge the gap between the operational semantics in $\text{CBPV}_{\text{OSum}}$ and PolyC^{ν} by proving the following adequacy theorem that says that the final behavior of terms in PolyC^{ν} is the same as the behavior of their translations:

Theorem 5.2 (Adequacy). *If* $\cdot \vdash M : A; \cdot$, *then* $M \Uparrow$ *if and only if* $[\![M]\!]_p \Uparrow$ *and* $M \Downarrow V$ *if and only if* $[\![M]\!]_p \Downarrow V'$ *and* $M \Downarrow \mho$ *if and only if* $[\![M]\!]_p \Downarrow \mho$.

The proof of the theorem follows by a forward *simulation* argument given in the appendix, adapting a similar CBPV simulation given by Levy [13].

6 GRADUALITY AND PARAMETRICITY

In this section we prove the central metatheoretic results of the paper: that our surface language, satisfies both graduality and parametricity theorems. Each of these is considered a technical challenge to prove: parametricity is typically proven by a logical relation and graduality is proven either by a simulation argument [19] or a logical relation [15, 16], so in the worst case this would require two highly technical developments. However, we show that this is not necessary: the logical relations proof for graduality is general enough that the parametricity theorem is a corollary of the *reflexivity* of the logical relation. This substantiates the analogy between parametricity and graduality originally proposed in [15].

The key to sharing this work is that we give a novel *relational* interpretation of type precision derivations. That is, our logical relation is indexed not by types, but by type precision derivations. For any derivation $A^{\sqsubseteq} : A_l \sqsubseteq A_r$, we define a relation $\mathcal{V}[\![A^{\sqsubseteq}]\!]$ between values of A_l and A_r. By taking the reflexivity case $A : A \sqsubseteq A$, we recover the parametricity logical relation. Previous logical relations proofs of graduality [15, 16] defined a logical relation relation indexed by types, and used casts to define a second relation based on type precision judgments, but the direct relational approach simplifies the proofs and immediately applies to parametricity as well.

6.1 Term Precision

To state the graduality theorem, we begin by formalizing the syntactic *term* precision relation. The intuition behind a precision relation $M \sqsubseteq M'$ is that M' is a (somewhat) dynamically typed term and we have changed some of its type annotations to be more precise, producing M. This is one of the main intended use cases for a gradual language: hardening the types of programs over time. Restated in a less directed way, a term M is (syntactically) more precise than M' when the types and annotations in M are more precise than M' and otherwise the terms have the same structure. We formalize this as a judgment $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$, where $\Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$ is a type precision derivation and $\Gamma^{\sqsubseteq} : \Gamma_l \sqsubseteq \Gamma_r$ and $\Gamma^{\sqsubseteq\prime} : \Gamma'_l \sqsubseteq \Gamma'_r$ are precision *contexts* and $\Gamma_l \vdash M_l : A_l; \Gamma'_l$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vdash (M_l :: B_l) \sqsubseteq (M_r :: B_r) : B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq} \qquad X \cong B^{\sqsubseteq} \in \Gamma^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \mathtt{seal}_X M_l \sqsubseteq \mathtt{seal}_X M_r : X}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq} \qquad X \cong B^{\sqsubseteq} \in \Gamma^{\sqsubseteq}, \Gamma_o^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \mathtt{unseal}_X M_l \sqsubseteq \mathtt{unseal}_X M_r : B^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}} \qquad \frac{\Gamma^{\sqsubseteq}, X \cong B^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq} \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vdash \mathsf{pack}^{\nu}(X \cong B_l, M_l) \sqsubseteq \mathsf{pack}^{\nu}(X \cong B_r, M_r) : \exists^{\nu} X.A^{\sqsubseteq}; \cdot}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq}, X, x : \mathrm{un}\exists^{\nu}(A^{\sqsubseteq}) \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \mathsf{unpack}\ (X, x) = M_l; N_l \sqsubseteq \mathsf{unpack}\ (X, x) = M_r; N_r : B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}|_X}$$

$$\frac{\Gamma^{\sqsubseteq}, X \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \Lambda^{\nu} X.M_l \sqsubseteq \Lambda^{\nu} X.M_r : \forall^{\nu} X.A^{\sqsubseteq}; \cdot} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq} \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vdash M_l\{X \cong B_l\} \sqsubseteq M_r\{X \cong B_r\} : \mathrm{un}\forall^{\nu}(A^{\sqsubseteq}); \Gamma_M^{\sqsubseteq}, X \cong B^{\sqsubseteq}}$$

Fig. 14. PolyG$^{\nu}$ Term Precision (fragment)

and $\Gamma_r \vdash M_r : A_r; \Gamma_r'$. A precision context $\Gamma^{\sqsubseteq}$ is like a precision derivation between two contexts: everywhere a type would be in an ordinary context, a precision derivation is used instead.

We show a representative sample of the term precision for the surface language in Figure 14, the full rules are given in the appendix. The rules are all completely structural: just check that the two terms have the same term constructor and all of the corresponding arguments of the rule are $\sqsubseteq$. The metafunctions dom, cod, un$\forall^{\nu}$, un$\exists^{\nu}$ are extended in the obvious way to work on precision derivations. Next, we define a similar notion of term precision for PolyC$^{\nu}$. We show a representative fragment in Figure 15, the full definition is in the appendix. The main difference is that we now include four rules involving casts: two for downcasts and two for upcasts. We can summarize all four by saying that if $M_l \sqsubseteq M_r$, then adding a cast to either M_l or M_r still maintains the left side is more precise than the right, as long as the type on the left is more precise than the right. Semantically, these are the most important rules of term precision, as they are the bridge between the worlds of type and term precision.

Then the key lemma is that the elaboration process from PolyG$^{\nu}$ to PolyC$^{\nu}$ preserves term precision. The proof is a straightforward by induction on term precision proofs, and is presented in the appendix.

Lemma 6.1. *If* $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$ *in the surface language, then* $\Gamma^{\sqsubseteq} \vdash M_l^+ \sqsubseteq M_r^+ : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$

6.2 Graduality Theorem

The graduality theorem states that if a term M is *syntactically* more precise than a term M', then M *semantically* refines the behavior of M': it may error, but otherwise it has the same behavior as M': if it diverges so does M' and if it terminates at V, M' terminates with some V' as well. If we think of M as the result of hardening the types of M', then this shows that hardening types semantically only increases the burden of runtime type checking and doesn't otherwise interfere with program behavior. We call this *operational* graduality, as we will consider some related notions later.

Theorem 6.2 (Operational Graduality). *If* $\cdot \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \cdot$, *then either* $M_l^+ \Downarrow \mathfrak{U}$ *or both terms diverge* $M_l^+, M_r^+ \Uparrow$ *or both terms terminate successfully* $M_l^+ \Downarrow V_l$ *and* $M_r^+ \Downarrow V_r$.

6.3 Logical Relation

The basic idea of the logical relations proof to proving graduality is to interpret a term precision judgment $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}$ in a *relational* manner. That is, to every type precision derivation $A^{\sqsubseteq} : A_l \sqsubseteq A_r$, we associate a relation $\mathcal{V}[\![A^{\sqsubseteq}]\!]$ between closed values of types A_l and A_r. Then we define a semantic version of the term precision judgment $\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}$ which says that

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \quad \Gamma, \Gamma' \vdash AB^{\sqsubseteq} : A \sqsubseteq B \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vdash \langle AB^{\sqsubseteq}\rangle\uparrow M_l \sqsubseteq M_r : BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \quad \Gamma, \Gamma' \vdash AB^{\sqsubseteq} : A \sqsubseteq B \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vdash \langle AB^{\sqsubseteq}\rangle\downarrow M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \quad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\uparrow M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \quad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\downarrow M_r : AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong A^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime\prime} \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma^{\sqsubseteq} \vdash \mathsf{hide}\ X \cong A_l; M_l \sqsubseteq \mathsf{hide}\ X \cong A_r; M_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, \Gamma^{\sqsubseteq\prime\prime}}$$

$$\frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathsf{seal}_X M_l \sqsubseteq \mathsf{seal}_X M_r : X; \Gamma^{\sqsubseteq\prime}} \qquad \frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : X; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathsf{unseal}_X M_l \sqsubseteq \mathsf{unseal}_X M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

$$\frac{\Gamma^{\sqsubseteq}, X \cong B^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \cdot}{\Gamma^{\sqsubseteq} \vdash \mathsf{pack}^{\nu}(X \cong B_l, M_l) \sqsubseteq \mathsf{pack}^{\nu}(X \cong B_r, M_r) : \exists^{\nu} X.A^{\sqsubseteq}; \cdot} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : \forall^{\nu} X.A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \quad \Gamma^{\sqsubseteq} \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vdash M_l\{X \cong B_l\} \sqsubseteq M_r\{X \cong B_r\} : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong B^{\sqsubseteq}}$$

$$\frac{\Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq}_M \vdash \Gamma^{\sqsubseteq}_N \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq}_M, \Gamma^{\sqsubseteq}_N \vdash B^{\sqsubseteq} \quad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : \exists^{\nu} X.A^{\sqsubseteq}; \Gamma^{\sqsubseteq}_M \quad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq}_M, X, x : A^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq}_N}{\Gamma^{\sqsubseteq} \vdash \mathsf{unpack}\ (X, x) = M_l; N_l \sqsubseteq \mathsf{unpack}\ (X, x) = M_r; N_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq}_M, \Gamma^{\sqsubseteq}_N} \qquad \frac{\Gamma^{\sqsubseteq}, X \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \cdot}{\Gamma^{\sqsubseteq} \vdash \Lambda^{\nu} X.M_l \sqsubseteq \Lambda^{\nu} X.M_r : \forall^{\nu} X.A^{\sqsubseteq}; \cdot}$$

Fig. 15. PolyC$^{\nu}$ Term Precision (fragment)

given inputs satisfying the relations in $\Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq}_o$, then either M_l will error, both sides diverge, or M_l and M_r will terminate with values in the relation $\mathcal{V}[\![A^{\sqsubseteq}]\!]$. We define this relation over $\mathrm{CBPV}_{\mathrm{OSum}}$ translations of PolyC$^{\nu}$ terms, rather than PolyC$^{\nu}$ terms because the operational semantics is simpler.

More precisely, we use the now well established technique of Kripke, step-indexed logical relations [1]. Because the language includes allocation of fresh type names at runtime, the set of values that should be in the relation grows as the store increases. This is modeled *Kripke* structure, which indexes the relation by a "possible world" that attaches invariants to the allocated cases. Because our language includes diverging programs (due to the open sum type), we need to use a *step-indexed* relation that decrements when pattern matching on OSum, and "times out" when the step index hits 0. Finally, following [15, 16], to model graduality we need to associate two relations to each type precision derivation: one which times out when the left hand hand term runs out of steps, but allows the right hand side to take any number of steps and vice-versa one that times out when the right runs out of steps.

Figure 16 includes some preliminary definitions we need for the logical relation. First, $\mathrm{Atom}_n[A_l, A_r]$ and $\mathrm{CAtom}_n[A_l, A_r]$ define the world-term-term triples that the relations are defined over. A relation $R \in \mathrm{Rel}_n[A_l, A_r]$ at stage n consists of triples of a world, and a value of type A_l and a value of type A_r (ignore the n for now) such that it is monotone in the world. The world $w \in \mathrm{World}_n$ contains the number of steps remaining $w.j$, the current state of each side $w.\Sigma_l$, $w.\Sigma_r$, and finally an interpretation of how the cases in the two stores are related $w.\eta$. An interpretation $\eta \in \mathrm{Interp}_n[\Sigma_l, \Sigma_r]$ consists of a cardinality $\eta.size$ which says how many cases are related and a function $\eta.f$ which says *which* cases are related, i.e., for each $i \in \eta.size$ it gives a pair of cases, one valid in the left hand store and one in the right. Finally, $\eta.\rho$ gives a relation between the types of these two cases. The final side-condition says this association is a *partial bijection*: a case on one side is associated to at most one case on the other side. Staging the relations and worlds is necessary due to a circularity here: a relation is (contravariantly) dependent on the current world, which contains relations. A

$$
\begin{array}{rcl}
\mathrm{Atom}_n[A_l, A_r] & = & \{(w, V_l, V_r) \mid w \in \mathrm{World}_n \wedge (w.\Sigma_l \mid \cdot \vdash V_l : A_l) \wedge (w.\Sigma_r \mid \cdot \vdash V_r : A_r)\} \\
\mathrm{CAtom}_n[A_l, A_r] & = & \{(w, V_l, V_r) \mid w \in \mathrm{World}_n \wedge w.\Sigma_l \mid \cdot \vdash M_l : FA_l \wedge w.\Sigma_r \mid \cdot \vdash M_r : FA_r\} \\
\mathrm{Rel}_n[A_l, A_r] & = & \{R \subseteq \mathrm{Atom}_n[A_l, A_r] \mid \forall (w, V_l, V_r) \in R, w' \sqsupseteq w.(w', V_l, V_r) \in R\} \\
\mathrm{World}_n & = & \{(j, \Sigma_l, \Sigma_r, \eta) \mid j < n \wedge \eta \in \mathrm{Interp}_j[\Sigma_l, \Sigma_r]\} \\
\mathrm{Interp}_n[\Sigma_l, \Sigma_r] & = & \{(size, f, \rho) \mid size \in \mathbb{N} \wedge f \in [size] \to ([\Sigma_l.size] \times [\Sigma_2.size]) \\
 & & \quad \wedge \rho : (i < size) \to Rel_n[\Sigma_l(f(i)); \Sigma_r(f(i))] \\
 & & \quad \wedge \forall i < j < size. f(i)_l \neq f(j)_l \wedge f(i)_r \neq f(j)_r\} \\
w' \sqsupset w. & = & (w' \sqsupseteq w) \wedge w'.j > w.j \\
w' \sqsupseteq w & = & w'.j \leq w.j \wedge w'.\Sigma_l \sqsupseteq w.\Sigma_l \wedge w'.\Sigma_r \sqsupseteq w.\Sigma_r \wedge w'.\eta \sqsupseteq \lfloor w.\eta \rfloor_{w'.j} \\
\Sigma' \sqsupseteq \Sigma & = & \Sigma'.size \geq \Sigma.size \wedge \forall i < \Sigma.size.\ \Sigma'(i) = \Sigma(i) \\
\eta' \sqsupseteq \eta & = & \eta'.size \geq \eta \wedge \forall i < \eta.size.\ \eta'.f(i) = \eta.f(i) \wedge \eta'.\rho(i) = \eta.\rho(i) \\
\triangleright R & = & \{(w, V_l, V_r) \mid \forall w' \sqsupset w.(w', V_l, V_r) \in R\} \\
\mathrm{Rel}_\omega[A_l, A_r] & = & \{R \subseteq \bigcup_{n \in \mathbb{N}} \mathrm{Atom}_n[A_l, A_r] \mid \forall n \in \mathbb{N}.\ \lfloor R \rfloor_n \in \mathrm{Rel}_n[A_l, A_r]\} \\
\lfloor \eta \rfloor_n & = & (\eta.size, \eta.f, \lambda i. \lfloor \rho(i) \rfloor_n) \\
\lfloor R \rfloor_n & = & \{(w, V_l, V_r) \mid w.j < n \wedge (w, V_l, V_r) \in R\} \\
\eta \vDash (\sigma_l, \sigma_r, R) & = & \exists i < \eta.size.\ \eta.f(i) = (\sigma_l, \sigma_r) \wedge \eta.\rho(i) = R\}
\end{array}
$$

Fig. 16. Logical Relation Auxiliary Definitions

relation in Rel_n is indexed by a World_n, but a World_n contains relations in $\mathrm{Rel}_{w.j}$, and $w.j < n$. In particular, $\mathrm{World}_0 = \emptyset$, so the definition is well-founded.

The next portion of the figure contains the definition of *world* extension $w' \sqsupseteq w$[2], representing the idea that w' is a possible "future" of w: the step index j is smaller and the states of the two sides have increased the number of allocated cases, but the old invariants are still there. We define strict extension $w' \sqsupset w$ to mean extension where the step has gotten strictly smaller. This allows us to define the *later* relation $\triangleright R$ which is used to break circular definitions in the logical relation. Next, we define our notion of non-indexed relation Rel_ω, which is what we quantify over in the interpretation of $\forall^\nu, \exists^\nu$. Then we define the restriction of interpretations and relations to a stage n. An infinitary relation can be "clamped" to any stage n using $\lfloor R \rfloor_n$. Finally, we define when two cases are related in an interpretation as $\eta \vDash (\sigma_l, \sigma_r, R)$.

Next, The top of Figure 17 contains the definition of the logical relation on values and computations. First, we write $\sim$ as a metavariable that ranges over two symbols: $\prec$ which indicates that we are counting steps on the left side, and $\succ$ which indicates we are counting steps on the right side. We then define the value relation $\mathcal{V}_n^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta \in \mathrm{Rel}_n[\delta_l(A_l), \delta_r(A_r)]$. Here γ maps the free term variables to pairs of values and δ maps free type variables to triples of two types and a relation between them. First, the definition for type variables looks up the relation in the relational substitution δ. Next, two values in ? are related when they are both injections into OSum, and the "payloads" of the injections are *later* related in the relation R which the world associates to the corresponding cases. The $\triangleright$ is used because we count pattern matching on OSum as a step. This also crucially lines up with the fact that pattern matching on the open sum type is the only reduction that consumes a step in our operational semantics. Note that this is a generalization of the logical relation definition for a recursive sum type, where each injection corresponds to a case of the sum. Here since the sum type is open, we must look in the world to see what cases are allocated. Next, the $\mathrm{tag}_G(A^{\sqsubseteq})$ case relates values on the left at some type A_l and values on the right of type ?. The definition states that the dynamically typed value must be an injection using the tag given by G, and that the payload of that injection must be related to V_l with the relation given by $A^{\sqsubseteq}$. This case splits into two because we are pattern matching on a value of the open sum type, and so in the $\succ$

[2]there is a clash of notation between precision $\sqsubseteq$ and world extension $\sqsupseteq$ but it should be clear which is meant at any time.

case we must decrement because we are consuming a step on the right, whereas in the $\prec$ case we do not decrement because we are only counting steps on the left.

The boolean, product and function cases are all standard for Kripke logical relations. The universal and existential cases are the most interesting. The $\forall^{\nu} X.B^{\sqsubseteq}$ case says that two values are related when in any future world, and any relation $R \in \mathrm{Rel}_{\omega}[A_l, A_r]$, and any pair of cases σ_l, σ_r that have $\lfloor R \rfloor_{w'.j}$ as their associated relation, if the values are instantiated with A_l, σ_l and A_r, σ_r respectively, then they behave as related computations. The intuition is that values of type $\forall^{\nu} X.B$ are parameterized by a type A and a tag for that type σ, but the relational interpretation of the type and tag must be the *same*. This is the key to proving the $\mathtt{seal}_X$ and $\mathtt{unseal}_X$ cases of graduality. The fresh existential is dual in the choice of relation, but the same in its treatment of the case σ.

Next, we define the relation on expressions. The two expression relations, $\mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]$ and $\mathcal{E}^{\succ}[\![A^{\sqsubseteq}]\!]$ capture the semantic idea behind graduality: either the left expression raises an error, or the two programs have the same behavior: diverge or return related values in $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$. However, to account for step-indexing, each is an *approximation* to this notion where $\mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]$ times out if the left side consumes all of the available steps $w.j$, and $\mathcal{E}^{\succ}[\![A^{\sqsubseteq}]\!]$ times out if the right side consumes all of the available steps. We define the *infinitary* version of the relations $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$ and $\mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]$ a the union of all of the level n approximations.

Next, we give the relational interpretation of environments. The interpretation of the empty environment are empty substitutions with a valid world w. Extending with a value variable $x : A^{\sqsubseteq}$ means extending γ with a pair of values related by $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$. For an abstract type variable X, first δ is extended with a pair of types and a relation between them. Then, γ must also be extended with a pair of *cases* encoding how these types are injected into the dynamic type. Crucially, just as with the $\forall^{\nu}, \exists^{\nu}$ value relations, these cases must be associated by w to the $w.j$ approximation of the *same* relation with which we extend δ. The interpretation of the *known* type variables $X \cong A^{\sqsubseteq}$ has the same basic structure, the key difference is that rather than using an arbitrary, δ is extended with the value relation $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$.

With all of that preparation finished, we finally define the semantic interpretation of the graduality judgment $\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$ in the bottom of Figure 17. First, it says that both $M_l \sqsubseteq\prec M_r$ and $M_l \sqsubseteq\succ M_r$ hold, where we define $\sqsubseteq\sim$ to mean that for any valid instantiation of the environments, we get related computations. We can then define the "logical" Graduality theorem, that syntactic term precision implies semantic term precision, briefly, $\vdash$ implies $\vDash$.

Theorem 6.3 (Logical Graduality). *If* $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$, *then* $\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$

The proof is by induction on the term precision derivation. Each case is proven as a separate lemma in the appendix. The cases of $\forall^{\nu}, \exists^{\nu}$, sealing and unsealing follow because the treatement of type variables between the value and environment relations is the same. In the hide case, the world is extended with $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$ as the relation between new cases. The cast cases are the most involved, following by two lemmas proven by induction over precision derivations: one for when the cast is on the left, and the other when the cast is on the right.

Finally, we prove the operational graduality theorem as a corollary of the logical graduality Theorem 6.3 and the adequacy Theorem 5.2. By constructing a suitable starting world w_{pre} that allocates the globally known tags, we ensure the operational graduality property holds for the code translated to $\mathrm{CBPV}_{\mathrm{OSum}}$, and then the simulation theorem implies the analogous property holds for the PolyC^{ν} operational semantics.

7 PARAMETRICITY AND FREE THEOREMS

Our relational approach to proving the graduality theorem is not only elegant, it also makes the theorem more general, and in particular it *subsumes* the parametricity theorem that we want for

$$
\begin{array}{rcl}
\mathcal{V}_n^{\sim}[\![X]\!]\gamma\delta & = & \lfloor \delta(X) \rfloor_n \\
\mathcal{V}_n^{\sim}[\![?]\!]\gamma\delta & = & \{(w, \mathtt{inj}_{\sigma_l}\ V_l, \mathtt{inj}_{\sigma_r}\ V_r) \in \mathrm{Atom}_n[?]\delta \mid w.\eta \vDash (\sigma_l, \sigma_r, R) \wedge (w, V_l, V_r) \in \triangleright R\} \\
\mathcal{V}_n^{\prec}[\![\mathrm{tag}_G(A^{\sqsubseteq})]\!]\gamma\delta & = & \{(w, V_l, \mathtt{inj}_{\gamma_r([\![G]\!]_{tag})}\ V_r) \in \mathrm{Atom}_n[\mathrm{tag}_G(A^{\sqsubseteq})]\delta \mid (w, V_l, V_r) \in \mathcal{V}_n^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta\} \\
\mathcal{V}_n^{\succ}[\![\mathrm{tag}_G(A^{\sqsubseteq})]\!]\gamma\delta & = & \{(w, V_l, \mathtt{inj}_{\gamma_r([\![G]\!]_{tag})}\ V_r) \in \mathrm{Atom}_n[\mathrm{tag}_G(A^{\sqsubseteq})]\delta \mid (w, V_l, V_r) \in (\triangleright \mathcal{V}^{\succ}[\![A^{\sqsubseteq}]\!]\gamma\delta)\} \\
\mathcal{V}_n^{\sim}[\![\mathbb{B}]\!]\gamma\delta & = & \{(w, \mathtt{true}, \mathtt{true}) \in \mathrm{Atom}_n[\mathbb{B}]\delta\} \cup \{(w, \mathtt{false}, \mathtt{false}) \in \mathrm{Atom}_n[\mathbb{B}]\delta\} \\
\mathcal{V}_n^{\sim}[\![A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}]\!]\gamma\delta & = & \{((V_{l1}, V_{l2}), (V_{r1}, V_{r2})) \in \mathrm{Atom}_n[A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}]\delta \\
 & & \mid (w, V_{l1}, V_{r1}) \in \mathcal{V}_n^{\sim}[\![A_1^{\sqsubseteq}]\!]\gamma\delta \wedge (w, V_{l2}, V_{r2}) \in \mathcal{V}_n^{\sim}[\![A_2^{\sqsubseteq}]\!]\gamma\delta\} \\
\mathcal{V}_n^{\sim}[\![A^{\sqsubseteq} \to B^{\sqsubseteq}]\!]\gamma\delta & = & \{(w, V_l, V_r) \in \mathrm{Atom}_n[A^{\sqsubseteq} \to B^{\sqsubseteq}]\delta \mid \forall w' \sqsupseteq w.(w', V_l', V_r') \in \mathcal{V}_n^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta. \\
 & & \quad (w', \mathtt{force}\ V_l\ V_l', \mathtt{force}\ V_r\ V_r') \in \mathcal{E}_n^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta\} \\
\mathcal{V}_n^{\sim}[\![\forall^{\nu} X.B^{\sqsubseteq}]\!]\gamma\delta & = & \{(w, V_l, V_r) \in \mathrm{Atom}_n[\forall^{\nu} X.A^{\sqsubseteq}]\delta \mid \\
 & & \quad \forall R \in \mathrm{Rel}_{\omega}[A_l, A_r].\forall w' \sqsupseteq w.\forall \sigma_l, \sigma_r.w'.\eta \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j}) \implies \\
 & & \quad (w', \mathtt{force}\ V_l\ [A_l]\ \sigma_l, \mathtt{force}\ V_r\ [A_r]\ \sigma_r) \in \mathcal{E}_n^{\sim}[\![B^{\sqsubseteq}]\!]\gamma'\delta' \\
 & & \quad (\text{where } \gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r), \text{ and } \delta' = \delta, X \mapsto (A_l, A_r, R))\} \\
\mathcal{V}_n^{\sim}[\![\exists^{\nu} X.B^{\sqsubseteq}]\!]\gamma\delta & = & \{(w, \mathtt{pack}\ (A_l, V_l), \mathtt{pack}\ (A_r, V_r)) \in \mathrm{Atom}_n[\exists^{\nu} X.B^{\sqsubseteq}]\delta \mid \\
 & & \quad \exists R \in \mathrm{Rel}_{\omega}[A_l, A_r].\forall w' \sqsupseteq w.\forall \sigma_l, \sigma_r.w'.\eta \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j}) \implies \\
 & & \quad (\mathtt{force}\ V_l\ \sigma_l, \mathtt{force}\ V_r\ \sigma_r) \in \mathcal{E}_n^{\sim}[\![B^{\sqsubseteq}]\!]\gamma'\delta' \\
 & & \quad (\text{where } \gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r), \text{ and } \delta' = \delta, X \mapsto (A_l, A_r, R))\} \\
\mathcal{E}_n^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta & = & \{(w, M_l, M_r) \in \mathrm{CAtom}_n[A^{\sqsubseteq}]\delta \mid (w.\Sigma_l, M_l) \mapsto^{w.j} \vee((w.\Sigma_l, M_l) \mapsto^{<w.j} (\Sigma_l', \mathfrak{V})) \\
 & & \vee(\exists w' \sqsupseteq w.(w', V_l, V_r) \in \mathcal{V}_n^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta. \\
 & & \quad (w.\Sigma_l, M_l) \mapsto^{w'.j-w.j} (w'.\Sigma_l, \mathtt{ret}\ V_l) \wedge (w.\Sigma_r, M_r) \mapsto^* (w'.\Sigma_r, \mathtt{ret}\ V_r))\} \\
\mathcal{E}_n^{\succ}[\![A^{\sqsubseteq}]\!]\gamma\delta & = & \{(w, M_l, M_r) \in \mathrm{CAtom}_n[A^{\sqsubseteq}]\delta \mid (w.\Sigma_r, M_r) \mapsto^{w.j} \vee((w.\Sigma_l, M_l) \mapsto^* (\Sigma_l', \mathfrak{V})) \\
 & & \vee\exists w' \sqsupseteq w.(w', V_l, V_r) \in \mathcal{V}_n^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta. \\
 & & \quad (w.\Sigma_l, M_l) \mapsto^* (w'.\Sigma_l, \mathtt{ret}\ V_l) \wedge (w.\Sigma_r, M_r) \mapsto^{w'.j-w.j} (w'.\Sigma_r, \mathtt{ret}\ V_r)\}
\end{array}
$$

$$
\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta = \bigcup_{n \in \mathbb{N}} \mathcal{V}_n^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta \qquad\qquad \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta = \bigcup_{n \in \mathbb{N}} \mathcal{E}_n^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta
$$

$$
\begin{array}{rcl}
\mathcal{G}^{\sim}[\![\cdot]\!] & = & \{(w, \emptyset, \emptyset) \mid \exists n.w \in \mathrm{World}_n\} \\
\mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}, x : A^{\sqsubseteq}]\!] & = & \{(w, (\gamma, x \mapsto (V_l, V_r)), \delta) \mid (w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!] \wedge (w, V_l, V_r) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta\} \\
\mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}, X]\!] & = & \{(w, (\gamma, c_X \mapsto (\sigma_l, \sigma_r)), \delta, X \mapsto (A_l, A_r, R)) \mid (w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!] \\
 & & \quad \wedge R \in \mathrm{Rel}_{\omega}[A_l, A_r] \wedge (\sigma_l, \sigma_r, \lfloor R \rfloor_{w.j}) \in w\} \\
\mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}, X \cong A^{\sqsubseteq}]\!] & = & \{(w, (\gamma, c_X \mapsto (\sigma_l, \sigma_r)), \delta, X \mapsto (A_l, A_r, \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta)) \mid (w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!] \\
 & & \quad w \vDash (\sigma_l, \sigma_r, \mathcal{V}_{w.j}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta)\}
\end{array}
$$

$$
\begin{array}{rcl}
\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} & = & \Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq_{\prec} M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \wedge \Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq_{\succ} M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \\
\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq_{\sim} M_r \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} & = & \forall (w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_p, \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime}]\!]. \\
 & & \quad (w, [\![M_l]\!][\gamma_l][\delta_l], [\![M_r]\!][\gamma_l][\delta_l]) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta
\end{array}
$$

Fig. 17. Graduality/Parametricity Logical Relation

the language, because we already assign arbitrary relations to abstract type variables. Then the parametricity theorem is just the reflexivity case of the graduality theorem.

Theorem 7.1 (Parametricity). *If* $\Gamma \vdash M : A; \Gamma'$, *then* $\Gamma \vDash M^+ \sqsubseteq M^+ : A; \Gamma'$.

To demonstrate that this really is a parametricity theorem, we show that from this theorem we can prove "free theorems" that are true in polymorphic languages. These free theorems are naturally stated in terms of *contextual equivalence*, the gold standard for operational equivalence of programs, which we define as both programs diverging, erroring, or terminating successfully when plugged into an arbitrary context. The appendix contains a formal definition.

To use our logical relation to prove contextual equivalence, we need the following lemma, which says that semantic term precision both ways is *sound* for PolyG$^{\nu}$ contextual equivalence.

$$\frac{M : \forall^{\nu} X.X \to X \qquad V_A : A \qquad V_B : B}{\lambda\_ : ?.\mathtt{unseal}_X(M_f\{X \cong A\}(\mathtt{seal}_X V_A)) \approx^{\mathrm{ctx}} \lambda\_ : ?.\mathtt{let}\ y = M\{X \cong B\}V_B; V_A}$$

$$\frac{M : \forall^{\nu} X.\forall^{\nu} Y.(X \times Y) \to (Y \times X) \qquad V_A : A \qquad V_B : B}{\begin{array}{l}\lambda\_ : ?.\mathtt{let}\ (y, x) = (M\{X \cong A\}\{Y \cong B\}(\mathtt{seal}_X V_A, \mathtt{seal}_Y V_B)); (\mathtt{unseal}_X x, \mathtt{unseal}_Y y) \\ \approx^{\mathrm{ctx}} \lambda\_ : ?.\mathtt{let}\ (y, x) = (M\{X \cong B\}\{Y \cong A\}(\mathtt{seal}_X V_B, \mathtt{seal}_Y V_A)); (\mathtt{unseal}_Y y, \mathtt{unseal}_X x)\end{array}}$$

$$\begin{array}{c}\mathtt{pack}^{\nu}(X \cong \mathbb{B}, (\mathtt{seal}_X \mathtt{true}, (\mathrm{NOT}, \lambda x : X.\mathtt{unseal}_X x))) \\ \approx^{\mathrm{ctx}} \mathtt{pack}^{\nu}(X \cong \mathbb{B}, (\mathtt{seal}_X \mathtt{true}, (\mathrm{NOT}, \lambda x : X.\mathrm{NOT}(\mathtt{unseal}_X x))))\end{array}$$

Fig. 18. Free Theorems

Lemma 7.2. *If* $\Gamma \vDash M_l \sqsubseteq M_2 \in A; \Gamma_M$ *and* $\Gamma \vDash M_2 \sqsubseteq M_1 \in A; \Gamma_M$, *then* $\Gamma \vDash M_l \approx^{ctx} M_2 \in A; \Gamma_M$.

We now substantiate that this is a parametricity theorem by proving a few contextual equivalence results, presented in Figure 18. The first equivalence shows that the behavior of any term with the "identity function type" $\forall^{\nu} X.X \to X$ must be independent of the input it is given. We place a λ on each side to delimit the scope of the X outward. Without the X (or a similar thunking feature like $\forall^{\nu}$ or $\exists^{\nu}$), the two programs would not have the same (effect) typing. In a more realistic language, this corresponds to wrapping each side in a module boundary. The next result shows that a function $\forall^{\nu} X.\forall^{\nu} Y.(X \times Y) \to (Y \times X)$, if it terminates, must flip the values of the pair, and furthermore whether it terminates, diverges or errors does not depend on the input values. Finally, we give an example using existential types. That shows that an abstract "flipper" which uses `true` for on and `false` for off in its internal state is equivalent to one using `false` and `true`, respectively as long as they return the same value in their "readout" function.

8 DISCUSSION AND RELATED WORK

Our use of abstract and known type variables was directly inspired by Neis et al. [14], which presents a language with a fresh type creation mechanism which they show enables parametric reasoning though the language overall does not satisfy a traditional parametricity theorem. This suggests an alternative language design, where $\forall$ and $\exists$ behave normally, and we directly add a `newtype` facility, analogous to that feature of Haskell, where a type created with newtype has a new case of the open sum type allocated for it. Such a language would not have known type variables or the inside-out scoping in $\forall^{\nu}$, but would also not be parametric by default. Instead, programmers would be able to create fresh types and know that they are abstract to other modules.

Siek et al. [19] claim that graduality demands that tag-checking functions like our `is(`$\mathbb{B}$`)?` form must error when applied to sealed values, and used this as a criticism of the design of Typed Racket. However, in our language, `is(`$\mathbb{B}$`)?` will simply return `false`, which matches Typed Racket's behavior. This is desirable behavior from a backwards-compatibility standpoint, because typically a runtime type check should be a *safe* operation in a dynamically typed language. Our language design makes the sealing explicit and so the graduality issue is avoided, which suggests our design may be more desirable as a language design than traditional polymorphism.

Our use of explicit sealing eliminates much of the complexity of prior logical relations [3, 24]. To accommodate dynamic conversion and evidence insertion, those relations adopted complex value relations for universal types that in turn restricted the ways in which they could treat type variables. By freeing ourselves of the need to deal with λB-style conversions, our logical relation supports existential types in addition to universals. Prior relations [3, 24] could not express the behavior of existentials due to limitations on what they could substitute for a type variable. While those relations could likely have been adapted to handle existential types, such modifications would be mutually exclusive with their approach to universal types, unlike ours.

Toro et al. [23] prove termination-insensitive noninterference, for a language with gradual security via a logical relations argument. Analogous to Toro et al. [24], applying the AGT approach produces a language that satisfies graduality but not noninterference, and so they tweak the language to satisfy noninterference but not graduality. It is possible that, as with this work, revisiting the *syntax* of the security language might allow for a design satisfying both properties.

REFERENCES

[1] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. 2009. State-Dependent Representation Independence. In *ACM Symposium on Principles of Programming Languages (POPL), Savannah, Georgia*.

[2] Amal Ahmed, Robert Bruce Findler, Jeremy Siek, and Philip Wadler. 2011. Blame for All. In *ACM Symposium on Principles of Programming Languages (POPL), Austin, Texas*. 201–214.

[3] Amal Ahmed, Dustin Jamner, Jeremy G. Siek, and Philip Wadler. 2017. Theorems for Free for Free: Parametricity, With and Without Types. In *International Conference on Functional Programming (ICFP), Oxford, United Kingdom*.

[4] Felipe Bañados Schwerter, Ronald Garcia, and Éric Tanter. 2014. A Theory of Gradual Effect Systems. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming (ICFP '14)*. 283–295.

[5] Giuseppe Castagna and Victor Lanvin. 2017. Gradual Typing with Union and Intersection Types. *Proc. ACM Program. Lang.* 1, ICFP, Article 41 (Aug. 2017), 28 pages. https://doi.org/10.1145/3110285

[6] Tim Disney and Cormac Flanagan. 2011. Gradual Information Flow Typing. In *Workshop on Script-to-Program Evolution (STOP)*.

[7] Luminous Fennell and Peter Thiemann. 2013. Gradual Security Typing with References. In *CSF*. IEEE Computer Society, 224–239.

[8] Ronald Garcia and Matteo Cimini. 2015. Principal Type Schemes for Gradual Programs *(POPL '15)*.

[9] Ronald Garcia, Alison M. Clark, and Éric Tanter. 2016. Abstracting Gradual Typing. In *ACM Symposium on Principles of Programming Languages (POPL)*.

[10] Atsushi Igarashi, Peter Thiemann, Vasco Vasconcelos, and Philip Wadler. 2017. Gradual Session Types. In *International Conference on Functional Programming (ICFP)*.

[11] Yuu Igarashi, Taro Sekiyama, and Atsushi Igarashi. 2017. On Polymorphic Gradual Typing. In *International Conference on Functional Programming (ICFP), Oxford, United Kingdom*.

[12] Nico Lehmann and Éric Tanter. 2017. Gradual Refinement Types. In *ACM Symposium on Principles of Programming Languages (POPL)*.

[13] Paul Blain Levy. 2003. *Call-By-Push-Value: A Functional/Imperative Synthesis*. Springer.

[14] Georg Neis, Derek Dreyer, and Andreas Rossberg. 2009. Non-Parametric Parametricity. In *International Conference on Functional Programming (ICFP)*. 135–148.

[15] Max S. New and Amal Ahmed. 2018. Graduality from Embedding-Projection Pairs. In *International Conference on Functional Programming (ICFP), St. Louis, Missouri*.

[16] Max S. New, Daniel R. Licata, and Amal Ahmed. 2019. Gradual Type Theory *(POPL '19)*.

[17] Amr Sabry and Matthias Felleisen. 1992. Reasoning about Programs in Continuation-Passing Style. In *Conf. on LISP and functional programming, LFP '92*.

[18] Ilya Sergey and Dave Clarke. 2012. Gradual Ownership Types. In *ESOP (Lecture Notes in Computer Science)*, Vol. 7211. Springer, 579–599.

[19] Jeremy Siek, Micahel Vitousek, Matteo Cimini, and John Tang Boyland. 2015. Refined Criteria for Gradual Typing. In *1st Summit on Advances in Programming Languages*.

[20] Jeremy G. Siek and Walid Taha. 2006. Gradual Typing for Functional Languages. In *Scheme and Functional Programming Workshop (Scheme)*. 81–92.

[21] Sam Tobin-Hochstadt and Matthias Felleisen. 2006. Interlanguage Migration: From Scripts to Programs. In *Dynamic Languages Symposium (DLS)*. 964–974.

[22] Sam Tobin-Hochstadt and Matthias Felleisen. 2008. The Design and Implementation of Typed Scheme. In *ACM Symposium on Principles of Programming Languages (POPL), San Francisco, California*.

[23] Matías Toro, Ronald Garcia, and Éric Tanter. 2018. Type-Driven Gradual Security with References. *ACM Transactions on Programming Languages and Systems* 40, 4 (Dec. 2018). http://doi.acm.org/10.1145/3229061

[24] Matías Toro, Elizabeth Labrada, and Éric Tanter. 2019. Gradual Parametricity, Revisited. *Proc. ACM Program. Lang.* 3, POPL, Article 17 (Jan. 2019), 30 pages. https://doi.org/10.1145/3290330

[25] Roger Wolff, Ronald Garcia, Éric Tanter, and Jonathan Aldrich. 2011. Gradual Typestate. In *Proceedings of the 25th European Conference on Object-oriented Programming (ECOOP'11)*.

[26] Ningning Xie, Xuan Bi, and Bruno C. d. S. Oliveira. 2018. Consistent Subtyping for All. In *Programming Languages and Systems*, Amal Ahmed (Ed.). Springer International Publishing, Cham, 3–30.

$$\vdash \cdot \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash A}{\vdash \Gamma, x : A} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, X} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash A}{\vdash \Gamma, X \cong A}$$

$$\Gamma \vdash ? \qquad \frac{X \in \Gamma}{\Gamma \vdash X} \qquad \Gamma \vdash \mathbb{B} \qquad \frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \times A_2} \qquad \frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \to A_2} \qquad \frac{\Gamma, X \vdash A}{\Gamma \vdash \exists^\nu X.A}$$

$$\frac{\Gamma, X \vdash A}{\Gamma \vdash \forall^\nu X.A}$$

Fig. 19. Well-formedness of Environments, Types

$$\Gamma^\sqsubseteq \vdash ? : ? \sqsubseteq ? \qquad \Gamma^\sqsubseteq \vdash \mathbb{B} : \mathbb{B} \sqsubseteq \mathbb{B} \qquad \frac{X \in \Gamma^\sqsubseteq}{\Gamma^\sqsubseteq \vdash X : X \sqsubseteq X} \qquad \frac{\Gamma^\sqsubseteq \vdash A^\sqsubseteq : A \sqsubseteq G}{\Gamma^\sqsubseteq \vdash \mathrm{tag}_G(A^\sqsubseteq) : A \sqsubseteq ?}$$

$$\frac{\Gamma^\sqsubseteq, X \vdash A^\sqsubseteq : A_l \sqsubseteq A_r}{\Gamma^\sqsubseteq \vdash \forall^\nu X.A^\sqsubseteq : \forall^\nu X.A_l \sqsubseteq \forall^\nu X.A_r} \qquad \frac{\Gamma^\sqsubseteq, X \vdash A^\sqsubseteq : A_l \sqsubseteq A_r}{\Gamma^\sqsubseteq \vdash \exists^\nu X.A^\sqsubseteq : \exists^\nu X.A_l \sqsubseteq \exists^\nu X.A_r}$$

$$\frac{\Gamma^\sqsubseteq \vdash A^\sqsubseteq : A_l \sqsubseteq A_r \qquad \Gamma^\sqsubseteq \vdash B^\sqsubseteq : B_l \sqsubseteq B_r}{\Gamma^\sqsubseteq \vdash A^\sqsubseteq \to B^\sqsubseteq : A_l \to B_l \sqsubseteq A_r \to B_r} \qquad \frac{\Gamma^\sqsubseteq \vdash A^\sqsubseteq : A_l \sqsubseteq A_r \qquad \Gamma^\sqsubseteq \vdash B^\sqsubseteq : B_l \sqsubseteq B_r}{\Gamma^\sqsubseteq \vdash A^\sqsubseteq \times B^\sqsubseteq : A_l \times B_l \sqsubseteq A_r \times B_r}$$

$$\begin{aligned} \cdot|_{\Gamma'} &= \cdot \\ (X \cong A^\sqsubseteq, \Gamma^\sqsubseteq)|_{\Gamma'} &= X \cong A^\sqsubseteq, (\Gamma^\sqsubseteq|_{\Gamma'}) && (\mathrm{FV}(A^\sqsubseteq) \cap \Gamma' = \emptyset) \\ (X \cong A^\sqsubseteq, \Gamma^\sqsubseteq)|_{\Gamma'} &= \Gamma^\sqsubseteq|_{\Gamma', X} && (\mathrm{FV}(A^\sqsubseteq) \cap \Gamma' \neq \emptyset) \end{aligned}$$

Fig. 20. Type Precision in a Precision Context, Restriction of a Precision Context

A SURFACE LANGUAGE

Well-formedness of environments and types is mutually defined:

B TYPE PRECISION

We present the definition of type precision with respect to a precision context in Figure 20.

With this presentation it is easy to see that

(1) There is at most one derivation of $\Gamma^\sqsubseteq \vdash A \sqsubseteq A'$
(2) ? is the most imprecise type.
(3) Precision is reflexive.
(4) Precision is transitive.

The latter 3 statements are actually ambiguous because we haven't said what $\Gamma^\sqsubseteq$ is in each situation. We don't use these operations in their full generality, but ok.

Lemma B.1. *If* $\Gamma^\sqsubseteq \vdash A^\sqsubseteq : A \sqsubseteq A'$ *and* $\Gamma^\sqsubseteq B^\sqsubseteq : A \sqsubseteq A'$ *then* $A^\sqsubseteq = B^\sqsubseteq$

$$
\begin{aligned}
A \sqcap ? &= A \\
? \sqcap B &= B \\
X \sqcap X &= X \\
\mathbb{B} \sqcap \mathbb{B} &= \mathbb{B} \\
(A_1 \times A_2) \sqcap (B_1 \times B_2) &= (A_1 \sqcap B_1) \times (A_2 \sqcap B_2) \\
(A_i \to A_o) \sqcap (B_i \to B_o) &= (A_i \sqcap B_i) \to (A_o \sqcap B_o) \\
(\exists^\nu X.A) \sqcap (\exists^\nu X.B) &= \exists^\nu X.(A \sqcap B) \\
(\forall^\nu X.A) \sqcap (\forall^\nu X.B) &= \forall^\nu X.(A \sqcap B)
\end{aligned}
$$

Fig. 21. Gradual Meets

Proof. By induction on derivations. If $A' \neq ?$, then only one rule applies and the proof follows by inductive hypotheses. If $A^\sqsubseteq = ? : ? \sqsubseteq ?$, the only other rule that could apply is $\mathrm{tag}_G(A^\sqsubseteq)$, which cannot apply because it is easy to see that $? \sqsubseteq G$ does not hold for any G. Finally, we need to show that if $A^\sqsubseteq = \mathrm{tag}_G(A^\sqsubseteq)$ and $B^\sqsubseteq = \mathrm{tag}_{G'}(A^\sqsubseteq)$ then $G = G'$ because it is easy to see if $A \sqsubseteq G$ and $A \sqsubseteq G'$ then $G = G'$. □

This is "identity expansion", the derivations are used in the parametricity theorem.

Lemma B.2. *If* $\Gamma \vdash A$ *is a well-formed type then* $\Gamma \vdash A : A \sqsubseteq A$.

Proof. By induction over A. Every type constructor is punned with its type precision constructor. □

This is "cut elimination":

Lemma B.3. *If* $\Gamma^\sqsubseteq \vdash AB^\sqsubseteq : A \sqsubseteq B$ *and* $\Gamma^\sqsubseteq \vdash BC^\sqsubseteq : B \sqsubseteq C$ *then we can construct a proof* $\Gamma^\sqsubseteq \vdash AC^\sqsubseteq : A \sqsubseteq C$.

Proof. By induction on $BC^\sqsubseteq$.

(1) If $BC^\sqsubseteq = ? : ? \sqsubseteq ?$, then the proof is just $AB^\sqsubseteq$
(2) If $BC^\sqsubseteq = \mathrm{tag}_G(BG^\sqsubseteq)$, then $BG^\sqsubseteq : B \sqsubseteq G$ so by inductive hypothesis, there is a proof $AG^\sqsubseteq : A \sqsubseteq G$ and the proof we need is $\mathrm{tag}_G(AG^\sqsubseteq)$.
(3) If $BC^\sqsubseteq = BC_1^\sqsubseteq \times BC_2^\sqsubseteq$, then it must also be the case that $AB^\sqsubseteq = AB_1^\sqsubseteq \times AB_2^\sqsubseteq$, and then our result is $AC_1^\sqsubseteq \times AC_2^\sqsubseteq$ where $AC_1^\sqsubseteq, AC_2^\sqsubseteq$ come from the inductive hypothesis.
(4) All other cases are analogous to the product.

□

Next, we (partially) define gradual meets $A \sqcap B$ in Figure 21. The meet is undefined if the case is missing.

Lemma B.4. *For every* $\Gamma \vdash A, B$, $\Gamma \vdash A \sqcap B$ *and there are precision derivations*

(1) $\Gamma \vdash A{\sqcap}^\sqsubseteq : A \sqcap B \sqsubseteq A$
(2) $\Gamma \vdash B{\sqcap}^\sqsubseteq : A \sqcap B \sqsubseteq B$

Such that for any $\Gamma \vdash C$ *with* $\Gamma \vdash CA^\sqsubseteq : C \sqsubseteq A$ *and* $\Gamma \vdash CB^\sqsubseteq : C \sqsubseteq B$, *there exists a derivation*

$$\Gamma \vdash C{\sqcap}^\sqsubseteq : C \sqsubseteq A \sqcap B$$

$$\frac{x : A \in \Gamma}{\Sigma; \Gamma \vdash x : A; \cdot} \qquad \frac{\Sigma; \Gamma \vdash M : A_l; \Gamma_M \qquad \Sigma; \Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Sigma; \Gamma \vdash \langle A^{\sqsubseteq} \rangle\!\uparrow M : A_r; \Gamma_M} \qquad \frac{\Sigma; \Gamma \vdash M : A_r; \Gamma_M \qquad \Sigma; \Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Sigma; \Gamma \vdash \langle A^{\sqsubseteq} \rangle\!\downarrow M : A_l; \Gamma_M}$$

$$\frac{\Sigma; \Gamma \vdash M : A; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M, x : A \vdash N : B; \Gamma_N}{\Sigma; \Gamma \vdash \mathtt{let}\ x = M; N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Sigma; \Gamma \vdash M : A; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o}{\Sigma; \Gamma \vdash \mathtt{seal}_X M : X; \Gamma_o}$$

$$\frac{\Sigma; \Gamma \vdash M : X; \Gamma_o \qquad X \cong A \in \Gamma, \Gamma_o}{\Sigma; \Gamma \vdash \mathtt{unseal}_X M : A; \Gamma_o} \qquad \frac{\Sigma; \Gamma \vdash M : ?; \Gamma_o \qquad \Sigma; \Gamma \vdash G}{\Sigma; \Gamma \vdash \mathtt{is}(G)?\ M : \mathbb{B}; \Gamma_o} \qquad \Sigma; \Gamma \vdash \mathtt{true} : \mathbb{B}; \cdot$$

$$\Sigma; \Gamma \vdash \mathtt{false} : \mathbb{B}; \cdot \qquad \frac{\Sigma; \Gamma \vdash M : \mathbb{B}; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M \vdash N_t : B; \Gamma_N \qquad \Sigma; \Gamma, \Gamma_M \vdash N_f : B; \Gamma_N}{\Sigma; \Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ N_t\ \mathtt{else}\ N_f : B; \Gamma_M, \Gamma_N}$$

$$\frac{\Sigma; \Gamma \vdash M_1 : A_1; \Gamma_1 \qquad \Sigma; \Gamma, \Gamma_1 \vdash M_2 : A_2; \Gamma_2}{\Sigma; \Gamma \vdash (M_1, M_2) : A_1 \times A_2; \Gamma_1, \Gamma_2}$$

$$\frac{\Sigma; \Gamma \vdash M : A_1 \times A_2; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M, x : A_1, y : A_2 \vdash N : B; \Gamma_N}{\Sigma; \Gamma \vdash \mathtt{let}\ (x, y)\ =\ M; N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Sigma; \Gamma, x : A \vdash M : B; \cdot}{\Sigma; \Gamma \vdash \lambda x : A.M : A \to B; \cdot}$$

$$\frac{\Sigma; \Gamma \vdash M : A \to B; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M \vdash N : A; \Gamma_N}{\Sigma; \Gamma \vdash M\ N : B; \Gamma_M, \Gamma_N} \qquad \frac{\Sigma; \Gamma, X \cong A \vdash M : B; \cdot}{\Sigma; \Gamma \vdash \mathsf{pack}^{\nu}(X \cong A, M) : \exists^{\nu} X.B; \cdot}$$

$$\frac{\Sigma; \Gamma \vdash M : \exists^{\nu} X.A; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M, X, x : A \vdash N : B; \Gamma_N \qquad \Sigma; \Gamma, \Gamma_M \vdash \Gamma_N \qquad \Sigma; \Gamma, \Gamma_M, \Gamma_N \vdash B}{\Sigma; \Gamma \vdash \mathsf{unpack}\ (X, x) = M; N : B; \Gamma_M, \Gamma_N}$$

$$\frac{\Sigma; \Gamma, X \vdash M : A; \cdot}{\Sigma; \Gamma \vdash \Lambda^{\nu} X.M : \forall^{\nu} X.A; \cdot} \qquad \frac{\Sigma; \Gamma \vdash M : \forall^{\nu} X.A; \Gamma_M \qquad \Sigma; \Gamma, \Gamma_M \vdash B}{\Sigma; \Gamma \vdash M\{X \cong B\} : A; \Gamma_M, X \cong B}$$

$$\frac{\Sigma; \Gamma \vdash M : \Gamma_M, X \cong A, \Gamma'_M \qquad \Sigma; \Gamma, \Gamma_M \vdash \Gamma'_M}{\Sigma; \Gamma \vdash \mathtt{hide}\ X \cong A; M; \Gamma_M, \Gamma'_M} \qquad \frac{\Sigma; \Gamma \vdash M : A; \Gamma_o \qquad \sigma : A \in \Sigma}{\Sigma; \Gamma \vdash \mathtt{seal}_{\sigma} M : \sigma; \Gamma_o}$$

$$\frac{\Sigma; \Gamma \vdash M : \sigma; \Gamma_o \qquad \sigma : A \in \Sigma}{\Sigma; \Gamma \vdash \mathtt{unseal}_{\sigma} M : A; \Gamma_o} \qquad \frac{\Sigma; \Gamma \vdash M : \forall^{\nu} X.A; \Gamma_M \qquad \sigma : A \in \Sigma}{\Sigma; \Gamma \vdash M\{\sigma \cong B\} : A; \Gamma_M}$$

$$\frac{\Sigma; \Gamma, X \cong A' \vdash \langle A^{\sqsubseteq} \rangle\!\updownarrow \ldots M : B; \cdot}{\Sigma; \Gamma \vdash \mathsf{pack}^{\nu}(X \cong A', [A^{\sqsubseteq} \updownarrow \ldots], M) : \exists^{\nu} X.B; \cdot}$$

Fig. 22. PolyC$^{\nu}$ Typing for Runtime Forms

C CAST CALCULUS

Some of the semantics in Figure 7 involve terms with σs in places we would expect Xs, in particular instantiations, seals, and unseals. These forms come about when we evaluate a `hide` and substitute σ for X. We also have our aforementioned intermediate form for pack casts. Figure 22 gives the static typing rules for runtime terms. Note that the typing of runtime terms depends on a given Σ. To reason about well-typed terms at runtime, we also thread a store through the rules in Figure 5.

$$\frac{\Sigma;\Delta;\Gamma \vdash V_T : \text{Case } A \qquad \Sigma;\Delta;\Gamma \vdash V : \text{OSum} \qquad \Sigma;\Delta;\Gamma, x : A \mid \cdot \vdash M : B \qquad \Sigma;\Delta;\Gamma \mid \cdot \vdash N : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{match}\ V_T\ \texttt{with}\ V\{\texttt{inj}\ x.M \mid N\} : B}$$

$$\frac{\sigma : A \in \Sigma \qquad \Delta \vdash A}{\Sigma;\Delta;\Gamma \vdash \sigma : \text{Case } A} \qquad \frac{\Delta \vdash A \qquad \Sigma;\Delta;\Gamma, x : \text{Case } A \mid \cdot \vdash M : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{newcase}_A\ x; M : B}$$

$$\frac{\Sigma;\Delta;\Gamma \vdash V_T : \text{Case } A \qquad \Sigma;\Delta;\Gamma \vdash V : A}{\Sigma;\Delta;\Gamma \vdash \texttt{inj}_{V_T}\ V : \text{OSum}} \qquad \frac{\Sigma;\Delta, X;\Gamma \mid \cdot \vdash M : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \Lambda X.M : \forall X.B}$$

$$\frac{\Sigma;\Delta;\Gamma \mid \Theta \vdash M : \forall X.B \qquad \Delta \vdash A}{\Sigma;\Delta;\Gamma \mid \Theta \vdash M[A] : B[A/X]} \qquad \frac{\Sigma;\Delta;\Gamma \vdash V : A[A'/X]}{\Sigma;\Delta;\Gamma \vdash \texttt{pack}(A', V)\ \texttt{as}\ \exists X.A : \exists X.A}$$

$$\frac{\Sigma;\Delta;\Gamma \vdash V : \exists X.A \qquad \Delta \vdash B \qquad \Sigma;\Delta, X;\Gamma, x : A \mid \cdot \vdash M : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{unpack}\ (X, x) = V; M : B} \qquad \frac{}{\Sigma;\Delta;\Gamma, x : A, \Gamma' \vdash x : A}$$

$$\frac{}{\Sigma;\Delta;\Gamma \mid \bullet : B \vdash \bullet : B} \qquad \frac{\Sigma;\Delta;\Gamma \vdash V : \mathbb{B} \qquad \Sigma;\Delta;\Gamma \mid \cdot \vdash M_1 : B \qquad \Sigma;\Delta;\Gamma \mid \cdot \vdash M_2 : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{if}\ V\ \texttt{then}\ M_1\ \texttt{else}\ M_2 : B}$$

$$\frac{}{\Sigma;\Delta;\Gamma \vdash \texttt{true} : \mathbb{B}} \qquad \frac{}{\Sigma;\Delta;\Gamma \vdash \texttt{false} : \mathbb{B}} \qquad \frac{\Sigma;\Delta;\Gamma \vdash V_1 : A_1 \qquad \Sigma;\Delta;\Gamma \vdash V_2 : A_2}{\Sigma;\Delta;\Gamma \vdash (V_1, V_2) : A_1 \times A_2}$$

$$\frac{\Sigma;\Delta;\Gamma \vdash V : A_1 \times A_2 \qquad \Sigma;\Delta;\Gamma, x : A_1, y : A_2 \mid \cdot \vdash M : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{let}\ (x, y)\ =\ V; M : B} \qquad \frac{\Sigma;\Delta;\Gamma \mid \cdot \vdash M : B}{\Sigma;\Delta;\Gamma \vdash \texttt{thunk}\ M : UB}$$

$$\frac{\Sigma;\Delta;\Gamma \vdash V : UB}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{force}\ V : B} \qquad \frac{\Sigma;\Delta;\Gamma \vdash V : A}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \texttt{ret}\ V : FA}$$

$$\frac{\Sigma;\Delta;\Gamma \mid \Theta \vdash M : FA \qquad \Sigma;\Delta;\Gamma \mid \cdot \vdash N : B}{\Sigma;\Delta;\Gamma \mid \Theta \vdash x \leftarrow M; N : B} \qquad \frac{\Sigma;\Delta;\Gamma \mid \Theta \vdash M : A \to B \qquad \Sigma;\Delta;\Gamma \vdash V : A}{\Sigma;\Delta;\Gamma \mid \Theta \vdash M\ V : B}$$

$$\frac{\Sigma;\Delta;\Gamma, x : A \mid \cdot \vdash V : B}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \lambda x : A.V : A \to B} \qquad \frac{}{\Sigma;\Delta;\Gamma \mid \cdot \vdash \mho : B}$$

Fig. 23. $\text{CBPV}_{\text{OSum}}$ Type System

$$
\begin{aligned}
S[\mho] &\mapsto \mho \\
\Sigma \triangleright S[\mathtt{newcase}_A\ x; M] &\mapsto \Sigma, \sigma : A \triangleright S[M[\sigma/x]] \\
S[\mathtt{match}\ \mathtt{inj}_\sigma\ V\ \mathtt{with}\ \sigma\{\mathtt{inj}\ x.M \mid N\}] &\mapsto S[M[V/x]] \\
S[\mathtt{match}\ \mathtt{inj}_{\sigma_1}\ V\ \mathtt{with}\ \sigma_2\{\mathtt{inj}\ x.M \mid N\}] &\mapsto S[N] \qquad (\text{where } \sigma_1 \neq \sigma_2) \\
S[\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ M\ \mathtt{else}\ N] &\mapsto S[M] \\
S[\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ M\ \mathtt{else}\ N] &\mapsto S[N] \\
S[\mathtt{let}\ (x, y) = (V_1, V_2); M] &\mapsto S[M[V_1/x, V_2/y]] \\
S[\mathtt{force}\ (\mathtt{thunk}\ M)] &\mapsto S[M] \\
S[\mathtt{unpack}\ (X, x) = \mathtt{pack}(A, V); M] &\mapsto S[M[A/X, V/x]] \\
S[(\Lambda X.M)[A]] &\mapsto S[M[A/X]] \\
S[(\lambda(x : A).M)\ V] &\mapsto S[M[V/x]] \\
S[x \leftarrow \mathtt{ret}\ V; N] &\mapsto S[N[V/x]]
\end{aligned}
$$

Fig. 24. $\mathrm{CBPV}_{\mathrm{OSum}}$ Operational Semantics

D CBPV

We also name the store generated by our preamble:

$$
\begin{aligned}
\Sigma_p &= (4, f) \\
f(0) &= \mathbb{B} \\
f(1) &= \mathrm{OSum} \times \mathrm{OSum} \\
f(2) &= U(\mathrm{OSum} \to F\mathrm{OSum}) \\
f(3) &= \exists X.U(\mathrm{Case}\ X \to F\mathrm{OSum}) \\
f(4) &= U(\forall X.\mathrm{Case}\ X \to F\mathrm{OSum})
\end{aligned}
$$

We define $\gamma_p : \Gamma_p$ to be a substitution that closes terms with respect to Γ_p using the store Σ_p:

$$
\begin{aligned}
\gamma_p(c_{\mathbb{B}}) &= 0 \\
\gamma_p(c_{\times}) &= 1 \\
\gamma_p(c_{\to}) &= 2 \\
\gamma_p(c^{\nu}_{\exists}) &= 3 \\
\gamma_p(c^{\nu}_{\forall}) &= 4
\end{aligned}
$$

We give the full translation from PolyC^{ν} to $\mathrm{CBPV}_{\mathrm{OSum}}$ in Figure 25. Added to the cases from the paper are the translation for types $\mathbb{B}$ and $A \times B$.

E TERM PRECISION

Figure 28 shows the full definition of term precision for PolyG^{ν}. Figures 29 and 30 shows the full definition of term precision for PolyC^{ν}.

Lemma E.1 (Casts are Monotone). *If* $A^{\sqsubseteq} : A_l \sqsubseteq A_r$ *and* $AB^{\sqsubseteq}_l : A_l \sqsubseteq B_l$ *and* $AB^{\sqsubseteq}_r : A_r \sqsubseteq B_r$ *and* $B^{\sqsubseteq} : B_l \sqsubseteq B_r$, *then*

(1) *If* $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq}$, *then* $\Gamma^{\sqsubseteq} \vdash \langle AB^{\sqsubseteq}_l \rangle\!\uparrow M_l \sqsubseteq \langle AB^{\sqsubseteq}_r \rangle\!\uparrow M_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq}_o$

(2) *If* $\Gamma^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq}_o$, *then* $\Gamma^{\sqsubseteq} \vdash \langle c \rangle\!\downarrow$ *inserts that* $AB^{\sqsubseteq}_l N_l \sqsubseteq \langle AB^{\sqsubseteq}_r \rangle\!\downarrow N_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq}_o$

$$
\begin{array}{lcl}
[\![x]\!] & = & \mathtt{ret}\ x \\
[\![\mathtt{let}\ x = M; N]\!] & = & x \leftarrow [\![M]\!]; [\![N]\!] \\
[\![\mho_A]\!] & = & \mho \\
[\![\mathtt{seal}_\alpha M]\!] & = & [\![M]\!] \\
[\![\mathtt{unseal}_\alpha M]\!] & = & [\![M]\!] \\
[\![\mathtt{inj}_G\ M]\!] & = & r \leftarrow [\![M]\!]; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ r \\
[\![\mathtt{is}(G)?\ M]\!] & = & r \leftarrow [\![M]\!]; \\
 & & \mathtt{match}\ r\ \mathtt{with}\ \mathrm{case}(G)\{\mathtt{inj}\ y.\mathtt{ret}\ \mathtt{true} \mid \mathtt{ret}\ \mathtt{false}\} \\
[\![\mathtt{hide}\ X \cong A; M]\!] & = & \mathtt{newcase}_{[\![A]\!]}\ c_X; [\![M]\!] \\
[\![\langle A^{\sqsubseteq}\rangle\updownarrow\ M]\!] & = & [\![A^{\sqsubseteq}]\!]\updownarrow[\![M]\!] \\
[\![\mathtt{true}]\!] & = & \mathtt{ret}\ \mathtt{true} \\
[\![\mathtt{false}]\!] & = & \mathtt{ret}\ \mathtt{false} \\
[\![\mathtt{if}\ M_1\ \mathtt{then}\ M_2\ \mathtt{else}\ M_3]\!] & = & r \leftarrow [\![M_1]\!]; \mathtt{if}\ r\ \mathtt{then}\ [\![M_2]\!]\ \mathtt{else}\ [\![M_3]\!] \\
[\![(M_1, M_2)]\!] & = & x_1 \leftarrow [\![M_1]\!]; x_2 \leftarrow [\![M_2]\!]; \mathtt{ret}\ (x_1, x_2) \\
[\![\mathtt{let}\ (x, y)\ =\ M; N]\!] & = & r \leftarrow [\![M]\!]; \mathtt{let}\ (x, y)\ =\ r; [\![N]\!] \\
[\![\mathtt{pack}^\nu(X \cong A, M)]\!] & = & \mathtt{ret}\ \mathtt{pack}(A, \mathtt{thunk}\ (\lambda c_X : \mathrm{Case}\ A.[\![M]\!])) \\
[\![\mathtt{pack}^\nu(X \cong A', [A_n^{\sqsubseteq} \updownarrow_n \ldots A_1^{\sqsubseteq} \updownarrow_1], M)]\!] & = & \mathtt{ret}\ \mathtt{pack}(A, \mathtt{thunk}\ (\lambda c_X : \mathrm{Case}\ A.M'_n)) \\
 & & \text{where } M'_0 = [\![M]\!] \\
 & & \text{and } M'_{i+1} = [\![A_{i+1}^{\sqsubseteq}]\!]\updownarrow_{i+1}[\mathtt{force}\ (\mathtt{thunk}\ (\lambda c_X : \mathrm{Case}\ A'.[\![M_i]\!]))\ c_X] \\
[\![\mathtt{unpack}\ (X, x) = M; N]\!] & = & r \leftarrow [\![M]\!]; \mathtt{unpack}\ (X, f) = r; \\
 & & \mathtt{newcase}_X\ c_X; x \leftarrow (\mathtt{force}\ f)\ c_X; [\![N]\!] \\
[\![\Lambda^\nu X.M]\!] & = & \mathtt{ret}\ (\mathtt{thunk}\ (\Lambda X.\lambda(c_X : \mathrm{Case}\ X).[\![M]\!])) \\
[\![M\{\alpha \cong A\}]\!] & = & f \leftarrow [\![M]\!]; (\mathtt{force}\ f)[A]\ (\mathrm{case}(\alpha)) \\
[\![\lambda(x : A).M]\!] & = & \mathtt{ret}\ \mathtt{thunk}\ \lambda(x : [\![A]\!]).[\![M]\!] \\
[\![M\ N]\!] & = & f \leftarrow [\![M]\!]; a \leftarrow [\![N]\!]; (\mathtt{force}\ f)\ a
\end{array}
$$

Fig. 25. PolyC$^\nu$ term translation

$$
\cdot : \cdot \sqsubseteq \cdot \qquad \frac{\Gamma^{\sqsubseteq} : \Gamma_l \sqsubseteq \Gamma_r \qquad \Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{(\Gamma^{\sqsubseteq}, x : A^{\sqsubseteq}) : \Gamma_l, x : A_l \sqsubseteq \Gamma_r, x : A_r} \qquad \frac{\Gamma^{\sqsubseteq} : \Gamma_l \sqsubseteq \Gamma_r \qquad \Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{(\Gamma^{\sqsubseteq}, X \cong A^{\sqsubseteq}) : \Gamma_l, X \cong A_l \sqsubseteq \Gamma_r, X \cong A_r}
$$

$$
\frac{\Gamma^{\sqsubseteq} : \Gamma_l \sqsubseteq \Gamma_r}{(\Gamma^{\sqsubseteq}, X) : \Gamma_l, X \sqsubseteq \Gamma_r, X}
$$

Fig. 26. Type Precision Contexts

Proof. (1) By the following derivation

$$
\frac{\dfrac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle AB_r^{\sqsubseteq}\rangle\updownarrow\ M_r : AB_{lr}^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}}{\Gamma^{\sqsubseteq} \vdash \langle AB_l^{\sqsubseteq}\rangle\updownarrow\ M_l \sqsubseteq \langle AB_r^{\sqsubseteq}\rangle\updownarrow\ M_r : B^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}
$$

Where $AB_{lr}^{\sqsubseteq} : A_l \sqsubseteq B_r$, which exists by transitivity lemma B.3.

(2) By the following derivation

$$
\frac{\dfrac{\Gamma^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \langle AB_l^{\sqsubseteq}\rangle\downarrow\ N_l \sqsubseteq N_r : AB_{lr}^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}}{\Gamma^{\sqsubseteq} \vdash \langle AB_l^{\sqsubseteq}\rangle\downarrow\ N_l \sqsubseteq \langle AB_r^{\sqsubseteq}\rangle\downarrow\ N_r : A^{\sqsubseteq}; \Gamma_o^{\sqsubseteq}}
$$

$$\begin{aligned}
\mathrm{cod}(A^{\sqsubseteq} \to B^{\sqsubseteq}) &= B^{\sqsubseteq} \\
\mathrm{cod}(?) &= ? \\
\mathrm{cod}(\mathrm{tag}_G(A^{\sqsubseteq} \to B^{\sqsubseteq})) &= B^{\sqsubseteq} \\
\pi_i(A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}) &= A_i^{\sqsubseteq} \\
\pi_i(?) &= ? \\
\pi_i(\mathrm{tag}_{?\times ?}(A_0^{\sqsubseteq} \times A_1^{\sqsubseteq})) &= A_i^{\sqsubseteq}
\end{aligned}$$

$$\begin{aligned}
\mathrm{un}\forall^{\nu}(\forall^{\nu} X.A^{\sqsubseteq}) &= A^{\sqsubseteq} \\
\mathrm{un}\forall^{\nu}(?) &= ? \\
\mathrm{un}\forall^{\nu}(\mathrm{tag}_{\forall^{\nu} X.?}(\forall^{\nu} X.A^{\sqsubseteq})) &= A^{\sqsubseteq} \\
\mathrm{un}\exists^{\nu}(\exists^{\nu} X.A^{\sqsubseteq}) &= A^{\sqsubseteq} \\
\mathrm{un}\exists^{\nu}(?) &= ? \\
\mathrm{un}\exists^{\nu}(\mathrm{tag}_{\exists^{\nu} X.?}(\exists^{\nu} X.A^{\sqsubseteq})) &= A^{\sqsubseteq}
\end{aligned}$$

Fig. 27. Metafunctions extended to type precision derivations

Where $AB^{\sqsubseteq}_{lr} : A_l \sqsubseteq B_r$, which exists by transitivity lemma B.3.

□

Lemma E.2 (Hide Monotonicity). *If* $\Gamma_1^{\sqsubseteq} : \Gamma_{l1} \sqsubseteq \Gamma_{r1}$ *and* $\Gamma_2^{\sqsubseteq} : \Gamma_{l2} \sqsubseteq \Gamma_{r2}$ *and* $\Gamma_{l1} \subseteq \Gamma_{l2}$ *and* $\Gamma_{r1} \subseteq \Gamma_{r2}$ *and* $M_l \sqsubseteq M_r$*, then*

$$\mathtt{hide}\ \Gamma_{l1} \subseteq \Gamma_{l2}; M_l \sqsubseteq \mathtt{hide}\ \Gamma_{r1} \subseteq \Gamma_{r2}; M_r$$

Proof. By induction over $\Gamma_2^{\sqsubseteq}$, each case is either congruence or trivial application of inductive hypothesis. □

Lemma E.3. *If* $\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$ *in the surface language, then* $\Gamma^{\sqsubseteq} \vdash M_l^+ \sqsubseteq M_r^+ : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}$

Proof. By induction on term precision derivations.

(1) Cast. By applying lemma E.1 twice.
(2) Var: Immediate
(3) Let: immediate
(4) seal: by the argument for the ascription case.
(5) unseal: There are three cases for $A^{\sqsubseteq}$: X, ? and $\mathrm{tag}_X(X)$. The first case is immediate. If $A^{\sqsubseteq} = ?$, we need to show

$$\mathtt{unseal}_X \langle \mathrm{tag}_X(X) \rangle\!\downarrow M_l^+ \sqsubseteq \mathtt{unseal}_X \langle \mathrm{tag}_X(X) \rangle\!\downarrow M_r^+$$

Which follows by $\mathtt{unseal}_X$ congruence and lemma E.1. For the final case we need to show

$$\mathtt{unseal}_X M_l^+ \sqsubseteq \mathtt{unseal}_X \langle \mathrm{tag}_X(X) \rangle\!\downarrow M_r^+$$

which follows by congruence for $\mathtt{unseal}_X$ and the downcast-right rule.
(6) tag-check $\mathtt{is}(G)?\ M_l \sqsubseteq \mathtt{is}(G)?\ M_r$: follows by congruence and lemma E.1.
(7) tru: Immediate
(8) fls: Immediate
(9) if: By if congruence, we need to show the condition and the two branches of the if are ordered.
   - For the condition there are three subcases $M_l \sqsubseteq M_r : A^{\sqsubseteq}$: either $\mathbb{B}$, ? or $\mathrm{tag}_{\mathbb{B}}(\mathbb{B})$. The ordering follows by the same argument as the $\mathtt{unseal}_X$ case.

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^\sqsubseteq, \Gamma^{\sqsubseteq\prime} \vdash B^\sqsubseteq : B_l \sqsubseteq B_r}{\Gamma^\sqsubseteq \vdash (M_l :: B_l) \sqsubseteq (M_r :: B_r) : B^\sqsubseteq; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq \qquad X \cong B^\sqsubseteq \in \Gamma^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathtt{seal}_X M_l \sqsubseteq \mathtt{seal}_X M_r : X}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_o^\sqsubseteq \qquad X \cong B^\sqsubseteq \in \Gamma^\sqsubseteq, \Gamma_o^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathtt{unseal}_X M_l \sqsubseteq \mathtt{unseal}_X M_r : B^\sqsubseteq; \Gamma_o^\sqsubseteq} \qquad \frac{x : A^\sqsubseteq \in \Gamma^\sqsubseteq}{\Gamma^\sqsubseteq \vdash x \sqsubseteq x : A^\sqsubseteq}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, x : A^\sqsubseteq \vdash N_l \sqsubseteq N_r : B^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathtt{let}\ x = M_l; N_l \sqsubseteq \mathtt{let}\ x = M_r; N_r : B; \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq} \qquad \Gamma^\sqsubseteq \vdash \mathtt{true} \sqsubseteq \mathtt{true} : \mathbb{B}; \cdot$$

$$\Gamma^\sqsubseteq \vdash \mathtt{false} \sqsubseteq \mathtt{false} : \mathbb{B}; \cdot$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq \vdash N_{lt} \sqsubseteq N_{rt} : B_t^\sqsubseteq; \Gamma_t^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq \vdash N_{lf} \sqsubseteq N_{rf} : B_f^\sqsubseteq; \Gamma_f^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathtt{if}\ M_l\ \mathtt{then}\ N_{lt}\ \mathtt{else}\ N_{lf} \sqsubseteq \mathtt{if}\ M_r\ \mathtt{then}\ N_{rt}\ \mathtt{else}\ N_{rf} : B_t^\sqsubseteq \sqcup B_f^\sqsubseteq; \Gamma_M^\sqsubseteq, \Gamma_t^\sqsubseteq \cap \Gamma_f^\sqsubseteq}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_{l1} \sqsubseteq M_{r1} : A_1^\sqsubseteq; \Gamma_1^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_1^\sqsubseteq \vdash M_{l2} \sqsubseteq M_{r2} : A_2^\sqsubseteq; \Gamma_2^\sqsubseteq}{\Gamma^\sqsubseteq \vdash (M_{l1}, M_{l2}) \sqsubseteq (M_{r1}, M_{r2}) : A_1^\sqsubseteq \times A_2^\sqsubseteq; \Gamma_1^\sqsubseteq, \Gamma_2^\sqsubseteq}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, x : \pi_0(A^\sqsubseteq), y : \pi_1(A_1^\sqsubseteq) \vdash N_l \sqsubseteq N_r : B^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathtt{let}\ (x,y)\ =\ M_l; N_l \sqsubseteq \mathtt{let}\ (x,y)\ =\ M_r; N_r : B^\sqsubseteq; \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq}$$

$$\frac{\Gamma^\sqsubseteq, x : A^\sqsubseteq \vdash M_l \sqsubseteq M_r : B^\sqsubseteq; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^\sqsubseteq \vdash A^\sqsubseteq : A_l \sqsubseteq A_r}{\Gamma^\sqsubseteq \vdash \lambda x : A_l.M_l \sqsubseteq \lambda x : A_r.M_r : A^\sqsubseteq \to B^\sqsubseteq; \cdot}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq \vdash N_l \sqsubseteq N_r : B^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vdash M_l\ N_l \sqsubseteq M_r\ N_r : \mathrm{cod}(A^\sqsubseteq); \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq}$$

$$\frac{\Gamma^\sqsubseteq, X \cong B^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_o^\sqsubseteq \qquad \Gamma^\sqsubseteq \vdash B^\sqsubseteq : B_l \sqsubseteq B_r}{\Gamma^\sqsubseteq \vdash \mathsf{pack}^\nu(X \cong B_l, M_l) \sqsubseteq \mathsf{pack}^\nu(X \cong B_r, M_r) : \exists^\nu X.A^\sqsubseteq; \cdot}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, X, x : \mathrm{un}\exists^\nu(A^\sqsubseteq) \vdash N_l \sqsubseteq N_r : B^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \mathsf{unpack}\ (X,x) = M_l; N_l \sqsubseteq \mathsf{unpack}\ (X,x) = M_r; N_r : B^\sqsubseteq; \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq|_X}$$

$$\frac{\Gamma^\sqsubseteq, X \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_o^\sqsubseteq}{\Gamma^\sqsubseteq \vdash \Lambda^\nu X.M_l \sqsubseteq \Lambda^\nu X.M_r : \forall^\nu X.A^\sqsubseteq; \cdot}$$

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq \vdash B^\sqsubseteq : B_l \sqsubseteq B_r}{\Gamma^\sqsubseteq \vdash M_l\{X \cong B_l\} \sqsubseteq M_r\{X \cong B_r\} : \mathrm{un}\forall^\nu(A^\sqsubseteq); \Gamma_M^\sqsubseteq, X \cong B^\sqsubseteq}$$

Fig. 28. PolyG$^\nu$ Term Precision

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash AB^{\sqsubseteq} : A \sqsubseteq B \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vdash \langle AB^{\sqsubseteq}\rangle\uparrow M_l \sqsubseteq M_r : BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash AB^{\sqsubseteq} : A \sqsubseteq B \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vdash \langle AB^{\sqsubseteq}\rangle\downarrow M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\uparrow M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\downarrow M_r : AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong A^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma^{\sqsubseteq} \vdash \mathtt{hide}\ X \cong A_l; M_l \sqsubseteq \mathtt{hide}\ X \cong A_r; M_r : B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, \Gamma^{\sqsubseteq\prime\prime}}$$

$$\frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathtt{seal}_X M_l \sqsubseteq \mathtt{seal}_X M_r : X; \Gamma^{\sqsubseteq\prime}} \qquad \frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : X; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathtt{unseal}_X M_l \sqsubseteq \mathtt{unseal}_X M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Fig. 29. PolyC$^{\nu}$ Term Precision Part 1

$$\frac{x : A^{\sqsubseteq} \in \Gamma^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash x \sqsubseteq x : A^{\sqsubseteq}} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma_M \qquad \Gamma^{\sqsubseteq}, \Gamma_M, x : A^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma_N}{\Gamma^{\sqsubseteq} \vdash \mathtt{let}\ x = M_l; N_l \sqsubseteq \mathtt{let}\ x = M_r; N_r : B^{\sqsubseteq}; \Gamma_M, \Gamma_N}$$

$$\Gamma^{\sqsubseteq} \vdash \mathtt{true} \sqsubseteq \mathtt{true} : \mathbb{B} \qquad\qquad \Gamma^{\sqsubseteq} \vdash \mathtt{false} \sqsubseteq \mathtt{false} : \mathbb{B}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : \mathbb{B}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq} \vdash N_{lt} \sqsubseteq N_{rt} : B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq} \vdash N_{lf} \sqsubseteq N_{rf} : B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \mathtt{if}\ M_l\ \mathtt{then}\ N_{lt}\ \mathtt{else}\ N_{lf} \sqsubseteq \mathtt{if}\ M_r\ \mathtt{then}\ N_{rt}\ \mathtt{else}\ N_{rf} : B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_{l1} \sqsubseteq M_{r1} : A_1^{\sqsubseteq}; \Gamma_1^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_1^{\sqsubseteq} \vdash M_{l2} \sqsubseteq M_{r2} : A_2^{\sqsubseteq}; \Gamma_2^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash (M_{l1}, M_{l2}) \sqsubseteq (M_{r1}, M_{r2}) : A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}; \Gamma_1^{\sqsubseteq}, \Gamma_2^{\sqsubseteq}}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq}, x : A_1^{\sqsubseteq}, y : A_2^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash \mathtt{let}\ (x, y)\ =\ M_l; N_l \sqsubseteq \mathtt{let}\ (x, y)\ =\ M_r; N_r : B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}}$$

$$\frac{\Gamma^{\sqsubseteq}, x : A^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : B^{\sqsubseteq}; \cdot \qquad \Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma^{\sqsubseteq} \vdash \lambda x : A_l.M_l \sqsubseteq \lambda x : A_l.M_l : A^{\sqsubseteq} \to B^{\sqsubseteq}; \cdot}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq} \to B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : A^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vdash M_l\, N_l \sqsubseteq M_r\, N_r : B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}}$$

$$\frac{\Gamma^{\sqsubseteq}, X \cong B^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \cdot}{\Gamma^{\sqsubseteq} \vdash \mathsf{pack}^{\nu}(X \cong B_l, M_l) \sqsubseteq \mathsf{pack}^{\nu}(X \cong B_r, M_r) : \exists^{\nu} X.A^{\sqsubseteq}; \cdot} \qquad \frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : \forall^{\nu} X.A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vdash B^{\sqsubseteq} : B_l \sqsubseteq}{\Gamma^{\sqsubseteq} \vdash M_l\{X \cong B_l\} \sqsubseteq M_r\{X \cong B_r\} : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : \exists^{\nu} X.A^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \begin{array}{c}\Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq} \vdash \Gamma_N^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq} \vdash B^{\sqsubseteq} \\ \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq}, X, x : A^{\sqsubseteq} \vdash N_l \sqsubseteq N_r : B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}\end{array}}{\Gamma^{\sqsubseteq} \vdash \mathtt{unpack}\ (X, x) = M_l; N_l \sqsubseteq \mathtt{unpack}\ (X, x) = M_r; N_r : B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}} \qquad \frac{\Gamma^{\sqsubseteq}, X \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \cdot}{\Gamma^{\sqsubseteq} \vdash \Lambda^{\nu} X.M_l \sqsubseteq \Lambda^{\nu} X.M_r : \forall^{\nu} X.A^{\sqsubseteq}; \cdot}$$

Fig. 30. PolyC$^{\nu}$ Term Precision Part 2

- The two branches follow by the same argument. We describe the true branch. We have by inductive hypothesis that $N^+_{tl} \sqsubseteq N^+_{tr}$ and we need to show

$$\langle B^{\sqsubseteq}_{tl}\rangle\!\downarrow \texttt{hide}\ \Gamma_{tl} \subseteq \Gamma_{tl} \cap \Gamma_{fl}; N^+_{tl} \sqsubseteq \langle B^{\sqsubseteq}_{tr}\rangle\!\downarrow \texttt{hide}\ \Gamma_{tr} \subseteq \Gamma_{tr} \cap \Gamma_{fr}; N^+_{tr}$$

  Which follows by lemmas E.1 and E.2.

(10) Pair intro: Immediate by inductive hypothesis.
(11) Pair elim: By similar argument to $\texttt{unseal}_X$
(12) By congruence and lemma E.2.
(13) Function application: similar argument to $\texttt{unseal}_X$
(14) $\exists^\nu$ introduction: by congruence and lemma E.2.
(15) $\forall^\nu$ elimination: by similar argument to $\texttt{unseal}_X$, and lemma E.2.
(16) $\forall^\nu$ introduciton: by congruence and lemma E.2.
(17) $\forall^\nu$ elimination: by similar argument to $\texttt{unseal}_X$ case.

□

F SIMULATION

We provide the full proof of simulation here, including all supporting lemmas. As outlined in the paper, this proof revolves around a translation relation that generalizes our translation function.

Lemma F.1 (Cast calculus dynamic semantics are deterministic). *If* $\Sigma \triangleright M \mapsto \Sigma_1 \triangleright M_1$ *and* $\Sigma \triangleright M \mapsto \Sigma_2 \triangleright M_2$ *then* $\Sigma_1 = \Sigma_2$ *and* $M_1 = M_2$.

Lemma F.2 (target language dynamic semantics are deterministic). *If* $\Sigma' \triangleright M' \mapsto \Sigma'_1 \triangleright M'_1$ *and* $\Sigma' \triangleright M' \mapsto \Sigma'_2 \triangleright M'_2$ *then* $\Sigma'_1 = \Sigma'_2$ *and* $M'_1 = M'_2$.

Lemma F.3. *If* $\Sigma; \Gamma \vdash M : A; \Gamma'$, *then* $M \rightsquigarrow^{CT} [\![\Sigma; \Gamma \vdash M]\!]$.

Proof. By induction on M. □

The proof of simulation requires knowledge about how substitutions translate. Value substitutions are straightforward due to the definition of our relation. Type substitutions behave differently depending on whether the type variable to be replaced is known.

Lemma F.4. *If* $\Sigma; \Gamma \vdash M \rightsquigarrow^{CT} M' : B; \Gamma'$ *and* $X \in \Gamma$ *and* $\sigma : A \in \Sigma$, *then* $M[\sigma/X] \rightsquigarrow^{CT} M'[[\![\Sigma; \Gamma \vdash A]\!]/X][\sigma/c_X]$.

Proof. By induction on the derivation of $M \rightsquigarrow^{CT} M'$. □

Lemma F.5. *If* $\Sigma; \Gamma \vdash M \rightsquigarrow^{CT} M' : B; \Gamma'$ *and* $X \cong A \in \Gamma, \Gamma'$ *and* $\sigma : A \in \Sigma$, *then* $M[\sigma/X] \rightsquigarrow^{CT} M'[\sigma/c_X]$.

Proof. By induction on the derivation of $M \rightsquigarrow^{CT} M'$. □

Lemma F.6. *If* $\Sigma; \Gamma, x : A', \Gamma' \vdash M \rightsquigarrow^{CT} M' : A; \Gamma''$ *and* $\Sigma; \Gamma \vdash V \rightsquigarrow^{CT} \texttt{ret}\ V' : A'; \cdot$, *then* $\Sigma; \Gamma, \Gamma' \vdash M[V/x] \rightsquigarrow^{CT} M'[V'/x] : A; \Gamma''$.

Proof. By induction on the derivation of $M \rightsquigarrow^{CT} M'$. □

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} x \leftarrow \mathtt{ret}\ V'; M' : A; \Gamma_1}{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M'[V'/x] : A; \Gamma_1}$$

$$\frac{\Sigma;\Gamma \vdash M_1 \rightsquigarrow^{CT} M_1' : A_1; \Gamma_1 \qquad \Sigma;\Gamma,\Gamma_1 \vdash M_2 \rightsquigarrow^{CT} M_2' : A_2; \Gamma_2}{\Sigma;\Gamma \vdash (M_1, M_2) \rightsquigarrow^{CT} x_1 \leftarrow M_1'; x_2 \leftarrow M_2'; \mathtt{ret}\ (x_1, x_2) : A_1 \times A_2; \Gamma_1, \Gamma_2}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : \mathbb{B}; \Gamma' \qquad \Sigma;\Gamma,\Gamma' \vdash M_1 \rightsquigarrow^{CT} M_1' : A; \Gamma'' \qquad \Sigma;\Gamma,\Gamma' \vdash M_2 \rightsquigarrow^{CT} M_2' : A; \Gamma''}{\Sigma;\Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 \rightsquigarrow^{CT} r \leftarrow M'; \mathtt{if}\ r\ \mathtt{then}\ M_1'\ \mathtt{else}\ M_2' : A; \Gamma', \Gamma''}$$

$$\frac{}{\Sigma;\Gamma \vdash \mathtt{true} \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{true} : \mathbb{B}; \cdot} \qquad \frac{}{\Sigma;\Gamma \vdash \mathtt{false} \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{false} : \mathbb{B}; \cdot} \qquad \frac{}{\Sigma;\Gamma \vdash \mathfrak{V} \rightsquigarrow^{CT} \mathfrak{V} : A; \cdot}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A; \Gamma' \qquad \Sigma;\Gamma,\Gamma', x : A \vdash N \rightsquigarrow^{CT} N' : B; \Gamma''}{\Sigma;\Gamma \vdash \mathtt{let}\ x = M; N \rightsquigarrow^{CT} x \leftarrow M'; N' : B; \Gamma', \Gamma''}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A_1 \times A_2; \Gamma' \qquad \Sigma;\Gamma,\Gamma', x : A_1, y : A_2 \vdash N \rightsquigarrow^{CT} N' : B; \Gamma''}{\Sigma;\Gamma \vdash \mathtt{let}\ (x, y)\ =\ M; N \rightsquigarrow^{CT} r \leftarrow M'; \mathtt{let}\ (x, y)\ =\ r; N' : B; \Gamma', \Gamma''}$$

$$\frac{\Sigma;\Gamma, X \cong A \vdash M \rightsquigarrow^{CT} M' : B; \cdot \qquad A' = [\![\Sigma;\Delta \vdash A]\!]}{\Sigma;\Gamma \vdash \mathsf{pack}^{\nu}(X \cong A, M) \rightsquigarrow^{CT} \mathtt{ret}\ \mathsf{pack}(A', \mathsf{thunk}\ \lambda(c_X : \mathrm{Case}\ A').M')\ \mathsf{as}\ [\![\Sigma;\Gamma \vdash \exists^{\nu} X.B]\!] : \exists^{\nu} X.B; \cdot}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : \exists^{\nu} X.A; \Gamma' \qquad \Sigma;\Gamma \vdash N \rightsquigarrow^{CT} N' : B; \Gamma'' \qquad \Gamma, \Gamma', \Gamma'' \vdash B}{\Sigma;\Gamma \vdash \mathsf{unpack}\ (X, x) = M; N \rightsquigarrow^{CT} \begin{array}{l} r \leftarrow M'; \mathsf{unpack}\ (X, f) = r; \\ \mathtt{newcase}_X\ c_X; x \leftarrow (\mathtt{force}\ f)\ c_X; N' \end{array} : B; \Gamma', \Gamma''}$$

$$\frac{\Sigma;\Gamma, X \vdash M \rightsquigarrow^{CT} M' : A; \cdot}{\Sigma;\Gamma \vdash \Lambda X.M \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{thunk}\ \Lambda X.\lambda(c_X : \mathrm{Case}\ X).M' : \forall^{\nu} X.A; \cdot}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : \forall^{\nu} X.B; \Gamma'}{\Sigma;\Gamma \vdash M\{X \cong A\} \rightsquigarrow^{CT} f \leftarrow M'; (\mathtt{force}\ f)[A]\ c_X : B; \Gamma', X \cong A}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : \forall^{\nu} X.B; \Gamma'}{\Sigma;\Gamma \vdash M\{\sigma \cong A\} \rightsquigarrow^{CT} f \leftarrow M'; (\mathtt{force}\ f)[A]\ \sigma : B[\sigma/X]; \Gamma'}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A \to B; \Gamma' \qquad \Sigma;\Gamma,\Gamma' \vdash N \rightsquigarrow^{CT} N' : A; \Gamma''}{\Sigma;\Gamma \vdash M\ N \rightsquigarrow^{CT} f \leftarrow M'; a \leftarrow N'; (\mathtt{force}\ f)\ a : B; \Gamma', \Gamma''}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : B; \cdot}{\Sigma;\Gamma \vdash \lambda x : A.M \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{thunk}\ \lambda x : [\![\Sigma;\Gamma \vdash A]\!].M' : A \to B; \cdot}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A; \Gamma' \qquad X \cong A \in \Gamma, \Gamma'}{\Sigma;\Gamma \vdash \mathtt{seal}_X M \rightsquigarrow^{CT} M' : X; \Gamma'} \qquad \frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : X; \Gamma' \qquad X \cong A \in \Gamma, \Gamma'}{\Sigma;\Gamma \vdash \mathtt{unseal}_X M \rightsquigarrow^{CT} M' : A; \Gamma'}$$

Fig. 31. PolyC$^{\nu}$ translation relation

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : G;\Gamma'}{\Sigma;\Gamma \vdash \mathtt{inj}_G\ M \rightsquigarrow^{CT} x \leftarrow M';\mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ x : ?;\Gamma'}$$

$$\frac{\Sigma;\Gamma, X \cong A \vdash M \rightsquigarrow^{CT} M' : B;\cdot \qquad A' = [\![\Sigma;\Delta \vdash A]\!]}{\Sigma;\Gamma \vdash \mathtt{pack}^{\nu}(X \cong A,[],M) \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{pack}(A',\mathtt{thunk}\ \lambda(c_X : \mathrm{Case}\ A').M')\ \mathtt{as}\ [\![\Sigma;\Gamma \vdash \exists^{\nu}X.B]\!] : \exists^{\nu}X.B;\cdot}$$

$$\frac{\Sigma;\Gamma \vdash \mathtt{pack}^{\nu}(X \cong A,[A_2 \updownarrow_2 \ldots],M) \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{pack}(A',\mathtt{thunk}\ \lambda(c_X : \mathrm{Case}\ A').M')\ \mathtt{as}\ \exists X.A'_1 : \exists^{\nu}X.A^{\sqsubseteq}_{1l};\cdot}{\begin{array}{c}\Sigma;\Gamma \vdash \mathtt{pack}^{\nu}(X \cong A,[A_1 \updownarrow, A_2 \updownarrow_2 \ldots],M) \\ \rightsquigarrow^{CT} \left(\begin{array}{ll}\mathtt{ret}\ \mathtt{pack}(A', & \mathtt{thunk}\ \lambda c_X : \mathrm{Case}\ A'. \\ & [\![\Sigma;\Gamma, X \cong A' \vdash A^{\sqsubseteq}]\!]\updownarrow \\ & \quad [(\mathtt{force}\ \mathtt{thunk}\ \lambda c_X : \mathrm{Case}\ A'.M')\ c_X])\ \mathtt{as}\ [\![\exists^{\nu}X.A^{\sqsubseteq}_{1r}]\!]\end{array}\right) : \exists^{\nu}X.A^{\sqsubseteq}_{1r};\cdot\end{array}}$$

$$\frac{\Sigma;\Gamma \vdash \mathtt{pack}^{\nu}(X \cong A,[A_2 \updownarrow_2 \ldots],M) \rightsquigarrow^{CT} \mathtt{ret}\ \mathtt{pack}(A',\mathtt{thunk}\ \lambda(c_X : \mathrm{Case}\ A').M')\ \mathtt{as}\ \exists X.A'_1 : \exists^{\nu}X.A^{\sqsubseteq}_{1r};\cdot}{\begin{array}{c}\Sigma;\Gamma \vdash \mathtt{pack}^{\nu}(X \cong A,[A_1 \downarrow, A_2 \updownarrow_2 \ldots],M) \\ \rightsquigarrow^{CT} \left(\begin{array}{ll}\mathtt{ret}\ \mathtt{pack}(A', & \mathtt{thunk}\ \lambda c_X : \mathrm{Case}\ A'. \\ & [\![\Sigma;\Gamma, X \cong A' \vdash A^{\sqsubseteq}]\!]\downarrow \\ & \quad [(\mathtt{force}\ \mathtt{thunk}\ \lambda c_X : \mathrm{Case}\ A'.M')\ c_X])\ \mathtt{as}\ [\![\exists^{\nu}X.A^{\sqsubseteq}_{1l}]\!]\end{array}\right) : \exists^{\nu}X.A^{\sqsubseteq}_{1l};\cdot\end{array}}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A^{\sqsubseteq}_{l};\Gamma' \qquad \Sigma;\Gamma \vdash A^{\sqsubseteq}}{\Sigma;\Gamma \vdash \langle A^{\sqsubseteq}\rangle\updownarrow\ M \rightsquigarrow^{CT} [\![\Sigma;\Gamma \vdash A^{\sqsubseteq}]\!]\updownarrow[M'] : A^{\sqsubseteq}_{r};\Gamma'} \qquad \frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : A^{\sqsubseteq}_{r};\Gamma' \qquad \Sigma;\Gamma \vdash A^{\sqsubseteq}}{\Sigma;\Gamma \vdash \langle A^{\sqsubseteq}\rangle\downarrow\ M \rightsquigarrow^{CT} [\![\Sigma;\Gamma \vdash A^{\sqsubseteq}]\!]\downarrow[M'] : A^{\sqsubseteq}_{l};\Gamma'}$$

$$\frac{\Sigma;\Gamma \vdash M \rightsquigarrow^{CT} M' : ?;\Gamma'}{\Sigma;\Gamma \vdash \mathtt{is}(G)?\ M \rightsquigarrow^{CT} \left(\begin{array}{l} r \leftarrow M'; \\ \mathtt{match}\ \mathrm{case}(G)\ \mathtt{with}\ r\{\mathtt{inj}\ y.\mathtt{ret}\ \mathtt{true} \mid \mathtt{ret}\ \mathtt{false}\}\end{array}\right) : \mathbb{B};\Gamma'}$$

Fig. 32. PolyC$^{\nu}$ translation relation (Continued)

We use a restricted form of reduction that only eliminates bind steps in our proof of simulation. This reduction gives us exactly the leeway we need to deal with the possibility of administrative redexes in our translation.

Definition F.7 (bind reduction). We define $\mapsto_b$ to be the least relation on closed terms such that $S[x \leftarrow \mathtt{ret}\ V';M'] \mapsto_b S[M'[V'/x]]$. Note that $\mapsto_b$ is a strict subset of $\mapsto$. We use $\mapsto_b^*$ to mean its reflexive, transitive closure.

Lemma F.8 (bind reduction normalization). *Let $\Sigma;\cdot \vdash M_1 : A$. There exists a unique M_2 such that $M_1 \mapsto_b^* M_2$ and M_2 does not take a bind reduction.*

Proof. By induction on the number of binds in M_1 not under thunks. If $M_1 = S[x \leftarrow \mathtt{ret}\ V;N_1]$ then $M_1 \mapsto_b N_1[V/x]$ and we conclude by the inductive hypothesis for $N_1[V/x]$. Otherwise, M_1 must not take a bind reduction, so $M_2 = M_1$ and we conclude by reflexivity. □

Lemma F.9 (bind reduction confluence). *Let $\Sigma;\cdot \vdash M_1 : A$, $\Sigma;\cdot \vdash M_2 : A$, and $\Sigma;\cdot \vdash M_3 : A$. If $M_1 \mapsto_b^* M_2$, $M_1 \mapsto_b^* M_3$, and M_3 does not take a bind reduction, then $M_2 \mapsto_b^* M_3$.*

Proof. By induction on $M_1 \mapsto_b^* M_2$. If $M_1 = M_2$, then we conclude since $M_1 \mapsto_b^* M_3$. Otherwise, by the definition of $\mapsto_b^*$, we have $M_1 \mapsto_b N_1 \mapsto_b^* M_2$. Since M_1 takes a bind reduction, $M_1 \neq M_3$ and so we must have $M_1 \mapsto_b N_2 \mapsto_b^* M_3$. Furthermore, by Lemma F.1 (deterministic semantics), we have $N_1 = N_2$. We then conclude by the inductive hypothesis for N_2, M_2, M_3. □

At the core of our simulation argument is the guarantee that, up to some bind reductions, any related terms have a related structure that follows the form of the translation function.

Lemma F.10 (canonical forms of the translation relation). *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : B$, *then either*

- $M = \Lambda X.N_1$ *and* $M' = \mathtt{ret}\ \mathtt{thunk}\ \Lambda X.\lambda(c_X : \mathit{Case}\ X).N_1'$
- $M = N_1\{\sigma \cong A\}$ *and* $(f \leftarrow N_1'; (\mathtt{force}\ f)[A]\ \sigma)[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{hide}\ X \cong A; N_1$ *and* $M' = \mathtt{newcase}_A\ c_X; N_1'$
- $M = (N_1, N_2)$ *and* $(x_1 \leftarrow N_1'; x_2 \leftarrow N_2'; \mathtt{ret}\ (x_1, x_2))[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{let}\ (x, y) = N_1; N_2$ *and* $(r \leftarrow N_1'; \mathtt{let}\ (x, y) = r; N_2')[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{true}$ *and* $M' = \mathtt{ret}\ \mathtt{true}$
- $M = \mathtt{false}$ *and* $M' = \mathtt{ret}\ \mathtt{false}$
- $M = \mathtt{if}\ N_1\ \mathtt{then}\ N_2\ \mathtt{else}\ N_3$ *and* $(r \leftarrow N_1'; \mathtt{if}\ r\ \mathtt{then}\ N_2'\ \mathtt{else}\ N_3')[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{let}\ x = N_1; N_2$ *and* $(x \leftarrow N_1'; N_2')[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathfrak{U}_A$ *and* $M' = \mathfrak{U}$
- $M = \mathtt{inj}_G\ N_1$ *and* $(x \leftarrow N_1'; \mathtt{inj}_{\mathit{case}(G)}\ x)[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \langle A^{\sqsubseteq} \rangle\!\updownarrow N_1$ *and* $[\![\Sigma; \cdot \vdash A^{\sqsubseteq}]\!]\!\updownarrow[N_1'][\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{is}(G)?\ N_1$ *and* $(r \leftarrow M'; \mathtt{match}\ \mathit{case}(G)\ \mathtt{with}\ r\{\mathtt{inj}\ y.\mathtt{ret}\ \mathtt{true} \mid \mathtt{ret}\ \mathtt{false}\})[\gamma_p] \mapsto_b^* M'[\gamma_p]$
- $M = \mathtt{seal}_\sigma N_1$ *and* $N_1' \mapsto_b^* M'$
- $M = \mathtt{unseal}_\sigma N_1$ *and* $N_1' \mapsto_b^* M'$
- $M = \mathtt{pack}^\nu(X \cong A, N_1)$ *and* $M' = \mathtt{ret}\ \mathtt{pack}(A, \mathtt{thunk}\ \lambda(c_X : \mathit{Case}\ A).N_1')\ \mathtt{as}\ [\![B]\!]$
- $M = \mathtt{pack}^\nu(X \cong A', [A^{\sqsubseteq} \updownarrow \ldots], N_1)$ *and* $M' = \mathtt{ret}\ \mathtt{pack}(A, \mathtt{thunk}\ \lambda(c_X : \mathit{Case}\ A).M_n')\ \mathtt{as}\ [\![B]\!]$ *where*

$$\begin{array}{rcl} M_0' & = & N_1' \\ M_{i+1}' & = & [\![\Sigma; \Delta \vdash A_{i+1}^{\sqsubseteq}]\!]\!\updownarrow_{i+1}[\mathtt{force}\ (\mathtt{thunk}\ (\lambda c_X : \mathit{Case}\ A'.[\![M_i]\!]))\ c_X] \end{array}$$

- $M = \mathtt{unpack}\ (X, x) = N_1; N_2$ *and* $M_{unp} \mapsto_b^* M'$ *where*

$$\begin{array}{rcl} M_{unp} = & r \leftarrow N_1'; \mathtt{unpack}\ (X, f) = r; \\ & \mathtt{newcase}_X\ c_X; x \leftarrow (\mathtt{force}\ f)\ c_X; N_2' \end{array}$$

for some $N_1, \ldots, N_n$ *and* $N_1', \ldots, N_n'$ *where* $N_i \rightsquigarrow^{CT} N_i'$.

Proof. By induction on the derivation of $M \rightsquigarrow^{CT} M'$. If the derivation ends in a congruence from the translation, the conclusion is immediate by reflexivity. We then have only the bind-reduction rule to consider. In this case, we have $M' = M''[V'/x]$ where $M \rightsquigarrow^{CT} x \leftarrow \mathtt{ret}\ V'; M''$. By the inductive hypothesis, we have the desired property for M and $x \leftarrow \mathtt{ret}\ V'; M''$. Then, since $(x \leftarrow \mathtt{ret}\ V'; M'')[\gamma_p] \mapsto_b M''[V'/x][\gamma_p] = M'[\gamma_p]$, we have what we need to show by transitivity. □

Lemma F.11. *Let* $\Sigma; \cdot \vdash V : A$. *If* $V \rightsquigarrow^{CT} M'$, *then* $M'[\gamma_p] \mapsto_b^* \mathtt{ret}\ V'[\gamma_p]$ *and* $V \rightsquigarrow^{CT} \mathtt{ret}\ V'$ *for some* V'.

$$\frac{}{\Sigma;\Gamma \mid \bullet : A \vdash [] \rightsquigarrow^{CT} \bullet : A;\cdot}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A \vdash E_1 \rightsquigarrow^{CT} S_1' : A_1;\Gamma_1 \qquad \Sigma;\Gamma,\Gamma_1 \vdash M_2 \rightsquigarrow^{CT} M_2' : A_2;\Gamma_2}{\Sigma;\Gamma \mid \bullet : A \vdash (E_1, M_2) \rightsquigarrow^{CT} x_1 \leftarrow S_1'; x_2 \leftarrow M_2'; \mathtt{ret}\ (x_1,x_2) : A_1 \times A_2;\Gamma_1,\Gamma_2}$$

$$\frac{\Sigma;\Gamma \vdash V_1 \rightsquigarrow^{CT} \mathtt{ret}\ V_1' : A_1;\Gamma_1 \qquad \Sigma;\Gamma,\Gamma_1 \mid \bullet : A \vdash E_2 \rightsquigarrow^{CT} S_2' : A_2;\Gamma_2}{\Sigma;\Gamma \mid \bullet : A \vdash (V_1, E_2) \rightsquigarrow^{CT} x_2 \leftarrow S_2'; \mathtt{ret}\ (V_1', x_2) : A_1 \times A_2;\Gamma_1,\Gamma_2}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : \mathbb{B};\Gamma' \qquad \Sigma;\Gamma,\Gamma' \vdash M_1 \rightsquigarrow^{CT} M_1' : A;\Gamma'' \qquad \Sigma;\Gamma,\Gamma' \vdash M_2 \rightsquigarrow^{CT} M_2' : A;\Gamma''}{\Sigma;\Gamma \mid \bullet : A' \vdash \mathtt{if}\ E\ \mathtt{then}\ M_1\ \mathtt{else}\ M_2 \rightsquigarrow^{CT} r \leftarrow S'; \mathtt{if}\ r\ \mathtt{then}\ M_1'\ \mathtt{else}\ M_2' : A;\Gamma',\Gamma''}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : A;\Gamma' \qquad \Sigma;\Gamma,\Gamma',x : A \vdash N \rightsquigarrow^{CT} N' : B;\Gamma''}{\Sigma;\Gamma \mid \bullet : A' \vdash \mathtt{let}\ x = E; N \rightsquigarrow^{CT} x \leftarrow S'; N' : B;\Gamma',\Gamma''}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A \vdash E \rightsquigarrow^{CT} S' : A_1 \times A_2;\Gamma' \qquad \Sigma;\Gamma,\Gamma',x : A_1, y : A_2 \vdash N \rightsquigarrow^{CT} N' : B;\Gamma''}{\Sigma;\Gamma \mid \bullet : A \vdash \mathtt{let}\ (x,y) = E; N \rightsquigarrow^{CT} r \leftarrow S'; \mathtt{let}\ (x,y) = r; N' : B;\Gamma',\Gamma''}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A \vdash E \rightsquigarrow^{CT} S' : \exists^{\nu} X.A;\Gamma' \qquad \Sigma;\Gamma \vdash N \rightsquigarrow^{CT} N' : B;\Gamma'' \qquad \Gamma,\Gamma',\Gamma'' \vdash B}{\Sigma;\Gamma \mid \bullet : A \vdash \mathtt{unpack}\ (X,x) = E; N \rightsquigarrow^{CT} \left(\begin{array}{l} r \leftarrow S'; \\ \mathtt{unpack}\ (X,f) = r; \\ \mathtt{newcase}_X\ c_X; \\ x \leftarrow (\mathtt{force}\ f)\ c_X; \\ N' \end{array}\right) : B;\Gamma',\Gamma''}$$

Fig. 33. PolyC$^{\nu}$ evaluation context translation relation

Proof. By induction on the derivation of $V \rightsquigarrow^{CT} M'$. All cases are immediate since $M' = \mathtt{ret}\ V'$ except the following:

- $(V_1, V_2) \rightsquigarrow^{CT} x_1 \leftarrow N_1'; x_2 \leftarrow N_2'; \mathtt{ret}\ (x_1,x_2)$ where $V_1 \rightsquigarrow^{CT} N_1'$ and $V_2 \rightsquigarrow^{CT} N_2'$: By the inductive hypothesis for each of these sub-derivations, we have $N_i'[\gamma_p] \mapsto_b^* \mathtt{ret}\ V_i'[\gamma_p]$ and $V_i \rightsquigarrow^{CT} \mathtt{ret}\ V_i'$. Then, by definition, we have $M'[\gamma_p] \mapsto_b^* \mathtt{ret}\ (V_1,V_2)[\gamma_p]$ and $(V_1,V_2) \rightsquigarrow^{CT} \mathtt{ret}\ (V_1,V_2)$ as we were required to show.

□

Since our dynamic semantics for PolyC$^{\nu}$ uses evaluation contexts, we need to be able to describe their transformation into CBPV$_{\mathrm{OSum}}$.

Lemma F.12 (Translation context decomposition). *If* $\Sigma;\cdot \vdash M_1 : A$ *and* $\Sigma;\cdot \vdash E[M_1] \rightsquigarrow^{CT} M' : B$, *then* $S[M_1'][\gamma_p] \mapsto_b^* M'[\gamma_p]$ *for some* S *and* M_1' *such that* $\Sigma;\cdot \vdash M_1 \rightsquigarrow^{CT} M_1' : A$ *and* $E \rightsquigarrow^{CT} S$.

Proof. By induction on the derivation of $E[M_1] \rightsquigarrow^{CT} M'$. All cases are straightforward except the bind rule. That case proceeds as follows: we have $E[M_1] \rightsquigarrow^{CT} x \leftarrow \mathtt{ret}\ V_1'; M''$ and, from the inductive hypothesis, some S, M_1' such that $S[M_1'][\gamma_p] \mapsto_b^* (x \leftarrow \mathtt{ret}\ V_1'; M'')[\gamma_p]$, $\Sigma;\cdot \vdash M_1 \rightsquigarrow^{CT} M_1' : A$, and $E \rightsquigarrow^{CT} S$. It then suffices to show that $S[M_1'][\gamma_p] \mapsto_b^* M''[V_1'/x][\gamma_p]$, which we have by definition. □

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : \forall^{v} X.B;\Gamma'}{\Sigma;\Gamma \mid \bullet : A' \vdash E\{X \cong A\} \rightsquigarrow^{CT} f \leftarrow S';(\texttt{force}\ f)[A]\ c_X : B;\Gamma', X \cong A}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : \forall^{v} X.B;\Gamma'}{\Sigma;\Gamma \mid \bullet : A' \vdash E\{\sigma \cong A\} \rightsquigarrow^{CT} f \leftarrow S';(\texttt{force}\ f)[A]\ \sigma : B[\sigma/X];\Gamma'}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : A \to B;\Gamma' \qquad \Sigma;\Gamma,\Gamma' \vdash N \rightsquigarrow^{CT} N' : A;\Gamma''}{\Sigma;\Gamma \mid \bullet : A' \vdash E\ N \rightsquigarrow^{CT} f \leftarrow S'; a \leftarrow N';(\texttt{force}\ f)\ a : B;\Gamma',\Gamma''}$$

$$\frac{\Sigma;\Gamma \vdash V \rightsquigarrow^{CT} \texttt{ret}\ V' : A \to B;\Gamma' \qquad \Sigma;\Gamma,\Gamma' \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : A;\Gamma''}{\Sigma;\Gamma \mid \bullet : A' \vdash V\ E \rightsquigarrow^{CT} a \leftarrow S';(\texttt{force}\ V')\ a : B;\Gamma',\Gamma''}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : A;\Gamma' \qquad X \cong A \in \Gamma,\Gamma'}{\Sigma;\Gamma \mid \bullet : A' \vdash \texttt{seal}_X E \rightsquigarrow^{CT} S' : X;\Gamma'} \qquad \frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : X;\Gamma' \qquad X \cong A \in \Gamma,\Gamma'}{\Sigma;\Gamma \mid \bullet : A' \vdash \texttt{unseal}_X E \rightsquigarrow^{CT} S' : A;\Gamma'}$$

$$\frac{\Sigma;\Gamma \mid \bullet : A' \vdash E \rightsquigarrow^{CT} S' : G;\Gamma'}{\Sigma;\Gamma \mid \bullet : A' \vdash \texttt{inj}_G\ E \rightsquigarrow^{CT} x \leftarrow S';\texttt{inj}_{\text{case}(G)}\ x : ?;\Gamma'}$$

$$\frac{\Sigma;\Gamma \mid \bullet : B \vdash E \rightsquigarrow^{CT} S' : A^{\sqsubseteq}_l;\Gamma' \qquad \Sigma;\Gamma \vdash A^{\sqsubseteq}}{\Sigma;\Gamma \mid \bullet : B \vdash \langle A^{\sqsubseteq}\rangle\!\uparrow E \rightsquigarrow^{CT} [\![\Sigma;\Gamma \vdash A^{\sqsubseteq}]\!]\!\uparrow [S'] : A^{\sqsubseteq}_r;\Gamma'}$$

$$\frac{\Sigma;\Gamma \mid \bullet : B \vdash E \rightsquigarrow^{CT} S' : A^{\sqsubseteq}_r;\Gamma' \qquad \Sigma;\Gamma \vdash A^{\sqsubseteq}}{\Sigma;\Gamma \mid \bullet : B \vdash \langle A^{\sqsubseteq}\rangle\!\downarrow E \rightsquigarrow^{CT} [\![\Sigma;\Gamma \vdash A^{\sqsubseteq}]\!]\!\downarrow [S'] : A^{\sqsubseteq}_l;\Gamma'}$$

$$\frac{\Sigma;\Gamma \mid \bullet : B \vdash E \rightsquigarrow^{CT} S' : ?;\Gamma'}{\Sigma;\Gamma \mid \bullet : B \vdash \texttt{is}(G)?\ E \rightsquigarrow^{CT} \left(\begin{array}{l} r \leftarrow S'; \\ \texttt{match case}(G)\ \texttt{with}\ r\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\} \end{array}\right) : \mathbb{B};\Gamma'}$$

Fig. 34. PolyC^{v} calculus evaluation context translation relation (Continued)

Lemma F.13 (Translation context plug). *If* $\Sigma;\cdot \vdash M_1 \rightsquigarrow^{CT} M_1' : A$ *and* $\Sigma;\cdot \mid \bullet : A \vdash E \rightsquigarrow^{CT} S : B$, *then* $\Sigma;\cdot \vdash E[M_1] \rightsquigarrow^{CT} S[M_1'] : B$.

Proof. By induction on the derivation of $E \rightsquigarrow^{CT} S$. □

Lemma F.14 (Translation stack decomposition). *If* $\Sigma;\cdot \vdash M \rightsquigarrow^{CT} S[M_1'] : A$ *then* $M = E[M_1]$ *for some* E *and* M_1 *such that* $E \rightsquigarrow^{CT} S$ *and* $M_1 \rightsquigarrow^{CT} M_1'$.

Proof. By induction on the derivation of $M \rightsquigarrow^{CT} S[M_1']$. □

Lemma F.15 (Bind reduction preserves translation). *If* $\Sigma;\cdot \vdash M \rightsquigarrow^{CT} M_1' : A$ *and* $M_1'[\gamma_p] \mapsto_b M_2'[\gamma_p]$ *then* $\Sigma;\cdot \vdash M \rightsquigarrow^{CT} M_2' : A$

Proof. By the definition of $\mapsto_b$, we have that $M_1' = S[x \leftarrow \texttt{ret}\ V'; M_3']$ and $M_2' = S[M_3'[V'/x]]$ for some S, V', M_3'. By Lemma F.14 (stack decomposition), we have $M = E[M_3]$, $E \rightsquigarrow^{CT} S$ and

$M_3 \rightsquigarrow^{CT} x \leftarrow \texttt{ret}\ V'; M_3'$. We then apply the bind reduction rule, so we have $M_3 \rightsquigarrow^{CT} M_3'[V'/x]$ and conclude by Lemma F.13 (translation context plug). □

We would like all PolyC$^{\nu}$ terms to make progress whenever their corresponding CBPV$_{\text{OSum}}$ translation evaluates. However, this is not always the case since some PolyC$^{\nu}$ terms step to terms with identical translations. However, the number of such steps is limited by the syntactic size of the term, defined below.

Definition F.16 (Type/Term size).

$$\begin{array}{lcl} |\texttt{let}\ x = M; N| & = & 1 + |N[M/x]| \\ |\texttt{tag}_G(A^{\sqsubseteq})| & = & 2 + |A^{\sqsubseteq}| \\ |\texttt{pack}^{\nu}(X \cong A', M)| & = & 2 + |M| \\ |C(\vec{A^{\sqsubseteq}}, \vec{M})| & = & 1 + \sum_{M \in \vec{M}} |M| + \sum_{A^{\sqsubseteq} \in \vec{A^{\sqsubseteq}}} |A^{\sqsubseteq}| \text{ otherwise} \end{array}$$

where C denotes any syntactic term or precision judgment constructor.

Theorem F.17 (Simulation). *If* $\Sigma; \cdot \vdash M_1 : A; \cdot$ *and* $\Sigma; \cdot \vdash M_1 \rightsquigarrow^{CT} M_1' : A; \cdot$ *and* $\Sigma \triangleright M_1 \mapsto \Sigma' \triangleright M_2$, *then* $\Sigma'; \cdot \vdash M_2 \rightsquigarrow^{CT} M_2' : A; \cdot$ *and either* $\Sigma_p, [\![\Sigma]\!] \triangleright M_1'[\gamma_p] \mapsto^{+} \Sigma_p, [\![\Sigma']\!] \triangleright M_2'[\gamma_p]$ *or* $\Sigma_p, [\![\Sigma]\!] \triangleright M_1'[\gamma_p] \mapsto^{*} \Sigma_p, [\![\Sigma']\!] \triangleright M_2'[\gamma_p]$ *and* $|M_1| > |M_2|$ *for some* M_2'.

Proof. Note that, by unique decomposition, we have some context E and redex M_3 such that $M_1 = E[M_3]$ and either $M_3 = \mathcal{U}_B$ or $M_3 \mapsto M_4$ for some M_4. By Lemma F.12 (translation context decomposition), we have some S, M_3' such that $S[M_3'][\gamma_p] \mapsto_b^* M_1'[\gamma_p]$ where $E \rightsquigarrow^{CT} S$ and $M_3 \rightsquigarrow^{CT} M_3'$. We proceed by cases on $\Sigma \triangleright M_1 \mapsto \Sigma' \triangleright M_2$. In each case, we first use Lemma F.10 (canonical forms of the translation) to determine the form of M_3'.

- Error:

  $$E[\mathcal{U}_B] \mapsto \mathcal{U}_{type(\Sigma; \cdot \vdash E[\mathcal{U}_B])} \text{ where } E \neq []$$

  We have $M_3' = \mathcal{U}$ and so $S[M_3'][\gamma_p] \mapsto \mathcal{U}$. Since $\Sigma; \cdot \vdash \mathcal{U}_{type(\Sigma; \cdot \vdash M_1)} \rightsquigarrow^{CT} \mathcal{U} : type(\Sigma; \cdot \vdash M_1)$, we have what we were required to show.
- Instantiation:

  $$E[(\Lambda X.N_1)\{\sigma \cong B\}] \mapsto E[N_1[\sigma/X]]$$

  Let $B' = [\![\Sigma; \cdot \vdash B]\!]$. We have

  $$(f \leftarrow \texttt{ret thunk}\ \Lambda X.\lambda(c_X : \text{Case } X).N_1'; (\texttt{force}\ f)[B']\ \sigma)[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$$

  for some N_1' such that $N_1 \rightsquigarrow^{CT} N_1'$. Then, by Lemma F.9 (bind confluence), we have $M_3'[\gamma_p] \mapsto_b^* ((\texttt{force thunk}\ \Lambda X.\lambda(c_X : \text{Case } X).N_1')[B']\ \sigma)[\gamma_p]$ and so by the dynamic semantics, we have the following:

  $$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[(\texttt{force thunk}\ \Lambda X.\lambda(c_X : \text{Case } X).N_1')[B']\ \sigma][\gamma_p] \\ \mapsto & S[(\Lambda X.\lambda(c_X : \text{Case } X).N_1')[B']\ \sigma][\gamma_p] \\ \mapsto & S[(\lambda(c_X : \text{Case } B').N_1'[B'/X])\ \sigma][\gamma_p] \\ \mapsto & S[N_1'[B'/X][\sigma/c_X]][\gamma_p] \end{array}$$

  Furthermore, by Lemma F.4, since $\sigma : B \in \Sigma$ and $\Sigma; X \vdash N_1 \rightsquigarrow^{CT} N_1' : A_1$, we have that $N_1[\sigma/X] \rightsquigarrow^{CT} N_1'[B'/X][\sigma/c_X]$. Then, by Lemma F.13 (translation context plug), we have what we were required to show.

- Hiding:

$$\Sigma \triangleright E[\texttt{hide } X \cong B; N] \mapsto \Sigma, \sigma : \text{Case } B \triangleright E[N[\sigma/X]]$$

We have $M_3' = \texttt{newcase}_B\ c_X; N'$ for some N' such that $N \rightsquigarrow^{CT} N'$. We then have the following by the operational semantics:

$$\begin{array}{rrl} & [\![\Sigma]\!]\triangleright & S[\texttt{newcase}_B\ c_X; N'][\gamma_p] \\ \mapsto & [\![\Sigma]\!], \sigma : \text{Case } B\triangleright & S[N'[\sigma/c_X]][\gamma_p] \end{array}$$

Then by Lemma F.5, since $\Sigma; \cdot \vdash N \rightsquigarrow^{CT} N' : A, X \cong B$, we have $N[\sigma/X] \rightsquigarrow^{CT} N'[\sigma/c_X]$, so we conclude by Lemma F.13 (translation context plug).

- Pack:

$$E[\texttt{pack}^\nu(X \cong A_1, N_1)] \mapsto E[\texttt{pack}^\nu(X \cong A_1, [], N_1)]$$

Note that $\texttt{pack}^\nu(X \cong A_1, N_1) \rightsquigarrow^{CT} M_3'$ iff $\texttt{pack}^\nu(X \cong A_1, [], N_1) \rightsquigarrow^{CT} M_3'$, so by Lemma F.13 (translation context plug), we may choose $M_2' = M_1'$. Then, it suffices to show that $|M_1| > |M_2|$. This holds since $|\texttt{pack}^\nu(X \cong A_1, N_1)| = 2 + |N_1| > 1 + |N_1| = |\texttt{pack}^\nu(X \cong A_1, [], N_1)|$.

- Unpack:

$$\begin{array}{rrl} & \Sigma\triangleright & E[\texttt{unpack } (X, x) = \texttt{pack}^\nu(X \cong A_1, [A^{\sqsubseteq} \updownarrow \ldots], N_1); N_2] \\ \mapsto & \Sigma, \sigma : \text{Case } A_1\triangleright & E[\texttt{let } x = \langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow \ldots N_1[\sigma/X]; N_2[\sigma/X]] \end{array}$$

Let $A' = [\![\Sigma; \cdot \vdash \exists^\nu X.B]\!]$ and $A_1' = [\![\Sigma; \cdot \vdash A_1]\!]$. We have

$$\begin{array}{rrl} & & r \leftarrow \texttt{ret pack}(A_1', \texttt{thunk } \lambda(c_X : \text{Case } A_1').N_{cst}'[\gamma_p])\ \texttt{as } A'; \\ & & \texttt{unpack } (X, f) = r; \\ & & \texttt{newcase}_X\ c_X; \\ & & x \leftarrow (\texttt{force } f)\ c_X; \\ & & N_2'[\gamma_p] \\ \mapsto_b^* & & M_3'[\gamma_p] \end{array}$$

where $N_{cst}' = [\![\Sigma; X \cong A_1 \vdash A^{\sqsubseteq}]\!]\updownarrow[\texttt{force } (\texttt{thunk } (\lambda c_X : \text{Case } A'.(\ldots N_1' \ldots)))\ c_X]$ and $N_i \rightsquigarrow^{CT} N_i'$. We then have the following by Lemma F.9 (bind confluence) and the operational semantics:

$$\begin{array}{rrl} & & M_3'[\gamma_p] \\ \mapsto_b^* & & \texttt{unpack } (X, f) = \texttt{pack}(A_1', \texttt{thunk } \lambda(c_X : \text{Case } A_1').N_{cst}'[\gamma_p])\ \texttt{as } A'; \\ & & \texttt{newcase}_X\ c_X; \\ & & x \leftarrow (\texttt{force } f)\ c_X; \\ & & N_2'[\gamma_p] \\ \mapsto & [\![\Sigma]\!]\triangleright & \texttt{newcase}_{A_1'}\ c_X; \\ & & x \leftarrow (\texttt{force thunk } \lambda(c_X : \text{Case } A_1').N_{cst}'[\gamma_p])\ c_X; \\ & & N_2'[\gamma_p][A_1'/X][\texttt{thunk } \lambda(c_X : \text{Case } A_1').N_{cst}'[\gamma_p]/f] \\ = & [\![\Sigma]\!]\triangleright & \texttt{newcase}_{A_1'}\ c_X; \\ & & x \leftarrow (\texttt{force thunk } \lambda(c_X : \text{Case } A_1').N_{cst}'[\gamma_p])\ c_X; \\ & & N_2'[\gamma_p][A_1'/X] \\ \mapsto^* & [\![\Sigma]\!], \sigma : A_1'\triangleright & x \leftarrow N_{cst}'[\gamma_p][\sigma/c_X]; \\ & & N_2'[\gamma_p][A_1'/X][\sigma/c_X] \\ \mapsto^* & [\![\Sigma]\!], \sigma : A_1'\triangleright & x \leftarrow [\![\Sigma; X \cong A_1 \vdash A^{\sqsubseteq}]\!]\updownarrow[\ldots N_1' \ldots][\gamma_p][\sigma/c_X]; \\ & & N_2'[\gamma_p][A_1'/X][\sigma/c_X] \end{array}$$

Note that, since $\Sigma; X \cong A_1' \vdash N_1 \rightsquigarrow^{CT} N_1'$, we have $\Sigma; X \cong A_1' \vdash \langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow \ldots N_1 \rightsquigarrow^{CT} [\![\Sigma; X \cong A_1 \vdash A^{\sqsubseteq}]\!]\updownarrow[\ldots N_1' \ldots]$ by the definition of the translation relation for casts. Furthermore, by Lemma F.5, since $\sigma : A_1 \in \Sigma$, we have $\langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow \ldots N_1[\sigma/X] \rightsquigarrow^{CT} [\![\Sigma; X \cong A_1 \vdash$

$A^{\sqsubseteq}[\![\Sigma; X \cong A_1 \vdash A^{\sqsubseteq}]\!]\updownarrow[\dots N_1' \dots][\sigma/c_X]$ and by Lemma F.4, since $\sigma : A_1 \in \Sigma$ and $\Sigma; X \vdash N_2 \rightsquigarrow^{CT} N_2' : A_2$, we have $N_2[\sigma/X] \rightsquigarrow^{CT} N_2'[A_1'/X][\sigma/c_X]$. Thus, we have

$$\texttt{let}\ x = \langle A^{\sqsubseteq}[\sigma/X]\rangle\updownarrow \dots N_1[\sigma/X]; N_2[\sigma/X] \rightsquigarrow^{CT} x \leftarrow [\![\Sigma; X \cong A_1 \vdash A^{\sqsubseteq}]\!]\updownarrow[\dots N_1' \dots][\sigma/c_X]; N_2'[A_1'/X][\sigma/c_X]$$

so we conclude by Lemma F.13 (translation context plug).

- function application

$$E[(\lambda x : A.N_1)\ V] \mapsto E[N_1[V/x]]$$

We have $(f \leftarrow \texttt{ret thunk}\ \lambda x : [\![\Sigma; \cdot \vdash A]\!].N_1'; a \leftarrow N_2'; \texttt{force}\ f\ a)[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $N_i \rightsquigarrow^{CT} N_i'$. By Lemma F.11 (value translation), we have some V' such that $N_2'[\gamma_p] \mapsto_b^* \texttt{ret}\ V'[\gamma_p]$ and $V \rightsquigarrow^{CT} \texttt{ret}\ V'$. Then, by Lemma F.9 (bind confluence), we have $M_3'[\gamma_p] \mapsto_b^* (\texttt{force (thunk}\ \lambda x : [\![\Sigma; \cdot \vdash A]\!].N_1')\ V')[\gamma_p]$. Then $M_3'[\gamma_p] \mapsto^+ N_1'[V'/x][\gamma_p]$ and by Lemma F.6 (value substitution translation), we have $\Sigma; \cdot \vdash N_1[V/x] \rightsquigarrow^{CT} N_1'[V'/x] : B$. Thus, we conclude by Lemma F.13 (translation context plug).

- If true:

$$E[\texttt{if true then}\ N_1\ \texttt{else}\ N_2] \mapsto E[N_1]$$

We have $(r \leftarrow \texttt{ret true}; \texttt{if}\ r\ \texttt{then}\ N_1'\ \texttt{else}\ N_2')[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$. where $N_i \rightsquigarrow^{CT} N_i'$. We then have the following by Lemma F.8 (bind normalization) and the operational semantics:

$$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[\texttt{if true then}\ N_1'\ \texttt{else}\ N_2'][\gamma_p] \\ \mapsto & S[N_1'][\gamma_p] \end{array}$$

We then conclude by Lemma F.13 (translation context plug) since $N_1 \rightsquigarrow^{CT} N_1'$.

- If false:

$$\texttt{if false then}\ N_1\ \texttt{else}\ N_2 \mapsto M_2$$

We have $(r \leftarrow \texttt{ret false}; \texttt{if}\ r\ \texttt{then}\ N_1'\ \texttt{else}\ N_2')[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$. where $N_i \rightsquigarrow^{CT} N_i'$. We then have the following by Lemma F.8 (bind normalization) and the operational semantics:

$$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[\texttt{if false then}\ N_1'\ \texttt{else}\ N_2'][\gamma_p] \\ \mapsto & S[N_2'][\gamma_p] \end{array}$$

We then conclude by Lemma F.13 (translation context plug) since $N_2 \rightsquigarrow^{CT} N_2'$.

- Pair elimination:

$$\texttt{let}\ (x, y) = (V_1, V_2); N \mapsto N[V_1/x][V_2/y]$$

We have $(r \leftarrow M''; \texttt{let}\ (x, y) = r; N')[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $(V_1, V_2) \rightsquigarrow^{CT} M''$ and $N \rightsquigarrow^{CT} N'$. Thus, by Lemma F.11 (value translation), we further have $M''[\gamma_p] \mapsto_b^* \texttt{ret}\ V'[\gamma_p]$ and $(V_1, V_2) \rightsquigarrow^{CT} \texttt{ret}\ V'$ for some V'. Note that by the type of V', we have $V' = (V_1', V_2')$ for some V_1', V_2'.

We then have the following by Lemma F.9 (bind confluence) and the operational semantics:

$$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[\texttt{let}\ (x, y) = (V_1', V_2'); N'][\gamma_p] \\ \mapsto & S[N'[V_1'/x][V_2'/y]][\gamma_p] \end{array}$$

By Lemma F.6, since $V_i \rightsquigarrow^{CT} \texttt{ret}\ V_i'$, we have $N[V_1/x][V_2/y] \rightsquigarrow^{CT} N'[V_1'/x][V_2'/y]$. We then conclude by Lemma F.13 (translation context plug).

- Let:

$$\texttt{let } x = V; N_2 \mapsto N_2[V/x]$$

We have $(x \leftarrow N_1'; N_2')[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V \rightsquigarrow^{CT} N_1'$ and $N_2 \rightsquigarrow^{CT} N_2'$. Then, by Lemma F.11 (value translation reduction), we have $N_1'[\gamma_p] \mapsto_b^* \texttt{ret } V'[\gamma_p]$ for some V' such that $V \rightsquigarrow^{CT} \texttt{ret } V'$. Thus, we have the following reduction:

$$S[x \leftarrow N_1'; N_2'][\gamma_p] \mapsto_b^* S[N_2'[V'/x]][\gamma_p]$$

By Lemma F.8 (bind normalization) there exists a unique M_4' such that

$$S[x \leftarrow N_1'; N_2'][\gamma_p] \mapsto_b^* M_4'[\gamma_p]$$

and $M_4'[\gamma_p]$ does not take a bind reduction. Then, by Lemma F.9 (bind reduction confluence), we have $M_1'[\gamma_p] \mapsto_b^* M_4'[\gamma_p]$ and $S[N_2'[V'/x]][\gamma_p] \mapsto_b^* M_4'[\gamma_p]$.
Note that, by Lemma F.6 (value substitution translation), we have $N_2[V/x] \rightsquigarrow^{CT} N_2'[V'/x]$ and thus by Lemma F.13 (translation context plug), we have $E[N_2[V/x]] \rightsquigarrow^{CT} S[N_2'[V'/x]]$. Then, by Lemma F.15 (bind reduction preserves translation), we have $E[N_2[V/x]] \rightsquigarrow^{CT} M_4'$. It finally suffices to show that $|M_1| > |M_2|$, which we have since by definition $|M_3| = 1 + |N_2[V/x]|$.

- Unseal:

$$E[\texttt{unseal}_\sigma \texttt{seal}_\sigma V] \mapsto E[V]$$

We have $N'[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V \rightsquigarrow^{CT} N'$ and so $S[N'][\gamma_p] \mapsto_b^* S[M_3'][\gamma_p]$. By Lemma F.8 (bind normalization) there exists a unique M_4' such that $S[N'][\gamma_p] \mapsto_b^* M_4'[\gamma_p]$ and $M_4'[\gamma_p]$ does not take a bind reduction. Then, by Lemma F.9 (bind confluence), we have $S[M_3'][\gamma_p] \mapsto_b^* M_4'[\gamma_p]$. Since, by Lemmas F.13 (translation context plug) and F.15 (bind reduction preserves translation), we have $E[V] \rightsquigarrow^{CT} M_4'$, it suffices to show that $|M_1| > |M_2|$. This holds since by definition $|M_3| = 1 + 1 + |V|$.

- Atomic cast:

$$E[\langle A^{\sqsubseteq} \rangle\!\updownarrow V] \mapsto E[V] \text{ where } A \in \{\mathbb{B}, \alpha, ?\}$$

The reasoning for this case is analogous to the prior case.

- Pair cast:

$$E[\langle A_1^{\sqsubseteq} \times A_2^{\sqsubseteq} \rangle\!\updownarrow (V_1, V_2)] \mapsto E[(\langle A_1^{\sqsubseteq} \rangle\!\updownarrow V_1, \langle A_2^{\sqsubseteq} \rangle\!\updownarrow V_2)]$$

We have $[\![\Sigma; \cdot \vdash A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}]\!]\updownarrow[x_1 \leftarrow N_1'; x_2 \leftarrow N_2'; \texttt{ret } (x_1, x_2)][\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V_i \rightsquigarrow^{CT} N_i'$. Then, by Lemma F.11 (value translation reduction), we have $N_i'[\gamma_p] \mapsto_b^* \texttt{ret } V_i'[\gamma_p]$ for some V_1', V_2' such that $V_i \rightsquigarrow^{CT} \texttt{ret } V_i'$. Thus, we have the following reductions:

$$\begin{array}{ll} & [\![\Sigma; \cdot \vdash A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}]\!]\updownarrow[x_1 \leftarrow N_1'; x_2 \leftarrow N_2'; \texttt{ret } (x_1, x_2)][\gamma_p] \\ \mapsto_b^* & [\![\Sigma; \cdot \vdash A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}]\!]\updownarrow[\texttt{ret } (V_1', V_2')][\gamma_p] \\ \mapsto_b^* & \texttt{let } (x_1, x_2) = (V_1', V_2')[\gamma_p]; \\ & x_1' \leftarrow [\![\Sigma; \Gamma \vdash A_1^{\sqsubseteq}]\!]\updownarrow[\texttt{ret } x_1][\gamma_p]; \\ & x_2' \leftarrow [\![\Sigma; \Gamma \vdash A_2^{\sqsubseteq}]\!]\updownarrow[\texttt{ret } x_2][\gamma_p]; \\ & \texttt{ret } (x_1', x_2') \end{array}$$

Let $M_4'[\gamma_p]$ be the latter term. By Lemma F.9 (bind confluence), we have $M_3'[\gamma_p] \mapsto_b^* M_4'[\gamma_p]$. Thus, we have this last reduction:

$$\begin{array}{ll} & S[M_3'][\gamma_p] \\ \mapsto_b^* & S[M_4'][\gamma_p] \\ \mapsto & S[x_1' \leftarrow [\![\Sigma; \Gamma \vdash A_1^{\sqsubseteq}]\!]\updownarrow[\texttt{ret } V_1']; \\ & \quad x_2' \leftarrow [\![\Sigma; \Gamma \vdash A_2^{\sqsubseteq}]\!]\updownarrow[\texttt{ret } V_2']; \\ & \quad \texttt{ret } (x_1', x_2')][\gamma_p] \end{array}$$

And, since $V_i \rightsquigarrow^{CT} \texttt{ret}\ V_i'$, we have that this result is related to M_2 by Lemma F.13 (translation context plug), so we conclude.

- Function cast:

$$E[(\langle A_1^{\sqsubseteq} \to A_2^{\sqsubseteq}\rangle\!\updownarrow\ V_1)\ V_2] \mapsto E[\langle A_2^{\sqsubseteq}\rangle\!\updownarrow\ (V_1\ \langle A_1^{\sqsubseteq}\rangle\!\updownarrow^{-}\ V_2)]$$

We have $(f \leftarrow [\![\Sigma;\cdot \vdash A_1^{\sqsubseteq} \to A_2^{\sqsubseteq}]\!]\!\updownarrow[N_1'];a \leftarrow N_2';\texttt{force}\ f\ a)[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V_i \rightsquigarrow^{CT} N_i'$. Then, by Lemma F.11 (value translation reduction), we have $N_i'[\gamma_p] \mapsto_b^* \texttt{ret}\ V_i'[\gamma_p]$ for some V_1', V_2' such that $V_i \rightsquigarrow^{CT} \texttt{ret}\ V_i'$. Thus, we have the following reduction:

$$\begin{array}{rl} & (f \leftarrow [\![\Sigma;\cdot \vdash A_1^{\sqsubseteq} \to A_2^{\sqsubseteq}]\!]\!\updownarrow[N_1'];a \leftarrow N_2';\texttt{force}\ f\ a)[\gamma_p] \\ \mapsto_b^* & (\texttt{force}\ (\texttt{thunk}\ \lambda y : [\![\Sigma;\Gamma \vdash A_{1l}^{\sqsubseteq}]\!].\\ & \qquad a \leftarrow [\![\Sigma;\Gamma \vdash A_1^{\sqsubseteq}]\!]\!\updownarrow^{-}[\texttt{ret}\ y];[\![\Sigma;\Gamma \vdash A_2^{\sqsubseteq}]\!]\!\updownarrow[\texttt{force}\ V_1'\ a])\ V_2')[\gamma_p] \\ \mapsto^+ & (a \leftarrow [\![\Sigma;\Gamma \vdash A_1^{\sqsubseteq}]\!]\!\updownarrow^{-}[\texttt{ret}\ V_2'];[\![\Sigma;\Gamma \vdash A_2^{\sqsubseteq}]\!]\!\updownarrow[\texttt{force}\ V_1'\ a])[\gamma_p] \end{array}$$

Let $M_4'[\gamma_p]$ be the latter term. Then we have $\langle A_2^{\sqsubseteq}\rangle\!\updownarrow\ (V_1\ \langle A_1^{\sqsubseteq}\rangle\!\updownarrow^{-}\ V_2 \rightsquigarrow^{CT} M_4'$ since $V_i \rightsquigarrow^{CT} \texttt{ret}\ V_i'$, applying the bind reduction rule to eliminate the first bind. We then conclude by Lemma F.13 (translation context plug).

- Universal cast:

$$E[(\langle \forall^{\nu} X.A^{\sqsubseteq}\rangle\!\updownarrow\ \Lambda X.N)\{\sigma \cong A_1\}] \mapsto E[\langle A^{\sqsubseteq}[\sigma/X]\rangle\!\updownarrow\ N[\sigma/X]]$$

Let $A_1' = [\![\Sigma;\cdot \vdash A_1]\!]$. We have

$$(x \leftarrow [\![\Sigma;\Gamma \vdash \forall^{\nu} X.A^{\sqsubseteq}]\!]\!\updownarrow[\texttt{ret thunk}\ \Lambda X.\lambda c_X : \text{Case}\ X.N'];\texttt{force}\ x\ [A_1']\ \sigma)[\gamma_p] \mapsto_b^* M_3'[\gamma_p]$$

where $N \rightsquigarrow^{CT} N'$. Let $V' = \texttt{thunk}\ \Lambda X.\lambda c_X : \text{Case}\ X.N'$. We then have the following reduction:

$$\begin{array}{rl} & (x \leftarrow [\![\Sigma;\cdot \vdash \forall^{\nu} X.A^{\sqsubseteq}]\!]\!\updownarrow[\texttt{ret}\ V'];\texttt{force}\ x\ [A_1']\ \sigma)[\gamma_p] \\ \mapsto_b^* & (\texttt{force}\ (\texttt{thunk}\ \Lambda X.\lambda c_X : \text{Case}\ X.[\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[\texttt{force}\ V'\ [X]\ c_X])\ [A_1']\ \sigma)[\gamma_p] \end{array}$$

Then, by Lemma F.9 (bind confluence) and the operational semantics, we have

$$\begin{array}{rl} & S[M_3'][\gamma_p] \\ \mapsto_b^* & S[\texttt{force}\ (\texttt{thunk}\ \Lambda X.\lambda c_X : \text{Case}\ X.[\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[\texttt{force}\ V'\ [X]\ c_X])\ [A_1']\ \sigma][\gamma_p] \\ \mapsto^+ & S[([\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[A_1'/X][\sigma/c_X])[\texttt{force}\ V'\ [A_1']\ \sigma]][\gamma_p] \\ \mapsto^+ & S[([\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[A_1'/X][\sigma/c_X])[N'[A_1'/X][\sigma/c_X]]][\gamma_p] \\ = & S[[\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[N'][A_1'/X][\sigma/c_X]][\gamma_p] \end{array}$$

Let $M_2'[\gamma_p]$ be the latter term. Then, by Lemma F.13 (translation context plug), and since $\langle A^{\sqsubseteq}[\sigma/X]\rangle\!\updownarrow\ N[\sigma/X] \rightsquigarrow^{CT} [\![\Sigma;X \vdash A^{\sqsubseteq}]\!]\!\updownarrow[N'][A_1'/X][\sigma/c_X]$, which we have by Lemma F.4 (type variable translation substitution) because $\Sigma;X \vdash N \rightsquigarrow^{CT} N' : B$ and $\sigma : A_1 \in \Sigma$, we may conclude.

- tag upcast:

$$E[\langle \text{tag}_G(A^{\sqsubseteq})\rangle\!\updownarrow\ V] \mapsto E[\texttt{inj}_G\ \langle A^{\sqsubseteq}\rangle\!\updownarrow\ V]$$

We have $[\![\Sigma;\cdot \vdash \text{tag}_G(A^{\sqsubseteq})]\!]\!\updownarrow[N'][\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V \rightsquigarrow^{CT} N'$. Then, by Lemma F.11 (value translation), we have some V' such that $N'[\gamma_p] \mapsto_b^* \texttt{ret}\ V'[\gamma_p]$ and $V \rightsquigarrow^{CT} \texttt{ret}\ V'$. Thus, we have the following reduction:

$$\begin{array}{rl} & [\![\Sigma;\cdot \vdash \text{tag}_G(A^{\sqsubseteq})]\!]\!\updownarrow[N'][\gamma_p] \\ \mapsto_b^* & [\![\Sigma;\cdot \vdash \text{tag}_G(A^{\sqsubseteq})]\!]\!\updownarrow[\texttt{ret}\ V'][\gamma_p] \\ = & (r \leftarrow [\![\Sigma;\cdot \vdash A^{\sqsubseteq}]\!]\!\updownarrow[\texttt{ret}\ V'];\texttt{inj}_{\text{case}(G)}\ r)[\gamma_p] \end{array}$$

Let the latter term be $M_4'[\gamma_p]$. By Lemma F.8 (bind normalization) there must exist some M_5' such that $S[M_4'][\gamma_p] \mapsto_b^* M_5'[\gamma_p]$ and $M_5'[\gamma_p]$ does not take a bind reduction. Then, by Lemma

F.9 (bind confluence), we have that $M_1'[\gamma_p] \mapsto_b^* M_5'[\gamma_p]$. Note that $E[\mathtt{inj}_G\ \langle A^{\sqsubseteq}\rangle\!\uparrow\ V] \rightsquigarrow^{CT} S[M_4']$ by Lemma F.13 (translation context plug). Then, by Lemma F.15 (bind reduction preserves translation), we have $E[\mathtt{inj}_G\ \langle A^{\sqsubseteq}\rangle\!\uparrow\ V] \rightsquigarrow^{CT} M_5'$. Thus, it suffices to show that $|\langle \mathrm{tag}_G(A^{\sqsubseteq})\rangle\!\uparrow\ V| > |\mathtt{inj}_G\ \langle A^{\sqsubseteq}\rangle\!\uparrow\ V|$, which we have since $|\langle \mathrm{tag}_G(A^{\sqsubseteq})\rangle\!\uparrow\ V| = 1+2+|A^{\sqsubseteq}|+|V|$.

- tag downcast:

$$E[\langle \mathrm{tag}_G(A^{\sqsubseteq})\rangle\!\downarrow\ \mathtt{inj}_G\ V] \mapsto E[\langle A^{\sqsubseteq}\rangle\!\downarrow\ V]$$

We have $[\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [r \leftarrow N'; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ r][\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V \rightsquigarrow^{CT} N'$. Then, by Lemma F.11 (value translation), we have some V' such that $N'[\gamma_p] \mapsto_b^* \mathtt{ret}\ V'[\gamma_p]$ and $V \rightsquigarrow^{CT} \mathtt{ret}\ V'$. Thus, we have the following reduction:

$$\begin{array}{ll} & [\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [r \leftarrow N'; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ r][\gamma_p] \\ \mapsto_b^* & [\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [\mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(G)}\ V'][\gamma_p] \\ \mapsto_b^* & \mathtt{match}\ \mathrm{case}(G)\ \mathtt{with}\ (\mathtt{inj}_{\mathrm{case}(G)}\ V')\{\mathtt{inj}\ y.[\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ y] \mid \mho\}[\gamma_p] \end{array}$$

Let $M_4'[\gamma_p]$ be the latter term. Then, by Lemma F.9 (bind confluence) and the operational semantics, we have the following reductions:

$$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[\mathtt{match}\ \mathrm{case}(G)\ \mathtt{with}\ (\mathtt{inj}_{\mathrm{case}(G)}\ V')\{\mathtt{inj}\ y.[\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ y] \mid \mho\}][\gamma_p] \\ \mapsto & S[[\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ V']][\gamma_p] \end{array}$$

Then, since $V \rightsquigarrow^{CT} \mathtt{ret}\ V'$, we have $\langle A^{\sqsubseteq}\rangle\!\downarrow\ V \rightsquigarrow^{CT} [\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ V'][\gamma_p]$ and we may conclude by Lemma F.13 (translation context plug).

- tag downcast error:

$$E[\langle \mathrm{tag}_G(A^{\sqsubseteq})\rangle\!\downarrow\ \mathtt{inj}_H\ V] \mapsto E[\mho_B] \text{ where } G \neq H$$

We have $[\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [r \leftarrow N'; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(H)}\ r][\gamma_p] \mapsto_b^* M_3'[\gamma_p]$ where $V \rightsquigarrow^{CT} N'$. Then, by Lemma F.11 (value translation), we have some V' such that $N'[\gamma_p] \mapsto_b^* \mathtt{ret}\ V'[\gamma_p]$ and $V \rightsquigarrow^{CT} \mathtt{ret}\ V'$. Thus, we have the following reduction:

$$\begin{array}{ll} & [\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [r \leftarrow N'; \mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(H)}\ r][\gamma_p] \\ \mapsto_b^* & [\![\Sigma; \Delta \vdash \mathrm{tag}_G(A^{\sqsubseteq})]\!]\downarrow [\mathtt{ret}\ \mathtt{inj}_{\mathrm{case}(H)}\ V'][\gamma_p] \\ \mapsto_b^* & \mathtt{match}\ \mathrm{case}(G)\ \mathtt{with}\ (\mathtt{inj}_{\mathrm{case}(H)}\ V')\{\mathtt{inj}\ y.[\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ y] \mid \mho\}[\gamma_p] \end{array}$$

Let $M_4'[\gamma_p]$ be the latter term. Then, by Lemma F.9 (bind confluence) and the operational semantics, we have the following reductions:

$$\begin{array}{ll} & M_1'[\gamma_p] \\ \mapsto_b^* & S[\mathtt{match}\ \mathrm{case}(G)\ \mathtt{with}\ (\mathtt{inj}_{\mathrm{case}(H)}\ V')\{\mathtt{inj}\ y.[\![\Sigma; \Gamma \vdash A^{\sqsubseteq}]\!]\downarrow [\mathtt{ret}\ y] \mid \mho\}][\gamma_p] \\ \mapsto & S[\mho][\gamma_p] \end{array}$$

Then, since $\mho_B \rightsquigarrow^{CT} \mho$, we may conclude by Lemma F.13 (translation context plug).

- Existential upcast:

$$E[\langle \exists^{\nu} X.A_1^{\sqsubseteq}\rangle\!\uparrow\ \mathtt{pack}^{\nu}(X \cong A, [A_2^{\sqsubseteq} \updownarrow \ldots], N_1)] \mapsto E[\mathtt{pack}^{\nu}(X \cong A, [A_1^{\sqsubseteq}\ \uparrow, A_2^{\sqsubseteq} \updownarrow \ldots], N_1)]$$

For some N_1' such that $N_1 \rightsquigarrow^{CT} N_1'$, we have

$$[\![\Sigma; \cdot \vdash \exists^{\nu} X.A_1^{\sqsubseteq}]\!]\uparrow [\mathtt{ret}\ \mathtt{pack}(A', \mathtt{thunk}\ \lambda c_X : \mathrm{Case}\ A'.N'_{cst})\ \mathtt{as}\ [\![\Sigma; \cdot \vdash \exists^{\nu} X.A_{1r}^{\sqsubseteq}]\!]][\gamma_p] \mapsto_b^* M_3'[\gamma_p]$$

where $N_{cst} = [\![\Sigma; X \cong A' \vdash A_2^{\sqsubseteq}]\!] \updownarrow [\texttt{force}\ (\texttt{thunk}\ (\lambda c_X : \text{Case}\ A'.(\ldots\ N_1'\ \ldots)))\ c_X]$ and $A' = [\![\Sigma; \cdot \vdash A]\!]$. By Lemma F.9 (bind confluence), we then have the following reductions:

$$\begin{array}{ll} & M_3'[\gamma_p] \\ \mapsto_b^* & \texttt{unpack}\ (Y, f) = \texttt{pack}(A', \texttt{thunk}\ \lambda c_X : \text{Case}\ A'.N_{cst}')\ \texttt{as}\ [\![\exists^\nu X.A_{1l}^{\sqsubseteq}]\!]; \\ & \texttt{ret}\ \texttt{pack}(Y, \texttt{thunk}\ \lambda c_X : \text{Case}\ Y.[\![\Sigma; Y, X \cong Y \vdash A_1^{\sqsubseteq}]\!] \updownarrow [\texttt{force}\ f\ c_X])\ \texttt{as}\ [\![\exists^\nu X.A_{1r}^{\sqsubseteq}]\!][\gamma_p] \\ \mapsto^+ & \texttt{ret}\ \texttt{pack}(A', \texttt{thunk}\ \lambda c_X : \text{Case}\ A'. \\ & \quad [\![\Sigma; X \cong A' \vdash A_1^{\sqsubseteq}]\!] \updownarrow [\texttt{force}\ (\texttt{thunk}\ \lambda c_X : \text{Case}\ A'.N_{cst}')\ c_X])\ \texttt{as}\ [\![\exists^\nu X.A_{1r}^{\sqsubseteq}]\!][\gamma_p] \end{array}$$

Let the above term be $M_4'[\gamma_p]$. We know that $\texttt{pack}^\nu(X \cong A', [A_1^{\sqsubseteq} \updownarrow, A_2^{\sqsubseteq} \updownarrow \ldots], N_1) \rightsquigarrow^{CT} M_4'$ since $N_1 \rightsquigarrow^{CT} N_1'$, so we conclude by Lemma F.13 (translation context plug).

- Existential downcast:

$$E[\langle \exists^\nu X.A_1^{\sqsubseteq} \rangle \downarrow\ \texttt{pack}^\nu(X \cong A, [A_2^{\sqsubseteq} \updownarrow \ldots], N_1)] \mapsto E[\texttt{pack}^\nu(X \cong A, [A_1^{\sqsubseteq} \downarrow, A_2^{\sqsubseteq} \updownarrow \ldots], N_1)]$$

This case proceeds analogously, replacing $A_{1r}^{\sqsubseteq}$ with $A_{1l}^{\sqsubseteq}$ and vice versa.

- Tag check true:

$$E[\texttt{is}(G)\texttt{?}\ \texttt{inj}_G\ V] \mapsto E[\texttt{true}]$$

We have

$$\begin{array}{ll} & (r \leftarrow x \leftarrow N'; \texttt{ret}\ \texttt{inj}_{\text{case}(G)}\ x; \\ & \texttt{match}\ \text{case}(G)\ \texttt{with}\ r\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\})[\gamma_p] \\ \mapsto_b^* & M_3'[\gamma_p] \end{array}$$

where $V \rightsquigarrow^{CT} N'$. Then, by Lemma F.11 (value translation), we have some V' such that $N'[\gamma_p] \mapsto_b^* \texttt{ret}\ V'[\gamma_p]$ and $V \rightsquigarrow^{CT} \texttt{ret}\ V'$. Thus, we have the following reduction:

$$\begin{array}{ll} & (r \leftarrow x \leftarrow N'; \texttt{ret}\ \texttt{inj}_{\text{case}(G)}\ x; \\ & \texttt{match}\ \text{case}(G)\ \texttt{with}\ r\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\})[\gamma_p] \\ \mapsto_b^* & (r \leftarrow \texttt{ret}\ \texttt{inj}_{\text{case}(G)}\ V'; \\ & \texttt{match}\ \text{case}(G)\ \texttt{with}\ r\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\})[\gamma_p] \\ \mapsto_b^* & (\texttt{match}\ \text{case}(G)\ \texttt{with}\ (\texttt{inj}_{\text{case}(G)}\ V')\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\})[\gamma_p] \end{array}$$

Then, by Lemma F.9 (bind confluence), we have

$$\begin{array}{ll} & M_3' \\ \mapsto_b^* & (\texttt{match}\ \text{case}(G)\ \texttt{with}\ (\texttt{inj}_{\text{case}(G)}\ V')\{\texttt{inj}\ y.\texttt{ret true} \mid \texttt{ret false}\})[\gamma_p] \\ \mapsto & \texttt{ret true} \end{array}$$

Since $\texttt{true} \rightsquigarrow^{CT} \texttt{ret true}$, we conclude by Lemma F.13 (translation context plug).

- Tag check false:

$$E[\texttt{is}(G)\texttt{?}\ \texttt{inj}_H\ V] \mapsto E[\texttt{false}] \text{ where } G \neq H$$

This case is analogous to the former except that since $G \neq H$, the translation produces $\texttt{false}$ and since $\texttt{false} \rightsquigarrow^{CT} \texttt{false}$, we conclude.

□

Lemma F.18 (Multi-step simulation). *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : A$ *and* $\Sigma \triangleright M \mapsto^* \Sigma_1 \triangleright M_1$, *then* $M_1 \rightsquigarrow^{CT} M_1'$ *and* $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_1]\!] \triangleright M_1'[\gamma_p]$ *for some* M_1'.

Proof. We proceed by induction on the number of steps n in $\Sigma \triangleright M \mapsto^* \Sigma_1 \triangleright M_1$.

- $n = 0$: Then $M_1 = M$, so we have $M_1 \rightsquigarrow^{CT} M'$ and $M'[\gamma_p] \mapsto^* M'[\gamma_p]$ by reflexivity.

- $n = n' + 1$: Then there exists some Σ_2, M_2 such that $\Sigma \triangleright M \mapsto^* \Sigma_2 \triangleright M_2 \mapsto \Sigma_1 \triangleright M_1$. By the inductive hypothesis for n', we then have some M_2' such that $M_2 \rightsquigarrow^{CT} M_2'$ and
$$\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_2]\!] \triangleright M_2'[\gamma_p]$$
Finally, by Theorem F.17 (simulation), since $M_2 \rightsquigarrow^{CT} M_2'$ and $\Sigma_2 \triangleright M_2 \mapsto \Sigma_1 \triangleright M_1$, there exists M_1' such that $M_1 \rightsquigarrow^{CT} M_1'$ and $\Sigma_p, [\![\Sigma_2]\!] \triangleright M_2'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_1]\!] \triangleright M_1'[\gamma_p]$. Therefore, $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_1]\!] \triangleright M_1'[\gamma_p]$ as we were required to show.

□

Lemma F.19 (Unique Decomposition). *If $\Sigma_1; \cdot \vdash M_1 : A$, then there exist unique E, M_2 such that $M_1 = E[M_2]$.*

Proof. By induction on M_1. □

Lemma F.20 (Progress). *If $\Sigma_1; \cdot \vdash M_1 : A$ then either $\Sigma_1 \triangleright M_1 \mapsto \Sigma_2 \triangleright M_2$, $M_1 = \mathfrak{U}$, or $M_1 = V$ for some V.*

Proof. If $M_1 = \mathfrak{U}$ or M_1 is a value, we conclude. Otherwise, by Lemma F.19 (unique decomposition), we have unique E, M_3 such that $M_1 = E[M_3]$. We proceed by cases on M_3.

- $M_3 = \mathfrak{U}$ Since $M_1 \neq \mathfrak{U}$, we have $E \neq []$, so $M_1 \mapsto \mathfrak{U}$ and we conclude.
- $M_3 = \texttt{hide}\ X \cong B; M_4$ Then $\Sigma_1 \triangleright M_3 \mapsto \Sigma_1, \sigma : B \triangleright M_4[\sigma/X]$, so we conclude since $M_2 \mapsto E[M_4[\sigma/X]]$.
- $M_3 = \texttt{if}\ V\ \texttt{then}\ M_4\ \texttt{else}\ M_5$ Since M_1 is well-typed, $V : \mathbb{B}$ and so, by Lemma F.10 (canonical forms), which we may use thanks to Lemma F.3, either $V = \texttt{true}$ or $V = \texttt{false}$. In the first case, $M_3 \mapsto M_4$ and in the second $M_3 \mapsto M_5$, so we conclude.
- $M_3 = \texttt{let}\ x = V; M_4$ Then $M_3 \mapsto M_4[V/x]$, so we conclude.
- $M_3 = \texttt{let}\ (x, y) = V; M_4$ Since M_1 is well-typed, $V : A_1 \times A_2$ for some A_1, A_2 and so, by Lemma F.10 (canonical forms), $V = (V_1, V_2)$ for some V_1, V_2. Then $M_3 \mapsto M_4[V_1/x][V_2/y]$, so we conclude.
- $M_3 = \texttt{pack}^\nu(X \cong B, M_4)$ Then $M_3 \mapsto \texttt{pack}^\nu(X \cong B, [], M_4)$, so we conclude.
- $M_3 = \texttt{unpack}\ (X, x) = V; M_4$ Since M_1 is well-typed, $V : \exists^\nu X B$ for some B and so, by Lemma F.10 (canonical forms), $V = \texttt{pack}^\nu(X \cong B, [], M_5)$ for some M_5. Then $\Sigma \triangleright M_3 \mapsto \Sigma, \sigma : B \triangleright \texttt{let}\ x = M_5[\sigma/X]; M_4[\sigma/X]$, so we conclude.
- $M_3 = V\{\sigma \cong B\}$ Since M_1 is well-typed, $V : \forall^\nu X A_1$ for some A_1 and so, by Lemma F.10 (canonical forms), $V = \Lambda^\nu X.M_4$ for some M_4 or $V = \langle \forall^\nu X.A^\sqsubseteq \rangle\!\updownarrow\ V_1$ for some $A^\sqsubseteq, V_1$. In the first case, $M_3 \mapsto M_4[\sigma/X]$ and in the second, $M_3 \mapsto \langle A^\sqsubseteq[\sigma/X] \rangle\!\updownarrow\ (V_1\{\sigma \cong B\})$.
- $M_3 = V_1\ V_2$ Since M_1 is well-typed, $V : A_1 \to A_2$ for some A_1, A_2 and so, by Lemma F.10 (canonical forms), $V_1 = \lambda x : A_1.M_4$ for some M_4 or $V = \langle A_1^\sqsubseteq \to A_2^\sqsubseteq \rangle\!\updownarrow\ V_1$ for some $A_1^\sqsubseteq, A_2^\sqsubseteq, V_1$. In the first case, $M_3 \mapsto M_4[V_2/x]$ and in the second, $M_3 \mapsto \langle A_2^\sqsubseteq \rangle\!\updownarrow\ (V_1\ \langle A_1^\sqsubseteq \rangle\!\updownarrow^-\ V_2)$.
- $M_3 = \texttt{unseal}_\sigma V$ Since M_1 is well-typed, $V : \sigma$ and so, by Lemma F.10 (canonical forms), $V = \texttt{seal}_\sigma V_1$ for some V_1. Then $M_3 \mapsto V_1$, so we conclude.
- $M_3 = \texttt{is}(G)?\ V$ Since M_1 is well-typed, $V : ?$ and so, by Lemma F.10 (canonical forms), $V = \texttt{inj}_H\ V_1$ for some H, V_1. If $H = G$, then $M_3 \mapsto \texttt{true}$, otherwise $M_3 \mapsto \texttt{false}$.
- $M_3 = \langle \exists^\nu X.A^\sqsubseteq \rangle\!\updownarrow\ V$ Since M_1 is well-typed, $V : \exists^\nu X.A_1$ for some A_1 and so, by Lemma F.10 (canonical forms), $V = \texttt{pack}^\nu(X \cong B, [A_2^\sqsubseteq \updownarrow_2 \ldots], M_4)$ for some $B, A_2^\sqsubseteq \updownarrow_2 \ldots, M_4$. Then $M_3 \mapsto \texttt{pack}^\nu(X \cong A_2, [A^\sqsubseteq \updownarrow, A_2^\sqsubseteq \updownarrow_2 \ldots], M_4)$.

- $M_3 = \langle A_1^{\sqsubseteq} \times A_2^{\sqsubseteq} \rangle\!\updownarrow V$ Since M_1 is well-typed, $V : B_1 \times B_2$ for some B_1, B_2 and so, by Lemma F.10 (canonical forms), $V = (V_1, V_2)$ for some V_1, V_2. Then $M_3 \mapsto (\langle A_1^{\sqsubseteq} \rangle\!\updownarrow V_1, \langle A_2^{\sqsubseteq} \rangle\!\updownarrow V_2)$.
- $M_3 = \langle A^{\sqsubseteq} \rangle\!\updownarrow V$ where $A^{\sqsubseteq} \in \{\mathbb{B}, \sigma, ?\}$ Then $M_3 \mapsto V$.
- $M_3 = \langle \mathrm{tag}_G A^{\sqsubseteq} \rangle\!\updownarrow V$ Then $M_3 \mapsto \mathtt{inj}_G\ \langle A^{\sqsubseteq} \rangle\!\updownarrow V$.
- $M_3 = \langle \mathrm{tag}_G A^{\sqsubseteq} \rangle\!\downarrow V$ Since M_1 is well-typed, $V : ?$ and so, by Lemma F.10 (canonical forms), $V = \mathtt{inj}_H\ V_1$ for some H, V_1. If $H = G$, then $M_3 \mapsto \langle A^{\sqsubseteq} \rangle\!\downarrow V$. Otherwise, $M_3 \mapsto \mathfrak{U}$.

□

Theorem F.21. *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : A$, *and* $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \Uparrow$, *then* $\Sigma \triangleright M \Uparrow$.

Proof. Assume the opposite, that $M \mapsto^* M_1$ and M_1 does not take a step. By Lemma F.18 (multi-step simulation), we have some M_1' such that $M_1 \rightsquigarrow^{CT} M_1'$ and $M'[\gamma_p] \mapsto^* M_1'[\gamma_p]$. Then, by Lemma F.20 (progress), $M_1 = \mathfrak{U}$ or $M_1 = V$ for some value V so we have two cases to consider:

- $M_1 = \mathfrak{U}$: By Lemma F.10 (canonical forms), $M_1' = \mathfrak{U}$ and we have $M'[\gamma_p] \mapsto^* \mathfrak{U}$, but $M'[\gamma_p] \Uparrow$, so we have a contradiction.
- $M_1 = V$: By Lemma F.11, we have $M'[\gamma_p] \mapsto^* M_1'[\gamma_p] \mapsto^* \mathtt{ret}\ V'$ for some value V', but $M'[\gamma_p] \Uparrow$, so again we have a contradiction.

□

Theorem F.22. *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : A$, *and* $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \Downarrow$, *then* $\Sigma \triangleright M \Downarrow$.

Proof. We proceed by induction first on the number of steps $M'[\gamma_p]$ takes and then on $|M|$. Assume that $\Sigma \triangleright M \mapsto \Sigma_1 \triangleright M_1$ since otherwise, we may conclude. By Theorem F.17 (simulation), we have some M_1' such that $M_1 \rightsquigarrow^{CT} M_1'$ and either $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^+ \Sigma_p, [\![\Sigma_1]\!] \triangleright M_1'[\gamma_p]$ or $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_1]\!] \triangleright M_1'[\gamma_p]$ and $|M| > |M_1|$. Note that since $M'[\gamma_p] \Downarrow$, by Lemma F.2 (target semantics deterministic) we have $M_1'[\gamma_p] \Downarrow$. We then have two cases to consider. If $M' \mapsto^+ M_1'$, then we conclude by the inductive hypothesis for $M_1'[\gamma_p] \Downarrow$. Otherwise, we have that $|M| > |M_1|$, so we conclude by the inductive hypothesis for $|M_1|$.

□

Lemma F.23. *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : A$, *and* $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \Downarrow \mathfrak{U}$, *then* $\Sigma \triangleright M \Downarrow \mathfrak{U}_A$.

Proof. By Theorem F.22 (target termination implies source termination), we have $\Sigma \triangleright M \Downarrow M_R$ for some M_R. If $M_R = \mathfrak{U}_A$, we conclude. Otherwise, $M_R = \mathtt{ret}\ V$ for some V. We prove this case by contradiction. By Lemma F.18 (multi-step simulation), We have some Σ_R, M_R' such that $M_R \rightsquigarrow^{CT} M_R'$ and $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_R]\!] \triangleright M_R'[\gamma_p]$. Then, by Lemma F.11 (value translation), we then have $M_R'[\gamma_p] \mapsto_b^* \mathtt{ret}\ V'[\gamma_p]$. However, by Lemma F.2 (target language deterministic), we have $M_R'[\gamma_p] \Downarrow \mathfrak{U}$ and we have a contradiction. □

Lemma F.24. *If* $\Sigma; \cdot \vdash M \rightsquigarrow^{CT} M' : A$, *and* $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \Downarrow \mathtt{ret}\ V'$, *then* $\Sigma \triangleright M \Downarrow V$.

Proof. By Theorem F.22 (target termination implies source termination), we have $\Sigma \triangleright M \Downarrow M_R$ for some M_R. If $M_R = V$ for some V, we conclude. Otherwise, $M_R = \mathfrak{U}_A$. We prove this case by contradiction. By Lemma F.18 (multi-step simulation), We have some Σ_R, M_R' such that $M_R \rightsquigarrow^{CT} M_R'$ and $\Sigma_p, [\![\Sigma]\!] \triangleright M'[\gamma_p] \mapsto^* \Sigma_p, [\![\Sigma_R]\!] \triangleright M_R'[\gamma_p]$. Then, by Lemma F.10 (canonical forms), we then have

$M'_R = \mho$. However, by Lemma F.2 (target language deterministic), we have $M'_R[\gamma_p] \Downarrow \texttt{ret}\ V'$ and we have a contradiction. □

G GRADUALITY

Definition G.1. We say γ, δ are valid instantiations of $\Gamma^{\sqsubseteq}$ in $\text{CBPV}_{\text{OSum}}$, written $(\gamma, \delta) \vDash \Gamma^{\sqsubseteq}$. when

- For each $i \in \{l, r\}$, there exists Σ_i such that for each $(x : A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}$, $\Sigma_i \mid \cdot \vdash \gamma_i(x) : [\![A_i]\!]$ when $\Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$ and for each $X \in \Gamma^{\sqsubseteq}$, $\cdot \vdash \delta_i(X)$ and $\Sigma_i \mid \cdot \vdash \gamma_i(c_X) : \text{Case}\ \delta_i(X)$.
- For each $X \in \Gamma^{\sqsubseteq}$, $\delta_R(X) \in \text{Rel}_\omega[\delta_l(X), \delta_r(X)]$.

Definition G.2. We define the extension of an interpretation η with a new association between seals as

$$\eta \boxplus (\sigma_l, \sigma_r, R) = (\eta.size + 1, (f, \eta.size \mapsto (\sigma_l, \sigma_r)), (\rho, \eta.size \mapsto R))$$

Logical Lemmas

Lemma G.3. *If $R \in Rel_\omega[A_l, A_r]$, then $\lfloor R \rfloor_n \in Rel_n[A_l, A_r]$*

Proof. Direct by definition. □

Lemma G.4. $\lfloor \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta \rfloor_n = \mathcal{V}^{\sim}_n[\![A^{\sqsubseteq}]\!]\gamma\delta$

Proof. Direct by definition. □

Lemma G.5. *If $R \in Rel_n[A_l, A_r]$, then $\triangleright R \in Rel_n[A_l, A_r]$*

Proof. If $(w, V_l, V_r) \in (\triangleright R)$ and $w' \sqsupseteq w$, then to show $(w', V_l, V_r) \in \triangleright R$, we need to show that for any $w'' \sqsupset w'$, that $(w'', V_l, V_r) \in R$, but this follows because $(w, V_l, V_r) \in \triangleright R$ and $w'' \sqsupset w' \sqsupseteq w$. □

Lemma G.6. *Let $\Gamma^{\sqsubseteq}$ be a well-formed context, with $\Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$. If $(\gamma, \delta) \vDash \Gamma^{\sqsubseteq}$, then $\mathcal{V}^{\sim}_n[\![A^{\sqsubseteq}]\!]\gamma\delta \in Rel_n[A_l, A_r]$.*

Proof. By induction on $A^{\sqsubseteq}$.

(1) X: by lemma G.3.
(2) ?: If $(w, \texttt{inj}_{\sigma_l}\ V_l, \texttt{inj}_{\sigma_r}\ V_r) \in \mathcal{V}^{\sim}_n[\![?]\!]\gamma\delta$ and $w' \sqsupseteq w$, then there exists $R \in \text{Rel}_n[A_l, A_r]$ with $w.\eta \vDash (\sigma_l, \sigma_r, R)$ and $(w, V_l, V_r) \in \triangleright R$. By definition of $\sqsupseteq$, we have that $w' \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j})$ so it is sufficient to show $(w, V_l, V_r) \in \triangleright \lfloor R \rfloor_{w'.j}$, which follows by lemma G.5 that later preserves monotonicity.
(3) $\text{tag}_G(A^{\sqsubseteq})$: by inductive hypothesis, using lemma G.5 in the $\succ$ case.
(4) $\mathbb{B}$: immediate
(5) $\times$: immediate by inductive hypothesis.
(6) $\to$: If $(w, V_l, V_r) \in \mathcal{V}^{\sim}_n[\![A^{\sqsubseteq} \to B^{\sqsubseteq}]\!]\gamma\delta$ and $w' \sqsupseteq w$. Then given $w'' \sqsupseteq w'$ and $(w'', V'_l, V'_r) \in \mathcal{V}^{\sim}_n[\![A^{\sqsubseteq}]\!]\gamma\delta$, we need to show $(w'', \texttt{force}\ V_l\, V'_l, \texttt{force}\ V_r\, V'_r) \mathcal{E}^{\sim}_n[\![B^{\sqsubseteq}]\!]\gamma\delta$, but this holds by relatedness of V_l, V_r because $w'' \sqsupseteq w$ by transitivity of world extension.
(7) $\forall^\nu, \exists^\nu$: similar to the $\to$ case.

□

Lemma G.7. *If $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$ and $w' \sqsupseteq w$, then $(w', \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$.*

Proof. By induction on $\Gamma^{\sqsubseteq}$, uses monotonicity of $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]$ □

Corollary G.8. $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta \in Rel_\omega[A_l, A_r]$

Lemma G.9 (Anti-Reduction). (1) *If $w' \sqsupseteq w$ and $(w.\Sigma_l, M_l) \mapsto^{w.j - w'.j} (w'.\Sigma_l, M'_l)$ and $(w.\Sigma_r, M_r) \mapsto^* (w'.\Sigma_r, M'_r)$ and $(w', M'_l, M'_r) \in \mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta$, then $(w, M_l, M_r) \in \mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta$.*

(2) *If* $w' \sqsupseteq w$ *and* $(w.\Sigma_l, M_l) \mapsto^{w.j-w'.j} (w'.\Sigma_l, M_l')$ *and* $(w.\Sigma_r, M_r) \mapsto^* (w'.\Sigma_r, M_r')$ *and* $(w', M_l', M_r') \in \mathcal{E}^{\succ}[\![A^{\sqsubseteq}]\!]\gamma\delta$*, then* $(w, M_l, M_r) \in \mathcal{E}^{\succ}[\![A^{\sqsubseteq}]\!]\gamma\delta$.

Proof. We do the $\prec$ case, the other is symmetric. By case analysis on $(w', M_l', M_r') \in \mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta$.

(1) If $w'.\Sigma_l, M_l' \mapsto^{w'.j+1}$, then $(w.\Sigma_l, M_l) \mapsto^{w.j-w'.j+w'.j+1}$ and $w.j - w'.j + w'.j + 1 = w.j + 1$.
(2) If $w'.\Sigma_l, M_l' \mapsto^{j} \Sigma_l', \mathfrak{V}$, with $j \leq w.j$, then $w.\Sigma_l, M_l \mapsto^{w.j-w'.j+j} \Sigma_l', \mathfrak{V}$ and $w.j - w'.j + j \leq w.j$ since $j - w'.j \leq 0$.
(3) Finally, if there is some $w'' \sqsupseteq w'$ and $(w'', V_l, V_r) \in \mathcal{V}^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta$ with $w'.\Sigma_l, M_l' \mapsto^{w'j.-w''.j} w''.\Sigma_l, \texttt{ret}\ V_l$ and $w'.\Sigma_r, M_r' \mapsto^* w'', \Sigma_r, \texttt{ret}\ V_r$, then $w.\Sigma_l, M_l \mapsto^{w.j-w'.j+w'.j-w''.j} w''.\Sigma_l, \texttt{ret}\ V_l''$ and $w.\Sigma_r, M_r \mapsto^* w'', \Sigma_r, \texttt{ret}\ V_r$ and $w.j - w'.j + w'.j - w''.j = w.j - w''.j$ so the result holds.

□

Lemma G.10 (Pure Anti-Reduction). *If* $(w, M_l', M_r') \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$ *and* $(w.\Sigma_l, M_l) \mapsto^0 (w.\Sigma_l, M_l')$ *and* $(w.\Sigma_r, M_r) \mapsto^0 (w.\Sigma_r, M_r')$*, then* $(w, M_l, M_r) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$.

Proof. Immediate corollary of anti-reduction lemma G.9 □

Lemma G.11 (Pure Forward Reduction). *If* $(w, M_l', M_r') \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$ *and* $(w.\Sigma_l, M_l) \mapsto^0 (w.\Sigma_l, M_l')$ *and* $(w.\Sigma_r, M_r) \mapsto^0 (w.\Sigma_r, M_r')$*, then* $(w, M_l, M_r) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$.

Proof. By determinism of evaluation. □

Lemma G.12 (Monadic bind). *If* $(w, M_l, M_r) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$ *and for all* $w' \sqsupseteq w$*, and* $(w', V_l, V_r) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$*,* $(w', S_l[\texttt{ret}\ V_l], S_r[\texttt{ret}\ V_r]) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$*, then* $(w, S_l[M_l], S_r[M_r]) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$.

Proof. We show the proof for $\mathcal{E}^{\prec}[\![A^{\sqsubseteq}]\!]$, the $\succ$ case is symmetric. By case analysis on $(w, M_l, M_r) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$.

(1) If $w.\Sigma_l, M_l \mapsto^{w.j+1}$, then $w.\Sigma_l, S[M_l] \mapsto^{w.j+1}$.
(2) If $w.\Sigma_l, M_l \mapsto^{j} w.\Sigma_l', \mathfrak{V}$, then $w.\Sigma_l, S[M_l] \mapsto^{j} w.\Sigma_l', \mathfrak{V}$.
(3) Otherwise there exists w' and $(w', V_l, V_r) \in \mathcal{V}^{\prec}[\![A^{\sqsubseteq}]\!]\gamma\delta$ with $w.\Sigma_l, M_l \mapsto^{w.j-w'.j} w'.\Sigma_l, \texttt{ret}\ V_l$ and $w.\Sigma_r, M_r \mapsto^* w'.\Sigma_r, \texttt{ret}\ V_r$. Then $w.\Sigma_l, S_l[M_l] \mapsto^{w.j-w'.j} w'.\Sigma_l, S[\texttt{ret}\ V_l]$ and $w.\Sigma_r, S_r[M_r] \mapsto^* w'.\Sigma_r, S_r[\texttt{ret}\ V_r]$, and the result follows by the assumption.

□

Pure evaluation is monotone.

Lemma G.13. *If* $\Sigma, M \mapsto^* \Sigma, N$*, then for any* $\Sigma' \sqsupseteq \Sigma$*,* $\Sigma', M \mapsto^* \Sigma', N$.

Clamping

Lemma G.14. *If* $(w, V_l, V_r) \in R$ *and* $w.j \leq n$*, then* $(w, V_l, V_r) \in \lfloor R \rfloor_n$.

Proof. Direct from definition. □

Tag-to-type

Lemma G.15. $\mathcal{V}_n^{\sim}[\![G]\!]\gamma\delta = \lfloor \delta_R([\![G]\!]_{tag}) \rfloor_n$

Proof. Direct from definition □

Lemma G.16 (Weakening). *If* $\Gamma^{\sqsubseteq} \vdash A^{\sqsubseteq}$ *and* $\Gamma^{\sqsubseteq} \subseteq \Gamma^{\sqsubseteq\prime}$ *and* $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$ *and* $(w, \gamma', \delta') \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq\prime}]\!]$*, where* $\gamma \subseteq \gamma'$ *and* $\delta \subseteq \delta'$*, then all of the following are true:*

$$\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta = \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma'\delta'$$
$$\mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta = \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma'\delta'$$

Proof. Straightforward, by induction over $\Gamma^{\sqsubseteq\prime}$. □

G.1 Cast Lemmas

To prove the cast left lemma, we need the following lemma that casts always either error or terminate with a value on well-typed inputs.

Lemma G.17 (Casts don't diverge). *If* $\Gamma \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$, *then for any* Σ, *and* $\Sigma \mid \cdot \vdash \gamma : [\![\Gamma]\!]$,

(1) *If* $\Sigma \mid \cdot \vdash V : A_l$, *then either* $\Sigma, [\![\langle A^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V_l][\gamma] \mapsto^* \Sigma, \mho$ *or* $\Sigma, [\![\langle A^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V_l][\gamma] \mapsto^* \Sigma, \mathtt{ret}\ V'$.
(2) *If* $\Sigma \mid \cdot \vdash V : A_r$, *then either* $\Sigma, [\![\langle A^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V][\gamma] \mapsto^* \Sigma, \mho$ *or* $\Sigma, [\![\langle A^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V][\gamma] \mapsto^* \Sigma, \mathtt{ret}\ V'$.

Proof. By induction on $A^{\sqsubseteq}$.

(1) If $A^{\sqsubseteq} \in \{?, \mathbb{B}\}$ the cast is trivial.
(2) Case $A^{\sqsubseteq} = \mathrm{tag}_G(AG^{\sqsubseteq})$:
   (a) The upcast definition expands as follows:

$$[\![\langle \mathrm{tag}_G(AG^{\sqsubseteq})\rangle\updownarrow]\!][\mathtt{ret}\ V][\gamma] = x \leftarrow [\![\langle AG^{\sqsubseteq}\rangle\updownarrow]\!][\mathtt{ret}\ V][\gamma]; \mathtt{ret}\ \mathtt{inj}_\sigma\ x$$

   where $\sigma = \gamma([\![G]\!]_{tag})$. By inductive hypothesis, $[\![\langle AG^{\sqsubseteq}\rangle\updownarrow]\!][\mathtt{ret}\ V][\gamma]$ either errors (in which case the whole term errors), or runs to a value V', in which case

$$x \leftarrow \mathtt{ret}\ V'; \mathtt{ret}\ \mathtt{inj}_\sigma\ x \mapsto^* \mathtt{ret}\ \mathtt{inj}_\sigma\ V'$$

   (b) The downcast definition expands as follows:

$$[\![\langle \mathrm{tag}_G(AG^{\sqsubseteq})\rangle\downarrow]\!][\mathtt{ret}\ V][\gamma] = x \leftarrow \mathtt{ret}\ V; \mathtt{match}\ x\ \mathtt{with}\ \sigma\{\mathtt{inj}\ y.[\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ y][\gamma] \mid \mho\}$$

   Since $\Sigma \mid \cdot V : \mathrm{OSum}$, $V = \mathtt{inj}_{\sigma'}\ V'$ for some $\sigma' \in \Sigma$.
   (i) If $\sigma' = \sigma$, then

$$\mathtt{match}\ \mathtt{inj}_\sigma\ V'\ \mathtt{with}\ \sigma\{\mathtt{inj}\ y.[\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ y][\gamma] \mid \mho\} \mapsto^1 [\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V'][\gamma]$$

   and then it follows by inductive hypothesis with $AG^{\sqsubseteq}$.
   (ii) If $\sigma' \neq \sigma$, then

$$\mathtt{match}\ \mathtt{inj}_\sigma\ V'\ \mathtt{with}\ \sigma\{\mathtt{inj}\ y.[\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ y][\gamma] \mid \mho\} \mapsto^1 \mho$$

   and the result holds.
(3) If $A^{\sqsubseteq} = A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}$, we consider the downcast case, the upcast is entirely symmetric. First,

$$\begin{array}{l}[\![\langle A_1^{\sqsubseteq} \times A_2^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V][\gamma] = x \leftarrow \mathtt{ret}\ V;\\ \qquad\qquad\qquad\qquad\quad \mathtt{let}\ (x_1, x_2) = x;\\ \qquad\qquad\qquad\qquad\quad y_1 \leftarrow [\![\langle A_1^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ x_1][\gamma];\\ \qquad\qquad\qquad\qquad\quad y_2 \leftarrow [\![\langle A_2^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ x_2][\gamma];\\ \qquad\qquad\qquad\qquad\quad \mathtt{ret}\ (y_1, y_2)\end{array}$$

   Next, since V is well-typed, $V = (V_1, V_2)$. Then,

$$\begin{array}{ll}x \leftarrow \mathtt{ret}\ V; & \mapsto^0 y_1 \leftarrow [\![\langle A_1^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V_1][\gamma];\\ \mathtt{let}\ (x_1, x_2) = x; & \quad y_2 \leftarrow [\![\langle A_2^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ V_2][\gamma];\\ y_1 \leftarrow [\![\langle A_1^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ x_1][\gamma]; & \quad \mathtt{ret}\ (y_1, y_2)\\ y_2 \leftarrow [\![\langle A_2^{\sqsubseteq}\rangle\downarrow]\!][\mathtt{ret}\ x_2][\gamma]; & \\ \mathtt{ret}\ (y_1, y_2) & \end{array}$$

Applying the inductive hypothesis to $A_1^{\sqsubseteq}$, either $[\![\langle A_1^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V_1][\gamma]$ errors (in which case the whole term errors), or it runs to a value V_1'. Then we need to show

$$y_2 \leftarrow [\![\langle A_2^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V_2][\gamma]; \\ \texttt{ret } (V_1', y_2)$$

errors or terminates. Applying the inductive hypothesis to $A_2^{\sqsubseteq}$, either $[\![\langle A_2^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V_2][\gamma]$ errors (in which case the whole term errors), or it runs to a value V_2'. Then the whole term runs to $\texttt{ret } (V_1', V_2')$.

(4) If $A^{\sqsubseteq} = A_i^{\sqsubseteq} \to A_o^{\sqsubseteq}$, we consider the downcast case (upcast is symmetric). The downcast definition expands as follows:

$$[\![\langle A_i^{\sqsubseteq} \to A_o^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V][\gamma] = f \leftarrow \texttt{ret } V; \texttt{ret thunk } \lambda x.[\![\langle A_o^{\sqsubseteq}\rangle\downarrow]\!][y \leftarrow [\![\langle A_i^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret } x][\gamma]; \texttt{force } f\ y][\gamma]$$

Which steps immediately to a value.

(5) If $A^{\sqsubseteq} = \forall^{v} X.A_o^{\sqsubseteq}$, then it follows by similar reasoning to the function case, that is, it immediately terminates.

(6) If $A^{\sqsubseteq} = \exists^{v} X.A_o^{\sqsubseteq}$, we consider the downcast case (upcast is symmetric). The definition expands as follows:

$$[\![\langle \exists^{v} X.A_o^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V][\gamma] = \texttt{unpack } (X, x) = \texttt{ret } V; \texttt{ret thunk } \lambda c_X.[\![\langle A_o^{\sqsubseteq}\rangle\downarrow]\!][\texttt{force } x\ c_X]$$

Which steps immediately to a value.

□

Lemma G.18 (Cast Right). *For any* $\Gamma^{\sqsubseteq} : \Gamma$, *if* $\Gamma^{\sqsubseteq} \vdash AC^{\sqsubseteq} : A \sqsubseteq C$ *and* $\Gamma^{\sqsubseteq} \vdash AB^{\sqsubseteq} : A \sqsubseteq B$, *and* $\Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C$, *Then if* $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$,

(1) *If* $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![AB^{\sqsubseteq}]\!]\gamma\delta$, *then* $(w, \texttt{ret } V_l, [\![\langle BC^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret } V_r][\gamma_r]) \in \mathcal{E}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$

(2) *If* $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$, *then* $(w, \texttt{ret } V_l, [\![\langle BC^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } V_r][\gamma_r]) \in \mathcal{E}^{\sim}[\![AB^{\sqsubseteq}]\!]\gamma\delta$

Proof. By induction on $BC^{\sqsubseteq}$.

(1) If $BC^{\sqsubseteq} \in \{\mathbb{B}, ?, X\}$, then the cast is trivial.

(2) If $BC^{\sqsubseteq} = \text{tag}_G(BG^{\sqsubseteq})$, then $AC^{\sqsubseteq} = \text{tag}_G(AG^{\sqsubseteq})$.

(a) For the upcast case, we are given that $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![AB^{\sqsubseteq}]\!]\gamma\delta$ and we need to prove that

$$(w, \texttt{ret } V_l, y \leftarrow [\![\langle BG^{\sqsubseteq}\rangle\updownarrow]\!][V_r][\gamma_r]; \texttt{ret inj}_G\ y) \in \mathcal{E}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

By inductive hypothesis, we know

$$(w, \texttt{ret } V_l, [\![\langle BG^{\sqsubseteq}\rangle\updownarrow]\!]V_r[\gamma_r]) \in \mathcal{E}^{\sim}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

We then use monadic bind (lemma G.12). Suppose $w' \sqsupseteq w$ and $(w', V_l', V_r') \in \mathcal{V}^{\sim}[\![AG^{\sqsubseteq}]\!]\gamma\delta$. We need to show that

$$(w', \texttt{ret } V_l', y \leftarrow \texttt{ret } V_r'; \texttt{ret inj}_G\ y) \in \mathcal{E}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

By anti reduction, it is sufficient to show

$$(w', V_l, \texttt{inj}_{\gamma_r(G)}\ V_r) \in \mathcal{V}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

(i) If $\sim = \prec$, we need to show $(w', V_l, V_r) \in \mathcal{V}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$, which follows by inductive hypothesis.

(ii) If $\sim = \succ$, we need to show $(w', V_l, V_r) \in \triangleright\mathcal{V}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$, that is for any $w'' \sqsupset w'$, $(w'', V_l, V_r) \in \mathcal{V}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$ which also follows by inductive hypothesis.

(b) For the downcast case, we know $(V_l, V_r) \in \mathcal{V}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]w\gamma$. Let $\sigma_r = \gamma_r([\![G]\!]_{tag})$.

(i) In the $\prec$ case, we know $V_r = \mathtt{inj}_{\sigma_r}\ V_r'$ and

$$(w, V_l, V_r') \in \mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

we need to show

$$(w, \mathtt{ret}\ V_l, \mathtt{match}\ (\mathtt{inj}_{\sigma_r}\ V_r')\ \mathtt{with}\ \sigma_r\{\mathtt{inj}\ x.[\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][x][\gamma_r] \mid \mho\}) \in \mathcal{E}^{\prec}[\![AB^{\sqsubseteq}]\!]\gamma\delta$$

the right hand side reduces to $[\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][V'']$, so by anti-reduction it is sufficient to show

$$(w, \mathtt{ret}\ V_l, [\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][V_r']) \in \mathcal{E}^{\prec}[\![AB^{\sqsubseteq}]\!]\gamma\delta$$

which follows by inductive hypothesis.

(ii) In the $\succ$ case, we know $V_r = \mathtt{inj}_{\sigma_r}\ V_r'$ and

$$(w, V, V_r') \in \triangleright(\mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta)$$

(note the $\triangleright$). We need to show

$$(w, \mathtt{ret}\ V_l, \mathtt{match}\ (\mathtt{inj}_{\sigma_r}\ V_r')\ \mathtt{with}\ \sigma_r\{\mathtt{inj}\ x.[\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][x][\gamma_r] \mid \mho\}) \in \mathcal{E}^{\succ}[\![AB^{\sqsubseteq}]\!]\gamma\delta$$

the right hand side takes 1 step to $[\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][V_r']$.

(A) If $w.j = 0$, then we are done.

(B) Otherwise, define $w' = (w.j - 1, w.\Sigma_l, w.\Sigma_r, \lfloor w.\eta \rfloor_{w.j-1})$. Then by anti-reduction, it is sufficient to show

$$(w', \mathtt{ret}\ V_l, [\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][V_r']) \in \mathcal{E}^{\succ}[\![AB^{\sqsubseteq}]\!]\gamma\delta$$

$$(w', \mathtt{ret}\ V_l, [\![\langle BG^{\sqsubseteq}\rangle\!\downarrow]\!][V_r']) \in \mathcal{E}^{\succ}[\![AB^{\sqsubseteq}]\!]$$

By inductive hypothesis, it is sufficient to show

$$(w', V_l, V_r') \in \mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

which follows by assumption because $w' \sqsupset w$.

(3) If $BC^{\sqsubseteq} = BC_1^{\sqsubseteq} \times BC_2^{\sqsubseteq}$, then by precision inversion also $AC^{\sqsubseteq} = AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}$ and $AB^{\sqsubseteq} = AB_1^{\sqsubseteq} \times AB_2^{\sqsubseteq}$. We consider the upcast case, the downcast case follows by an entirely analogous argument.

Given $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![BC_1^{\sqsubseteq} \times BC_2^{\sqsubseteq}]\!]\gamma\delta$, we need to show

$$(w, \mathtt{ret}\ V_l, [\![\langle BC_1^{\sqsubseteq} \times BC_2^{\sqsubseteq}\rangle\!\updownarrow]\!][\mathtt{ret}\ V_l][\gamma_r]) \in \mathcal{E}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]\gamma\delta$$

Expanding definitions, and applying anti-reductino , this reduces to showing

$$\begin{array}{ll}(w, \mathtt{ret}\ V_l, \mathtt{let}\ (y_1, y_2) = V_r; &) \in \mathcal{E}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]\gamma\delta \\ \quad z_1 \leftarrow [\![\langle BC_1^{\sqsubseteq}\rangle\!\downarrow]\!][\mathtt{ret}\ y_1][\gamma_r]; & \\ \quad z_2 \leftarrow [\![\langle BC_2^{\sqsubseteq}\rangle\!\downarrow]\!][\mathtt{ret}\ y_2][\gamma_r]; & \\ \quad \mathtt{ret}\ (z_1, z_2) & \end{array}$$

Since $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![BC_1^{\sqsubseteq} \times BC_2^{\sqsubseteq}]\!]\gamma\delta$, we know

$$V_l = (V_{l1}, V_{l2}) \qquad V_r = (V_{r1}, V_{r2})$$

$$(V_{l1}, V_{r1}) \in \mathcal{V}^{\sim}[\![BC_1^{\sqsubseteq}]\!]\gamma\delta \qquad (V_{l2}, V_{r2}) \in \mathcal{V}^{\sim}[\![BC_2^{\sqsubseteq}]\!]\gamma\delta$$

So after a reduction we need to show

$$\begin{array}{ll}(w, \mathtt{ret}\ V_l, z_1 \leftarrow [\![\langle BC_1^{\sqsubseteq}\rangle\!\downarrow]\!][\mathtt{ret}\ V_{r1}][\gamma_r];) & \in \mathcal{E}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]\gamma\delta \\ \quad z_2 \leftarrow [\![\langle BC_2^{\sqsubseteq}\rangle\!\downarrow]\!][\mathtt{ret}\ V_{r2}][\gamma_r]; & \\ \quad \mathtt{ret}\ (z_1, z_2) & \end{array}$$

By forward reduction, it is sufficient to prove the following, (which is amenable to monadic bind):

$$\begin{array}{l}(w, z \leftarrow \texttt{ret}\ V_{l1};_1, z_1 \leftarrow [\![\langle BC_1^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ V_{r1}][\gamma_r];) \in \mathcal{E}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]\gamma\delta w\gamma \\ \quad z_2 \leftarrow \texttt{ret}\ V_{l2};\ z_2 \leftarrow [\![\langle BC_2^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ V_{r2}][\gamma_r]; \\ \quad \texttt{ret}\ (z_1, z_2) \quad \texttt{ret}\ (z_1, z_2)\end{array}$$

We then apply monadic bind with the inductive hypothesis for $BC_1^{\sqsubseteq}$. Given $w' \sqsupseteq w$ and $(V'_{l1}, V'_{r1}) \in \mathcal{V}^{\sim}[\![AC_1^{\sqsubseteq}]\!]\gamma\delta$ the goal reduces to

$$\begin{array}{l}(w', z_2 \leftarrow \texttt{ret}\ V_{l2};, z_2 \leftarrow [\![\langle BC_2^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ V_{r2}][\gamma_r];) \in \mathcal{E}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]\gamma\delta \\ \quad \texttt{ret}\ (V'_{l1}, z_2) \quad \texttt{ret}\ (V'_{r1}, z_2)\end{array}$$

We then apply another monadic bind with the inductive hypothesis for $BC_2^{\sqsubseteq}$. Given $w'' \sqsupseteq w'$ and $(V'_{l2}, V'_{r2}) \in \mathcal{V}^{\sim}[\![AC_2^{\sqsubseteq}]\!]w'\gamma$, the goal reduces to

$$(w'', (V'_{l1}, V'_{l2}), (V'_{r1}, V'_{r2})) \in \mathcal{V}^{\sim}[\![AC_1^{\sqsubseteq} \times AC_2^{\sqsubseteq}]\!]w''\gamma$$

which follows immediately by our assumptions from monadic bind.

(4) If $BC^{\sqsubseteq} = BC_i^{\sqsubseteq} \to BC_o^{\sqsubseteq}$, then by precision inversion also $AC^{\sqsubseteq} = AC_i^{\sqsubseteq} \to AC_o^{\sqsubseteq}$ and $AB^{\sqsubseteq} = AB_i^{\sqsubseteq} \to AB_o^{\sqsubseteq}$. We consider the upcast case, the downcast case follows by an entirely analogous argument.
Given $(V_l, V_r) \in \mathcal{V}^{\sim}[\![BC_i^{\sqsubseteq} \to BC_o^{\sqsubseteq}]\!]\gamma\delta$, we need to show

$$(w, \texttt{ret}\ V_l, [\![\langle BC_i^{\sqsubseteq} \to BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ V_r][\gamma_r]) \in \mathcal{E}^{\sim}[\![AC_i^{\sqsubseteq} \to AC_o^{\sqsubseteq}]\!]\gamma\delta$$

Expanding definitions, this reduces to showing

$$\begin{array}{l}(w, V_l, \texttt{thunk}\ (\lambda x.y \leftarrow [\![\langle BC_i^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ x][\gamma_r];)) \in \mathcal{V}^{\sim}[\![AC_i^{\sqsubseteq} \to AC_o^{\sqsubseteq}]\!]\gamma\delta \\ \qquad\qquad z \leftarrow \texttt{force}\ V_r y; \\ \qquad\qquad [\![\langle BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ z][\gamma_r]\end{array}$$

Let $w' \sqsupseteq w$ be a future world and $(w', V_{li}, V_{ri}) \in \mathcal{V}^{\sim}[\![AC_i^{\sqsubseteq}]\!]\gamma\delta$. Then our goal reduces to showing

$$\begin{array}{l}(w', \texttt{force}\ V_l V_{li}, y \leftarrow [\![\langle BC_i^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ V_{ri}][\gamma_r];) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma\delta \\ \qquad\qquad z \leftarrow \texttt{force}\ V_r y; \\ \qquad\qquad [\![\langle BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ z][\gamma_r]\end{array}$$

by forward reduction, it is sufficient to show

$$\begin{array}{l}(w', y \leftarrow V_{li};\ \ , y \leftarrow [\![\langle BC_i^{\sqsubseteq}\rangle{\downarrow}]\!][\texttt{ret}\ V_{ri}][\gamma_r];) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma\delta \\ \quad \texttt{force}\ V_l y\ z \leftarrow \texttt{force}\ V_r y; \\ \qquad\qquad [\![\langle BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ z][\gamma_r]\end{array}$$

We then use the inductive hypothesis on $BC_i^{\sqsubseteq}$ (which applies because of downward-closure) and monadic bind: assume $w'' \sqsupseteq w'$ and $(w'', V'_{li}, V'_{ri}) \in \mathcal{V}^{\sim}[\![AC_i^{\sqsubseteq}]\!]\gamma\delta$. We need to show

$$\begin{array}{l}(w'', \texttt{force}\ V_l V'_{li}, z \leftarrow \texttt{force}\ V_r V'_{ri};\quad) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma\delta \\ \qquad\qquad [\![\langle BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ z][\gamma_r]\end{array}$$

We apply monadic bind again, noting that the applications are related by assumption and downward closure. Assume $w''' \sqsupseteq w''$ and $(w''', V_{lo}, V_{ro}) \in \mathcal{V}^{\sim}[\![AB_o^{\sqsubseteq}]\!]\gamma\delta$. By anti-reduction , the goal reduces to showing

$$(\texttt{ret}\ V_o, [\![\langle BC_o^{\sqsubseteq}\rangle{\updownarrow}]\!][\texttt{ret}\ V'_o][\gamma]) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma\delta$$

which follows by inductive hypothesis for $BC_o^{\sqsubseteq}$.

(5) If $BC^{\sqsubseteq} = \forall^{\nu} X.BC_o^{\sqsubseteq}$, then by precision inversion also $AC^{\sqsubseteq} = \forall^{\nu} X.AC_o^{\sqsubseteq}$ and $AB^{\sqsubseteq} = \forall^{\nu} X.AB_o^{\sqsubseteq}$. We consider the upcast case, the downcast case follows by an entirely analogous argument. Given $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![\forall^{\nu} X.BC_o^{\sqsubseteq}]\!]\gamma\delta$, we need to show

$$(w, \texttt{ret}\ V_l, [\![\langle \forall^{\nu} X.BC_o^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret}\ V_r][\gamma_r]) \in \mathcal{E}^{\sim}[\![\forall^{\nu} AC_o^{\sqsubseteq}]\!]\gamma\delta$$

Expanding definitions and applying anti-reduction, this reduces to showing

$$(w, V_l, V_r') \in \mathcal{V}^{\sim}[\![\forall^{\nu} AC_o^{\sqsubseteq}]\!]\gamma\delta$$

where

$$V_r' = \texttt{thunk}\ (\Lambda X.\lambda c_X : \text{Case}\ X.[\![\langle BC_o^{\sqsubseteq}\rangle\downarrow]\!][(\texttt{force}\ V_r)\, X\, c_X][\gamma_r])$$

Let $w' \sqsupseteq w$, $R \in Rel[A_l, A_r]$, and $w.\eta \vDash (\sigma_l, \sigma_r, \lfloor R\rfloor_{w.j})$, then we need to show that

$$(w', (\texttt{force}\ V_l)\, A_l\, \sigma_l, (\texttt{force}\ V_r')\, A_r\, \sigma_r) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma'\delta'$$

where $\gamma' = (\gamma, c_X \mapsto (\sigma_l, \sigma_r))$ and $\delta' = (\delta, X \mapsto (A_l, A_r, R))$ which reduces in 0 steps to showing

$$(w', (\texttt{force}\ V_l)\, A_l\, \sigma_l, [\![\langle BC_o^{\sqsubseteq}\rangle\downarrow]\!][(\texttt{force}\ V_r)\, A_r\, \sigma][\gamma_r']) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma'\delta'$$

by noting that by definition, $\gamma_r' = \gamma_r, c_X \mapsto \sigma_r$.
Then, we invoke monadic bind using $(w', (\texttt{force}\ V_l)\, A_l\, \sigma_l, (\texttt{force}\ V_r)\, A_r\, \sigma_r) \in \mathcal{E}^{\sim}[\![BC_o^{\sqsubseteq}]\!]\gamma'\delta'$. Let $w'' \sqsupseteq w'$ and $(w'', V_{lo}, V_{ro}) \in \mathcal{V}^{\sim}[\![BC_o^{\sqsubseteq}]\!]\gamma'\delta'$. We then need to show

$$(w'', \texttt{ret}\ V_{lo}, [\![\langle BC_o^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret}\ V_{ro}][\gamma_r'] \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma'\delta')$$

which follows by inductive hypothesis.

(6) If $BC^{\sqsubseteq} = \exists^{\nu} X.BC_o^{\sqsubseteq}$, then by precision inversion also $AC^{\sqsubseteq} = \exists^{\nu} X.AC_o^{\sqsubseteq}$ and $AB^{\sqsubseteq} = \exists^{\nu} X.AB_o^{\sqsubseteq}$. We consider the upcast case, the downcast case follows by an entirely analogous argument. Given $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![\exists^{\nu} X.BC_o^{\sqsubseteq}]\!]\gamma\delta$, we need to show

$$(w, \texttt{ret}\ V_l, [\![\langle \exists^{\nu} X.BC_o^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret}\ V_r][\gamma_r]) \in \mathcal{E}^{\sim}[\![\exists^{\nu} AC_o^{\sqsubseteq}]\!]\gamma\delta$$

Expanding definitions and applying anti-reduction, this reduces to showing

$$(w, \texttt{ret}\ V_l, \texttt{unpack}\ (X, y) = \texttt{ret}\ V_r;\ \texttt{ret pack}(X, ())\texttt{thunk}\ (\lambda c_X : \text{Case}\ X.[\![\langle BC_o^{\sqsubseteq}\rangle\updownarrow]\!][(\texttt{force}\ y)\, c_X][\gamma_r])) \in \mathcal{E}^{\sim}[\![\exists^{\nu}.AC_o^{\sqsubseteq}]\!]\gamma\delta$$

By definition of $\mathcal{V}^{\sim}[\![\exists^{\nu} X.BC_o^{\sqsubseteq}]\!]\gamma\delta$, we know

$$V_l = \texttt{pack}(A_l, V_l')$$

$$V_r = \texttt{pack}(A_r, V_r')$$

and there is an associated relation $R \in \text{Rel}_{\omega}[A_l, A_r]$. Then the goal reduces to showing

$$(\texttt{pack}(A_l, V_l'), \texttt{pack}(A_l, ()\texttt{thunk}\ (\lambda c_X.[\![\langle BC_o^{\sqsubseteq}\rangle\updownarrow]\!][(\texttt{force}\ V_r')\, c_X][\gamma_r]))) \in \mathcal{V}^{\sim}[\![\exists^{\nu} X.AC_o^{\sqsubseteq}]\!]\gamma\delta$$

we choose R as the relation for X, and then we need to show (after applying anti-reduction) that for any $w' \sqsupseteq w$, $w' \vDash (\sigma_l, \sigma_r, \lfloor R\rfloor_{w'.j})$ that

$$(w', (\texttt{force}\ V_l')\, \sigma, [\![\langle BC_o^{\sqsubseteq}\rangle\updownarrow]\!][(\texttt{force}\ V_r')\, \sigma][\gamma_r']) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma'\delta'$$

where $\gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r)$. and $\delta' = \delta, X \mapsto (A_l, A_r, R)$. We use the relatedness assumption and monadic bind again. Then we are given $w'' \sqsupseteq w'$, and $(V_{lo}, V_{ro}) \in \mathcal{V}^{\sim}[\![AB_o^{\sqsubseteq}]\!]\gamma'\delta'$ and need to show

$$(w'', \texttt{ret}\ V_{lo}, [\![\langle BC_o^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret}\ V_{ro}]) \in \mathcal{E}^{\sim}[\![AC_o^{\sqsubseteq}]\!]\gamma'\delta'$$

which follows by inductive hypothesis.

□

Lemma G.19 (Cast Left). *For any* $\Gamma^{\sqsubseteq} : \Gamma, \Gamma^{\sqsubseteq} \vdash AC^{\sqsubseteq} : A \sqsubseteq C, \Gamma \vdash AB^{\sqsubseteq} : A \sqsubseteq B, \Gamma^{\sqsubseteq} \vdash BC^{\sqsubseteq} : B \sqsubseteq C$ *and* $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$,

(1) *If* $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![BC^{\sqsubseteq}]\!]\gamma\delta$, *then* $(w, [\![\langle AB^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret}\ V_l][\gamma_l], \texttt{ret}\ V_r) \in \mathcal{E}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$
(2) *If* $(V_l, V_r) \in \mathcal{V}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$, *then* $(w, [\![\langle AB^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l], \texttt{ret}\ V_r) \in \mathcal{E}^{\sim}[\![BC^{\sqsubseteq}]\!]\gamma\delta$

Proof. By nested induction on $AB^{\sqsubseteq}$ and $AC^{\sqsubseteq}$, i.e., if $AB^{\sqsubseteq}$ becomes smaller $AC^{\sqsubseteq}$ can be anything but if $AC^{\sqsubseteq}$ becomes smaller, then $AB^{\sqsubseteq}$ must stay the same.

(1) If $AC^{\sqsubseteq} \in \{\mathbb{B}, AC_0^{\sqsubseteq} \times AC_1^{\sqsubseteq}, AC_i^{\sqsubseteq} \to AC_o^{\sqsubseteq}, \forall^{\nu} X.AC_o^{\sqsubseteq}, \exists^{\nu} X.AC_o^{\sqsubseteq}\}$, then $AB^{\sqsubseteq}$ has the same top-level connective, and the proof is symmetric to the case of lemma G.18, which always makes $AB^{\sqsubseteq}$ and $AC^{\sqsubseteq}$ smaller in uses of the inductive hypothesis.
(2) If $AC^{\sqsubseteq} = ?$, then also $AB^{\sqsubseteq} = BC^{\sqsubseteq} = ?$ and the cast is trivial.
(3) If $AC^{\sqsubseteq} = \mathrm{tag}_G(AG^{\sqsubseteq})$, there are two cases: either $BC^{\sqsubseteq} = ?$ or $BC^{\sqsubseteq} = \mathrm{tag}_G(BG^{\sqsubseteq})$.
  (a) If $BC^{\sqsubseteq} = ?$, then $AB^{\sqsubseteq} = \mathrm{tag}_G(AG^{\sqsubseteq}) = AC^{\sqsubseteq}$. Define $\sigma_l = \gamma_l([\![G]\!]_{tag})$, $\sigma_r = \gamma_r([\![G]\!]_{tag})$.
    (i) In the upcast case, we know $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![\mathrm{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$. In which case, $V_r = \texttt{inj}_{\sigma_r}\ V_r'$
      (A) In the $\prec$ case, we know $(w, V_l, V_r') \in \mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta$, and we need to show

$$(w, (x \leftarrow [\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l]; \texttt{ret}\ \texttt{inj}_{\sigma_l}\ x), \texttt{ret}\ \texttt{inj}_{\sigma_r}\ V_r') \in \mathcal{E}^{\prec}[\![?]\!]\gamma\delta$$

which by forward reduction is equivalent to showing

$$(w, (x \leftarrow [\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l]; \texttt{ret}\ \texttt{inj}_{\sigma_l}\ x), x \leftarrow \texttt{ret}\ V_r'; \texttt{ret}\ \texttt{inj}_{\sigma_r}\ x) \in \mathcal{E}^{\prec}[\![?]\!]\gamma\delta$$

By inductive hypothesis, we know

$$(w, [\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l], \texttt{ret}\ V_r') \in \delta_R([\![G]\!]_{tag})$$

so can we apply monadic bind. Let $w' \sqsupseteq w$, and $(w', V_l', V_r'') \in \mathcal{V}^{\prec}[\![G]\!]\gamma\delta$. Then we need to show (after applying anti-reduction)

$$(w', \texttt{inj}_{\sigma_l}\ V_l', \texttt{inj}_{\sigma_r}\ V_r'') \in \mathcal{V}^{\prec}[\![?]\!]\gamma\delta$$

To do this, we need to give a relation R such that $w' \vDash (\sigma_l, \sigma_r, R)$ and $(w', V_l', V_r'') \in \triangleright R$. Since $\gamma(c_{[\![G]\!]_{tag}}) = (\sigma_l, \sigma_r)$, we know $R = \lfloor \delta_R([\![G]\!]_{tag})\rfloor_{w'.j}$. And we need to show that for any $w'' \sqsupseteq w'$, that $(w'', V_l', V_r'') \in \lfloor R\rfloor_{w'.j}$. Which follows by monotonicity because $w'' \sqsupseteq w$.

      (B) The $\succ$ case is slightly more complicated. This time we only know we know $(w, V_l, V_r') \in \triangleright\mathcal{V}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$ (note the $\triangleright$), and we need to show

$$(w, (x \leftarrow [\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l]; \texttt{ret}\ \texttt{inj}_{\sigma_l}\ x), \texttt{ret}\ \texttt{inj}_{\sigma_r}\ V_r') \in \mathcal{E}^{\succ}[\![?]\!]\gamma\delta$$

By lemma G.17, we know $[\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l]$ either runs to error or terminates. If it runs to an error then our goal holds. Otherwise, let $[\![\langle AG^{\sqsubseteq}\rangle\uparrow]\!][\texttt{ret}\ V_l][\gamma_l] \mapsto^* \texttt{ret}\ V_l'$. Applying anti-reduction, we need to show

$$(w, \texttt{inj}_{\sigma_l}\ V_l', \texttt{inj}_{\sigma_r}\ V_r') \in \mathcal{V}^{\succ}[\![?]\!]\gamma\delta$$

by the same reasoning as above, we need to show

$$(w, V_l', V_r') \in \triangleright\lfloor \delta_R([\![G]\!]_{tag})\rfloor_{w.j}$$

Let $w' \sqsupset w$. We need to show

$$(w', V_l', V_r') \in \lfloor \delta_R([\![G]\!]_{tag})\rfloor_{w.j}$$

By our assumption, we know

$$(w', V_l, V_r') \in \mathcal{V}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

so by inductive hypothesis, we know

$$(w', [\![\langle AG^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret } V_l][\gamma_l], \texttt{ret } V_r') \in \mathcal{E}^{\succ}[\![G]\!]\gamma\delta = \delta_R([\![G]\!]_{tag})$$

And since we know $w'.\Sigma_l, [\![\langle AG^{\sqsubseteq}\rangle\updownarrow]\!][\texttt{ret } V_l][\gamma_l] \mapsto^* w'.\Sigma_l, \texttt{ret } V_l'$ by lemma G.17, this means

$$(w', V_l', V_r') \in \mathcal{V}^{\succ}[\![G]\!]\gamma\delta = \delta_R([\![G]\!]_{tag})$$

so the result follows by lemma G.14.

(ii) For the downcast case, we know $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![?]\!]\gamma\delta$ which means there exists σ_l, σ_r, R with $w.\eta \vDash (\sigma_l, \sigma_r, R)$ and $V_l = \texttt{inj}_{\sigma_l}\ V_l'$ and $V_r = \texttt{inj}_{\sigma_r}\ V_r'$ and

$$(w, V_l', V_r') \in \triangleright R$$

Expanding the definition of the cast and applying anti-reduction, we need to show

$$(w, \texttt{match } V_l \texttt{ with } \sigma_l\{\texttt{inj } x.[\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } x] \mid \mho\}, \texttt{ret } V_r) \in \mathcal{E}^{\sim}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

If $\gamma_l([\![G]\!]_{tag}) = \sigma_l$, the left side errors and the result holds. Otherwise,

$$\texttt{match } V_l \texttt{ with } \sigma_l\{\texttt{inj } x.[\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!][\texttt{ret } x] \mid \mho\} \mapsto^1 [\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!]\texttt{ret } V_l'$$

(A) In the $\succ$ case, we need to show

$$(w, [\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!]\texttt{ret } V_l', \texttt{ret } V_r) \in \mathcal{E}^{\succ}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

Which follows by inductive hypothesis if we can show

$$(w, V_l', V_r) \in \mathcal{V}^{\succ}[\![\text{tag}_G(G)]\!]\gamma\delta$$

which reduces to showing

$$(w, V_l', V_r') \in \triangleright\mathcal{V}^{\succ}[\![G]\!]\gamma\delta$$

So let $w' \sqsupset w$. We need to show

$$(w', V_l', V_r') \in \mathcal{V}^{\succ}[\![G]\!]\gamma\delta$$

But we know $(w', V_l', V_r') \in R$ where $R = \lfloor\delta_R([\![G]\!]_{tag})\rfloor_{w.j}$ so the result follows by lemma G.15.

(B) In the $\prec$ case, we check $w.j$

- If $w.j = 0$, then the left side takes 1 step so the result holds.
- Otherwise, define $w' = (w.j - 1, w.\Sigma_l, w.\Sigma_r, \lfloor w.\eta\rfloor_{w.j-1})$. Then it is sufficient to show

$$(w', [\![\langle AG^{\sqsubseteq}\rangle\downarrow]\!]\texttt{ret } V_l', \texttt{ret } V_r) \in \mathcal{E}^{\prec}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

To apply the inductive hypothesis we need to show

$$(w', V_l', \texttt{ret } V_r) \in \mathcal{V}^{\prec}[\![\text{tag}_G(G)]\!]\gamma\delta$$

Unrolling definitions, it is sufficient to show

$$(w', V_l', V_r') \in \mathcal{V}^{\prec}[\![G]\!]\gamma\delta$$

But we know already that $(w', V_l', V_r') \in R$ where $R = \lfloor\delta_R([\![G]\!]_{tag})\rfloor_{w.j}$ so the result follows.

(b) Finally, if $BC^{\sqsubseteq} = \text{tag}_G(BG^{\sqsubseteq})$, then $AC^{\sqsubseteq} = \text{tag}_G(AG^{\sqsubseteq})$. We consider the downcast case, the upcast case is entirely symmetric. Let $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![\text{tag}_G(BG^{\sqsubseteq})]\!]\gamma\delta$. Then we know $V_r = \texttt{inj}_{\sigma_r}\ V_r'$ where $\sigma_r = \gamma_r([\![G]\!]_{tag})$.

(i) If $\sim = \prec$, we furthermore know $(w, V_l, V_r') \in \mathcal{V}^{\prec}[\![BG^{\sqsubseteq}]\!]\gamma\delta$. We need to show that

$$(w, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l], \texttt{ret inj}_{\sigma_r}\ V_r') \in \mathcal{E}^{\prec}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

by forward reduction it is sufficient to show

$$(w, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l], x \leftarrow \texttt{ret } V_r'; \texttt{ret inj}_{\sigma_r}\ x) \in \mathcal{E}^{\prec}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

We know by inductive hypothesis that

$$(w, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l], \texttt{ret } V_r)$$

so we can apply monadic bind. Let $w' \sqsupseteq w$, and $(w', V_l', V_r'') \in \mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta$. Then we need to show (after applying anti-reduction) that

$$(w', V_l', \texttt{inj}_{\sigma_r}\ V_r'') \in \mathcal{V}^{\prec}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

Which, unrolling the definition, is

$$(w', V_l', V_r'') \in \mathcal{V}^{\prec}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

which was our assumption.

(ii) If $\sim = \succ$, we only know $(w, V_l, V_r') \in \triangleright\mathcal{V}^{\succ}[\![BG^{\sqsubseteq}]\!]\gamma\delta$ (note the $\triangleright$).

$$(w, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l], \texttt{ret inj}_{\sigma_r}\ V_r') \in \mathcal{E}^{\succ}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

By lemma G.17, the left hand side either errors or terminates with a value.

(A) If $w.\Sigma_l, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l] \mapsto^* w.\Sigma_l, \mho$, then the result holds.

(B) If $w.\Sigma_l, [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l] \mapsto^* w.\Sigma_l, \texttt{ret } V_l'$, then we need to show that

$$(w, V_l', \texttt{inj}_{\sigma_r}\ V_r') \in \mathcal{V}^{\succ}[\![\text{tag}_G(AG^{\sqsubseteq})]\!]\gamma\delta$$

Which unrolls to

$$(w, V_l', V_r') \in \triangleright\mathcal{V}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

So let $w' \sqsupset w$. We need to show

$$(w', V_l', V_r') \in \mathcal{V}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

By inductive hypothesis, we know

$$(w', [\![\langle AB^{\sqsubseteq}\rangle\!\downarrow]\!][\texttt{ret } V_l][\gamma_l], \texttt{ret } V_r') \in \mathcal{E}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

and so by determinism of evaluation, we know

$$(w', V_l', V_r') \in \mathcal{V}^{\succ}[\![AG^{\sqsubseteq}]\!]\gamma\delta$$

so the result holds.

□

G.2 Compatibility Lemmas

Lemma G.20.

$$\Gamma_1^{\sqsubseteq}, x : A^{\sqsubseteq}, \Gamma_2^{\sqsubseteq} \vDash x \sqsubseteq\sim x \in A^{\sqsubseteq}; \cdot$$

Proof. We need to show

$$(w, \texttt{ret } V_l, \texttt{ret } V_r) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

where $V_i = \gamma_i(x)$. Since both sides are values, it is sufficient to show

$$(w, \gamma_l(x), \gamma_r(x)) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

By definition of $\mathcal{G}^{\sim}[\![\Gamma_1^{\sqsubseteq}, x : A^{\sqsubseteq}, \Gamma_2^{\sqsubseteq}]\!]$, we know $\gamma = \gamma_1, x \mapsto (V_l, V_r), \gamma_2$ and $\delta = \delta_1, \delta_2$ where $(w, \gamma_1, \delta_2) \in \mathcal{G}^{\sim}[\![\Gamma_1^{\sqsubseteq}]\!]$ and $(w, V_l, V_r) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma_1\delta_1$.

Then the result follows because $\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma_1\delta_1 = \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$ by Lemma G.16 □

Lemma G.21. *Generation compatibility.*

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq\sim M_r \in B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong A^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vDash A^{\sqsubseteq} : A_l \sqsubseteq A_r}{\Gamma^{\sqsubseteq} \vDash \mathtt{hide}\ X \cong A_l; M_l \sqsubseteq\sim \mathtt{hide}\ X \cong A_r; M_r \in B^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, \Gamma^{\sqsubseteq\prime\prime}}$$

Proof. The translation is defined as

$$[\![\mathtt{hide}\ X \cong A_i; M_i]\!] = \mathtt{newcase}_{[\![A_i]\!]}\ c_X; [\![M_i]\!]$$

We need to show

$$(w, \mathtt{newcase}_{[\![A_l]\!][\delta_l]}\ c_X; [\![M_l]\!][\gamma_l][\delta_l], \mathtt{newcase}_{[\![A_r]\!][\delta_r]}\ c_X; [\![M_r]\!][\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

Define $w' = (w.j, (w.\Sigma_l, [\![A_l]\!][\delta]), (w.\Sigma_r, [\![A_r]\!][\delta]), \eta\boxplus(w.\Sigma_l.size, w.\Sigma_r.size, \lfloor\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta\rfloor_w.j))$. Then

$$w.\Sigma_l, \mathtt{newcase}_{[\![A_l]\!][\delta_l]}\ c_X; [\![M_l]\!][\gamma_l][\delta_l] \mapsto^0 w'.\Sigma_l, [\![M_l]\!][\gamma_l][\delta_l][w'.\Sigma_l.size/c_X]$$

and similarly for the right side

$$w.\Sigma_r, \mathtt{newcase}_{[\![A_r]\!][\delta_r]}\ c_X; [\![M_r]\!][\gamma_r][\delta_r] \mapsto^0 w'.\Sigma_r, [\![M_r]\!][\gamma_r][\delta_r][w'.\Sigma_r.size/c_X]$$

Then by the anti-reduction lemma G.9, it is sufficient to show

$$(w', [\![M_l]\!][\gamma_l][\delta_l][w'.\Sigma_l.size/c_X], [\![M_r]\!][\gamma_r][\delta_r][w'.\Sigma_r.size/c_X]) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

Define $\gamma' = \gamma, c_X \mapsto (w'.\Sigma_l.size, w'.\Sigma_r.size)$ and $\delta' = \delta, X \mapsto (\delta_l(A_l), \delta_r(A_r), \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta\gamma\delta)$. Then $(w', \gamma', \delta') \in \mathcal{G}^{\sim}[\![\Gamma_p, \Gamma^{\sqsubseteq\prime}, X \cong A, \Gamma^{\sqsubseteq\prime\prime}]\!]$ since $(w', \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_p, \Gamma^{\sqsubseteq\prime}, \Gamma^{\sqsubseteq\prime\prime}]\!]$ (by monotonicity lemma G.7). Furthermore, $[\![M_i]\!][\gamma_i][\delta_i][w'.\Sigma_i.size/c_X] = [\![M_i]\!][\gamma_i'][\delta_i']$, so the result is equivalent to

$$(w', [\![M_l]\!][\gamma_l'][\delta_l'], [\![M_r]\!][\gamma_r'][\delta_r']) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

which is equivalent to showing

$$(w', [\![M_l]\!][\gamma_l'][\delta_l'], [\![M_r]\!][\gamma_r'][\delta_r']) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma'\delta'$$

by lemma G.16. Finally this result holds by applying the hypothesis. □

Lemma G.22 (Compatibility: Upcast Left).

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vDash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vDash AB^{\sqsubseteq} : A \sqsubseteq B \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vDash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vDash \langle AB^{\sqsubseteq}\rangle\!\uparrow M_l \sqsubseteq M_r \in BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Proof. We need to show

$$(w, [\![\langle AB^{\sqsubseteq}\rangle\!\uparrow]\!][[\![M_l]\!]][\gamma_l][\delta_l], [\![M_r]\!][\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![BC^{\sqsubseteq}]\!]\gamma\delta$$

By assumption, we know

$$(w, [\![M_l]\!][\gamma_l][\delta_l], [\![M_r]\!][\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$$

So we apply monadic bind. Let $w' \sqsupseteq w$ and $(w', V_l, V_r) \in \mathcal{V}^{\sim}[\![AC^{\sqsubseteq}]\!]\gamma\delta$. We need to show

$$(w', [\![\langle AB^{\sqsubseteq}\rangle\!\uparrow]\!][\mathtt{ret}\ V_l][\gamma_l], \mathtt{ret}\ V_r) \in \mathcal{E}^{\sim}[\![BC^{\sqsubseteq}]\!]\gamma\delta$$

which follows by lemma G.19. □

Lemma G.23 (Compatibility: Downcast Left).

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in BC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash AB^{\sqsubseteq} : A \sqsubseteq B \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash BC^{\sqsubseteq} : B \sqsubseteq C}{\Gamma^{\sqsubseteq} \vDash \langle AB^{\sqsubseteq}\rangle\!\downarrow M_l \sqsubseteq M_r \in AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Proof. By same argument as lemma G.22. □

Lemma G.24 (Compatibility: Upcast Right).

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\!\uparrow M_r \in AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Proof. By same argument as lemma G.22, but using lemma G.18. □

Lemma G.25.

$$\frac{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : AC^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AC^{\sqsubseteq} : A \sqsubseteq C \qquad \Gamma, \Gamma' \vdash BC^{\sqsubseteq} : B \sqsubseteq C \qquad \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \vdash AB^{\sqsubseteq} : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq \langle BC^{\sqsubseteq}\rangle\!\downarrow M_r : AB^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Proof. By same argument as lemma G.22, but using lemma G.18. □

Lemma G.26. *If* $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_1^{\sqsubseteq}, X \cong A^{\sqsubseteq}, \Gamma_2^{\sqsubseteq}]\!]$ *of* $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_1^{\sqsubseteq}, X, \Gamma_2^{\sqsubseteq}]\!]$, *then*

$$\mathcal{V}^{\sim}[\![X]\!]\gamma\delta = \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

Proof. By definition, $\gamma = \gamma_1, c_X \mapsto (\sigma_l, \sigma_r), \gamma_2$ and $\delta = \delta_1, X \mapsto (A_l, A_r, R), \delta_2$, where $(w, \gamma_1, \delta_1) \in \mathcal{G}^{\sim}[\![\Gamma_1^{\sqsubseteq}]\!]$. Then

$$\mathcal{V}^{\sim}[\![X]\!]\gamma\delta = \delta(X) = \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma_1\delta_1$$

So it is sufficient to show

$$\mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma_1\delta_1 = \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

which follows by Lemma G.16. □

Lemma G.27. *Seal*

$$\frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathtt{seal}_X M_l \sqsubseteq \mathtt{seal}_X M_r : X; \Gamma^{\sqsubseteq\prime}}$$

Proof. Assume $(M_l[\gamma_l][\delta_l], M_r[\gamma_l][\delta_l]) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$. We need to show

$$(M_l[\gamma_l][\delta_l], M_r[\gamma_l][\delta_l]) \in \mathcal{E}^{\sim}[\![X]\!]\gamma\delta$$

This follows immediately from Lemma G.26 □

Lemma G.28. *UnSeal*

$$\frac{(X \cong A^{\sqsubseteq}) \in \Gamma^{\sqsubseteq}, \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vdash M_l \sqsubseteq M_r : X; \Gamma^{\sqsubseteq\prime}}{\Gamma^{\sqsubseteq} \vdash \mathtt{unseal}_X M_l \sqsubseteq \mathtt{unseal}_X M_r : A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}}$$

Proof. By same reasoning as the seal case. □

Lemma G.29.

$$\Gamma_i \vdash \mathtt{true} \sqsubseteq\sim \mathtt{true} : \mathbb{B} \qquad\qquad \Gamma_i \vdash \mathtt{false} \sqsubseteq\sim \mathtt{false} : \mathbb{B}$$

Proof. The result $(w, \mathtt{ret\ true}, \mathtt{ret\ true}) \in \mathcal{E}^{\sim}[\![\mathbb{B}]\!]\gamma\delta$ follows because $(w, \mathtt{true}, \mathtt{true}) \in \mathcal{V}^{\sim}[\![\mathbb{B}]\!]\gamma\delta$. Similarly for `false`. □

Lemma G.30.

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in \mathbb{B}; \Gamma_0^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_0^{\sqsubseteq} \vDash N_{lt} \sqsubseteq N_{rt} \in B^{\sqsubseteq}; \Gamma^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_0^{\sqsubseteq} \vDash N_{lf} \sqsubseteq N_{rf} \in B^{\sqsubseteq}; \Gamma^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vDash \mathtt{if}\ M_l\ \mathtt{then}\ N_{lt}\ \mathtt{else}\ N_{lf} \sqsubseteq \mathtt{if}\ M_r\ \mathtt{then}\ N_{rt}\ \mathtt{else}\ N_{rf} \in B^{\sqsubseteq}; \Gamma_0^{\sqsubseteq}, \Gamma^{\sqsubseteq}}$$

Proof. Define $M'_l = [\![M_l]\!][\gamma_l][\delta_l]$ and similarly for the rest of the subterms. Then we need to show

$$(w, x \leftarrow M'_l; \mathtt{case}\ x\{x_t.N'_{lt} \mid x_f.N'_{lf}\}, x \leftarrow M'_r; \mathtt{case}\ x\{x_t.N'_{rt} \mid x_f.N'_{rf}\}) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

By assumption and weakening, $(w, M'_l, M'_r) \in \mathcal{E}^{\sim}[\![\mathbb{B}]\!]\gamma\delta$, and we apply monadic bind. Suppose $w' \sqsupseteq w$ and $(w', V_l, V_r) \in \mathcal{V}^{\sim}[\![\mathbb{B}]\!]\gamma\delta$. We need to show

$$(w, x \leftarrow \mathtt{ret}\ V_l; \mathtt{case}\ x\{x_t.N'_{lt} \mid x_f.N'_{lf}\}, x \leftarrow \mathtt{ret}\ V_r; \mathtt{case}\ x\{x_t.N'_{rt} \mid x_f.N'_{rf}\}) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

By definition either $V_l = V_r = \mathtt{true}$ or $V_l = V_r = \mathtt{false}$. WLOG assume it is $\mathtt{true}$. Then

$$x \leftarrow \mathtt{ret}\ V_l; \mathtt{case}\ x\{x_t.N'_{lt} \mid x_f.N'_{lf}\} \mapsto^0 N'_{lt}$$

and similarly for the right side. By anti-reduction (lemma G.9), it is sufficient to show

$$(w', N'_{lt}, N'_{rt}) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

which follows by hypothesis. □

Lemma G.31. *Product intro*

$$\frac{\Gamma^{\sqsubseteq} \vDash M_{l0} \sqsubseteq M_{r0} \in A_0^{\sqsubseteq}; \Gamma_0^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_0^{\sqsubseteq} \vDash M_{l1} \sqsubseteq M_{r1} \in A_1^{\sqsubseteq}; \Gamma_1^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vDash (M_{l0}, M_{l1}) \sqsubseteq (M_{r0}, M_{r1}) \in A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}; \Gamma_0^{\sqsubseteq}, \Gamma_1^{\sqsubseteq}}$$

Proof. We need to show that

$$\begin{array}{l}(w, x \leftarrow [\![M_{l0}]\!][\gamma_l][\delta_l];, x \leftarrow [\![M_{r0}]\!][\gamma_r][\delta_r];) \in \mathcal{E}^{\sim}[\![A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}]\!]\gamma\delta \\ \quad y \leftarrow [\![M_{l1}]\!][\gamma_l][\delta_l]; \quad y \leftarrow [\![M_{r1}]\!][\gamma_r][\delta_r]; \\ \quad \mathtt{ret}\ (x, y) \qquad\qquad \mathtt{ret}\ (x, y)\end{array}$$

By inductive hypothesis and weakening, we know $(w, [\![M_{l0}]\!][\gamma_l][\delta_l], [\![M_{r0}]\!][\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![A_0^{\sqsubseteq}]\!]\gamma\delta$. Applying monadic bind we get some $w' \sqsupseteq w$ and $(w', V_{l0}, V_{r0}) \in \mathcal{V}^{\sim}[\![A_0^{\sqsubseteq}]\!]\gamma\delta$ and applying anti-reduction, we need to show

$$\begin{array}{l}(w', y \leftarrow [\![M_{l1}]\!][\gamma_l][\delta_l];, y \leftarrow [\![M_{r1}]\!][\gamma_r][\delta_r];) \in \mathcal{E}^{\sim}[\![A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}]\!]\gamma\delta \\ \quad \mathtt{ret}\ (V_{l0}, y) \qquad\qquad \mathtt{ret}\ (V_{r0}, y)\end{array}$$

By inductive hypothesis, weakening and monotonicity, we know $(w', [\![M_{l1}]\!][\gamma_l][\delta_l], [\![M_{r1}]\!][\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![A_1^{\sqsubseteq}]\!]\gamma\delta$. Applying monadic bind we get some $w'' \sqsupseteq w'$ and $(w'', V_{l1}, V_{r1}) \in \mathcal{V}^{\sim}[\![A_1^{\sqsubseteq}]\!]\gamma\delta$ and applying anti-reduction we need to show

$$(w'', (V_{l0}, V_{l1}), (V_{r0}, V_{r1})) \in \mathcal{V}^{\sim}[\![A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}]\!]\gamma\delta$$

That is that for each $i \in \{0, 1\}$ that

$$(w'', V_{li}, V_{ri}) \in \mathcal{V}^{\sim}[\![A_i^{\sqsubseteq}]\!]\gamma\delta$$

which follows by assumption and monotonicity. □

Lemma G.32. *Product elim*

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A_0^{\sqsubseteq} \times A_1^{\sqsubseteq}; \Gamma_M^{\sqsubseteq} \qquad \Gamma^{\sqsubseteq}, \Gamma_M^{\sqsubseteq}, x : A_0^{\sqsubseteq}, y : A_1^{\sqsubseteq} \vDash N_l \sqsubseteq N_r \in B^{\sqsubseteq}; \Gamma_N^{\sqsubseteq}}{\Gamma^{\sqsubseteq} \vDash \mathtt{let}\ (x, y) = M_l; N_l \sqsubseteq \mathtt{let}\ (x, y) = M_r; N_r \in B^{\sqsubseteq}; \Gamma_M^{\sqsubseteq}, \Gamma_N^{\sqsubseteq}}$$

Proof. We need to show

$$\begin{array}{l}(w, z \leftarrow [\![M_l]\!][\gamma_l][\delta_l];, z \leftarrow [\![M_r]\!][\gamma_r][\delta_r];) \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta \\ \quad \mathtt{let}\ (x, y) = z; \quad \mathtt{let}\ (x, y) = z; \\ \quad [\![N_l]\!][\gamma_l][\delta_l] \qquad [\![N_r]\!][\gamma_r][\delta_r]\end{array}$$

First, by inductive hypothesis and weakening, we know

$$(w, [\![M_l]\!][\gamma_l][\delta_l], [\![M_r]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![A_0^\sqsubseteq \times A_1^\sqsubseteq]\!]\gamma\delta$$

Applying monadic bind, we get some $w' \sqsupseteq w$ with $(w', V_{l0}, V_{r0}) \in \mathcal{V}^\sim[\![A_0^\sqsubseteq]\!]\gamma\delta$ and $(w', V_{l1}, V_{r1}) \in \mathcal{V}^\sim[\![A_1^\sqsubseteq]\!]\gamma\delta$, and applying anti-reduction we need to show

$$(w', [\![N_l]\!][\gamma_l'][\delta_l], [\![N_r]\!][\gamma_r'][\delta_r]) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$$

Where we define $\gamma' = \gamma, x \mapsto (V_{l0}, V_{r0}), y \mapsto (V_{l1}, V_{r1})$. By weakening, it is sufficient to show

$$(w', [\![N_l]\!][\gamma_l'][\delta_l], [\![N_r]\!][\gamma_r'][\delta_r]) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma'\delta$$

which follows by inductive hypothesis if we can show

$$(w', \gamma', \delta) \in \mathcal{G}^\sim[\![\Gamma_p, \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq, x : A_0^\sqsubseteq, y : A_1^\sqsubseteq]\!]$$

which follows by definition and monotonicity. □

Lemma G.33. *Fun intro*

$$\frac{\Gamma^\sqsubseteq, x : A^\sqsubseteq \vDash M_l \sqsubseteq M_r \in B^\sqsubseteq; \cdot}{\Gamma^\sqsubseteq \vDash \lambda x : A_l.M_l \sqsubseteq \lambda x : A_l.M_l \in A^\sqsubseteq \to B^\sqsubseteq; \cdot}$$

Proof. It is sufficient to show

$$(w, \mathsf{thunk}\ (\lambda x : A_l[\delta_l].M_l'), \mathsf{thunk}\ (\lambda x : A_r[\delta_r].M_l')) \in \mathcal{V}^\sim[\![A^\sqsubseteq \to B^\sqsubseteq]\!]\gamma\delta$$

where $M_l' = [\![M_l]\!][\gamma_l][\delta_l]$ and similarly define M_r'. Suppose $w' \sqsupseteq w$ and $(w', V_l, V_r) \in \mathcal{V}^\sim[\![A^\sqsubseteq]\!]\gamma\delta$. We need to show

$(w', \mathsf{force}\ (\mathsf{thunk}\ (\lambda x : A_l[\delta_l].[\![M_l]\!][\gamma_l][\delta_l]))\, V_l, \mathsf{force}\ (\mathsf{thunk}\ (\lambda x : A_r[\delta_r].[\![M_r]\!][\gamma_r][\delta_r]))\, V_r) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$

By anti-reduction it is sufficient to show

$$(w', [\![M_l]\!][\gamma_l'][\delta_l], [\![M_r]\!][\gamma_r'][\delta_r]) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$$

where $\gamma' = \gamma, x \mapsto (V_l, V_r)$. By monotonicity, we have $(w', \gamma', \delta) \in \mathcal{G}^\sim[\![\Gamma_p, \Gamma^\sqsubseteq]\!]$ so the result follows by hypothesis. □

Lemma G.34.

$$\frac{\Gamma^\sqsubseteq \vdash M_l \sqsubseteq M_r : A^\sqsubseteq \to B^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq \vdash N_l \sqsubseteq N_r : A^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vdash M_l\, N_l \sqsubseteq M_r\, N_r : B^\sqsubseteq; \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq}$$

Proof. Define $M_l' = [\![M_l]\!][\gamma_l][\delta_l]$, etc. We need to show

$$(w, f \leftarrow M_l'; x \leftarrow N_l'; \mathsf{force}\ f\, x f \leftarrow M_r'; x \leftarrow N_r'; \mathsf{force}\ f\, x) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$$

We apply monadic bind with $(w, M_l', M_r') \in \mathcal{E}^\sim[\![A^\sqsubseteq \to B^\sqsubseteq]\!]\gamma\delta$ which holds by hypothesis and weakening. Suppose $w' \sqsupseteq w$, with anti-reduction it is sufficient to show

$$(w', x \leftarrow N_l'; \mathsf{force}\ V_l\, x, x \leftarrow N_r'; \mathsf{force}\ V_r\, x) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$$

where $(w', V_l, V_r) \in \mathcal{V}^\sim[\![A^\sqsubseteq \to B^\sqsubseteq]\!]\gamma\delta$. Then we apply monadic bind with $(w', N_l', N_r') \in \mathcal{E}^\sim[\![A^\sqsubseteq]\!]\gamma\delta$ which holds by hypothesis, weakening and monotonicity. Suppose $w'' \sqsupseteq w'$; with anti-reduction it is sufficient to show

$$(w'', \mathsf{force}\ V_l\, V_l', \mathsf{force}\ V_r\, V_r') \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta$$

Which follows by definition of the value relation and transitivity of $\sqsupseteq$. □

Lemma G.35. *Forall intro*

$$\frac{\Gamma^{\sqsubseteq}, X \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \cdot}{\Gamma^{\sqsubseteq} \vDash \Lambda^{\nu} X.M_l \sqsubseteq \Lambda^{\nu} X.M_r \in \forall^{\nu} X.A^{\sqsubseteq}; \cdot}$$

Proof. Define $M_i' = [\![M_i]\!][\gamma_i][\delta_i]$ It is sufficient to show

$$(w, \mathsf{thunk}\ (\Lambda X.\lambda c_X : \mathrm{Case}\ X.M_l'), \mathsf{thunk}\ (\Lambda X.\lambda c_X : \mathrm{Case}\ X.M_r')) \in \mathcal{V}^{\sim}[\![\forall^{\nu} X.A^{\sqsubseteq}]\!]\gamma\delta$$

Given $w' \sqsupseteq w$, and $R \in \mathrm{Rel}_n[B_l, B_r]$ and $w'.\eta \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j})$, we need to show

$(w', \mathsf{force}\ \mathsf{thunk}\ (\Lambda X.\lambda c_X : \mathrm{Case}\ X.M_l')\, B_l\, \sigma_l \mathsf{force}\ \mathsf{thunk}\ (\Lambda X.\lambda c_X : \mathrm{Case}\ X.M_r')\, B_r\, \sigma_r) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma'\delta'$

where $\gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r)$ and $\delta' = \delta, X \mapsto (B_l, B_r, R)$. By anti-reduction it is sufficient to show

$$(w', [\![M_l]\!][\gamma_l'][\delta_l'], [\![M_r]\!][\gamma_r'][\delta_r']) \in \mathcal{V}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma'\delta'$$

which follows by hypothesis if since $(w', \gamma', \delta') \in \mathcal{G}^{\sim}[\![\Gamma_p, \Gamma^{\sqsubseteq}, X]\!]$ by definition and monotonicity. □

Lemma G.36. *Forall elim*

$$\frac{\Gamma^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in \forall^{\nu} X.A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime} \qquad \Gamma^{\sqsubseteq} \vDash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vDash M_l\{X \cong B_l\} \sqsubseteq M_r\{X \cong B_r\} \in A^{\sqsubseteq}; \Gamma^{\sqsubseteq\prime}, X \cong B^{\sqsubseteq}}$$

Proof. By definition, instantiation translates as

$$[\![M_i\{X \cong B_i\}]\!] = f \leftarrow [\![M_i]\!]; \mathsf{force}\ f\ [\![B_i]\!]\ c_X$$

Define $M_i' = [\![M_i]\!][\gamma_i][\delta_i]$, then we need to show

$$(w, f \leftarrow M_l'; \mathsf{force}\ f\ [\![B_l]\!][\delta_l]\, \gamma_l(c_X), f \leftarrow M_r'; \mathsf{force}\ f\ [\![B_r]\!][\delta_r]\, \gamma_r(c_X)) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

We know $(w, M_l', M_r') \in \mathcal{E}^{\sim}[\![\forall^{\nu} X.A^{\sqsubseteq}]\!]\gamma\delta$ by hypothesis and weakening (lemma G.16), so we can apply monadic bind. Suppose $w' \sqsupseteq w$, then by anti-reduction it is sufficient to show

$$(w', \mathsf{force}\ V_l\ [\![B_l]\!][\delta_l]\, \gamma_l(c_X), \mathsf{force}\ V_r\ [\![B_r]\!][\delta_r]\, \gamma_r(c_X)) \in \mathcal{E}^{\sim}[\![A^{\sqsubseteq}]\!]\gamma\delta$$

where $(w', V_l, V_r) \in \mathcal{V}^{\sim}[\![\forall^{\nu} X.A^{\sqsubseteq}]\!]\gamma\delta$. Thus it is sufficient to give some relation $R \in \mathrm{Rel}_n[[\![B_l]\!][\delta_l], [\![B_r]\!][\delta_r]]$ such that $w' \vDash (\gamma_l(c_X), \gamma_r(c_X), R)$. By definition of $\mathcal{G}^{\sim}[\![\Gamma_p, \Gamma^{\sqsubseteq\prime}, X \cong B^{\sqsubseteq}]\!]$, this holds for $R = \mathcal{V}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$. □

Lemma G.37.

$$\frac{\Gamma^{\sqsubseteq}, X \cong B^{\sqsubseteq} \vDash M_l \sqsubseteq M_r \in A^{\sqsubseteq}; \cdot \qquad \Gamma^{\sqsubseteq} \vdash B^{\sqsubseteq} : B_l \sqsubseteq B_r}{\Gamma^{\sqsubseteq} \vDash \mathsf{pack}^{\nu}(X \cong B_l, M_l) \sqsubseteq \mathsf{pack}^{\nu}(X \cong B_r, M_r) \in \exists^{\nu} X.A^{\sqsubseteq}; \cdot}$$

Proof. Recall the translation is

$$[\![\mathsf{pack}^{\nu}(X \cong B_i, M_i)]\!] = \mathsf{ret}\ (\mathsf{pack}([\![B_i]\!], \mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ [\![B_i]\!].[\![M_i]\!])\ \mathsf{as}\ [\![\exists^{\nu} X.A_i]\!])$$

where $\Gamma^{\sqsubseteq}, X \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$. So it is sufficient to show

$(w, \mathsf{pack}([\![B_l]\!][\delta_l], \mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ ([\![B_l]\!][\delta_l]).[\![M_l]\!][\gamma_l][\delta_l])\ \mathsf{as}\ [\![\exists^{\nu} X.A^{\sqsubseteq}]\!][\delta_l], \mathsf{pack}([\![B_r]\!][\delta_r], \mathsf{thunk}\ \lambda c_X : \mathrm{Cas}$

For the relation, we pick $R = \mathcal{V}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$. Let $w' \sqsupseteq w$ and σ_l, σ_r be seals such that $w'.\eta \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j})$. Then we need to show

$(w, \mathsf{force}\ (\mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ ([\![B_l]\!][\delta_l]).[\![M_l]\!][\gamma_l][\delta_l])\, \sigma_l, \mathsf{force}\ (\mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ ([\![B_r]\!][\delta_r]).[\![M_r]\!][\gamma_r][\delta_r])\, \sigma_r$

where

$$\gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r)$$
$$\delta' = \delta, X \mapsto ([\![B_l]\!][\delta_l], [\![B_l]\!][\delta_l], R)$$

By anti reduction this reduces to showing

$$(w', [\![M_l]\!][\gamma_l'][\delta_l'], [\![M_r]\!][\gamma_r'][\delta_r']) \in \mathcal{E}^\sim[\![A^\sqsubseteq]\!]\gamma'\delta'$$

which follows by assumption. □

Lemma G.38.

$$\frac{\Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq \vDash \Gamma_N^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq \vDash B^\sqsubseteq \qquad \Gamma^\sqsubseteq \vDash M_l \sqsubseteq M_r \in \exists^\nu X.A^\sqsubseteq; \Gamma_M^\sqsubseteq \qquad \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, X, x : A^\sqsubseteq \vDash N_l \sqsubseteq N_r \in B^\sqsubseteq; \Gamma_N^\sqsubseteq}{\Gamma^\sqsubseteq \vDash \mathsf{unpack}\ (X, x) = M_l; N_l \sqsubseteq \mathsf{unpack}\ (X, x) = M_r; N_r \in B^\sqsubseteq; \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq}$$

Proof. Recall the translation:

$$\begin{aligned}[\![\mathsf{unpack}\ (X, x) = M_i; N_i]\!] = p \leftarrow [\![M_i]\!];\ & \\ & \mathsf{unpack}\ (X, f) = p; \\ & \mathsf{newcase}_X\ c_X; \\ & x \leftarrow (\mathtt{force}\ f\ c_X); \\ & [\![N_i]\!]\end{aligned}$$

Let $(w, \gamma, \delta) \in \mathcal{G}^\sim[\![\Gamma_p, \Gamma^\sqsubseteq, \Gamma_M^\sqsubseteq, \Gamma_N^\sqsubseteq]\!]$. By assumption (and weakening), we know $(w, [\![M_l]\!][\gamma_l][\delta_l], [\![M_r]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![\exists^\nu X.A^\sqsubseteq]\!]$. We apply monadic bind. Let $w' \sqsupseteq w$ and $(w', V_{\exists^\nu l}, V_{\exists^\nu r}) \in \mathcal{V}^\sim[\![\exists^\nu X.A^\sqsubseteq]\!]$.

Then $V_{\exists^\nu l} = \mathsf{pack}(A_{Xl}, V_{fl})$ and $V_{\exists^\nu r} = \mathsf{pack}(A_{Xr}, V_{fr})$ with some associated relation $R \in \mathrm{Rel}[A_{Xl}, A_{Xr}]$. Then by anti-reduction we need to show

$$\begin{array}{lll}(w', \mathsf{newcase}_{A_{Xl}}\ c_X; & , \mathsf{newcase}_{A_{Xr}}\ c_X; &) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma\delta \\ \quad x \leftarrow (\mathtt{force}\ V_{fl}\ c_X); & x \leftarrow (\mathtt{force}\ V_{fr}\ c_X); & \\ \quad [\![N_l]\!][\gamma_l][\delta_l] & [\![N_r]\!][\gamma_l][\delta_l] & \end{array}$$

Define

$$\begin{aligned} w'' &= (w'.j, (w'.\Sigma_l, A_{Xl}), (w'.\Sigma_r, A_{Xr}), w'.\eta \uplus (A_{Xl}, A_{Xr}, \lfloor R \rfloor_{w'.j})) \\ \gamma' &= \gamma, c_X \mapsto (w'.\Sigma_l.size, w'.\Sigma_r.size) \\ \delta' &= \delta, X \mapsto (A_{Xl}, A_{Xr}, R) \end{aligned}$$

Then by anti-reduction it is sufficient to show

$$(w'', (x \leftarrow (\mathtt{force}\ V_{fl}\ \sigma_l); [\![N_l]\!][\gamma_l'][\delta_l']), (x \leftarrow (\mathtt{force}\ V_{fr}\ \sigma_r); [\![N_r]\!][\gamma_r'][\delta_r'])) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma'\delta'$$

By assumption (and monotonicity), $(w'', \mathtt{force}\ V_{fl}\ \sigma_l, \mathtt{force}\ V_{fr}\ \sigma_r) \in \mathcal{E}^\sim[\![A^\sqsubseteq]\!]\gamma'\delta'$, so we can apply monadic bind. Let $w''' \sqsupseteq w''$ and $(w''', V_{Al}, V_{Ar}) \in \mathcal{V}^\sim[\![A^\sqsubseteq]\!]\gamma'\delta'$. Then after anti-reduction we need to show

$$(w''', [\![N_l]\!][\gamma_l''][\delta_l'], [\![N_r]\!][\gamma_r''][\delta_r']) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma'\delta'$$

where $\gamma'' = \gamma', x \mapsto (V_{Al}, V_{Ar})$. By weakening this is equivalent to

$$(w''', [\![N_l]\!][\gamma_l''][\delta_l'], [\![N_r]\!][\gamma_r''][\delta_r']) \in \mathcal{E}^\sim[\![B^\sqsubseteq]\!]\gamma''\delta'$$

which follows by assumption. □

H PARAMETRICITY

To establish this, we first need to show that the semantic precision relation is a congruence, which follows by induction using the compatibility lemmas.

Lemma H.1. *If* $C : (\Gamma \vdash \cdot : A; \Gamma_o) \Rightarrow (\Gamma' \vdash \cdot : B; \Gamma_o')$ *and* $\Gamma \vDash M_l \sqsubseteq M_r \in A; \Gamma_o$, *then* $\Gamma' \vDash C[M_l] \sqsubseteq C[M_r] \in B; \Gamma_o'$.

Proof. By induction on C, using a corresponding compatibility lemma in each case. □

We define contextual error approximation, following [15]:

Definition H.2. Let $\Gamma \vdash M_1 : A; \Gamma_M$ and $\Gamma \vdash M_2 : A; \Gamma_M$. Then we say M_1 (contextually) *error approximates* M_2, written $\Gamma \vDash M_1 \sqsubseteq^{ctx} M_2 \in A; \Gamma_M$ when for any context $C : (\Gamma \vdash A; \Gamma_M) \Rightarrow (\cdot \vdash B; \cdot)$, all of the following hold:

(1) If $C[M_1] \Uparrow$ then $C[M_2] \Uparrow$
(2) If $C[M_1] \Downarrow V_1$ then there exists V_2 such that $C[M_2] \Downarrow V_2$.

Definition H.3. If $\Gamma \vdash M_1 : A; \Gamma_M$ and $\Gamma \vdash M_2 : A; \Gamma_M$, then M_1 and M_2 are *contextually equivalent*, $\Gamma \vDash M_1 \approx^{\text{ctx}} M_2 \in A; \Gamma_M$, when for any context $C : (\Gamma \vdash A; \Gamma_M) \Rightarrow (\cdot \vdash B; \cdot)$, both diverge $C[M_1], C[M_2] \Uparrow$, or error $C[M_1], C[M_2] \Downarrow \mathfrak{U}$ or terminate successfully $C[M_1] \Downarrow V_1$, $C[M_2] \Downarrow V_2$.

From syntactic type safety, it is clear that mutual error approximation implies equivalence:

Lemma H.4. *If* $\Gamma \vDash M_1 \sqsubseteq^{ctx} M_2 : A; \Gamma_M$ *and* $\Gamma \vDash M_2 \sqsubseteq^{ctx} M_1 : A; \Gamma_M$, *then* $\Gamma \vDash M_1 \approx^{ctx} M_2 : A; \Gamma_M$

Proof. The first two cases are direct. For the third case, we know by type safety that there are only 3 possibilities for a closed term's behavior: $C[M_i] \Uparrow$, $C[M_i] \Downarrow V_i$ or $C[M_i] \Downarrow \mathfrak{U}$. If $C[M_1] \Downarrow \mathfrak{U}$, then it is not the case that $C[M_1] \Uparrow$ or $C[M_1] \Downarrow V_1$ but by the first two cases that means that it is not the case that $C[M_2] \Uparrow$ or $C[M_1] \Downarrow V$, so it must be the case that $C[M_2] \Downarrow \mathfrak{U}$. The opposite direction follows by symmetry. □

Lemma H.5.

$$\frac{\Gamma \vDash M_l \sqsubseteq M_2 \in A; \Gamma_M}{\Gamma \vDash M_l \sqsubseteq^{ctx} M_2 \in A; \Gamma_M}$$

Proof. Let C be an appropriately typed closing context. By the congruence Lemma H.1 we know

$$\cdot \vDash C[M_1] \sqsubseteq C[M_2] \in B; \cdot$$

Define

$$\begin{aligned}
\eta_p^{\sim}(n) = \emptyset &\boxplus (\mathbb{B}, \mathbb{B}, \mathcal{V}_n^{\sim}[\![\mathbb{B}]\!]\emptyset\emptyset) \boxplus (\text{OSum} \times \text{OSum}, \text{OSum} \times \text{OSum}, \mathcal{V}_n^{\sim}[\![? \times ?]\!]\emptyset\emptyset) \\
&\boxplus (U(\text{OSum} \to F\text{OSum}), U(\text{OSum} \to F\text{OSum}), \lfloor \mathcal{V}_n^{\sim}[\![? \to ?]\!]\emptyset\emptyset \rfloor_{\downarrow}) \\
&\boxplus (\exists X.U(\text{Case } X \to F\text{OSum}), \exists X.U(\text{Case } X \to F\text{OSum}), \mathcal{V}_n^{\sim}[\![\exists^{\nu} X.?]\!]\emptyset\emptyset) \\
&\boxplus (U(\forall X.\text{Case } X \to F\text{OSum}), \forall X.U(\text{Case } X \to F\text{OSum}), \mathcal{V}_n^{\sim}[\![\forall^{\nu} X.?]\!]\emptyset\emptyset) \\
w_p^{\sim}(n) = (n, \Sigma_p, \Sigma_p, \eta_p^{\sim}(n)) & \\
\delta_p^{\sim} = \emptyset, \text{Bool} &\mapsto (\mathbb{B}, \mathbb{B}, \mathcal{V}^{\sim}[\![\mathbb{B}]\!]\emptyset\emptyset), \text{Times} \mapsto (\text{OSum} \times \text{OSum}, \text{OSum} \times \text{OSum}, \mathcal{V}^{\sim}[\![? \times ?]\!]\emptyset\emptyset), \\
&\text{Fun} \mapsto (U(\text{OSum} \to F\text{OSum}), U(\text{OSum} \to F\text{OSum}), \mathcal{V}^{\sim}[\![? \to ?]\!]\emptyset\emptyset), \\
&\text{Exist} \mapsto (\exists X.U(\text{Case } X \to F\text{OSum}), \exists X.U(\text{Case } X \to F\text{OSum}), \mathcal{V}^{\sim}[\![\exists^{\nu} X.?]\!]\emptyset\emptyset), \\
&\text{Forall} \mapsto (U(\forall X.\text{Case } X \to F\text{OSum}), \forall X.U(\text{Case } X \to F\text{OSum}), \mathcal{V}^{\sim}[\![\forall^{\nu} X.?]\!]\emptyset\emptyset)
\end{aligned}$$

Then we can see that $(w_p^{\sim}, \gamma_p, \delta_p^{\sim}) \in \mathcal{G}^{\sim}[\![\Gamma_p]\!]$. So we know that

$$(w_p^{\sim}(n), [\![C[M_1]]\!][\gamma_p], [\![C[M_2]]\!][\gamma_p] \in \mathcal{E}^{\sim}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

(noting that $[\![C[M_i]]\!][\gamma_p][\delta_p^{\sim}] = [\![C[M_i]]\!][\gamma_p]$).

- If $C[M_1] \Uparrow$, then by the simulation theorem we know $\Sigma_p, [\![C[M_1]]\!][\gamma_p] \Uparrow$. Again by simulation, to show $C[M_2] \Uparrow$, it is sufficient to show that $\Sigma_p, [\![C[M_2]]\!][\gamma_p] \Uparrow$. We will show that for every n, $\Sigma_p, [\![C[M_2]]\!][\gamma_p] \mapsto^n \Sigma_n, N_n$ for some Σ_n, N_n. We know

  $$(w_p^{>}(n), [\![C[M_1]]\!][\gamma_p], [\![C[M_2]]\!][\gamma_p]) \in \mathcal{E}^{>}[\![B^{\sqsubseteq}]\!]\gamma\delta$$

  we proceed by the cases of $\mathcal{E}^{>}[\![B^{\sqsubseteq}]\!]\gamma\delta$

- If $w_p^>(n).\Sigma_r, [\![[C[M_2]][\gamma_p]]\!] \mapsto^{w_p^>(n).j}$ we are done because $w_p^>(n).\Sigma_r = \Sigma_p$ and $w_p^>(n).j = n$.
- If $w_p^>(n).\Sigma_l, [\![[C[M_1]][\gamma_p]]\!] \mapsto^* \Sigma', \mho$ we have a contradiction because $\Sigma_p, [\![[C[M_1]][\gamma_p]]\!] \Uparrow$
- If $w_p^>(n).\Sigma_l, [\![[C[M_1]][\gamma_p]]\!] \mapsto^* \Sigma', \texttt{ret}\ V_1$, we also have a contradiction for the same reason.

• If $C[M_1] \Downarrow V_1$ then by simulation we know $\Sigma_p, [\![C[M_1]]\!] \mapsto^n \Sigma', \texttt{ret}\ V_1'$ for some n, V_1. Furthermore, to show $C[M_2] \Downarrow V_2$ it is sufficient by simulation to show $\Sigma_p, [\![C[M_2]]\!] \mapsto^* \texttt{ret}\ V_2'$. We know

$$(w_p^<(n), [\![C[M_1]]\!][\gamma_p], [\![C[M_2]]\!][\gamma_p]) \in \mathcal{E}^<[\![B^{\sqsubseteq}]\!]\gamma\delta$$

we proceed by the cases of $\mathcal{E}^<[\![B^{\sqsubseteq}]\!]\gamma\delta$.

- If $w_p^<(n).\Sigma_l, [\![C[M_1]]\!][\gamma_p] \mapsto^{w_p^<(n).j+1}$ or $w_p^<(n).\Sigma_l, [\![C[M_1]]\!][\gamma_p] \mapsto^j \mho$ this contradicts the fact that $\Sigma_p, [\![C[M_1]]\!] \mapsto^n \Sigma', \texttt{ret}\ V_1'$.
- Otherwise, $w_p^<(n).\Sigma_r, [\![C[M_2]]\!][\gamma_p] \mapsto^* \Sigma', \texttt{ret}\ V_2'$, so the result holds because $w_p^<(n).\Sigma_r = \Sigma_p$.

□

$$\frac{M : \forall^v X.X \to X \qquad V_A : A \qquad V_B : B}{\lambda\_ : ?.\texttt{unseal}_X(M_f\{X \cong A\}(\texttt{seal}_X V_A)) \approx^{ctx} \lambda\_ : ?.\texttt{let}\ y = M\{X \cong B\}V_B; V_A}$$

Theorem H.6.

Proof. There are 4 cases: $\sqsubseteq<, \sqsubseteq>, >\sqsupseteq$ and $<\sqsupseteq$ but they are all by a similar argument. Let $(w, \gamma, \delta) \in \mathcal{G}^\sim[\![\Gamma_p, \Gamma]\!]$. We need to show

$$(w, \texttt{ret thunk}\ \lambda\_ : ?.\ \texttt{newcase}_{[\![A]\!][\delta_l]}\ c_X;\ [\![\texttt{unseal}_X(M\{X \cong A\}(\texttt{seal}_X V_A))]\!][\gamma_l][\delta_l], \texttt{ret thunk}\ \lambda\_ : ?.\ \texttt{newcase}_{[\![A]\!][\delta_r]}\ c_X;\ [\![\texttt{let}\ y = M\{X \cong B\}V_B; V_A]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

Given some $w' \sqsupseteq w$ and (irrelevant) values (w', V_{duml}, V_{dumr}), we need to show (after applying anti-reduction) that

$$(w', \texttt{newcase}_{[\![A]\!][\delta_l]}\ c_X;\ [\![\texttt{unseal}_X(M\{X \cong A\}(\texttt{seal}_X V_A))]\!][\gamma_l][\delta_l], \texttt{newcase}_{[\![A]\!][\delta_r]}\ c_X;\ [\![\texttt{let}\ y = M\{X \cong B\}V_B; V_A]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

Each side allocates a new case and we need to pick a relation with which to instantiate it. By Lemmas F.11 and F.3, we have that $[\![V_A]\!][\gamma_l][\delta_l] \mapsto^0 \texttt{ret}\ V_{Al}$ and $[\![V_B]\!][\gamma_r][\delta_r] \mapsto^0 \texttt{ret}\ V_{Br}$ for some V_{Al} and V_{Br}. We define R to be the "singleton" relation:

$$R = \{(w, V_{Al}, V_{Br}) \in \text{Atom}[[\![A]\!][\delta_l], [\![B]\!][\delta_r]] \mid w \sqsupseteq w'\}$$

Then we define w'' to be the world extended with $\lfloor R \rfloor_{w'.j}$:

$$w'' = (w'.j, w'.\Sigma_l, [\![A]\!][\delta_l], w'.\Sigma_r[\delta_r], w'.\eta \boxplus (w'.\Sigma_l.size, w'.\Sigma_r.size, \lfloor R \rfloor_{w'.j})$$

Then clearly $w'' \sqsupseteq w'$ and

$$w'.\Sigma_l, \texttt{newcase}_{[\![A]\!][\delta_l]}\ c_X;\ [\![\texttt{unseal}_X(M\{X \cong A\}(\texttt{seal}_X V_A))]\!][\gamma_l][\delta_l] \mapsto^0 w''.\Sigma_l, [\![\texttt{unseal}_X(M\{X \cong A\}(\texttt{seal}_X V_A))]\!][\gamma_l'][\delta_l]$$

and similarly for the right hand side:

$$w'.\Sigma_r, \texttt{newcase}_{[\![A]\!][\delta_r]}\ c_X;\ [\![\texttt{let}\ y = M\{X \cong B\}V_B; V_A]\!][\gamma_r][\delta_r] \mapsto^0 w''.\Sigma_r, [\![\texttt{let}\ y = M\{X \cong B\}V_B; V_A]\!][\gamma_r'][\delta_r]$$

where $\gamma' = \gamma, c_X \mapsto (w'.\Sigma_l.size, w'.\Sigma_r.size)$. Expanding definitions (and noting that c_X is free in M), we need to show

$$(w'', S_l[[\![M]\!][\gamma_l'][\delta_l]], S_2[[\![M]\!][\gamma_r'][\delta_r]]) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

where

$$\begin{aligned} S_l = f \leftarrow (z \leftarrow \bullet; \texttt{force}\ z([\![A]\!][\delta_l])\,\gamma_l'(c_X));\\ x \leftarrow [\![V_A]\!][\gamma_l][\delta_l];\\ \texttt{force}\ f\ x \end{aligned}$$

and

$$S_2 = y \leftarrow \left(\begin{array}{l} \\ f \leftarrow (z \leftarrow \bullet; \texttt{force}\ z([\![B]\!][\delta_l])\,\gamma_r'(c_X));\\ x \leftarrow [\![V_B]\!][\gamma_r][\delta_r];\\ \texttt{force}\ f\ x \end{array}\right); [\![V_A]\!][\gamma_r][\delta_r]$$

We apply monadic bind. Let $w''' \sqsupseteq w''$ and $(w''', V_l, V_r) \in \mathcal{V}^\sim[\![\forall^v X.X \to X]\!]\gamma\delta$. After anti-reduction we need to show

$$(w''', \begin{array}{l} f \leftarrow \texttt{force}\ V_l([\![A]\!][\delta_l])\,\gamma_l'(c_X);, S_r' \\ x \leftarrow [\![V_A]\!][\gamma_l][\delta_l];\\ \texttt{force}\ f\ x \end{array} \left[\begin{array}{l} f \leftarrow \texttt{force}\ V_r([\![B]\!][\delta_l])\,\gamma_r'(c_X);\\ x \leftarrow [\![V_B]\!][\gamma_r][\delta_r];\\ \texttt{force}\ f\ x \end{array}\right]) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

where

$$S_r' = y \leftarrow \bullet; [\![V_A]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

By weakening it is equivalent to showing the terms are in the relation $\mathcal{E}^\sim[\![A]\!]\gamma'\delta'$ where $\delta' = ([\![A]\!][\delta_l], [\![B]\!][\delta_r], R)$. By definition of $\mathcal{V}^\sim[\![\forall^v X.X \to X]\!]$, we have

$$(w''', \texttt{force}\ V_l([\![A]\!][\delta_l])\,\gamma_l'(c_X), \texttt{force}\ V_r([\![B]\!][\delta_r])\,\gamma_r'(c_X)) \in \mathcal{E}^\sim[\![X \to X]\!]\gamma'\delta'$$

So we apply monadic bind. Let $w'''' \sqsupseteq w'''$ and let $(w'''', V_{fl}, V_{fr}) \in \mathcal{V}^\sim[\![X \to X]\!]\gamma'\delta'$. Applying anti-reduction, we need to show

$$(w'''', \texttt{force}\ V_{fl}\ V_{Al}, y \leftarrow \texttt{force}\ V_{fr}\ V_{Br}; [\![V_A]\!][\gamma_r][\delta_r]) \in \mathcal{E}^\sim[\![A]\!]\gamma'\delta'$$

By definition of $\mathcal{V}^\sim[\![X \to X]\!], (w'''', \texttt{force}\ V_{fl}\ V_{Al}, \texttt{force}\ V_{fr}\ V_{Br}) \in \mathcal{E}^\sim[\![X]\!]\gamma'\delta'$ if $(w'''', V_{Al}, V_{Br}) \in \mathcal{V}^\sim[\![X]\!]\gamma'\delta'$ which holds because $(w'''', V_{Al}, V_{Br}) \in R$. By Lemmas F.11 and F.3, we have that $[\![V_A]\!][\gamma_r][\delta_r] \mapsto^0 \texttt{ret}\ V_{Ar}$ for some V_{Ar}. Applying monadic bind a final time we get $w_{fin} \sqsupseteq w''''$ and $(w_{fin}, V_{Al}', V_{Br}') \in \mathcal{V}^\sim[\![X]\!]\gamma'\delta'$ and after anti-reduction need to show that

$$(w_{fin}, \texttt{ret}\ V_{Al}', \texttt{ret}\ V_{Ar}) \in \mathcal{E}^\sim[\![A]\!]\gamma'\delta'$$

By weakening this is equivalent to

$$(w_{fin}, \texttt{ret}\ V_{Al}', \texttt{ret}\ V_{Ar}) \in \mathcal{E}^\sim[\![A]\!]\gamma\delta$$

By the definition of $\mathcal{V}^\sim[\![X]\!]\gamma'\delta'$, $V_{Al}' = V_{Al}$, so this follows by reflexivity; specifically that $\Gamma \vDash V_A \sqsubseteq V_A \in A; \cdot$. This is because, by the definition of $\mathcal{E}^\sim[\![A]\!]\gamma\delta$, since $V_A[\gamma_l][\delta_l] \mapsto^0 \texttt{ret}\ V_{Al}$ and $V_A[\gamma_r][\delta_r] \mapsto^0 \texttt{ret}\ V_{Ar}$, those values are related. □

Theorem H.7.

$$\frac{M : \forall^{\nu} X.\forall^{\nu} Y.(X \times Y) \to (Y \times X) \qquad V_A : A \qquad V_B : B}{\begin{array}{l} \lambda\_ : ?.\texttt{let}\ (y,x) = (M\{X \cong A\}\{Y \cong B\}\,(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)); (\texttt{unseal}_X x, \texttt{unseal}_Y y) \\ \approx^{ctx} \lambda\_ : ?.\texttt{let}\ (y,x) = (M\{X \cong B\}\{Y \cong A\}\,(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)); (\texttt{unseal}_Y y, \texttt{unseal}_X x) \end{array}}$$

Proof. We show the $\sqsubseteq\sim$ case, the $\sim\sqsupseteq$ case is symmetric. Let $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_p, \Gamma]\!]$. Define the following terms

$$N_l = [\![\texttt{let}\ (y,x) = (M\{X \cong A\}\{Y \cong B\}\,(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)); (\texttt{unseal}_X x, \texttt{unseal}_Y y)]\!]$$
$$N_r = [\![\texttt{let}\ (y,x) = (M\{X \cong B\}\{Y \cong A\}\,(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)); (\texttt{unseal}_Y y, \texttt{unseal}_X x)]\!]$$

Then we need to show for any $w_1 \sqsupseteq w$ that

$(w_1, \texttt{newcase}_A\ c_X; \texttt{newcase}_B\ c_Y; N_l[\gamma_l][\delta_l], \texttt{newcase}_B\ c_X; \texttt{newcase}_A\ c_Y; N_r[\gamma_r][\delta_r]) \in \mathcal{E}^{\sim}[\![A \times B]\!]\gamma\delta$

By Lemmas F.11 and F.3, we have that $[\![V_A]\!][\gamma_i][\delta_i] \mapsto^0 \texttt{ret}\ V_{Ai}$ and $[\![V_B]\!][\gamma_i][\delta_i] \mapsto^0 \texttt{ret}\ V_{Bi}$ for some V_{Ai} and V_{Bi}. Define

$$A_i = [\![A]\!][\delta_i]$$
$$B_i = [\![B]\!][\delta_i]$$
$$R_X = \{(w, V_{Al}, V_{Br}) \in \mathrm{Atom}[A_l, B_r] \mid w \sqsupseteq w_1\}$$
$$R_Y = \{(w, V_{Bl}, V_{Ar}) \in \mathrm{Atom}[B_l, A_r] \mid w \sqsupseteq w_1\}$$
$$w_2 = (w_1.j, (w_1.\Sigma_l, A_l, B_l), (w_1.\Sigma_r, B_r, A_r), (w_1.\eta \uplus (A_l, B_r, \lfloor R_X \rfloor_{w_1.j}) \uplus (B_l, A_r, \lfloor R_Y \rfloor_{w_1.j})))$$
$$\sigma_{Xi} = w_1.\Sigma_i.size$$
$$\sigma_{Yi} = w_1.\Sigma_i.size + 1$$
$$\gamma' = \gamma, c_X \mapsto (\sigma_{Xl}, \sigma_{Xr}), c_Y \mapsto (\sigma_{Yl}, \sigma_{Yr})$$
$$\delta' = \delta, X \mapsto (A_l, B_r, R_X), Y \mapsto (B_l, A_r, R_Y)$$

Then it is sufficient to show that

$$(w_2, N_l[\gamma'_l][\delta'_l], N_r[\gamma'_r][\delta'_r]) \in \mathcal{E}^{\sim}[\![A \times B]\!]\gamma\delta$$

which is equivalent by weakening to

$$(w_2, N_l[\gamma'_l][\delta'_l], N_r[\gamma'_r][\delta'_r]) \in \mathcal{E}^{\sim}[\![A \times B]\!]\gamma'\delta'$$

Next, note that by reflexivity,

$$(w_2, [\![M]\!][\gamma'_l][\delta'_l], [\![M]\!][\gamma'_r][\delta'_r]) \in \mathcal{E}^{\sim}[\![\forall^{\nu} X.\forall^{\nu} Y.(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$$

and

$$N_i = [\![S_i]\!][[\![M]\!]][\gamma'_i][\delta'_i]$$

for stacks

$$S_l = \texttt{let}\ (y,x) = (\bullet\{X \cong A\}\{Y \cong B\}\,(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)); (\texttt{unseal}_X x, \texttt{unseal}_Y y)$$
$$S_r = \texttt{let}\ (y,x) = (\bullet\{X \cong B\}\{Y \cong A\}\,(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)); (\texttt{unseal}_Y y, \texttt{unseal}_X x)$$

So we can apply monadic bind. Let $w_3 \sqsupseteq w_2$ and $(w_3, V_{\forall^{\nu} l}, V_{\forall^{\nu} r}) \in \mathcal{V}^{\sim}[\![\forall^{\nu} X.\forall^{\nu} Y.(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$. Then by anti-reduction it is sufficient to show

$$(w_3, [\![S'_l]\!][\texttt{force}\ V_{\forall^{\nu} l}\ A_l\ \sigma_{Xl}], [\![S'_r]\!][\texttt{force}\ V_{\forall^{\nu} r}\ B_r\ \sigma_{Xr}]) \in \mathcal{E}^{\sim}[\![A \times B]\!]\gamma'\delta'$$

for stacks

$$S_l' = \texttt{let}\ (y,x) = (\bullet\{Y \cong B\}\,(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)); (\texttt{unseal}_X x, \texttt{unseal}_Y y)$$
$$S_r' = \texttt{let}\ (y,x) = (\bullet\{Y \cong A\}\,(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)); (\texttt{unseal}_Y y, \texttt{unseal}_X x)$$

By definition, we know $(w_3, \texttt{force}\ V_{\forall^v l}\, A_l\, \sigma_{Xl}, \texttt{force}\ V_{\forall^v r}\, B_r\, \sigma_{Xr}) \in \mathcal{E}^\sim[\![\forall^v Y.(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$, so we can apply monadic bind again. Let $w_4 \sqsupseteq w_3$ and $(w_4, V'_{\forall^v l}, V'_{\forall^v r}) \in \mathcal{V}^\sim[\![\forall^v Y.(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$. Then after anti-reduction we need to show

$$(w_4, [\![S_l'']\!][\texttt{force}\ V'_{\forall^v l}\, B_l\, \sigma_{Yl}], [\![S_r'']\!][\texttt{force}\ V'_{\forall^v r}\, A_r\, \sigma_{Yr}]) \in \mathcal{E}^\sim[\![A \times B]\!]\gamma'\delta'$$

for stacks

$$S_l'' = \texttt{let}\ (y,x) = (\bullet\,(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)); (\texttt{unseal}_X x, \texttt{unseal}_Y y)$$
$$S_r'' = \texttt{let}\ (y,x) = (\bullet\,(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)); (\texttt{unseal}_Y y, \texttt{unseal}_X x)$$

Similarly to before, we know $(w_4, \texttt{force}\ V'_{\forall^v l}\, B_l\, \sigma_{Yl}, \texttt{force}\ V'_{\forall^v r}\, A_r\, \sigma_{Yr}) \in \mathcal{E}^\sim[\![(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$ so we can apply monadic bind again. Let $w_5 \sqsupseteq w_r$ and $(w_5, V_{fl}, V_{fr}) \in \mathcal{V}^\sim[\![(X \times Y) \to (Y \times X)]\!]\gamma'\delta'$. Next, note that

$$[\![(\texttt{seal}_X V_A, \texttt{seal}_Y V_B)]\!][\gamma_l'][\delta_l'] \mapsto^0 \texttt{ret}\ (V_{Al}, V_{Bl})$$
$$[\![(\texttt{seal}_X V_B, \texttt{seal}_Y V_A)]\!][\gamma_r'][\delta_r'] \mapsto^0 \texttt{ret}\ (V_{Br}, V_{Ar})$$

So by anti-reduction we need to show

$$(w_5, [\![S_l'''']\!][\texttt{force}\ V'_{\forall^v l}\,(V_{Al}, V_{Bl})], [\![S_r'''']\!][\texttt{force}\ V'_{\forall^v r}\,(V_{Br}, V_{Ar})]) \in \mathcal{E}^\sim[\![A \times B]\!]\gamma'\delta'$$

where

$$S_l''' = \texttt{let}\ (y,x) = \bullet; (\texttt{unseal}_X x, \texttt{unseal}_Y y)$$
$$S_r''' = \texttt{let}\ (y,x) = \bullet; (\texttt{unseal}_Y y, \texttt{unseal}_X x)$$

To show that

$$(w_5, \texttt{force}\ V'_{\forall^v l}\,(V_{Al}, V_{Bl}), \texttt{force}\ V'_{\forall^v r}\,(V_{Br}, V_{Ar})) \in \mathcal{E}^\sim[\![Y \times X]\!]\gamma'\delta'$$

It is sufficient to show

$$(w_5, (V_{Al}, V_{Bl}), (V_{Br}, V_{Ar})) \in \mathcal{E}^\sim[\![X \times Y]\!]\gamma'\delta'$$

which follows because $(w_5, V_{Al}, V_{Br}) \in R_X$ and $(w_5, V_{Br}, V_{Ar}) \in R_Y$. Then we can apply monadic bind a final time. Let $w_{fin} \sqsupseteq w_5$ and $(w_{fin}, V_{ol}, V_{or}) \in \mathcal{V}^\sim[\![Y \times X]\!]\gamma'\delta'$. Then we need to show

$$(w_{fin}, [\![S_l''']\!][\texttt{ret}\ V_{ol}], [\![S_r''']\!][\texttt{ret}\ V_{or}]) \in \mathcal{E}^\sim[\![A \times B]\!]\gamma'\delta'$$

First, by definition, it must be the case that $V_{ol} = (V_{Bl}, V_{Al})$ and $V_{or} = (V_{Ar}, V_{Br})$. Then by anti-reduction we need to show

$$(w_{fin}, \texttt{ret}\ (V_{Al}, V_{Bl}), \texttt{ret}\ (V_{Ar}, V_{Br})) \in \mathcal{E}^\sim[\![A \times B]\!]\gamma'\delta'$$

which means we need to show

$$(w_{fin}, V_{Al}, V_{Ar}) \in \mathcal{V}^\sim[\![A]\!]\gamma'\delta'$$

and

$$(w_{fin}, V_{Bl}, V_{Br}) \in \mathcal{V}^\sim[\![B]\!]\gamma'\delta'$$

which follows by reflexivity and monotonicity. □

Lemma H.8.

$$\begin{array}{c}\mathsf{pack}^{\nu}(X \cong \mathbb{B}, (\mathsf{seal}_X \mathtt{true}, (\mathit{NOT}, \lambda x : X.\mathsf{unseal}_X x))) \\ \approx^{ctx} \mathsf{pack}^{\nu}(X \cong \mathbb{B}, (\mathsf{seal}_X \mathtt{true}, (\mathit{NOT}, \lambda x : X.\mathit{NOT}(\mathsf{unseal}_X x))))\end{array}$$

Proof. We do the $\sqsubseteq\sim$ case, the $\sim\sqsupseteq$ case is symmetric. Let $(w, \gamma, \delta) \in \mathcal{G}^{\sim}[\![\Gamma_p]\!]$. The goal reduces to showing

$$\begin{pmatrix} w, & \mathsf{pack}(\mathbb{B}, (\mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ X.[\![(\mathsf{seal}_X \mathtt{true}, (\mathrm{NOT}, (\lambda x : X.\mathsf{unseal}_X x)))]\!])), \\ & \mathsf{pack}(\mathbb{B}, \mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ X.[\![(\mathsf{seal}_X \mathtt{false}, (\mathrm{NOT}, \lambda x : X.\mathrm{NOT}\,(\mathsf{unseal}_X x)))]\!]) \end{pmatrix} \\ \in \mathcal{V}^{\sim}[\![\exists^{\nu} X.X \times ((X \to X) \times (X \to \mathbb{B}))]\!]\gamma\delta$$

The relation we pick is $R = \{(w', \mathtt{true}, \mathtt{false}) \in \mathrm{Atom}[\mathbb{B}, \mathbb{B}]\} \cup \{(w', \mathtt{false}, \mathtt{true}) \in \mathrm{Atom}[\mathbb{B}, \mathbb{B}]\}$ Then we need to show for any future $w' \sqsupseteq w$ and $w' \vDash (\sigma_l, \sigma_r, \lfloor R \rfloor_{w'.j})$, that

$$\begin{pmatrix} w', & (\mathtt{force}\ (\mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ X.[\![(\mathsf{seal}_X \mathtt{true}, (\mathrm{NOT}, \lambda x : X.\mathsf{unseal}_X x))]\!])\ \sigma_l), \\ & (\mathtt{force}\ \mathsf{thunk}\ \lambda c_X : \mathrm{Case}\ X.[\![(\mathsf{seal}_X \mathtt{true}, (\mathrm{NOT}, \lambda x : X.\mathrm{NOT}\,(\mathsf{unseal}_X x)))]\!]\ \sigma_r) \end{pmatrix} \\ \in \mathcal{E}^{\sim}[\![X \times ((X \to X) \times (X \to \mathbb{B}))]\!]\gamma'\delta'$$

where

$$\gamma' = \gamma, c_X \mapsto (\sigma_l, \sigma_r)$$
$$\delta' = \delta, X \mapsto (\mathbb{B}, \mathbb{B}, R)$$

And after applying anti-reduction, by monadic bind, we need to show the following 3 things:

$$(w', \mathtt{true}, \mathtt{false}) \in \mathcal{V}^{\sim}[\![X]\!]\gamma'\delta'$$

$$(w', [\![\mathrm{NOT}]\!][\gamma_l'][\delta_l'], [\![\mathrm{NOT}[\gamma_r'][\delta_r']]\!]) \in \mathcal{E}^{\sim}[\![X \to X]\!]\gamma'\delta'$$

$$(w', [\![\lambda x : X.\mathsf{unseal}_X x]\!], [\![\lambda x : X.\mathrm{NOT}\,(\mathsf{unseal}_X x)]\!]) \in \mathcal{E}^{\sim}[\![X \to \mathbb{B}]\!]\gamma'\delta'$$

(1) First, $(w', \mathtt{true}, \mathtt{false}) \in \mathcal{V}^{\sim}[\![X]\!]\gamma'\delta'$ follows directly from the definition of $\delta_R(X) = R$.
(2) Second, let $w'' \sqsupseteq w'$ and $(w'', V_l, V_r) \in \mathcal{V}^{\sim}[\![X]\!]\gamma'\delta'$. Then we need to show

$$(w'', \mathtt{force}\ V_{NOT}[\gamma_l'][\delta_l']\, V_l, \mathtt{force}\ V_{NOT}[\gamma_r'][\delta_r']\, V_r) \in \mathcal{E}^{\sim}[\![X]\!]\gamma'\delta'$$

where $[\![\mathrm{NOT}]\!] = \mathtt{ret}\ V_{NOT}$. There are two cases: either $V_l = \mathtt{true}$ and $V_r = \mathtt{false}$ or vice-versa. In either case, NOT swaps the two values and the result holds.
(3) Finally, let $w'' \sqsupseteq w'$ and $(w'', V_l, V_r) \in \mathcal{V}^{\sim}[\![X]\!]\gamma'\delta'$. Then we need to show,

$$(w'', \mathtt{force}\ V_{fl}\ V_l, \mathtt{force}\ V_{fr}\ V_r) \in \mathcal{E}^{\sim}[\![\mathbb{B}]\!]\gamma'\delta'$$

where $[\![\lambda x : X.\mathsf{unseal}_X x]\!] = \mathtt{ret}\ V_{fl}$ and $[\![\lambda x : X.\mathrm{NOT}\,(\mathsf{unseal}_X x)]\!] = \mathtt{ret}\ V_{fr}$. By definition of R, either $V_l = \mathtt{true}$ and $V_r = \mathtt{false}$ or vice-versa. In either case, both sides evaluate to $\mathtt{ret}\ V_l$, and we need to show

$$(w'', V_l, V_l) \in \mathcal{V}^{\sim}[\![\mathbb{B}]\!]\gamma'\delta'$$

which follows by definition.

□

Finally we prove Lemma H.1 that states that semantic term precision is a congruence.

Proof. By induction over C. In ever non-cast case we use the corresponding compatibility rule. The two casts cases are precisely dual.

- If $C = \langle A^{\sqsubseteq} \rangle\!\uparrow C'$, where $\Gamma, \Gamma'_o \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$ then we need to show
$$\Gamma' \vDash \langle A^{\sqsubseteq} \rangle\!\uparrow C'[M_l] \sqsubseteq \langle A^{\sqsubseteq} \rangle\!\uparrow C'[M_r] \in A_r; \Gamma'_o$$
First we use the upcast-left compatibility rule, meaning we need to show
$$\Gamma' \vDash C'[M_l] \sqsubseteq \langle A^{\sqsubseteq} \rangle\!\uparrow C'[M_r] \in A^{\sqsubseteq}; \Gamma'_o$$
Next, we use the upcast-right compatibility rule, meaning we need to show
$$\Gamma' \vDash C'[M_l] \sqsubseteq C'[M_r] \in A_l; \Gamma'_o$$
which follows by inductive hypothesis.
- If $C = \langle A^{\sqsubseteq} \rangle\!\downarrow C'$, where $\Gamma, \Gamma'_o \vdash A^{\sqsubseteq} : A_l \sqsubseteq A_r$ then we need to show
$$\Gamma' \vDash \langle A^{\sqsubseteq} \rangle\!\downarrow C'[M_l] \sqsubseteq \langle A^{\sqsubseteq} \rangle\!\downarrow C'[M_r] \in A_l; \Gamma'_o$$
First we use the downcast-right compatibility rule, meaning we need to show
$$\Gamma' \vDash \langle A^{\sqsubseteq} \rangle\!\downarrow C'[M_l] \sqsubseteq C'[M_r] \in A^{\sqsubseteq}; \Gamma'_o$$
Next, we use the downcast-left compatibility rule, meaning we need to show
$$\Gamma' \vDash C'[M_l] \sqsubseteq C'[M_r] \in A^{\sqsubseteq}; \Gamma'_o$$
which follows by inductive hypothesis.

□