Equational Theories with Recursive Types (Extended Version)

Christopher A. Stone and Andrew P. Schoonmaker

2005

Abstract

Studies of equivalence for recursive types often consider impoverished type systems, where the equational theory is generated only by the fold/unfold rule $\mu X.\, T(X) \equiv T(\mu X.\, T(X))$. Recursive types have been applied in much richer contexts, including systems with β and η-equivalence, but without any guarantee that the implementations are correct. Though there are plausible ways to adapt standard recursive-type algorithms to richer equational theories, Colazzo and Ghelli observed that two "obvious" ways of extending the algorithm in a different direction (adding universally-quantified types) both fail. Extended systems may not even be formally specified; combining $\beta\eta$-equivalence with coinductive equivalence of recursive types requires care to avoid inconsistency.

In this paper we both define and analyze coinductive equivalence for recursive types combined with other common equational principles. We start by adding pairing and projection, allowing even pairs to be recursively defined. (This permits direct definitions for collections of mutually-recursive types.) We show that our definition yields a decidable theory with all the expected equational properties.

We then extend the system with first-order (non-recursive) type operators and β-equivalence, and show the same equational and decidability properties hold. Finally we add extensionality for both pairs and functions, obtaining a coinductively-defined theory of recursive types with $\beta\eta$-equivalence.

1 Introduction

A number of researchers have studied theories of equivalence or subtyping for recursive types [AC93, AF96, BH97, GLP02]. With only a few exceptions — typically adding isomorphisms such as associativity and commutativity of products [PZ00, Fio04, DPR05] — recursive types are studied in isolation, and the only nontrivial equivalences arise from recursive types and the so-called *fold/unfold* rule.

In practice, however, this may not be enough. For example, recursive types can be useful in the presence of parameterized types [BCP99], yet studies of recursive types generally omit type operators. Similarly, the FLINT and TILT implementations of Standard ML [LS98, VDP$^+$03] have used a slightly more restrictive variant of recursive type equivalence (based on an *unfold/unfold* rule), combined with β and η-equivalence and primitive notions of mutually-recursive types.

A common assumption is that algorithms designed for the simple system of recursive types will continue to work when the type system changes. However, Colazzo and Ghelli [CG99] observed that in the presence of universally quantified types (and hence of bound variables that cannot be eliminated through unfolding), the most obvious ways to extend the algorithm either fail to terminate or are unsound.

The goal of this paper is to verify that existing ideas from the study of equivalence of recursive types do extend to richer equational theories. We begin by reviewing some basic results used to study the simple system of recursive types, based on the presentation of Gapeyev et al. [GLP02]. We show how to extend the coinductive definition of equivalence to include pairs of types and projection operators (along with recursive definitions of pairs), and show that a sound, complete, and terminating equivalence algorithm can still be obtained. This extension allows direct definitions of mutually-recursive types, without specialized primitives or unintuitive encodings.

The approach for pairs then extends further to include type operators and β-reduction, provided that kinds are restricted to first order (no arrows in negative positions). We finally add extensionality principles: pointwise-equivalent functions are equivalent, and componentwise-equivalent pairs are equivalent. We thus obtain a coinductive theory of $\beta\eta$-equivalence with recursive types.

2 Review

2.1 Recursive Types

There are two traditional frameworks for recursive types, differing in how the type $\mu X.\, T(X)$ relates to the equation $X = T(X)$. In the *isorecursive* approach, recursive types induce no interesting type equivalences. The type $\mu X.\, T(X)$ is isomorphic to but not equal to $T(\mu X.\, T(X))$, and there are inverse term-level operators

$$\begin{array}{lcl} \mathtt{fold}_{\mu X.\, T(X)} & : & T(\mu X.\, T(X)) \;\rightarrow\; \mu X.\, T(X) \\ \mathtt{unfold}_{\mu X.\, T(X)} & : & \mu X.\, T(X) \;\rightarrow\; T(\mu X.\, T(X)) \end{array}$$

witnessing this isomorphism. Often these operators have no observable run-time effects, but their presence simplifies type checking [VDP$^+$03], just as systems with explicit type coercions are generally easier to type check than systems with implicit subsumption.

However, explicit coercions can be unwieldy at times. The *equirecursive* approach defines the type $\mu X.\, T(X)$ to be equal to $T(\mu X.\, T(X))$. Explicit `fold` and `unfold` term operators become unnecessary, but this immediately leads to a non-trivial equational theory of types.

Given the decision to study an equirecursive system, as we will do here, there is still a choice whether equivalence should be defined inductively (as usual in the absence of recursive types) or coinductively. Coinductive equivalence is often motivated by a view of recursive types as finite representations of potentially infinite μ-free types. Thus, $\mu X.\, \mathtt{int}\!\rightarrow\! X$ is a finite representation of the infinite type

$$\mathtt{int}\!\rightarrow\!\mathtt{int}\!\rightarrow\!\mathtt{int}\!\rightarrow\!\cdots.$$

The same infinite type can be represented by $\mu X.\, \mathtt{int}\!\rightarrow\!\mathtt{int}\!\rightarrow\! X$, because repeated unfoldings approach the same limit. Though no finite sequence of foldings and unfoldings can make the two recursive types identical (i.e., the types are not inductively equivalent), they have the same limit and so coinductively we have

$$\mu X.\, \mathtt{int}\!\rightarrow\! X \equiv \mu X.\, \mathtt{int}\!\rightarrow\!\mathtt{int}\!\rightarrow\! X.$$

The coinductive approach tends to be more useful than the inductive approach, as it safely equates more types. For example, suppose we have two separately-defined recursive types T_1 and

T_2. If instead we were encode the two as mutually-recursive types — the degenerate case where the types could refer to each other but don't — we typically obtain results that are coinductively but not inductively equivalent to the original T_1 and T_2. (Mutually-recursive definitions are described in more detail in Section 4.) We thus consider coinductive equivalence here.

2.2 Coinduction

We first review what if means to define equivalence coinductively. By following the presentation of Gapeyev et al. [GLP02], we can work directly with syntactic types, rather than defining types as reasoning about infinite trees (and reduction steps for infinite trees).

Assume $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ is a function from the subsets of $\mathcal{U}$ to the subsets of $\mathcal{U}$. If F is monotone then it has a unique greatest fixed point, written νF, satisfying $\nu F = F(\nu F)$. A defining property of this greatest fixed point is the following:

Definition 1 (Principle of Coinduction)
Assume F is monotone, so that νF exists. If $\mathcal{A} \subseteq F(\mathcal{A})$ then $\mathcal{A} \subseteq \nu F$.

The premise of the Principle of Coinduction can be weakened; $\mathcal{A} \subseteq \nu F$ if $\mathcal{A}$ is a subset of $F(\mathcal{A})$, or of the larger set $F(\mathcal{A} \cup F(\mathcal{A}))$, or of the still larger set $F(\mathcal{A} \cup F(\mathcal{A} \cup F(\mathcal{A})))$, and so on. Pushing this idea to the limit we obtain the following generalization:

Proposition 2 (Extended Principle of Coinduction)
Let F be a monotone function on sets, and $\mathcal{A}$ be a set. Define

$$F^{+\mathcal{A}}(X) := \mathcal{A} \cup F(X).$$

Then $\mathcal{A} \subseteq F(\nu F^{+\mathcal{A}})$ if and only if $\mathcal{A} \subseteq \nu F$.

Proof: Assume $\mathcal{A} \subseteq F(\nu F^{+\mathcal{A}})$. Then $\nu F^{+\mathcal{A}} = \mathcal{A} \cup F(\nu F^{+\mathcal{A}}) \subseteq F(\nu F^{+\mathcal{A}})$. By the Principle of Coinduction $\nu F^{+\mathcal{A}} \subseteq \nu F$, and thus $\mathcal{A} \subseteq \nu F$.

Conversely, assume $\mathcal{A} \subseteq \nu F$. Since pointwise $F \subseteq F^{+\mathcal{A}}$ we have $\nu F \subseteq \nu F^{+\mathcal{A}}$. Then by monotonicity $\mathcal{A} \subseteq \nu F = F(\nu F) \subseteq F(\nu F^{+\mathcal{A}})$. ∎

Corollary 3
Assume F is a monotone function on sets. If $\mathcal{A} \subseteq F(\mathcal{A}) \cup F(F(\mathcal{A}))$ then $\mathcal{A} \subseteq \nu F$. More generally, if $\mathcal{A} \subseteq \bigcup_{n\geq 1} F^n(\mathcal{A})$ then $\mathcal{A} \subseteq \nu F$.

Proof: Assume $\mathcal{A} \subseteq \bigcup_{n\geq 1} F^n(\mathcal{A})$. An easy inductive argument shows that $F^n(\mathcal{A}) \subseteq \nu F^{+\mathcal{A}}$ for every $n \geq 0$, and so using monotonicity, $\mathcal{A} \subseteq \bigcup_{n\geq 1} F^n(\mathcal{A}) = \bigcup_{n\geq 0} F(F^n(\mathcal{A})) \subseteq \bigcup_{n\geq 0} F(\nu F^{+\mathcal{A}}) = F(\nu F^{+\mathcal{A}})$. Therefore $\mathcal{A} \subseteq \nu F$ by Proposition 2. ∎

The application of greatest fixed points to type equivalence is that equivalence for recursive types can be defined coinductively as the greatest fixed point (rather than the more usual inductively-defined least fixed point) of the inference rules:

$$\frac{}{T \equiv T} \tag{1}$$

$$\frac{T_1 \equiv S_1 \qquad T_2 \equiv S_2}{T_1 {\to} T_2 \equiv S_1 {\to} S_2} \tag{2}$$

$$\frac{\{\mu X.\, T/X\}T \equiv S}{\mu X.\, T \equiv S} \tag{3}$$

$$\frac{T \equiv \{\mu X.\, S/X\}S}{T \equiv \mu X.\, S} \tag{4}$$

Here the notation $\{S/X\}T$ denotes the capture-avoiding substitution of S for free occurrences of X in the type T.

These rules can be viewed as a function from a set of premises to the set of those conclusions derivable in one step:

$$\begin{aligned} F_\mu(\mathcal{J}) := \; & \{\, (T \;\equiv\; T) \mid \text{for all types } T\} \\ \cup \; & \{\, (T_1{\to}T_2 \;\equiv\; S_1{\to}S_2) \mid (T_1 \;\equiv\; S_1) \in \mathcal{J} \text{ and } (T_2 \;\equiv\; S_2) \in \mathcal{J}\} \\ \cup \; & \{\, (\mu X.\, T \;\equiv\; S) \mid (\{\mu X.\, T/X\}T \;\equiv\; S) \in \mathcal{J}\} \\ \cup \; & \{\, (T \;\equiv\; \mu X.\, S) \mid (T \;\equiv\; \{\mu X.\, S/X\}S) \in \mathcal{J}\} \end{aligned}$$

The usual inductive interpretation of inference rules corresponds to the least fixed point of F_μ, while the coinductive interpretation is the greatest fixed point νF_μ. To assert a coinductive equivalence judgment $T \equiv S$ is to say that $(T \;\equiv\; S) \in \nu F_\mu$.

2.3 Algorithms

In some cases the decidability of membership in a greatest fixed point can be determined directly from properties of the generating function [GLP02].

A monotone function $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ is said to be *invertible* if for all $x \in \mathcal{U}$, the collection of sets sufficient to produce x,

$$\mathit{suff}[F](x) := \{\mathcal{A} \subseteq \mathcal{U} \mid x \in F(\mathcal{A})\}$$

is either empty or has a minimum element with respect to inclusion. When F is invertible, we define

$$\mathit{support}[F](x) := \begin{cases} \min \mathit{suff}[F](x) & \text{if } \mathit{suff}[F](x) \neq \emptyset \\ \uparrow & \text{otherwise} \end{cases}$$

In the context of functions mapping premises to conclusions, invertibility corresponds to proof search being deterministic. The support of a judgment is then the unique, minimal set of premises required to prove that judgment.

Given an invertible $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ and $x \in \mathcal{U}$ and $\mathcal{A} \in 2^{\mathcal{U}}$, let

$$\begin{aligned} \mathit{pred}[F](x) \quad &:= \quad \begin{cases} \emptyset & \text{if } \mathit{support}[F](x) = \uparrow \\ \mathit{support}[F](x) & \text{otherwise} \end{cases} \\ \mathit{pred}[F](\mathcal{A}) \quad &:= \quad \bigcup_{x \in \mathcal{A}} \mathit{pred}[F](x) \end{aligned}$$

The set of elements reachable from a set $\mathcal{A}$ is

$$\mathit{reachable}[F](\mathcal{A}) := \bigcup_{n \geq 0} \mathit{pred}[F]^n(\mathcal{A}).$$

An invertible function $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ is said to be *finite-state* if for all $x \in \mathcal{U}$, the set $\mathit{reachable}[F](\{x\})$ is finite. In the context of inference rules, this means that proof search finds only finitely many judgments before looping or terminating.

Proposition 4
If $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ is invertible and finite-state then membership in νF is decidable.

Proposition 4 can be proved constructively by presenting a sound, complete, and terminating algorithm; several are available. The following simple (though not most efficient) algorithm is a generalization of the subtyping algorithm of Amadio and Cardelli [AC93]:

```
$\mathit{gfp}^{ac}[F](\mathcal{A}, x) :=$  if $x \in \mathcal{A}$ then $\mathit{true}$
                     else if $\mathit{support}[F](x) = \uparrow$ then $\mathit{false}$
                     else $\bigwedge_{y \in \mathcal{A}'} \mathit{gfp}^{ac}[F](\mathcal{A} \cup \{x\}, y)$
                         where $\mathcal{A}' := \mathit{support}[F](x)$
```

If F is invertible and finite-state, then $\mathit{gfp}^{ac}[F](\emptyset, x) = \mathit{true}$ if $x \in \nu F$, and $\mathit{gfp}^{ac}[F](\emptyset, x) = \mathit{false}$ otherwise. (More generally, $\mathit{gfp}^{ac}[F](\mathcal{A}, x)$ tests x for membership in $\nu F^{+\mathcal{A}}$.) The correctness proof for this algorithm appears in Appendix A.

The function F_μ defined above is not invertible, since an equivalence $\mu X.\, T \equiv \mu Y.\, S$ can follow either from $\{\mu X.\, T/X\}T \equiv \mu Y.\, S$ or from $\mu X.\, T \equiv \{\mu Y.\, S/Y\}S$. However, the closely-related "algorithmic" variant

$$\begin{aligned} F^a_\mu(\mathcal{J}) := \; & \{ (\texttt{int} \;\equiv\; \texttt{int}) \} \\ \cup \; & \{ (X \;\equiv\; X) \mid \text{for all variables } X \} \\ \cup \; & \{ (T_1{\to}T_2 \;\equiv\; S_1{\to}S_2) \mid (T_1 \;\equiv\; S_1) \in \mathcal{J} \text{ and } (T_2 \;\equiv\; S_2) \in \mathcal{J} \} \\ \cup \; & \{ (\mu X.\, T_1 \;\equiv\; S) \mid (\{\mu X.\, T_1/X\}T_1 \;\equiv\; S) \in \mathcal{J} \} \\ \cup \; & \{ (T \;\equiv\; \mu X.\, S_1) \mid T \text{ is not of the form } \mu X.\, T_1 \text{ and} \\ & \qquad\qquad (T \;\equiv\; \{\mu X.\, S_1/X\}S_1) \in \mathcal{J} \} \end{aligned}$$

is both invertible and finite-state. Further, we have $\nu F^a_\mu = \nu F_\mu$ [GLP02], so we can use it to test equivalence. Instantiating the general algorithm for membership in the greatest fixed point with the function F^a_μ, we have that $T \equiv S$ if and only if $\mathit{gfp}^{ac}[F^a_\mu](\emptyset, (T \;\equiv\; S))$ returns true, where expanding the definition we have:

```
$\mathit{gfp}^{ac}[F^a_\mu](\mathcal{A}, (T \;\equiv\; S)) =$
  if $(T \;\equiv\; S) \in \mathcal{A}$ then $\mathit{true}$
  else if $T = \texttt{int}$ and $S = \texttt{int}$ then $\mathit{true}$
  else if $T = X$ and $S = X$ then $\mathit{true}$
  else if $T = T_1{\to}T_2$ and $S = S_1{\to}S_2$ then
        $\mathit{gfp}^{ac}[F^a_\mu](\mathcal{A} \cup \{(T \;\equiv\; S)\}, (T_1 \;\equiv\; S_1)) \wedge$
         $\mathit{gfp}^{ac}[F^a_\mu](\mathcal{A} \cup \{(T \;\equiv\; S)\}, (T_2 \;\equiv\; S_2))$
  else if $T = \mu X.\, T_1$ then
        $\mathit{gfp}^{ac}[F^a_\mu](\mathcal{A} \cup \{(T \;\equiv\; S)\}, (\{\mu X.\, T_1/X\}T_1 \;\equiv\; S))$
  else if $S = \mu X.\, S_1$ then
        $\mathit{gfp}^{ac}[F^a_\mu](\mathcal{A} \cup \{(T \;\equiv\; S)\}, (T \;\equiv\; \{\mu X.\, S_1/X\}S_1))$
  else $\mathit{false}$.
```

3 Recursive Types and Pairing

Applications of recursive types often require mutually-recursive types. Theoretical studies of recursive types usually ignore this issue, as mutual recursion can be encoded in terms of nested recursion (see Section 4). For the purposes of clarity and implementation efficiency, however, it may be worthwhile to have a language of types that can directly handle mutual recursion. We therefore extend the standard calculus of recursive types with pairs of types and projections from such pairs. We allow not just pairs of recursively-defined types, but recursively-defined pairs as well.

This extension is also of interest because it represents a particularly simple but non-trivial extension of the traditional equational theory for recursive types.

3.1 Syntax

The syntax of the type system is specified by the following grammar:

$$
\begin{array}{rcl}
K, L & ::= & \mathtt{*} \\
 & | & K{\times}K \\
 & & \\
S, T, U & ::= & \mathtt{int} \\
 & | & X \mid Y \mid Z \mid \cdots \\
 & | & T{\to}T \\
 & | & \mu X{::}L.\, T \\
 & | & \langle T, T\rangle \\
 & | & \pi_1\, T \\
 & | & \pi_2\, T
\end{array}
$$

The kind system distinguishes proper types of kind `*` from type-level pairs. The pair $\langle T_1, T_2\rangle$ is a collection of two types and will have a kind of the form $K_1{\times}K_2$. (Note that $\langle T_1, T_2\rangle$ is not the proper type that would classify a pair of values; such a type $T_1{\times}T_2$ would have kind `*`. Types of pairs could be added, but since they are equationally very similar to types of the form $T_1{\to}T_2$ they have been omitted.)

The notation $FV(T)$ denotes the set of free variables in T, where $\mu X{::}L.\, S$ binds X in S. Types are identified up to renaming of bound variables.

3.2 Weak Head Reduction

Our generalization of the "unfolding" transformation for recursive types is weak head reduction. To define this relation we use the concept of an *elimination context* E. These contexts are defined inductively according to the following grammar:

$$E ::= \bullet \mid \pi_1\, E \mid \pi_2\, E$$

Every elimination context contains a single *hole*, written $\bullet$. If E is an elimination context, we write $E[T]$ for the type obtained by replacing the hole in E with T. For example, if $E = \pi_1\,(\pi_2\, \bullet)$ then $E[\pi_1\, X] = \pi_1\,(\pi_2\,(\pi_1\, X))$. A type of the form $E[X]$ or $E[\mathtt{int}]$, an elimination context applied to a variable or constant, is called a *path* and denoted P.

The *weak head reduction* relation $\leadsto$ on types is defined by two axioms:

$$\begin{array}{lcl} E[\pi_i\,\langle T_1, T_2\rangle] & \leadsto & E[T_i] \\ E[\mu X{::}L.\,T] & \leadsto & E[\{\mu X{::}L.\,T/X\}T]. \end{array}$$

Unfolding and projections may occur inside projections. Thus, when

$$S := \mu X{::}{*}{\times}{*}.\,\langle \texttt{int}{\to}\pi_2\,X, \pi_1\,X{\to}\texttt{int}\rangle$$

we have $\pi_2\,S \leadsto \pi_2\,\langle \texttt{int}{\to}\pi_2\,S, \pi_1\,S{\to}\texttt{int}\rangle \leadsto \pi_1\,S{\to}\texttt{int}$. We use $\leadsto^*$ to denote the reflexive, transitive closure of this relation, and write $T \not\leadsto$ when T is weak head normal (cannot be reduced).

3.3 Well-Formedness

Well-formedness of types is relative to a typing (or in this system, kinding) context Γ, defined by the following grammar:

$$\begin{array}{rcl} \Gamma & ::= & \cdot \\ & \mid & \Gamma, X{::}K \end{array}$$

Contexts can be treated as partial functions from variables to their kinds. When $\mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2) = \emptyset$, we write Γ_1, Γ_2 to be the concatenation of the two contexts.

The well-formedness judgment for types is defined inductively, as usual, by the following sequence of inference rules:

$$\frac{}{\Gamma \vdash \texttt{int} :: *} \tag{5}$$

$$\frac{X \in \mathrm{dom}(\Gamma)}{\Gamma \vdash X :: \Gamma(X)} \tag{6}$$

$$\frac{\Gamma \vdash T_1 :: * \qquad \Gamma \vdash T_2 :: *}{\Gamma \vdash T_1{\to}T_2 :: *} \tag{7}$$

$$\frac{\Gamma, X{::}L \vdash T :: L}{\Gamma \vdash \ \mu X{::}L.T \ :: L} \tag{8}$$

$$\frac{\Gamma \vdash T_1 :: K_1 \qquad \Gamma \vdash T_2 :: K_2}{\Gamma \vdash \langle T_1, T_2\rangle :: K_1{\times}K_2} \tag{9}$$

$$\frac{\Gamma \vdash T :: K_1{\times}K_2}{\Gamma \vdash \pi_i\,T :: K_i} \tag{10}$$

Proposition 5 (Basic Properties of Well-Formedness)

1. *If* $\Gamma \vdash T :: K_1$ *and* $\Gamma \vdash T :: K_2$ *then* $K_1 = K_2$.

2. *If* $\Gamma \vdash T :: K$ *then* $FV(T) \subseteq dom(\Gamma)$.

3. *If* $\Gamma_1, \Gamma_3 \vdash T :: K$ *and* $dom(\Gamma_1, \Gamma_3) \cap dom(\Gamma_2) = \emptyset$ *then* $\Gamma_1, \Gamma_2, \Gamma_3 \vdash T :: K$.

4. *If* $\Gamma_1, Y{::}K', \Gamma_2 \vdash T :: K$ *and* $\Gamma \vdash T' :: K'$ *then* $\Gamma_1, \Gamma_2 \vdash \{T'/Y\}T :: K$.

Proof: By induction on derivations. ∎

Proposition 6 (Characterization of Kinds)

1. *If* $\Gamma \vdash T :: *$ *then* $T \leadsto T'$ *or* T *is a path or* $T = T_1 \to T_2$*.*

2. *If* $\Gamma \vdash T :: K_1 \times K_2$ *then* $T \leadsto T'$ *or* T *is a path or* $T = \langle T_1, T_2 \rangle$*.*

Beyond the well-formedness rules, we follow usual practice for equirecursive types by requiring that *all* types considered are contractive, a global syntactic restriction discussed further in the following section.

3.4 Contractiveness

Equirecursive recursive types are often motivated as finite representations of (potentially infinite) μ-free types, the limit of repeated unfoldings. Thus, $\mu X{::}*.\, \texttt{int} \to X$ unfolds to $\texttt{int} \to \texttt{int} \to \texttt{int} \to \cdots$, while $\mu X{::}*.\, X \to X$ unfolds to $(\cdots \to \cdots) \to (\cdots \to \cdots)$, an infinite binary tree where every node is →, while $\mu X{::}*.\, \texttt{int}$ unfolds to simply `int`. Not all syntactic recursive types correspond to such μ-free types, though; the type $\mu X{::}*.\, X$ unfolds only to itself. Such types are typically forbidden; types that correspond to trees are said to be contractive.

Although we do not define equivalence in terms of infinite trees, in order to obtain a sound theory we must make a similar syntactic restriction. Otherwise, coinduction and Rule 3 would imply that $(\mu X.\, X) \equiv S$ holds for *every* type S.

In simple systems contractiveness can be syntactically enforced in the grammar itself, for example by requiring that the body of every recursive type be a function arrow [BH97]. Here we formalize the intuition that μ-bound variables should appear only inside → by using a notion of *unguarded variables.* The unguarded variables of a type T, written $UV(T)$, are defined by:

$$\begin{array}{rcl}
UV(\texttt{int}) & := & \emptyset \\
UV(T_1 \to T_2) & := & \emptyset \\
UV(X) & := & \{X\} \\
UV(\pi_i\, T) & := & UV(T) \\
UV(\langle T_1, T_2 \rangle) & := & UV(T_1) \cup UV(T_2) \\
UV(\mu X{::}L.\, T) & := & UV(T) \setminus \{X\}
\end{array}$$

A type is then said to be *contractive* if every occurrence of a recursive type $\mu X{::}L.\, T$ satisfies $X \notin UV(T)$. Thus, the type $\pi_2\, (\mu Y{::}*{\times}*.\, \langle \pi_2\, Y, \pi_1\, Y \rangle)$ (which reduces to itself in four steps) is not contractive since $Y \in UV(\langle \pi_2\, Y, \pi_1\, Y \rangle)$.

Contractiveness is still purely syntactic, is not context-sensitive, and is preserved by capture-avoiding substitutions and by reductions. Following convention, we assume for the rest of the paper that all types mentioned are contractive.[1]

[1] Crary et al. [CHP99] give a closely-related definition of contractiveness for well-formed types, but they expand the set of contractive types to include every type provably equivalent to some type that is syntactically contractive in our sense. In the absence of defined type variables this extra flexibility does not appear useful, especially compared to the complexity introduced by defining contractiveness mutually with of well-formedness and type equivalence. Though we do rule out types such as $\mu X{::}*.\, \pi_1\, \langle X \to X, X \rangle$ that their approach would allow, one could write the syntactically-contractive type $\mu X{::}*.\, X \to X$ in the first place. We conjecture that equivalence with their more general notion of contractiveness would be a conservative extension of equivalence as defined here.

We depend on two properties of contractive types: any subcomponent of a contractive type is itself contractive (by definition of contractiveness), and weak head reduction of contractive types terminates.

There are several ways to prove the latter property. For example, we can explicitly define a nonnegative "height" measure for recursive types that is strictly reduced by weak head reduction. (Simpler type systems can just count the outermost μ's in a recursive type.)

$$\begin{array}{rcl}
\mathit{height}(\texttt{int}) & := & 0 \\
\mathit{height}(T_1{\to}T_2) & := & 0 \\
\mathit{height}(X) & := & 0 \\
\mathit{height}(\pi_1\,T) & := & \mathit{height}(T) \\
\mathit{height}(\pi_2\,T) & := & \mathit{height}(T) \\
\mathit{height}(\langle T_1,T_2\rangle) & := & 1+\max(\mathit{height}(T_1),\mathit{height}(T_2)) \\
\mathit{height}(\mu X{::}K.\,T) & := & 1+\mathit{height}(T)
\end{array}$$

Proposition 7
If $X \notin UV(T)$ then $\mathit{height}(\{T'/X\}T) = \mathit{height}(T)$.

Proof: By induction on T.

- Case: $T = \texttt{int}$. Both heights are zero.
- Case: $T = T_1{\to}T_2$. Both heights are zero.
- Case: $T = Y$. Then $Y \neq X$ since $Y \in UV(T)$, so both heights are zero.
- Case: $T = \pi_i\,S$. By the inductive hypothesis, since $X \notin UV(\pi_i\,S) = UV(S)$, we have $\mathit{height}(\{T'/X\}(\pi_i\,S)) = \mathit{height}(\{T'/X\}S) = \mathit{height}(S) = \mathit{height}(\pi_i\,S)$.
- Case: $T = \langle S_1, S_2\rangle$. Since $X \notin UV(T) = UV(S_1) \cup UV(S_2)$, by induction $\mathit{height}(\{T'/X\}\langle S_1,S_2\rangle) = 1+\max(\mathit{height}(\{T'/X\}S_1), \mathit{height}(\{T'/X\}S_2)) = 1+\max(\mathit{height}(S_1),\mathit{height}(S_2)) = 1+\mathit{height}(\langle S_1,S_2\rangle)$.
- Case: $T = \mu Y{::}L.\,S$, where without loss of generality $X \neq Y$ and $Y \notin FV(T')$. Then $X \notin UV(\mu Y{::}L.\,S) = UV(S)\backslash\{Y\}$, so by the inductive hypothesis we have $\mathit{height}(\{T'/X\}(\mu Y{::}L.\,S)) = 1+\mathit{height}(\{T'/X\}S) = 1+\mathit{height}(S) = \mathit{height}(\mu Y{::}L.\,S)$.∎

Proposition 8 (Basic Properties of Reduction)

1. *If $T \rightsquigarrow S_1$ and $T \rightsquigarrow S_2$ then $S_1 = S_2$.*
2. *If $T \rightsquigarrow T'$ then $E[T] \rightsquigarrow E[T']$.*
3. *If $T \rightsquigarrow T'$ then $(\{S/X\}T) \rightsquigarrow (\{S/X\}T')$.*
4. *If T is contractive and $T \rightsquigarrow T'$ then $\mathit{height}(T) > \mathit{height}(T')$.*
5. *If $\Gamma \vdash T :: K$ and $T \rightsquigarrow T'$ then $\Gamma \vdash T' :: K$.*
6. *If $T \rightsquigarrow T'$ then $FV(T) \supseteq FV(T')$.*

Proof:

1. By definition of $\leadsto$.

2. By definition of $\leadsto$.

3. There are two cases:

   - Case: $T = E[\pi_i\,\langle T_1, T_2\rangle]$ and $T' = E[T_i]$. Then we have that $\{S/X\}T = E[\pi_i\,\langle(\{S/X\}T_1), (\{S/X\}T_2)$ $E[\{S/X\}T_i] = \{S/X\}T'$.
   - Case: $T = E[\mu Y{::}L.\,T_1]$ and $T' = E[\{\mu Y{::}L.\,T_1/Y\}T_1]$. Without loss of generality, $X{\neq}Y$ and $Y{\notin}FV(S)$. Then $\{S/X\}T = E[\{S/X\}(\mu Y{::}L.\,T_1)] = E[\mu Y{::}L.\,(\{S/X\}T_1)] \leadsto E[\{\mu Y{::}L.\,(\{S/X\}T_1)/Y\}(\{S/X\}T_1)] = E[\{S/X\}(\{\mu Y{::}L.\,T_1/Y\}T_1)] = \{S/X\}T'$.

4. Again there are two cases.

   - Case: $T = E[\pi_i\,\langle T_1, T_2\rangle]$ and $T'{=}E[T_i]$. Then $\mathit{height}(T) = \mathit{height}(\langle T_1, T_2\rangle) = 1 + \max(\mathit{height}(T_1), \mathit{height}(T_2)) > \mathit{height}(T_i) = \mathit{height}(E[T_i]) = \mathit{height}(T')$.
   - Case: $T = E[\mu X{::}\,.\,S]$ and $T' = E[\{\mu X{::}\,.\,S/X\}S]$. T is contractive, so by Proposition 7 $\mathit{height}(T) = \mathit{height}(\mu X{::}\,.\,S) = 1 + \mathit{height}(S) > \mathit{height}(S) = \mathit{height}(\{\mu X{::}\,.\,S/X\}S) = \mathit{height}(E[\{\mu X{::}\,.\,S/X\}S]) = \mathit{height}(T')$. $\blacksquare$

5. By inversion of the typing rules and application of Rule 10 it suffices to consider the cases in which $E = \bullet$.

   - Case: $T = \pi_i\,\langle T_1, T_2\rangle$ and $T' = T_i$. By inversion of Rule 10 and Rule 9 we know that $K = K_i$ where $\Gamma \vdash T_i :: K_i$.
   - Case: $T = \mu X{::}L.\,S$ and $T' = \{\mu X{::}L.\,S/X\}S$. By inversion of Rule 8 we have $L = K$ and $\Gamma, X{::}L \vdash S :: L$. Thus by Proposition 5, we have $\Gamma \vdash \{\mu X{::}L.\,S/X\}S :: K$ as desired. $\blacksquare$

6. By definition of reduction.

3.5 Defining Type Equivalence

The following collection of inference rules defines equivalence of well-formed types. In contrast to the definition of the well-formedness judgment, these rules are to be interpreted *coinductively*, where the universe of potential equivalences is

$$U_{eq} := \{(\Gamma \vdash T \equiv S :: K) \bigm| \Gamma \vdash T :: K \text{ and } \Gamma \vdash S :: K\}.$$

U_{eq} is intended as an upper bound and contains judgments that are not provable.

$$\frac{\Gamma \vdash T :: K}{\Gamma \vdash T \equiv T :: K} \tag{11}$$

$$\frac{\Gamma \vdash T_1 \equiv S_1 :: * \qquad \Gamma \vdash T_2 \equiv S_2 :: *}{\Gamma \vdash T_1{\to}T_2 \equiv S_1{\to}S_2 :: *} \tag{12}$$

$$\begin{array}{rl}
F_\pi(\mathcal{J}) := & \{\, (\Gamma \vdash P \equiv P :: K) \mid \Gamma \vdash P :: K \} \\
\cup & \{\, (\Gamma \vdash T_1 {\to} T_2 \equiv S_1 {\to} S_2 :: *) \mid \\
& \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: *) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J} \} \\
\cup & \{\, (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow T' \text{ and } (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J} \text{ and } \Gamma \vdash T :: K \} \\
\cup & \{\, (\Gamma \vdash T \equiv S :: K) \mid S \rightsquigarrow S' \text{ and } (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J} \text{ and } \Gamma \vdash S :: K \} \\
\cup & \{\, (\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1 {\times} K_2) \mid \\
& \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: K_1) \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \mathcal{J} \}
\end{array}$$

Figure 1: Generating Function for Equivalence with Pairs and Projections

$$\frac{\Gamma \vdash E[\{\mu X{::}L.\, T/X\}T] \equiv S :: K \qquad \Gamma \vdash \mu X{::}L.\, T :: L}{\Gamma \vdash E[\mu X{::}L.\, T] \equiv S :: K} \tag{13}$$

$$\frac{\Gamma \vdash T \equiv E[\{\mu X{::}L.\, S/X\}S] :: K \qquad \Gamma \vdash \mu X{::}L.\, S :: L}{\Gamma \vdash T \equiv E[\mu X{::}L.\, S] :: K} \tag{14}$$

$$\frac{\Gamma \vdash E[T_i] \equiv S :: K \qquad \Gamma \vdash T_{3-i} :: K'}{\Gamma \vdash E[\pi_i\, \langle T_1, T_2 \rangle] \equiv S :: K} \tag{15}$$

$$\frac{\Gamma \vdash T \equiv E[S_i] :: K \qquad \Gamma \vdash S_{3-i} :: K'}{\Gamma \vdash T \equiv E[\pi_i\, \langle S_1, S_2 \rangle] :: K} \tag{16}$$

$$\frac{\Gamma \vdash T_1 \equiv S_1 :: K_1 \qquad \Gamma \vdash T_2 \equiv S_2 :: K_2}{\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1 {\times} K_2} \tag{17}$$

The corresponding generating function $F_\pi : 2^{U_{eq}} {\to} 2^{U_{eq}}$ appears in Figure 1. The well-formedness constraints ensure that F_π does map $2^{U_{eq}}$ to $2^{U_{eq}}$. F_π is monotone, and hence has a greatest fixed point νF_π. We take this set of judgments as the formal definition of equivalence, i.e., we say that $\Gamma \vdash T \equiv S :: K$ if and only if $(\Gamma \vdash T \equiv S :: K) \in \nu F_\pi$.

There are two differences between F_π and the inference rules above. One is an inessential notational convenience: weak head reduction has been used to merge Rules 13 and 15 into a single line in the definition of F_π, and similarly to merge Rules 14 and 16.

The other change is more substantive. Though Rule 11 states that any type is equal to itself, the definition F_π builds in reflexivity only for paths (which include the type `int` and projections from variables). This change makes F_π easier to work with, yet does not change the greatest fixed point: Rule 11 is admissible.

We try to avoid including admissible rules in our generating functions. Some additions are innocuous, but adding admissible rules to the generating function can actually change the greatest fixed point. For example, symmetric and transitive closure rules are admissible but were purposely omitted from definition of equivalence. If we were to augment the definition of $F_\pi(\mathcal{J})$ with the line

$$\cup \; \{ \; (\Gamma \vdash T \equiv S :: K) \mid (\Gamma \vdash S \equiv T :: K) \in \mathcal{J} \; \}$$

then the greatest fixed point would suddenly be U_{eq} itself, equating *all* types of the same kind. Explicitly requiring transitive closure would make equivalence similarly inconsistent.

Instead, the rules have been carefully designed to obtain an equivalence relation. For example, we have both Rules 13 and 14 to maintain symmetry, and the two rules each build in a nontrivial step of transitivity (compared to simply having the two axioms $\mu X{::}L.\,T \equiv \{\mu X{::}L.\,T/X\}T$ and $\{\mu X{::}L.\,T/X\}T \equiv \mu X{::}L.\,T$).

The presence of elimination contexts in a declarative definition of equivalence might also be surprising, but they seem necessary to obtain all the desired equivalences. Consider the equation $\pi_1\,(\mu X{::}{*}{\times}{*}.\,\langle \texttt{int}, \texttt{int}\rangle) \equiv \texttt{int}$. If the elimination contexts E were dropped from Rules 13–16 then this equation would not be coinductively provable because it would not match the conclusion of any inference rule, even if we added more congruence rules. (In an inductive presentation this equivalence could follow from an appeal to the transitive rule.)

3.6 Properties of Type Equivalence

By definition νF_π is closed under weak head expansion (as long as well-formedness is preserved). Less obviously, it is closed under reduction as well.

Proposition 9

1. *If $\Gamma \vdash T' \equiv S' :: K$, $T \rightsquigarrow^* T'$, $S \rightsquigarrow^* S'$, $\Gamma \vdash T :: K$, and $\Gamma \vdash S :: K$ then $\Gamma \vdash T \equiv S :: K$.*

2. *If $\Gamma \vdash T \equiv S :: K$, $T \rightsquigarrow^* T'$, and $S \rightsquigarrow^* S'$ then $\Gamma \vdash T' \equiv S' :: K$.*

Proof:

1. By Proposition 8, all reducts along the way from T to T' and from S to S' are well-formed. By repeatedly applying the fact that by definition νF_π is closed under single well-formed weak head expansions, we obtain the desired result.

2. By induction on *height*(T) + *height*(S), and cases on the justification for $(\Gamma \vdash T \equiv S :: K) \in \nu F_\pi = F_\pi(\nu F_\pi)$.

   - Case: $T \not\rightsquigarrow$ and $S \not\rightsquigarrow$. Then $T = T'$ and $S = S'$, so the desired result is true by assumption.
   - Case: $\Gamma \vdash T \equiv S :: K$ because $T \rightsquigarrow U$, $\Gamma \vdash T :: K$, and $\Gamma \vdash U \equiv S :: K$. If $U \rightsquigarrow^* T'$ then $\Gamma \vdash T' \equiv S' :: K$ follows inductively. Otherwise, since reduction is deterministic $T = T'$, and the inductive hypothesis yields $\Gamma \vdash U \equiv S' :: K$, so that $(\Gamma \vdash T' \equiv S' :: K) \in F_\pi(\nu F_\pi) = \nu F_\pi$.
   - Case: $\Gamma \vdash T \equiv S :: K$ because $S \rightsquigarrow U$, $\Gamma \vdash S :: K$, and $\Gamma \vdash T \equiv U :: K$. Analogous to the previous case. ∎

Then we can show that $\equiv$ as defined is both a partial equivalence relation and a congruence.

Proposition 10 (Reflexivity)
If $\Gamma \vdash T :: K$ then $\Gamma \vdash T \equiv T :: K$.

Proof: We want to show that $I \subseteq \nu F_\pi$, where $I := \{(\Gamma \vdash T \equiv T :: K) \mid \Gamma \vdash T :: K\}$. By Corollary 3 it suffices to show $I \subseteq F_\pi(I) \cup F_\pi(F_\pi(I))$. Let $(\Gamma \vdash T \equiv T :: K) \in I$ be given, and consider the possible cases, given that T is well-formed.

- Case: $T = P$. Then $(\Gamma \vdash P \equiv P :: K) \in F_\pi(\emptyset) \subseteq F_\pi(I)$.

- Case: $T = T_1 {\to} T_2$ and $K = *$. Then $\Gamma \vdash T_1 :: *$ and $\vdash T_2 :: *$, so $(\Gamma \vdash T_1 \equiv T_1 :: *) \in I$ and $(\Gamma \vdash T_2 \equiv T_2 :: *) \in I$. Thus $(\Gamma \vdash T_1 {\to} T_2 \equiv T_1 {\to} T_2 :: *) \in F_\pi(I)$.

- Case: $T = \langle T_1, T_2 \rangle$ and $K = K_1 {\times} K_2$. Then $\vdash T_1 :: K_1$ and $\vdash T_2 :: K_2$, so $(\Gamma \vdash T_1 \equiv T_1 :: K_1) \in I$ and $(\Gamma \vdash T_2 \equiv T_2 :: K_2) \in I$. Thus $(\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle T_1, T_2 \rangle :: K_1 {\times} K_2) \in F_\pi(I)$.

- Case: $T \leadsto T'$ for some T'. Then by Proposition 8, $\vdash T' :: K$ as well, and so $(\Gamma \vdash T' \equiv T' :: K) \in I$. Thus $(\Gamma \vdash T \equiv T' :: K) \in F_\pi(I)$, so $(\Gamma \vdash T \equiv T :: K) \in F_\pi(F_\pi(I))$.

The fold/unfold rule and the projection rule for pairs hold:

Corollary 11
If $\Gamma \vdash T :: K$ *and* $T \leadsto T'$ *then* $\Gamma \vdash T \equiv T' :: K$.

Proof: By Propositions 10 and 9. ∎

Proposition 12 (Transitivity)
If $\Gamma \vdash T_1 \equiv T_2 :: K$ *and* $\Gamma \vdash T_2 \equiv T_3 :: K$ *then* $\Gamma \vdash T_1 \equiv T_3 :: K$.

Proof: We must show that νF_π is transitively closed, i.e., $\mathit{TR}(\nu F_\pi) \subseteq \nu F_\pi$, where

$$\mathit{TR}(\mathcal{J}) := \{(\Gamma \vdash T_1 \equiv T_3 :: K) \mid \exists T_2. (\Gamma \vdash T_1 \equiv T_2 :: K), (\Gamma \vdash T_2 \equiv T_3 :: K) \in \mathcal{J}\}$$

is the transitive-closure operator. It suffices to show that $\mathit{TR}(\nu F_\pi) \subseteq F_\pi(\mathit{TR}(\nu F_\pi))$, because then $\mathit{TR}(\nu F_\pi) \subseteq \nu F_\pi$ follows by coinduction. Assume $(\Gamma \vdash T_1 \equiv T_3 :: K) \in \mathit{TR}(\nu F_\pi)$ because $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\pi$ and $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\pi$. We must show $(\Gamma \vdash T_1 \equiv T_3 :: K) \in F_\pi(\mathit{TR}(\nu F_\pi))$, and this follows by induction on $\mathit{height}(T_2)$ and cases on the justifications for the two assumed equivalences.

- Case: $T_1 = T_2 = T_3 = P$. Trivial.

- Case: $T_1 = T_1' {\to} T_1''$, $T_2 = T_2' {\to} T_2''$, $T_3 = T_3' {\to} T_3''$, and $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\pi$ and $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\pi$ because $(\Gamma \vdash T_1' \equiv T_2' :: *)$, $(\Gamma \vdash T_1'' \equiv T_2'' :: *)$, $(\Gamma \vdash T_2' \equiv T_3' :: *)$, $(\Gamma \vdash T_2'' \equiv T_3'' :: *) \in \nu F_\pi$. Then $(\Gamma \vdash T_1' \equiv T_3' :: *) \in \mathit{TR}(\nu F_\pi)$ and $(\Gamma \vdash T_1'' \equiv T_3'' :: *) \in \mathit{TR}(\nu F_\pi)$, so $(\Gamma \vdash T_1' {\to} T_1'' \equiv T_3' {\to} T_3'' :: *) \in F_\pi(\mathit{TR}(\nu F_\pi))$.

- Case: $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\pi$ because $(\Gamma \vdash T_1' \equiv T_2 :: K) \in \nu F_\pi$, $T_1 \leadsto T_1'$, and $\Gamma \vdash T_1 :: K$. Then $(\Gamma \vdash T_1' \equiv T_3 :: K) \in \mathit{TR}(\nu F_\pi)$, so $(\Gamma \vdash T_1 \equiv T_3 :: K) \in F_\pi(\mathit{TR}(\nu F_\pi))$.

- Case: $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\pi$ because $(\Gamma \vdash T_2 \equiv T_3' :: K) \in \nu F_\pi$, $T_3 \leadsto T_3'$, and $\Gamma \vdash T_3 :: K$. Then $(\Gamma \vdash T_1 \equiv T_3' :: K) \in \mathit{TR}(\nu F_\pi)$, so $(\Gamma \vdash T_1 \equiv T_3' :: K) \in F_\pi(\mathit{TR}(\nu F_\pi))$.

- Case: $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\pi$ because $(\Gamma \vdash T_1 \equiv T_2' :: K) \in \nu F_\pi$, $T_2 \leadsto T_2'$, and $\Gamma \vdash T_2 :: K$. By Proposition 9, $(\Gamma \vdash T_2' \equiv T_3 :: K) \in \nu F_\pi$, By Proposition 8 we know that $\mathit{height}(T_2) > \mathit{height}(T_2')$, so the induction hypothesis applies and $(\Gamma \vdash T_1 \equiv T_3 :: K) \in F_\pi(\mathit{TR}(\nu F_\pi))$.

- Case: $T_1 = \langle T_1', T_1'' \rangle$, $T_2 = \langle T_2', T_2'' \rangle$, $T_3 = \langle T_3', T_3'' \rangle$, $K = K' {\times} K''$, and $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\pi$ and $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\pi$ because $(\Gamma \vdash T_1' \equiv T_2' :: K')$, $(\Gamma \vdash T_1'' \equiv T_2'' :: K'')$, $(\Gamma \vdash T_2' \equiv T_3' :: K')$, $(\Gamma \vdash T_2'' \equiv T_3'' :: K'') \in \nu F_\pi$. Then $(\Gamma \vdash T_1' \equiv T_3' :: K') \in \mathit{TR}(\nu F_\pi)$ and $(\Gamma \vdash T_1'' \equiv T_3'' :: K'') \in \mathit{TR}(\nu F_\pi)$, so $(\Gamma \vdash \langle T_1', T_1'' \rangle \equiv \langle T_3', T_3'' \rangle :: K' {\times} K'') \in F_\pi(\mathit{TR}(\nu F_\pi))$. ∎

$$\begin{array}{rll}
F^a_\pi(\mathcal{J}) := & \{ (\Gamma \vdash P \equiv P :: K) \mid \Gamma \vdash P :: K \} & \\
\cup & \{ (\Gamma \vdash T_1 {\to} T_2 \equiv S_1 {\to} S_2 :: *) \mid & \\
 & \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: *) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J} \} & \\
\cup & \{ (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow T',\ (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J}, \text{ and } \Gamma \vdash T :: K \} & \\
\cup & \{ (\Gamma \vdash T \equiv S :: K) \mid T \not\rightsquigarrow\ ,\ S \rightsquigarrow S',\ (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J} \text{ and } \Gamma \vdash S :: K \} & \\
\cup & \{ (\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1 {\times} K_2) \mid & \\
 & \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: K_1) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \mathcal{J} \} &
\end{array}$$

Figure 2: Algorithmic Generating Function for Equivalence with Pairs and Projections

Proposition 13 (Symmetry)
If $\Gamma \vdash T \equiv S :: K$ *then* $\Gamma \vdash S \equiv T :: K$.

Proof: Similar to the previous proposition; by coinduction we can show that $SY(\nu F_\pi) \subseteq F_\pi(SY(\nu F_\pi))$ where $SY(\cdot)$ is the symmetric closure operator. ∎

Proposition 14 (Congruence)

1. *If* $\Gamma \vdash T \equiv S :: K_1 {\times} K_2$ *then* $\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i$.

2. *If* $\Gamma, X{::}L \vdash S_1 \equiv S_2 :: L$ *then* $\Gamma \vdash \mu X{::}L.\, S_1 \equiv \mu X{::}L.\, S_2 :: L$.

Proof: Simplified versions of the proofs for Propositions 34 and 38. ∎ ∎

3.7 Decidability of Equivalence

Decidability of equivalence in the presence of pairs is not a trivial corollary of decidability for recursive types in isolation. We cannot first reduce all projections and then proceed as before; as shown in Section 3.2, unfoldings can introduce new opportunities to project, while projections can introduce new opportunities to unfold. The termination of weak head reduction does not automatically guarantee that finite-state properties still hold. A priori, each unfolding could yield more projections from more pairs, and we might never see the same types twice.

To show that equivalence is decidable, we want an invertible, finite-state function $F^a_\pi : U_{eq} {\to} U_{eq}$ such that $\nu F^a_\pi = \nu F_\pi$. Such a function appears in Figure 2; it differs from F_π only in one line, where we require T to be weak head normal before allowing $S \rightsquigarrow S'$ as a justification.

Proposition 15
F^a_π *is invertible.*

Proof: By inspection of Figure 2. ∎

F_π and F^a_π have the same greatest fixed point. Intuitively, if we wish to know whether two types are equal it does not matter which one we weak head reduce first; completely reducing the left-hand type before starting on the right-hand type, as suggested by proof search using F^a_π, is thus a sound and deterministic strategy.

Proposition 16
$\nu F_\pi = \nu F^a_\pi$.

Proof: Since pointwise $F^a_\pi \subseteq F_\pi$, we know $\nu F^a_\pi \subseteq \nu F_\pi$. To show $\nu F_\pi \subseteq \nu F^a_\pi$ it suffices to show $\nu F_\pi \subseteq F^a_\pi(\nu F_\pi)$. Since the definitions of F_π and F^a_π differ only in one line, there is only one interesting case:

- Case: $(\Gamma \vdash T \equiv S :: K) \in \nu F_\pi$ because $(\Gamma \vdash T \equiv S' :: K) \in \nu F_\pi$ and $S \rightsquigarrow S'$ and $\Gamma \vdash S :: K$. There are two subcases:
  - Subcase: $T \not\rightsquigarrow$. Then $(\Gamma \vdash T \equiv S :: K) \in F^a_\pi(\nu F_\pi)$.
  - Subcase: $T \rightsquigarrow T'$. By Proposition 9, $(\Gamma \vdash T' \equiv S :: K) \in \nu F_\pi$, so since $\Gamma \vdash T :: K$ we have $(\Gamma \vdash T \equiv S :: K) \in F^a_\pi(\nu F_\pi)$.∎

F^a_π is finite-state. Following Brandt and Henglein [BH97] and Gapeyev et al. [GLP02], we define two sets of "subterms" of types. We say that T is a *top-down subterm* of S if $T \sqsubseteq S$ is provable from the rules in Figure 3. The top-down subterms are recognizable as the types that we might see in some comparison while running a proof-search algorithm.

Proposition 17 (Top-Down Transitivity)
If $T_1 \sqsubseteq T_2$ and $T_2 \sqsubseteq T_3$ then $T_1 \sqsubseteq T_3$.

Proof: By induction on the proof of $T_2 \sqsubseteq T_3$. ∎

Proposition 18

1. *If $(\Gamma' \vdash T' \equiv S' :: K') \in pred[F^a_\pi](\Gamma \vdash T \equiv S :: K)$ then $T' \sqsubseteq T$ and $S' \sqsubseteq S$ and $\Gamma = \Gamma'$.*
2. *If $(\Gamma'' \vdash T'' \equiv S'' :: K'') \in reachable[F^a_\pi](\{\Gamma \vdash T \equiv S :: K\})$ then $T'' \sqsubseteq T$ and $S'' \sqsubseteq S$ and $\Gamma = \Gamma''$.*

Proof:

1. By definition of F^a_π.
2. By Part 1 and Proposition 17. ∎

Next, we define the *bottom-up subterms*, also shown in Figure 3. The rules for T being a bottom-up subterm of S, written $T \preceq S$, are nearly the same except for the difference between Rule 21 and Rule 27. (Despite the notation, this relation has nothing to do with subtyping.) The key advantage of the bottom-up formulation is that the set of bottom-up subterms of every type is finite, a fact easily shown by induction. Because of Rule 21, *a priori* this might not be the case for the top-down subterms.

Lemma 19

1. *If $T \preceq S$ then $FV(T) \subseteq FV(S)$.*
2. *The set $\{ S \mid S \preceq T \}$ is finite for every type T.*

Proof:

1. By induction on the proof of $T \preceq S$.
2. By induction on T.∎

$$\frac{}{T \sqsubseteq T} \tag{18}$$

$$\frac{T \sqsubseteq S_i}{T \sqsubseteq S_1 {\to} S_2} \tag{19}$$

$$\frac{T \sqsubseteq S_i}{T \sqsubseteq \langle S_1, S_2 \rangle} \tag{20}$$

$$\frac{T \sqsubseteq E[\{\mu X{::}L.\, S/X\}S]}{T \sqsubseteq E[\mu X{::}L.\, S]} \tag{21}$$

$$\frac{T \sqsubseteq E[S_i]}{T \sqsubseteq E[\pi_i\, \langle S_1, S_2 \rangle]} \tag{22}$$

$$\frac{T \sqsubseteq S}{T \sqsubseteq \pi_i\, S} \tag{23}$$

$$\frac{}{T \preceq T} \tag{24}$$

$$\frac{T \preceq S_i}{T \preceq S_1 {\to} S_2} \tag{25}$$

$$\frac{T \preceq S_i}{T \preceq \langle S_1, S_2 \rangle} \tag{26}$$

$$\frac{T \preceq E[S]}{\{\mu X{::}L.\, S/X\}T \preceq E[\mu X{::}L.\, S]} \tag{27}$$

$$\frac{T \preceq E[S_i]}{T \preceq E[\pi_i\, \langle S_1, S_2 \rangle]} \tag{28}$$

$$\frac{T \preceq S}{T \preceq \pi_i\, S} \tag{29}$$

Figure 3: Top-Down and Bottom-Up Subterms

We would like to relate the top-down and bottom-up subterms. In the absence of pairing and projection, every top-down subterm is a bottom-up subterm, and hence the top-down subterms are finite in number [GLP02]. Here this is no longer true. For example, put

$$U := \mu X{::}{\ast}{\times}{\ast}.\, \langle \pi_1\, X {\to} \pi_2\, X, \pi_2\, X {\to} \pi_1\, X \rangle.$$

The type $\pi_1\, U$ has $\pi_2\, \langle \pi_1\, U {\to} \pi_2\, U, \pi_2\, U {\to} \pi_1\, U \rangle$ as a top-down subterm, but not as a bottom-up subterm.

However, it turns out that every top-down subterm is a weak head reduct of some bottom-up subterm. For example, the above top-down subterm is a reduct of the bottom-up subterm $\pi_2\, U$.

We show the this relationship using two lemmas characterizing the bottom-up subterm relation.

Lemma 20
If $T \preceq \pi_i\, S$ then $\pi_i\, S \rightsquigarrow^* T$ or $T \preceq S$.

Proof: By induction on the proof of the assumption, and cases on the last rule used.

- Case: $T \preceq \pi_i\, S$ because $T = \pi_i\, S$. Then $\pi_i\, S \rightsquigarrow^* \pi_i\, S = T$.
- Case: $T \preceq \pi_i\, S$ because $T \preceq S$. Trivial.
- Case: $T \preceq \pi_i\, S$ because $S = \langle S_1, S_2 \rangle$ and $T \preceq S_i$. Then by Rule 26, $T \preceq \langle S_1, S_2 \rangle = S$.
- Case: $T \preceq \pi_i\, S$ because $S = E[\pi_j\, \langle S_1, S_2 \rangle]$ and $T \preceq \pi_i\, (E[S_j])$. By the inductive hypothesis there are two subcases:
  - Subcase: $\pi_i\, (E[S_j]) \rightsquigarrow^* T$. Then $\pi_i\, S = \pi_i\, (E[\pi_j\, \langle S_1, S_2 \rangle]) \rightsquigarrow \pi_i\, (E[S_j]) \rightsquigarrow^* T$.

- Subcase: $T \preceq E[S_j]$. Then $T \preceq E[\pi_j \langle S_1, S_2\rangle] = S$ by Rule 28.

- Case: $T \preceq \pi_i\, S$ because $T = \{\mu X{::}L.\, S'/X\}T'$ and $S = E[\mu X{::}L.\, S']$ and $T' \preceq \pi_i\,(E[S'])$. By the inductive hypothesis there are two subcases:
  - Subcase: $\pi_i\,(E[S']) \rightsquigarrow^* T'$. Then using Proposition 8 we have $\pi_i\, S = \pi_i\,(E[\mu X{::}L.\, S']) \rightsquigarrow \pi_i\,(E[\{\mu X{::}L.\, S'/X\}S']) = \{\mu X{::}L.\, S'/X\}(\pi_i\,(E[S'])) \rightsquigarrow^* \{\mu X{::}L.\, S'/X\}T' = T$.
  - Subcase: $T' \preceq E[S']$. Then by Rule 27 we have $T = \{\mu X{::}L.\, S'/X\}T' \preceq E[\mu X{::}L.\, S'] = S$.∎

Lemma 21
If $S \preceq (\{U/X\}T)$ then either $S \preceq U$ or there exists $T' \preceq T$ with $(\{U/X\}T') \rightsquigarrow^* S$.

Proof: By induction on T. If $S = (\{U/X\}T)$ then we can choose $T' = T$, so assume otherwise.

- Case: $T = X$. Then $S \preceq (\{U/X\}X) = U$ by assumption.
- Case: $T = \texttt{int}$ or $T = Y \neq X$. Cannot happen, since then $S = T$ and we assumed $S \neq \{U/X\}T = T$.
- Case: $T = \langle T_1, T_2\rangle$. Then $S \preceq (\{U/X\}T) = \langle(\{U/X\}T_1), (\{U/X\}T_2)\rangle$. By inspection of the definition of $\preceq$, there are only two possibilities:
  - Subcase: $S \preceq (\{U/X\}T_1)$. By the inductive hypothesis either $S \preceq U$, in which case we are done, or else there exists T_1' with $T_1' \preceq T_1$ and $(\{U/X\}T_1') \rightsquigarrow^* S$. By Rule 26 we have $T_1' \preceq \langle T_1, T_2\rangle = T$.
  - Subcase: $S \preceq (\{U/X\}T_2)$. Similar.
- Case: $T = T_1 \to T_2$. Analogous to the previous case.
- Case: $T = \pi_i\, T_1$. Then $S \preceq \{U/X\}T = \pi_i\,(\{U/X\}T_1)$. By Lemma 20 there are two possibilities:
  - Subcase: $\pi_i\,(\{U/X\}T_1) \rightsquigarrow^* S$. We can put $T' = T = \pi_i\, T_1$.
  - Subcase: $S \preceq \{U/X\}T_1$. By the inductive hypothesis, either $S \preceq U$, in which case we are done, or else there exists T_1' with $T_1' \preceq T_1$ and $(\{U/X\}T_1') \rightsquigarrow^* S$. By Rule 29 we have $T_1' \preceq \pi_i\, T_1 = T$.
- Case: $T = \mu Y{::}L.\, T_1$. Without loss of generality we may assume that $X \neq Y$ and $Y \notin FV(U)$, so $S \preceq \{U/X\}(\mu Y{::}L.\, T_1) = \mu Y{::}L.\,(\{U/X\}T_1)$. By inversion of Rule 27, there exists a type S_1 such that $S_1 \preceq (\{U/X\}T_1)$ and $S = \{(\mu Y{::}L.\, \{U/X\}T_1)/Y\}S_1$. By the inductive hypothesis (since T_1 is a subterm of T), there are two possibilities:
  - Subcase: $S_1 \preceq U$. Then by Lemma 19 we have $FV(S_1) \subseteq FV(U)$, and hence $Y \notin FV(S_1)$ which implies $S = S_1 \preceq U$.
  - Subcase: There exists a type T_2 such that $T_2 \preceq T_1$ and $(\{U/X\}T_2) \rightsquigarrow^* S_1$. By Proposition 8 we have that $\{(\mu Y{::}L.\, \{U/X\}T_1)/Y\}(\{U/X\}T_2) \rightsquigarrow^* \{(\mu Y{::}L.\, \{U/X\}T_1)/Y\}S_1$. That is, $\{U/X\}(\{\mu Y{::}L.\, T_1/Y\}T_2) \rightsquigarrow^* S$. We can take $T' = \{\mu Y{::}L.\, T_1/Y\}T_2$, since by Rule 27 and $T_2 \preceq T_1$ we have $T' = \{\mu Y{::}L.\, T_1/Y\}T_2 \preceq \mu Y{::}L.\, T_1 = T$.∎

Proposition 22
Every top-down subterm of a type is a weak head reduct of a bottom-up subterm of that type: if $S \sqsubseteq T$ *then there exists* S' *such that* $S' \preceq T$ *and* $S' \rightsquigarrow^* S$.

Proof: By induction on the proof that $S \sqsubseteq T$, and cases on the last rule used. Because the definitions of $\sqsubseteq$ and $\preceq$ differ only in one rule (Rule 27 vs. Rule 21), there is only one interesting case; the rest follow directly from the inductive hypothesis.

- Case: $T = E[\mu X{::}L.\, T_1]$ and $S \sqsubseteq T$ because $S \sqsubseteq E[\{\mu X{::}L.\, T_1/X\}T_1]$. By the inductive hypothesis, there exists S'_1 such that $S'_1 \preceq E[\{\mu X{::}L.\, T_1/X\}T_1]$ and $S'_1 \rightsquigarrow^* S$. Then E has no free variables, so $S'_1 \preceq \{\mu X{::}L.\, T_1/X\}(E[T_1])$ and hence by Lemma 21 there are two possibilities:
  - Subcase: $S'_1 \preceq \mu X{::}L.\, T_1$. Then we can take $S' = S'_1$ because using Rule 29 we have $S'_1 \preceq E[\mu X{::}L.\, T_1] = T$.
  - Subcase: There exists T'_1 such that $T'_1 \preceq E[T_1]$ and $(\{\mu X{::}L.\, T_1/X\}T'_1) \rightsquigarrow^* S'_1$. Put $S' = (\{\mu X{::}L.\, T_1/X\}T'_1)$, so that $S' \rightsquigarrow^* S'_1 \rightsquigarrow^* S$. By Rule 27, $S' = (\{\mu X{::}L.\, T_1/X\}T'_1) \preceq E[\mu X{::}L.\, T_1] = T$.∎

Corollary 23

1. *The set* $\{\, S' \mid \exists S \preceq T.\ S \rightsquigarrow^* S' \,\}$ *is finite for every type* T.

2. *The set* $\{\, S \mid S \sqsubseteq T \,\}$ *is finite for every type* T.

Proof:

1. By Proposition 8, weak head reduction is deterministic and strongly normalizing, so there can be only a finite number of reducts of the elements of the finite set $\{S \mid S \preceq T\}$.

2. By Part 1 and Proposition 22.∎

Corollary 24
F^a_π *is finite-state*

Proof: By Proposition 18 and Corollary 23, given any judgment $\Gamma \vdash T \equiv S :: K$, the judgments reachable by working backwards through F^a_π involve a finite set of pairs of types. Further, the reachable judgments all have the same context Γ, and the classifying kind is determined uniquely by Γ and the types being compared. Hence the set of reachable judgments is finite. ∎

Corollary 25
Membership in νF_π *(that is, type equivalence) is decidable.*

Proof: By Corollary 24 and Proposition 4. ∎

Though we could directly apply $\mathit{gfp}^{ac}[F^a_\pi](\emptyset, \cdot)$ to decide equivalence, the algorithm can be further simplified. If the original two types being compared are well-formed, then all other comparisons done by the algorithm will automatically involve well-formed types and so there is no need to explicitly check well-formedness. Further, since the typing context never changes and the classifying kinds in each judgment are uniquely determined by the types being compared, the accumulator set $\mathcal{A}$ needs to contain only pairs of types as in Section 2.3, rather than the general judgment 4-tuple.

4 Mutual Recursion

Various ways of taking mutually-recursive types as primitive have been introduced [HS97b, HS97a, CS02], but usually these arise in isorecursive systems. The reason may be that with coinductive equivalence and the fold-unfold rule, Bekić's Theorem shows that mutually-recursive types are definable using simple μ-types [Win93].

For example, if we want types X_1 and X_2 satisfying

$$\begin{aligned} X_1 &\equiv T_1(X_1, X_2) \\ X_2 &\equiv T_2(X_1, X_2) \end{aligned}$$

then we can take

$$\begin{aligned} X_1 &:= \mu Y_1{::}{*}.\, T_1(Y_1, \mu Y_2{::}{*}.\, T_2(Y_1, Y_2)) \\ X_2 &:= \mu Y_2{::}{*}.\, T_2(\mu Y_1{::}{*}.\, T_1(Y_1, Y_2), Y_2). \end{aligned}$$

A more direct definition (involving T_1 and T_2 only once each) would be

$$\begin{aligned} X' &:= \mu Y{::}{*}{\times}{*}.\, \langle T_1(\pi_1\, Y, \pi_2\, Y), T_2(\pi_1\, Y, \pi_2\, Y)\rangle \\ X_1 &:= \pi_1\, X' \\ X_2 &:= \pi_2\, X', \end{aligned}$$

especially if simple syntactic sugar were used, allowing the first definition to be written $X' := \mu\langle Y_1{::}{*}, Y_2{::}{*}\rangle.\langle T_1(Y_1, Y_2), T_2(Y_1, Y_2)\rangle$.

Either definition is acceptable in νF_π, as they are coinductively equivalent:

Proposition 26
Let $T_1(X_1, X_2)$ and $T_2(X_1, X_2)$ be two types such that $\Gamma, X_1{::}{*}, X_2{::}{*} \vdash T_1(X_1, X_2) :: *$ and $\Gamma, X_1{::}{*}, X_2{::}{*} \vdash T_2(X_1, X_2) :: *$. Then

$$\begin{aligned} &\Gamma \vdash \pi_1\, (\mu\langle Y_1{::}{*}, Y_2{::}{*}\rangle.\ \langle T_1(Y_1, Y_2), T_2(Y_1, Y_2)\rangle) \equiv \\ &\qquad\qquad \mu Y_1{::}{*}.\, T_1(Y_1, \mu Y_2{::}{*}.\, T_2(Y_1, Y_2)) :: * \\ &\Gamma \vdash \pi_2\, (\mu\langle Y_1{::}{*}, Y_2{::}{*}\rangle.\ \langle T_1(Y_1, Y_2), T_2(Y_1, Y_2)\rangle) \equiv \\ &\qquad\qquad \mu Y_2{::}{*}.\, T_2(\mu Y_1{::}{*}.\, T_2(Y_1, Y_2), Y_2) :: *. \end{aligned}$$

Proof: Execution of the equivalence algorithm. ■

Proposition 26 still holds in νF_{uu} (i.e., using the unfold/unfold rule), but neither definition yields types satisfying $X_1 \equiv T_1(X_1, X_2)$ and $X_2 \equiv T_2(X_1, X_2)$ in general. (Recall that the unfold/unfold rule will never equate a recursive type and a non-recursive type.) What we do get are types X_1 and X_2 isomorphic to $T_1(X_1, X_2)$ and $T_2(X_1, X_2)$, respectively.

5 Adding Type Abstractions

Next, we consider the addition of type abstractions and β-equivalence. In general, type systems with bound variables (other than those bound in recursive types, so that the limit of unfolding still contains bound variables) can be tricky when combined with recursive types [CG99, GP04]. A key problem is that the definition of equivalence is no longer obviously finite-state, because premises of equivalence rules need not have the same typing context as the conclusion; consider the standard rule for equivalence of λ-abstractions. We might get into an infinite loop in which the same pairs of types appear (or the same up to renamings of variables) but all the judgments differ because they have different variables in the typing context. Colazzo and Ghelli [CG99] showed that attempting to short-circuit such loops by naively merging multiple bindings of what was originally a single bound variable could lead to incorrect results.

We sidestep this problem by restricting the language to first order, forbidding kind arrows in negative positions. All function arguments are then proper types (or tuples of proper types) and hence after some finite number of "outer" lambdas no more bound variables will enter the context during the algorithm's search process; all further type abstractions will be entirely β-reduced away.

5.1 Extending the Syntax

The syntax of the system with type operators is as follows:

$$\begin{array}{rll} L & ::= & * \mid L{\times}L \\ K & ::= & L \mid K{\times}K \mid L{\Rightarrow}K \\ S,T,U & ::= & \texttt{int} \\ & \mid & X \mid Y \mid Z \mid \cdots \\ & \mid & T_1{\to}T_2 \mid \mu X{::}L.\,T \\ & \mid & \langle T,T\rangle \mid \pi_1\,T \mid \pi_2\,T \\ & \mid & \lambda X{::}L.T \\ & \mid & T\,T \\ E & ::= & \bullet \mid \pi_1\,E \mid \pi_2\,E \mid E\,T \end{array}$$

We now use L to denote the kinds of (tuples of) proper types, and K to denote an arbitrary kind. Thus, μ-types cannot be type operators. However, type operators can accept or return recursive proper types, or even recursively-defined pairs. This is enough to handle most examples of ML-like datatypes. (In ML, `list` is a type operator that when given a proper type such as `int` returns a recursive proper type classifying lists of integers).

The definition of contractiveness remains unchanged, once we extend the definition of unguarded variables:

$$\begin{array}{rcl} UV(T_1\,T_2) & := & UV(T_1) \cup UV(T_2) \\ UV(\lambda X{::}L.T) & := & UV(T) \setminus \{X\}. \end{array}$$

We still require that all types be syntactically contractive.

5.2 Extending Well-Formedness

The existing well-formedness rules can remain. The additional two rules are completely standard, given the first-order restriction:

$$\frac{\Gamma, X{::}L \vdash T :: K}{\Gamma \vdash\ \lambda X{::}L.T\ ::\ L{\Rightarrow}K} \tag{30}$$

$$\frac{\Gamma \vdash T_1 :: L{\Rightarrow}K \qquad \Gamma \vdash T_2 :: L}{\Gamma \vdash T_1\,T_2 :: K} \tag{31}$$

The expected properties continue to hold:

Proposition 27 (Basic Properties of Well-Formedness)

1. *If* $\Gamma \vdash T :: K_1$ *and* $\Gamma \vdash T :: K_2$ *then* $K_1 = K_2$.

2. *If* $\Gamma \vdash T :: K$ *then* $FV(T) \subseteq dom(\Gamma)$.

3. *If* $\Gamma_1, \Gamma_3 \vdash T :: K$ *and* $dom(\Gamma_1, \Gamma_3) \cap dom(\Gamma_2) = \emptyset$ *then* $\Gamma_1, \Gamma_2, \Gamma_3 \vdash T :: K$.

4. *If* $\Gamma_1, Y{::}K', \Gamma_2 \vdash T :: K$ *and* $\Gamma \vdash T' :: K'$ *then* $\Gamma_1, \Gamma_2 \vdash \{T'/Y\}T :: K$.

5.3 Extending Reduction

The weak head reduction relation is extended to reduce function applications:

$$\begin{array}{lcl} E[(\lambda X{::}L.T)\, S] & \rightsquigarrow & E[\{S/X\}T] \\ E[\pi_i\, \langle T_1, T_2\rangle] & \rightsquigarrow & E[T_i] \\ E[\mu X{::}L.\, T] & \rightsquigarrow & E[\{\mu X{::}L.\, T/X\}T] \end{array}$$

Proposition 28 (Characterization of Kinds)

1. *If* $\Gamma \vdash T :: *$ *then* $T \rightsquigarrow T'$ *or* T *is a path or* $T = T_1 \to T_2$.

2. *If* $\Gamma \vdash T :: K_1 \times K_2$ *then* $T \rightsquigarrow T'$ *or* T *is a path or* $T = \langle T_1, T_2 \rangle$.

3. *If* $\Gamma \vdash T :: K_1 \Rightarrow K_2$ *then* $T \rightsquigarrow T'$ *or* T *is a path or* $T = \lambda X{::}L_1.T_2$.

Instead of defining a height metric and using it to show that weak head reduction must terminate, it is more convenient at this point to go the other direction. We show that weak head reduction terminates, and then define height as the number of steps required.

The translation function $|\cdot|$ maps each type into a type without μ.

$$\begin{array}{lcl} |\texttt{int}| & := & \texttt{int} \\ |T_1 \to T_2| & := & \texttt{int} \\ |X| & := & X \\ |\pi_i\, T| & := & \pi_i\, |T| \\ |\langle T_1, T_2 \rangle| & := & \langle |T_1|, |T_2| \rangle \\ |\mu X{::}L.\, T| & := & (\lambda X{::}*.|T|)\, \texttt{int} \\ |T_1\, T_2| & := & |T_1|\, |T_2| \\ |\lambda X{::}L.T| & := & \lambda X{::}L.|T| \end{array}$$

Then $|\cdot|$ maps weak head reduction sequences in the system with recursive types into reduction sequences in the simply-typed (or in this case, simply-kinded) lambda calculus with pairs. The two key steps are the translation of a recursive type into a β-redex (ensuring that every step of weak head normalization, including unfolding, becomes one application or projection), and the replacement of arrow types by `int` (a type with no free variables). These, in combination with the requirement that types are contractive, ensure that $|\mu X{::}L.\, T| \rightsquigarrow_\beta |\{\mu X{::}L.\, T/X\}T|$.

Proposition 29

1. *If* $\Gamma \vdash T :: K$ *then* $\Gamma \vdash |T| :: K$ *and* $|T|$ *has no* μ*'s.*

2. $UV(T) = FV(|T|)$.

3. *If $T \leadsto S$ then $|T| \leadsto_{\beta\pi} |S|$, where $\leadsto_{\beta\pi}$ is weak head reduction without unfolding (i.e., reducing projections and applications only).*

4. *Thus, $\leadsto$ is normalizing for well-formed types.*

Proof:

1. By induction on the proof of the assumption.

2. By induction on T.

3. By cases on $T \leadsto S$.

4. The types without μ are members of a simply-typed lambda calculus with pairs, which is well-known to be strongly normalizing under $\leadsto_{\beta\pi}$. Thus by Part 3, weak head reduction for the original types must terminate. ∎

We then define *height*(T) to be the (finite) number of reduction steps required to reduce T to a weak head-normal form.

Finally, the same properties listed in Proposition 8 continue to hold:

Proposition 30 (Basic Properties of Reduction)

1. *If $T \leadsto S_1$ and $T \leadsto S_2$ then $S_1 = S_2$.*

2. *If $T \leadsto T'$ then $E[T] \leadsto E[T']$.*

3. *If $T \leadsto T'$ then $(\{S/X\}T) \leadsto (\{S/X\}T')$.*

4. *If T is contractive and $T \leadsto T'$ then height$(T) >$ height(T').*

5. *If $\Gamma \vdash T :: K$ and $T \leadsto T'$ then $\Gamma \vdash T' :: K$.*

6. *If $T \leadsto T'$ then $FV(T) \supseteq FV(T')$.*

5.4 Extending Type Equivalence

Figure 4 shows a generating function F_λ whose fixed point νF_λ is an appropriate definition of equivalence in the presence of type operators. Asserting $\Gamma \vdash T \equiv S :: K$ now means that $(\Gamma \vdash T \equiv S :: K) \in \nu F_\lambda$. The definition of F_λ corresponds to the addition of three new rules, still to be interpreted coinductively:

$$\frac{\Gamma \vdash E[\{T_2/X\}T_1] \equiv S :: K \qquad \Gamma \vdash E[(\lambda X{::}L.T_1)\, T_2] :: K}{\Gamma \vdash E[(\lambda X{::}L.T_1)\, T_2] \equiv S :: K} \tag{32}$$

$$\frac{\Gamma \vdash T \equiv E[\{S_2/X\}S_1] :: K \qquad \Gamma \vdash E[(\lambda X{::}L.S_1)\, S_2] :: K}{\Gamma \vdash T \equiv E[(\lambda X{::}L.S_1)\, S_2] :: K} \tag{33}$$

$$\frac{\Gamma, X{::}L \vdash T \equiv S :: K}{\Gamma \vdash \lambda X{::}L.T \equiv \lambda X{::}L.S :: L{\Rightarrow}K} \tag{34}$$

$$
\begin{array}{rl}
F_\lambda(\mathcal{J}) := & \{\, (\Gamma \vdash \mathtt{int} \equiv \mathtt{int} :: *) \mid \text{for all } \Gamma \} \\
\cup & \{\, (\Gamma \vdash X \equiv X :: K) \mid \Gamma \vdash X :: K \} \\
\cup & \{\, (\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \mid (\Gamma \vdash P_1 \equiv P_2 :: K_1{\times}K_2) \in \mathcal{J} \} \\
\cup & \{\, (\Gamma \vdash P_1\, T_1 \equiv P_2\, T_2 :: K) \mid \\
 & \qquad\qquad (\Gamma \vdash P_1 \equiv P_2 :: L{\Rightarrow}K) \in \mathcal{J} \text{ and } (\Gamma \vdash T_1 \equiv T_2 :: L) \in \mathcal{J} \} \\
\cup & \{\, (\Gamma \vdash T_1{\to}T_2 \equiv S_1{\to}S_2 :: *) \mid (\Gamma \vdash T_1 \equiv S_1 :: *), (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J} \} \\
\cup & \{\, (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow T' \text{ and } (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J} \text{ and } \Gamma \vdash T :: K \} \\
\cup & \{\, (\Gamma \vdash T \equiv S :: K) \mid S \rightsquigarrow S' \text{ and } (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J} \text{ and } \Gamma \vdash S :: K \} \\
\cup & \{\, (\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1{\times}K_2) \mid \\
 & \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: K_1) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \mathcal{J} \} \\
\cup & \{\, (\Gamma \vdash \lambda X{::}L.T \equiv \lambda X{::}L.S :: L{\Rightarrow}K) \mid (\Gamma, X{::}L \vdash T \equiv S :: K) \in \mathcal{J} \}
\end{array}
$$

Figure 4: Equivalence with Type Abstractions

as well as replacing the reflexive case of path equivalence with the following four more general rules (since once elimination contexts contain types, path equivalence is no longer just syntactic equality):

$$\frac{}{\Gamma \vdash \mathtt{int} \equiv \mathtt{int} :: *} \tag{35}$$

$$\frac{}{\Gamma \vdash X \equiv X :: \Gamma(X)} \tag{36}$$

$$\frac{\Gamma \vdash P_1 \equiv P_2 :: K_1{\times}K_2}{\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i} \tag{37}$$

$$\frac{\Gamma \vdash P_1 \equiv P_2 :: L{\Rightarrow}K \qquad \Gamma \vdash T_1 \equiv T_2 :: L}{\Gamma \vdash P_1\, T_1 \equiv P_2\, T_2 :: K} \tag{38}$$

Proposition 31

1. *If* $\Gamma \vdash T' \equiv S' :: K$, $T \rightsquigarrow^* T'$, $S \rightsquigarrow^* S'$, $\Gamma \vdash T :: K$, *and* $\Gamma \vdash S :: K$ *then* $\Gamma \vdash T \equiv S :: K$.

2. *If* $\Gamma \vdash T \equiv S :: K$, $T \rightsquigarrow^* T'$, *and* $S \rightsquigarrow^* S'$ *then* $\Gamma \vdash T' \equiv S' :: K$.

Proof: Same argument as for Proposition 9. ∎

As before, equivalence as defined by membership in νF_λ is reflexive, symmetric, and transitive.

Proposition 32

1. *If* $\Gamma \vdash T :: K$ *then* $\Gamma \vdash T \equiv T :: K$.

2. *If* $\Gamma \vdash T_1 \equiv T_2 :: K$ *and* $\Gamma \vdash T_2 \equiv T_3 :: K$ *then* $\Gamma \vdash T_1 \equiv T_3 :: K$.

3. *If* $\Gamma \vdash T \equiv S :: K$ *then* $\Gamma \vdash S \equiv T :: K$.

Proof:

1. Following the pattern of Proposition 10.

2. Following the pattern of Proposition 12.

3. Following the pattern of Proposition 13. ∎

Corollary 33
If $\Gamma \vdash T :: K$ *and* $T \rightsquigarrow T'$ *then* $\Gamma \vdash T \equiv T' :: K$*.*

Proof: Assume $\Gamma \vdash T :: K$ and $T \rightsquigarrow T'$. By Proposition 30, $\Gamma \vdash T' :: K$. By reflexivity, then $(\Gamma \vdash T' \equiv T' :: K) \in \nu F_\lambda$. Hence $(\Gamma \vdash T' \equiv T :: K) \in F_\pi(\nu F_\lambda) = \nu F_\lambda$. ∎ ∎ Similar admissible rules follow as well, e.g.,

Proposition 34 (Congruence for Projections)
If $\Gamma \vdash T \equiv S :: K_1{\times}K_2$ *then* $\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i$*.*

Proof: By induction on $\mathit{height}(T) + \mathit{height}(S)$.

- Case: $(\Gamma \vdash T \equiv S :: K_1{\times}K_2) \in \nu F_\lambda$ with $T = P_1$, $S = P_2$. Then immediately $(\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \in F_\lambda(\nu F_\lambda) = \nu F_\lambda$.
- Case: $T = \langle T_1, T_2\rangle$, $S = \langle S_1, S_2\rangle$, $(\Gamma \vdash T_1 \equiv S_1 :: K_1) \in \nu F_\lambda$, and $(\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \nu F_\lambda$. Then $\Gamma \vdash \pi_i\, T :: K_i$, $\Gamma \vdash \pi_i\, S :: K_i$, $\pi_i\, T \rightsquigarrow T_i$, and $\pi_i\, S \rightsquigarrow S_i$, so $(\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i) \in F_\lambda(F_\lambda(\nu F_\lambda)) = \nu F_\lambda$.
- Case: $T \rightsquigarrow T'$, $(\Gamma \vdash T' \equiv S :: K_1{\times}K_2) \in \nu F_\lambda$ and $\Gamma \vdash T :: K_1{\times}K_2$. Then by the inductive hypothesis, $(\Gamma \vdash \pi_i\, T' \equiv \pi_i\, S :: K_i) \in \nu F_\lambda$. Then $\Gamma \vdash \pi_i\, T :: K_i$ and $\pi_i\, T \rightsquigarrow \pi_i\, T'$, so $(\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i) \in F_\lambda(\nu F_\lambda) = \nu F_\lambda$.
- Case: $S \rightsquigarrow S'$, $(\Gamma \vdash T \equiv S' :: K_1{\times}K_2) \in \nu F_\lambda$ and $\Gamma \vdash S :: K_1{\times}K_2$. Similar to the previous case.∎

Proposition 35 (Weakening)
If $\Gamma_1, \Gamma_3 \vdash T \equiv S :: K$ *and* $\mathit{dom}(\Gamma_1, \Gamma_3) \cap \mathit{dom}(\Gamma_2) = \emptyset$ *then* $\Gamma_1, \Gamma_2, \Gamma_3 \vdash T \equiv S :: K$*.*

Proof: Let

$$\begin{aligned} W(\mathcal{J}) := \{&(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 \equiv T_2 :: K) \mid \\ &(\Gamma_1, \Gamma_3 \vdash T_1 \equiv T_2 :: K) \in \mathcal{J} \text{ and } \mathrm{dom}(\Gamma_1, \Gamma_3) \cap \mathrm{dom}(\Gamma_2) = \emptyset\}. \end{aligned}$$

By Corollary 3, proving $W(\nu F_\lambda) \subseteq \bigcup_{n\geq 1} F_\lambda^n(W(\nu F_\lambda))$ suffices to get $W(\nu F_\lambda) \subseteq \nu F_\lambda$. Assume $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 \equiv T_2 :: K) \in W(\nu F_\lambda)$ because $\Gamma_1, \Gamma_3 \vdash T_1 \equiv T_2 :: K$ and $\mathrm{dom}(\Gamma_1, \Gamma_3) \cap \mathrm{dom}(\Gamma_2) = \emptyset$. The desired result follows by induction on $\mathit{height}(T_1) + \mathit{height}(T_2)$.

- Case: $T_1 = T_2 = \texttt{int}$ and $K = *$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash \texttt{int} \equiv \texttt{int} :: *) \in F_\lambda(\emptyset) \subseteq F_\lambda(W(\nu F_\lambda))$.
- Case: $T_1 = T_2 = X$ and $K = (\Gamma_1, \Gamma_3)(X)$. Then $(\Gamma_1, \Gamma_3)(X) = (\Gamma_1, \Gamma_2, \Gamma_3)(X)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash X \equiv X :: K) \in F_\lambda(\emptyset) \subseteq F_\lambda(W(\nu F_\lambda))$.
- Case: $T_1 = \pi_i\, P_1$, $T_2 = \pi_i\, P_2$, and $K = K_i$, where $\Gamma_1, \Gamma_3 \vdash P_1 \equiv P_2 :: K_1{\times}K_2$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash P_1 \equiv P_2 :: K_1{\times}K_2) \in W(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \in F_\lambda(W(\nu F_\lambda))$.

- Case: $T_1 = P_1\, S_1$, $T_2 = P_2\, S_2$, where $\Gamma_1, \Gamma_3 \vdash P_1 \equiv P_2 :: L{\Rightarrow}K$ and $\Gamma_1, \Gamma_3 \vdash S_1 \equiv S_2 :: L$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash P_1 \equiv P_2 :: L{\Rightarrow}K) \in W(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash S_1 \equiv S_2 :: L) \in W(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash P_1\, S_1 \equiv P_2\, S_2 :: K) \in F_\lambda(W(\nu F_\lambda))$.

- Case: $T_1 = T_1'{\to}T_1''$, $T_2 = T_2'{\to}T_2''$, and $K = *$, where $\Gamma_1, \Gamma_3 \vdash T_1' \equiv T_2' :: *$ and $\Gamma_1, \Gamma_3 \vdash T_1'' \equiv T_2'' :: *$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1' \equiv T_2' :: *) \in W(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1'' \equiv T_2'' :: *) \in W(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1'{\to}T_1'' \equiv T_2'{\to}T_2'' :: *) \in F_\lambda(W(\nu F_\lambda))$.

- Case: $\Gamma_1, \Gamma_3 \vdash T_1' \equiv T_2 :: K$ with $T_1 \rightsquigarrow T_1'$ and $\Gamma_1, \Gamma_3 \vdash T_1 :: K$. By the inductive hypothesis $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1' \equiv T_2 :: K) \in \bigcup_{n\geq 1} F_\lambda^n(W(\nu F_\lambda))$. Since by Proposition 27 we have $\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 :: K$, it follows that $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 \equiv T_2 :: K) \in \bigcup_{n\geq 1} F_\lambda(F_\lambda^n(W(\nu F_\lambda))) \subseteq \bigcup_{n\geq 1} F_\lambda^n(W(\nu F_\lambda))$.

- Case: $\Gamma_1, \Gamma_3 \vdash T_1 \equiv T_2' :: K$ with $T_2 \rightsquigarrow T_2'$ and $\Gamma_1, \Gamma_3 \vdash T_2 :: K$. Same argument as the previous case.

- Case: $T_1 = \langle T_1', T_1'' \rangle$, $T_2 = \langle T_2', T_2'' \rangle$, and $K = K' \times K''$, where $\Gamma_1, \Gamma_3 \vdash T_1' \equiv T_2' :: K'$ and $\Gamma_1, \Gamma_3 \vdash T_1'' \equiv T_2'' :: K''$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1' \equiv T_2' :: K') \in W(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1'' \equiv T_2'' :: K'') \in W(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1'{\to}T_1'' \equiv T_2'{\to}T_2'' :: K' \times K'') \in F_\lambda(W(\nu F_\lambda))$.

- Case: $T_1 = \lambda Y{::}L.T_1'$, $T_2 = \lambda Y{::}L.T_2'$, and $K = L'{\Rightarrow}K''$, where $\Gamma_1, \Gamma_3, Y{::}L' \vdash T_1' \equiv T_2' :: K''$. Without loss of generality we can assume $Y \notin \mathrm{dom}(\Gamma_2)$, in which case $(\Gamma_1, \Gamma_2, \Gamma_3, Y{::}L' \vdash T_1' \equiv T_2' :: K'') \in W(\nu F_\lambda)$, and so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash \lambda Y{::}L.T_1' \equiv \lambda Y{::}L.T_2' :: L'{\Rightarrow}K'') \in F_\lambda(W(\nu F_\lambda))$. ■

Proposition 36 (Functionality)
Put

$$H(\mathcal{J}) := \{(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K) \mid (\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv S_1 :: K) \in \mathcal{J} \text{ and } (\Gamma_1 \vdash T_2 \equiv S_2 :: L) \in \mathcal{J}\}.$$

1. $\nu F_\lambda \subseteq H(\nu F_\lambda)$.

2. $H(\nu F_\lambda) \subseteq \nu F_\lambda$.

3. If $\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv S_1 :: K$ and $\Gamma_1 \vdash T_2 \equiv S_2 :: L$ then $\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K$.

Proof:

1. Assume $\Gamma \vdash T_1 \equiv T_2 :: K$. Pick $X \notin \mathrm{dom}(\Gamma)$; by Proposition 35 we have $\Gamma, X{::}* \vdash T_1 \equiv T_2 :: K$. Since $\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *$, we have $\Gamma \vdash T_1 \equiv T_2 :: K \in H(\nu F_\lambda)$.

2. We wish to show that $H(\nu F_\lambda) \subseteq \nu F_\lambda$. By the Coinduction Principle, it suffices to show that $H(\nu F_\lambda) \subseteq F_\lambda(H(\nu F_\lambda))$. Assume that $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K) \in H(\nu F_\lambda)$ because $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv S_1 :: K) \in \nu F_\lambda$ and $(\Gamma_1 \vdash T_2 \equiv S_2 :: K_2) \in \nu F_\lambda$. We proceed by cases on the justification for $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv S_1 :: K) \in \nu F_\lambda = F_\lambda(\nu F_\lambda)$. By Proposition 27, in all cases we have $\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 :: K$ and $\Gamma_1, \Gamma_2 \vdash \{S_2/Y\}S_1 :: K$.

   - Case: $T_1 = T_2 = \texttt{int}$ and $K = *$. Then $(\Gamma_1, \Gamma_2 \vdash \texttt{int} \equiv \texttt{int} :: *) \in F_\lambda(\emptyset) \subseteq F_\lambda(H(\nu F_\lambda))$.

- Case: $T_1 = E_1[X]$ and $T_2 = E_2[X]$.

  If $X = Y$ then since $E_1[X]$ and $E_2[X]$ are well-formed and L contains no arrows, we know that E_1 and E_2 consist only of projections (and hence, have no free variables). In this case, by definition of F_λ, $E_1 = E_2$; we will denote this common elimination context by E. By repeated use of Proposition 34 we have $\Gamma_1 \vdash E[T_2] \equiv E[S_2] :: K$, and by Proposition 35, Part 1, and monotonicity we have $(\Gamma_1, \Gamma_2 \vdash E[T_2] \equiv E[S_2] :: K) \in \nu F_\lambda = F_\lambda(\nu F_\lambda) \subseteq F_\lambda(H(\nu F_\lambda))$.

  Otherwise, when $X \neq Y$ there are three subcases:

  - Subcase: $E_1 = E_2 = \bullet$ and $K = (\Gamma_1, \Gamma_2)(X)$. Then $(\Gamma_1, \Gamma_2 \vdash X \equiv X :: K) \in F_\lambda(\emptyset) \subseteq F_\lambda(H(\nu F_\lambda))$.
  - Subcase: $E_1 = \pi_j\, E'_1$, $E_2 = \pi_j\, E'_2$, and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash E'_1[X] \equiv E'_2[X] :: K_1{\times}K_2) \in \nu F_\lambda$ with $K = K_j$. Then we have $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}(E'_1[X]) \equiv \{S_2/Y\}(E'_2[X]) :: K_1{\times}K_2) \in H(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}(\pi_j\, E'_1[X]) \equiv \{S_2/Y\}(\pi_j\, E'_2[X]) :: K) \in F_\lambda(H(\nu F_\lambda))$ as desired.
  - Subcase: $E_1 = (E'_1[X])\, U_1$, $E_2 = (E'_1[X])\, U_2$, $(\Gamma_1, Y{::}L, \Gamma_2 \vdash E'_1[X] \equiv E'_2[X] :: L'{\Rightarrow}K) \in \nu F_\lambda$, and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash U_1 \equiv U_2 :: L') \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}(E'_1[X]) \equiv \{S_2/Y\}(E'_2[X]) :: L'{\Rightarrow}K) \in H(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}U_1 \equiv \{S_2/Y\}U_2 :: L') \in H(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}((E'_1[X])\, U_1 \equiv \{S_2/Y\}((E'_2[X])\, U_2) :: K) \in F_\lambda(H(\nu F_\lambda))$ as desired.

- Case: $T_1 = T'_1{\to}T''_1$ and $S_1 = S'_1{\to}S''_1$ and $K = *$, with $(\Gamma_1, Y{::}K_2, \Gamma_2 \vdash T'_1 \equiv S'_1 :: *) \in \nu F_\pi$ and $(\Gamma_1, Y{::}K_2, \Gamma_2 \vdash T''_1 \equiv S''_1 :: *) \in \nu F_\pi$.

  Then $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T'_1 \equiv \{S_2/Y\}S'_1 :: *) \in H(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T''_1 \equiv \{S_2/Y\}S''_1 :: *) \in H(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}(T'_1{\to}T''_1) \equiv \{S_2/Y\}(S'_1{\to}S''_1) :: *) \in F_\lambda(H(\nu F_\lambda))$.

- Case: $T_2 \rightsquigarrow T'_2$ and $(\Gamma \vdash T'_1 \equiv S_1 :: K) \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T'_1 \equiv \{S_2/Y\}S_1 :: K) \in H(\nu F_\lambda)$ and by Proposition 30 we have $\{T_2/Y\}T_1 \rightsquigarrow \{T_2/Y\}T'_1$, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K) \in F_\lambda(H(\nu F_\lambda))$.

- Case: $S_1 \rightsquigarrow S'_1$ and $(\Gamma \vdash T_1 \equiv S'_1 :: K) \in \nu F_\lambda$. Same argument as for the previous case.

- Case: $T_1 = \langle T'_1, T''_1\rangle$ and $S_1 = \langle S'_1, S''_1\rangle$ and $K = K'{\times}K''$, with $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T'_1 \equiv S'_1 :: K') \in \nu F_\lambda$ and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T''_1 \equiv S''_1 :: K'') \in \nu F_\lambda$.

  Then $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T'_1 \equiv \{S_2/Y\}S'_1 :: *) \in H(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T''_1 \equiv \{S_2/Y\}S''_1 :: *) \in H(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}(T'_1{\to}T''_1) \equiv \{S_2/Y\}(S'_1{\to}S''_1) :: *) \in F_\lambda(H(\nu F_\lambda))$.

- Case: $T_1 = \lambda X{::}L'.T'_1$ and $T_2 = \lambda X{::}L'.T'_2$ and $(\Gamma_1, Y{::}L, \Gamma_2, X{::}L' \vdash T'_1 \equiv T'_2 :: K')$ with $K = L'{\Rightarrow}K'$ Without loss of generality $X \neq Y$ and $X \notin FV(T_2) \cup FV(S_2)$. Then $(\Gamma_1, \Gamma_2, X{::}L' \vdash \{T_2/Y\}T'_1 \equiv \{S_2/Y\}S'_1 :: K') \in H(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \lambda X{::}L'.(\{T_2/Y\}T'_1) \equiv \lambda X{::}L'.(\{S_2/Y\}S'_1) :: K) \in F_\lambda(H(\nu F_\lambda))$.

3. This is a restatement of Part 2.∎

We can then prove the admissibility of the standard rule that applying equals to equals yields equals.

Proposition 37 (Congruence for Applications)
If $\Gamma \vdash T_1 \equiv S_1 :: L {\Rightarrow} K$ and $\Gamma \vdash T_2 \equiv S_2 :: L$ then $\Gamma \vdash T_1\, T_2 \equiv S_1\, S_2 :: K$.

Proof: By induction on $\mathit{height}(T_1\, T_2) + \mathit{height}(S_1\, S_2)$.

- Case: $T_1 = P_1$ and $S_1 = P_2$. Then $(\Gamma_2 \vdash P_1\, T_2 \equiv P_2\, S_2 :: K) \in F_\lambda(\nu F_\lambda) = \nu F_\lambda$.
- Case: $T_1 \rightsquigarrow T_1'$ and $(\Gamma \vdash T_1' \equiv T_2 :: L{\Rightarrow}K) \in \nu F_\lambda$. Then $T_1\, T_2 \rightsquigarrow T_1'\, T_2$ and inductively $(\Gamma_2 \vdash T_1'\, T_2 \equiv S_1\, S_2 :: K) \in \nu F_\lambda$, so $(\Gamma_2 \vdash T_1\, T_2 \equiv S_1\, S_2 :: K) \in F_\lambda(\nu F_\lambda) = \nu F_\lambda$.
- Case: $S_1 \rightsquigarrow T_2'$ and $(\Gamma \vdash T_1 \equiv T_2' :: L{\Rightarrow}K) \in \nu F_\lambda$. Similar.
- Case: $T_1 = \lambda X{::}L.T_1'$, $S_1 = \lambda X{::}L.T_2'$, and $(\Gamma, X{::}L \vdash T_1' \equiv T_2' :: K) \in \nu F_\lambda$. Then $(\Gamma \vdash \{T_2/X\}T_1' \equiv \{S_2/X\}T_2' :: K) \in \nu F_\lambda$ by Proposition 36. Thus $(\Gamma \vdash (\lambda X{::}L.T_1')\, T_2 \equiv (\lambda X{::}L.T_2')\, S_2 :: K) \in F_\lambda(F_\lambda(\nu F_\lambda)) = \nu F_\lambda$. $\blacksquare$

Proposition 38 (Congruence for Recursive Types)
Let

$$\begin{aligned} C(\mathcal{J}) :=\{ & (\Gamma_1, \Gamma_2 \vdash \{\mu X{::}L.\, S_1/Y\}T_1 \equiv \{\mu X{::}L.\, S_2/Y\}T_2 :: K) \mid \\ & (\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv T_2 :: K),\ (\Gamma_1, X{::}L \vdash S_1 \equiv S_2 :: L) \in \mathcal{J}, \\ & X \notin \mathit{dom}(\Gamma_1), Y \notin \mathit{dom}(\Gamma_1, \Gamma_2)\ \}. \end{aligned}$$

1. *$C(\nu F_\lambda) \subseteq \nu F_\lambda$.*
2. *If $\Gamma, X{::}L \vdash S_1 \equiv S_2 :: L$ then $\Gamma \vdash \mu X{::}L.\, S_1 \equiv \mu X{::}L.\, S_2 :: L$.*

Proof:

1. By Corollary 3 it suffices to show that $C(\nu F_\lambda) \subseteq F_\lambda(C(\nu F_\lambda)) \cup F_\lambda(F_\lambda(C(\nu F_\lambda)))$.

   Assume $(\Gamma_1, \Gamma_2 \vdash \{\mu X{::}L.\, S_1/Y\}T_1 \equiv \{\mu X{::}L.\, S_2/Y\}T_2 :: K) \in C(\nu F_\lambda)$ because $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv T_2 :: K) \in \nu F_\lambda = F_\lambda(\nu F_\lambda)$ and $(\Gamma_1, X{::}L \vdash S_1 \equiv S_2 :: L) \in \nu F_\lambda$.

   We will abbreviate the substitutions $\{\mu X{::}L.\, S_1/Y\}$ and $\{\mu X{::}L.\, S_2/Y\}$ as σ_1 and σ_2 respectively. Immediately, we have $\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1 :: K$ and $\Gamma_1, \Gamma_2 \vdash \sigma_2 T_2 :: K$, and we want to show that $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1 \equiv \sigma_2 T_2 :: K) \in F_\lambda(C(\nu F_\lambda)) \cup F_\lambda(F_\lambda(C(\nu F_\pi)))$.

   - Case: $T_1 = T_2 = \texttt{int}$. Then $(\Gamma_1, \Gamma_2 \vdash \texttt{int} \equiv \texttt{int} :: *) \in F_\lambda(\emptyset) \subseteq F_\lambda(C(\nu F_\lambda))$.
   - Case: $T_1 = E_1[Y]$ and $T_2 = E_2[Y]$.
     Since $E_1[Y]$ and $E_2[Y]$ are well-formed and the kind L of Y contains no arrows, we know that E_1 and E_2 must consist only of projections with no applications. In this case, by definition of F_λ, $E_1 = E_2$; we will denote this common elimination context by E.
     Since $(\Gamma_1, X{::}L \vdash S_1 \equiv S_2 :: L) \in \nu F_\lambda$, by repeated use of Proposition 34 we obtain $(\Gamma_1, X{::}L \vdash E[S_1] \equiv E[S_2] :: K) \in \nu F_\lambda$. By Proposition 35 we have $(\Gamma_1, X{::}L, \Gamma_2 \vdash E[S_1] \equiv E[S_2] :: K) \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash E[\{\mu X{::}L.\, S_1/X\}S_1] \equiv E[\{\mu X{::}L.\, S_2/X\}S_2] :: K) \in C(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash E[\mu X{::}L.\, S_1] \equiv E[\mu X{::}L.\, S_2] :: K) \in F_\lambda(F_\lambda(C(\nu F_\lambda)))$.
   - Case: $T_1 = E_1[Z]$ and $T_2 = E_2[Z]$ with $X' \neq Y$.
     There are three subcases.

$$\frac{T \sqsubseteq E[\{S_2/X\}S_1]}{T \sqsubseteq E[(\lambda X{::}L.S_1)\,S_2]} \tag{39}$$

$$\frac{T \preceq E[S_1] \qquad X \notin FV(E)}{\{S_2/X\}T \preceq E[(\lambda X{::}L.S_1)\,S_2]} \tag{41}$$

$$\frac{T \sqsubseteq S_i}{T \sqsubseteq S_1\,S_2} \tag{40}$$

$$\frac{T \preceq S_i}{T \preceq S_1\,S_2} \tag{42}$$

Figure 5: Additional Top-Down and Bottom-Up Subterms

- Subcase: $E_1 = E_2 = \bullet$ and $K = (\Gamma_1, \Gamma_2)(X)$. Then $(\Gamma_1, \Gamma_2 \vdash X \equiv X :: K) \in F_\lambda(\emptyset) \subseteq F_\lambda(C(\nu F_\lambda))$.
- Subcase: $E_1 = \pi_j\, E_1'$ and $E_2 = \pi_j\, E_2'$ and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash E_1'[Z] \equiv E_2'[Z] :: K_1{\times}K_2) \in \nu F_\lambda$ with $K = K_j$. Then $(\Gamma_1, \Gamma_2 \vdash \sigma_1(E_1'[Z]) \equiv \sigma_2(E_2'[Z]) :: K_1{\times}K_2) \in C(\nu F_\lambda)$, so that $(\Gamma_1, \Gamma_2 \vdash \sigma_1(\pi_j\, E_1'[Z]) \equiv \sigma_2(\pi_j\, E_2'[Z]) :: K) \in F_\lambda(C(\nu F_\lambda))$.
- Subcase: $E_1 = E_1'\, U_1$ and $E_2 = E_2'\, U_2$ and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash E_1'[Z] \equiv E_2'[Z] :: L'{\Rightarrow}K) \in \nu F_\lambda$ and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash U_1 \equiv U_2 :: L') \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash \sigma_1(E_1')[Z] \equiv \sigma_2(E_2')[Z] :: L'{\Rightarrow}K) \in C(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2 \vdash \sigma_1 U_1 \equiv \sigma_2 U_2 :: L') \in C(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1((E_1'[Z])\, U_1) \equiv \sigma_2((E_2'[Z])\, U_2) :: K) \in F_\lambda(C(\nu F_\lambda))$.

- Case: $T_1 \leadsto T_1'$ and $(\Gamma \vdash T_1' \equiv T_2 :: K) \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1' \equiv \sigma_1 T_2 :: K) \in C(\nu F_\lambda)$ and by Proposition 30 we have $\sigma_1 T_1 \leadsto \sigma_1 T_1'$, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1 \equiv \sigma_2 T_2 :: K) \in F_\lambda(C(\nu F_\lambda))$, as desired.
- Case: $T_2 \leadsto T_2'$ and $(\Gamma \vdash T_1 \equiv T_2' :: K) \in \nu F_\lambda$. Similar.
- Case: $T_1 = \langle T_1', T_1'' \rangle$ and $T_2 = \langle T_1', T_1'' \rangle$ and $K = K'{\times}K''$, where we have $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1' \equiv T_2' :: K') \in \nu F_\lambda$ and $(\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1'' \equiv T_2'' :: K'') \in \nu F_\lambda$. Then $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1' \equiv \sigma_2 T_2' :: K') \in C(\nu F_\lambda)$ and $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1'' \equiv \sigma_2 T_2'' :: K'') \in C(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1(\langle T_1', T_1'' \rangle) \equiv \sigma_2(\langle T_2', T_2'' \rangle) :: K'{\times}K'') \in F_\lambda(C(\nu F_\lambda))$.
- Case: $T_1 = T_1'{\to}T_1''$ and $T_2 = T_1'{\to}T_1''$ and $K = *$. Similar.
- Case: $T_1 = \lambda Z{::}L'.T_1'$ and $T_2 = \lambda Z{::}L'.T_2'$ and $(\Gamma_1, Y{::}L, \Gamma_2, Z{::}L' \vdash T_1' \equiv T_2' :: K') \in \nu F_\lambda$ with $K = L'{\Rightarrow}K'$. Without loss of generality, $Z \neq Y$ and $Z \notin FV(S_1) \cup FV(S_2)$. Then $(\Gamma_1, \Gamma_2, Z{::}L' \vdash \sigma_1 T_1' \equiv \sigma_2 T_2' :: K') \in C(\nu F_\lambda)$, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1(\lambda Z{::}L'.T_1') \equiv \sigma_2(\lambda Z{::}L'.T_2') :: K) \in F_\lambda(C(\nu F_\lambda))$.

2. If $\Gamma, X{::}L \vdash S_1 \equiv S_2 :: L$ then by Part 1 we have $(\Gamma \vdash \{\mu X{::}L.\, S_1/X\}S_1 \equiv \{\mu X{::}L.\, S_2/X\}S_2 :: L) \in \nu F_\lambda$ and hence $(\Gamma \vdash \mu X{::}L.\, S_1 \equiv \mu X{::}L.\, S_2 :: L) \in F_\lambda(F_\lambda(\nu F_\lambda)) = \nu F_\lambda$.

5.5 Decidability

Since we do not allow recursively-defined type operators, unfolding a β-normal type yields a β-normal type. This suggests we can compare types by β-normalizing and then using the previous algorithm. Unfortunately, it is not immediately obvious that this algorithm would be complete. We instead show that the techniques used for pairs can be reapplied for functions, a fact interesting in itself.

We can extend the definition of bottom-up and top-down subterms, as shown in Figure 5. We also add to Rules 21 and 27 the requirement that $X \notin FV(E)$, which is always possible by renaming of bound variables.

The following two lemmas continue to hold in the extended system:

Lemma 39
If $T \preceq \pi_i\, S$ then $\pi_i\, S \leadsto^* T$ or $T \preceq S$.

Proof: By induction on the proof of the assumption.

- Case: $T \preceq \pi_i\, E[(\lambda X{::}L.S_1)\, S_2]$ because $T = \{S_2/X\}T'$ and $T' \preceq \pi_i\, (E[S_1])$ where without loss of generality $X \notin FV(E)$. By the inductive hypothesis there are two subcases:
  - Subcase: $\pi_i\, (E[S_1]) \leadsto^* T'$. Then using Proposition 30 and $X \notin FV(E)$, we have $\pi_i\, (E[(\lambda X{::}L.S_1)\, S_2]) \leadsto \pi_i\, (E[\{S_2/X\}S_1]) = \{S_2/X\}(\pi_i\, (E[S_1])) \leadsto^* \{S_2/X\}T' = T$.
  - Subcase: $T' \preceq E[S_1]$. Then by Rule 41 we have that $T = \{S_2/X\}T' \preceq E[(\lambda X{::}L.S_1)\, S_2] = S$.

The remaining cases follow as in Lemma 20. ∎

Lemma 40
Assume $\Gamma \vdash U :: L$ and $\Gamma, X{::}L \vdash T :: K$. If $S \preceq \{U/X\}T$ then either $S \preceq U$ or there exists $T' \preceq T$ with $\{U/X\}T' \leadsto^* S$.

Proof: By induction on T.

- Case: $T = \lambda Y{::}L_1.T_2$. Then $T = \{U/X\}T$ since abstractions have no proper subterms, and we can choose $T' = T$.
- Case: $T = T_1\, T_2$. There are several subcases depending why $S \preceq \{U/X\}T = (\{U/X\}T_1)\, (\{U/X\}T_2)$.
  - Subcase: $S \preceq \{U/X\}T$ because $S \preceq \{U/X\}T_1$ by Rule 42. By the inductive hypothesis, either $S \preceq U$, or else there exists $T' \preceq T_1$ such that $\{U/X\}T' \leadsto^* S$. In the latter case, by Rule 42 we have $T' \preceq T_1\, T_2 = T$.
  - Subcase: $S \preceq \{U/X\}T$ because $S \preceq \{U/X\}T_2$ by Rule 42. Similar.
  - Subcase: $S \preceq \{U/X\}T$ using Rule 28 because $\{U/X\}T_1 = E'[\pi_i\, \langle T_1', T_2' \rangle]$ and $S \preceq (E'[T_i'])\, (\{U/X\}T_2)$.
    * Subsubcase: T_1 is of the form $E''[X]$ (i.e., the pair $\langle T_1', T_2' \rangle$ and possibly the π_i came from U), so $\{U/X\}T = ((\{U/X\}E'')[U])\, (\{U/X\}T_2)$. But this is impossible; by assumption U has no arrows in its kind, so $\{U/X\}T_1 = (\{U/X\}E'')[U]$ has no arrows in its kind, which contradicts the fact that by Proposition 27 we know that the application of this to $\{U/X\}T_2$ (i.e., the application $\{U/X\}(T_1\, T_2)$) is well-formed in the context Γ.
    * Subsubcase: $T_1 = E''[\pi_i\, \langle T_1'', T_2'' \rangle]$, $E' = \{U/X\}E''$, $T_1' = \{U/X\}T_1''$, and $T_2' = \{U/X\}T_2''$. Then we have that $S \preceq (E'[T_i'])\, (\{U/X\}T_2) = \{U/X\}((E''[T_i''])\, T_2)$. By the inductive hypothesis, either $S \preceq U$ and we are done, or else there exists $T' \preceq (E''[T_i''])\, T_2$ with $\{U/X\}T' \leadsto^* S$. In the latter case, by Rule 28 we have $T' \preceq (E''[\pi_i\, \langle T_1'', T_2'' \rangle])\, T_2 = T_1\, T_2 = T$, as required.

- Subcase: $\{U/X\}T = E'[(\lambda Y{::}L_3.T_3')\,T_4']$ and $T = \{T_4'/Y\}S'$ with $S' \preceq E'[T_3']$. Without loss of generality $X \neq Y$ and $Y \notin FV(U)$. Since $\Gamma \vdash U :: L$, we know U itself not a type abstraction and hence it must be the case that $T = E''[(\lambda Y{::}L_3.T_3'')\,T_4'']$ (where T_4'' may or may not be T_2) with $E' = \{U/X\}E''$, $T_3' = \{U/X\}T_3''$ and $T_4' = \{U/X\}T_4''$. As $S' \preceq \{U/X\}(E''[T_3''])$, by the inductive hypothesis there are two possibilities:
  - ∗ Subsubcase: $S' \preceq U$. Then $FV(S') \subseteq FV(U)$ and hence $T = \{T_4'/Y\}S' = S' \preceq U$.
  - ∗ Subsubcase: there exists $T_3' \preceq E''[T_3'']$ such that $\{U/X\}T_3' \leadsto^* S'$. Put $T' = \{T_4''/Y\}T_3'$. Then we have $\{U/X\}T' = \{U/X\}(\{T_4''/Y\}T_3') = \{(\{U/X\}T_4'')/Y\}(\{U/X\}T_3') \leadsto^* \{(\{U/X\}T_4'')/Y\}S' = \{T_4'/Y\}S' = T$. Since $T_3' \preceq E''[T_3'']$, by Rule 41 we have that $T' = \{T_4''/Y\}T_3'' \preceq E''[(\lambda Y{::}L.T_3'')\,T_4''] = T$.
- Subcase: $\{U/X\}T = E'[\mu Y{::}L.\,T_3']$ and $T = \{\mu Y{::}L.\,T_3'/Y\}S'$ with $S' \preceq E'[T_3']$. This cannot occur; by well-formedness constraints no type of the form $E'[\mu Y{::}L.\,T_3']$ can be an application (since the kind of $\mu Y{::}L.\,T_3'$ contains no arrows).

The remaining cases follow as in Lemma 21. ∎

Proposition 41
Every top-down subterm of a well-formed type is a weak head reduct of a bottom-up subterm: if $\Gamma \vdash S :: K$ and $T \sqsubseteq S$ then there exists T' such that $T' \preceq S$ and $T' \leadsto^* T$.

Proof: By induction on the proof that $T \sqsubseteq S$, and cases on the last rule used. The definitions of $\sqsubseteq$ and $\preceq$ differ only in two rules (Rule 27 vs. Rule 21, and Rule 41 vs. Rule 39). These are the only interesting cases:

- Case: $S = E[\mu X{::}L.\,S_1]$ and $T \sqsubseteq S$ because $T \sqsubseteq E[\{\mu X{::}L.\,S_1/X\}S_1]$, where without loss of generality $X \notin FV(E)$. By the inductive hypothesis, there exists T_1' such that $T_1' \preceq E[\{\mu X{::}L.\,S_1/X\}S_1] = \{\mu X{::}L.\,S_1/X\}(E[S_1])$ and $T_1' \leadsto^* T$. Then by Lemma 40 there are two possibilities:
  - Subcase: $T_1' \preceq \mu X{::}L.\,S_1$. Then we can take $T' = T_1'$ because by well-formedness E cannot contain applications, and so repeated use of Rule 29 yields $S' = T_1' \preceq E[\mu X{::}L.\,S_1] = S$.
  - Subcase: There exists S_1' such that $S_1' \preceq E[S_1]$ and $(\{\mu X{::}L.\,S_1/X\}S_1') \leadsto^* T_1'$. Then we can take $T' = (\{\mu X{::}L.\,S_1/X\}S_1')$ because then $T' \leadsto^* T_1' \leadsto^* T$ and by Rule 27 we have $T' = (\{\mu X{::}L.\,S_1/X\}S_1') \preceq E[\mu X{::}L.\,S_1] = S$.
- Case: $S = E[(\lambda X{::}L.S_1)\,S_2]$ and $T \sqsubseteq S$ because $T \sqsubseteq E[\{S_2/X\}S_1]$; further, without loss of generality $X \notin FV(E)$. Then by the induction hypothesis there exists T_1' such that $T_1' \preceq E[\{S_2/X\}S_1] = \{S_2/X\}(E[S_1])$ and $T_1' \leadsto^* T$. By Lemma 40 there are two possibilities:
  - Subcase: $T_1' \preceq S_2$. Then we can take $T' = T_1'$ because using Rules 29 and 42 we have $T_1' \preceq E[(\lambda X{::}L.S_1)\,S_2] = S$.
  - Subcase: There exists S_1' such that $S_1' \preceq E[S_1]$ and $(\{S_2/X\}S_1') \leadsto^* T_1'$. Put $T' = (\{S_2/X\}S_1')$, so that $T' \leadsto^* T_1' \leadsto^* T$ and by Rule 41 we have $T' = (\{S_2/X\}S_1') \preceq E[(\lambda X{::}L.S_1)\,S_2] = S$.

$$\begin{aligned}
F^a_{\lambda-}(\mathcal{J}) := {}& \{\, (\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *) \mid \text{for all } \Gamma \} \\
\cup {}& \{\, (\Gamma \vdash X \equiv X :: K) \mid K = \Gamma(X) \} \\
\cup {}& \{\, (\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \mid (\Gamma \vdash P_1 \equiv P_2 :: K_1{\times}K_2) \in \mathcal{J} \} \\
\cup {}& \{\, (\Gamma \vdash P_1\, T_1 \equiv P_2\, T_2 :: K) \mid \\
& \qquad (\Gamma \vdash P_1 \equiv P_2 :: L{\Rightarrow}K) \in \mathcal{J} \text{ and } (\Gamma \vdash T_1 \equiv T_2 :: L) \in \mathcal{J} \} \\
\cup {}& \{\, (\Gamma \vdash T_1{\to}T_2 \equiv S_1{\to}S_2 :: *) \mid \\
& \qquad (\Gamma \vdash T_1 \equiv S_1 :: *) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J} \} \\
\cup {}& \{\, (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow T' \text{ and } (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J} \text{ and } \Gamma \vdash T :: K \} \\
\cup {}& \{\, (\Gamma \vdash T \equiv S :: K) \mid T \not\rightsquigarrow,\ S \rightsquigarrow S',\ (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J},\ \text{and } \Gamma \vdash S :: K \} \\
\cup {}& \{\, (\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1{\times}K_2) \mid \\
& \qquad (\Gamma \vdash T_1 \equiv S_1 :: K_1) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \mathcal{J} \} \\[1ex]
F^a_{\lambda}(\mathcal{J}) := {}& F^a_{\lambda-}(\mathcal{J}) \\
\cup {}& \{\, (\Gamma \vdash \lambda X{::}L.T \equiv \lambda X{::}L.S :: L{\Rightarrow}K) \mid (\Gamma, X{::}L \vdash T \equiv S :: K) \in \mathcal{J} \}
\end{aligned}$$

Figure 6: Invertible Equivalence with Type Abstractions

The remaining cases follow directly from the inductive hypothesis. ∎

Proposition 42

1. *The set $\{\, T \mid T \preceq S \}$ is finite for every well-formed type S.*
2. *The set $\{\, T' \mid \exists T \preceq S.\ T \rightsquigarrow^* T' \,\}$ is finite for every well-formed type S.*
3. *The set $\{\, T \mid T \sqsubseteq S \,\}$ is finite for every well-formed type S.*

Proof:

1. By induction on the type S.
2. By Proposition 30 weak head reduction is deterministic and strongly normalizing, so there can be only a finite number of reducts of the elements of the finite set $\{T \mid T \preceq S\}$.
3. By Part 1 and Proposition 41.∎

Next, Figure 6 defines two invertible generating functions. The function F^a_λ, though written as a combination of two parts, differs from F_λ only in the left-to-right ordering of reductions. As for F_π and F^a_π before, the two functions have the same fixed point, for the same reasons.

Proposition 43
$\nu F_\lambda = \nu F^a_\lambda$.

Unfortunately, though F^a_λ is invertible we cannot directly use finiteness of top-down subterms to show that it is also finite-state. When comparing two type abstractions the predecessor has a different context, so repeating the same pair of types is no guarantee that the whole judgment has been repeated.

Therefore, we temporarily restrict attention to the function $F^a_{\lambda-}$, which never changes typing contexts. The finite state property for $F^a_{\lambda-}$ follows as for F^a_π.

Proposition 44
The $\sqsubseteq$ relation is transitive: if $T_1 \sqsubseteq T_2$ and $T_2 \sqsubseteq T_3$ then $T_1 \sqsubseteq T_3$.

Proof: By induction on the proof of $T_2 \sqsubseteq T_3$. ∎

Proposition 45

1. *If $(\Gamma' \vdash T' \equiv S' :: K') \in \mathit{pred}[F^a_{\lambda-}](\Gamma \vdash T \equiv S :: K)$ then $T' \sqsubseteq T$ and $S' \sqsubseteq S$ and $\Gamma = \Gamma'$.*
2. *If $(\Gamma' \vdash T' \equiv S' :: K') \in \mathit{reachable}[F^a_{\lambda-}](\{\Gamma \vdash T \equiv S :: K\})$ then $T' \sqsubseteq T$ and $S' \sqsubseteq S$ and $\Gamma = \Gamma'$.*

Proof:

1. By definition of $F^a_{\lambda-}$.
2. By Part 1 and Proposition 44.∎

Corollary 46
$F^a_{\lambda-}$ is finite-state.

We would like to show next that F^a_{λ} and $F^a_{\lambda-}$ agree on comparisons at the base kind $*$, but this is insufficient as a coinductive hypothesis; checking the equivalence of types at kind $*$ may require comparisons at higher kinds. For example, checking a judgment of the form $X{::}(*{\times}*){\Rightarrow}* \vdash X\langle T_1, T_2\rangle \equiv X\, T_3 :: *$ requires comparing X with itself at kind $(*{\times}*){\Rightarrow}*$ and comparing $\langle T_1, T_2\rangle$ with T_3 at kind $*{\times}*$. However, we can guarantee that when starting with a comparison at kind $*$, if we reach a comparison at a kind $K_1{\times}K_2$ then neither K_1 nor K_2 contain arrows (pair kinds can only arise when comparing function arguments) and if we compare at a kind $L{\Rightarrow}K$ then we are doing so because we are comparing two paths being applied to arguments. In all such cases, the difference between F^a_{λ} and $F^a_{\lambda-}$ is irrelevant:

Lemma 47
Put

$$\begin{array}{rl}\mathcal{A} := & \{\, (\Gamma \vdash T \equiv S :: K) \in U_{eq} \mid \\ & \qquad \textit{K is arrow-free, or T and S both reduce to paths.}\,\}\end{array}$$

1. *If $J \in \mathcal{A}$ then $\mathit{pred}[F^a_{\lambda-}](J) = \mathit{pred}[F^a_{\lambda}](J) \subseteq \mathcal{A}$.*
2. *If $J \in \mathcal{A}$ then $\mathit{reachable}[F^a_{\lambda}](\{J\}) = \mathit{reachable}[F^a_{\lambda-}](\{J\}) \subseteq \mathcal{A}$.*

Proof:

1. By definition of F^a_{λ} and $F^a_{\lambda-}$.
2. By induction and Part 1. ∎

Proposition 48
F^a_{λ} is finite-state.

$$
\begin{aligned}
F_\eta(\mathcal{J}) := {} & \{\, (\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *) \mid \text{for all } \Gamma \} \\
& \cup \{\, (\Gamma \vdash X \equiv X :: K) \mid \Gamma \vdash X :: K \} \\
& \cup \{\, (\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \mid (\Gamma \vdash P_1 \equiv P_2 :: K_1 {\times} K_2) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash P_1\, T_1 \equiv P_2\, T_2 :: K) \mid \\
& \qquad\qquad (\Gamma \vdash P_1 \equiv P_2 :: L {\Rightarrow} K) \in \mathcal{J} \text{ and } (\Gamma \vdash T_1 \equiv T_2 :: L) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash T_1 {\to} T_2 \equiv S_1 {\to} S_2 :: K) \mid \\
& \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: K) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: K) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: K) \mid T \leadsto T',\ (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J},\ \text{and } \Gamma \vdash T :: K \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: K) \mid S \leadsto S',\ (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J},\ \text{and } \Gamma \vdash S :: K \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: K_1 {\times} K_2) \mid \text{at least one of } T \text{ and } S \text{ is not a path}, \\
& \qquad\qquad (\Gamma \vdash \pi_1\, T \equiv \pi_1\, S :: K_1) \in \mathcal{J},\ \text{and } (\Gamma \vdash \pi_2\, T \equiv \pi_2\, S :: K_2) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: L_1 {\Rightarrow} K_2) \mid \text{at least one of } T \text{ and } S \text{ is not a path}, \\
& \qquad\qquad Z \notin FV(T) {\cup} FV(S),\ \text{and } (\Gamma, Z{::}L_1 \vdash T\, Z \equiv S\, Z :: K_2) \in \mathcal{J} \}
\end{aligned}
$$

Figure 7: Generating Function for Equivalence with Extensionality

Proof: We show that $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T \equiv S :: K\})$ is finite, by induction on K. Assume $\Gamma \vdash T :: K$ and $\Gamma \vdash T :: S$. Without loss of generality we may assume that T and S are weak head normal; otherwise a finite number of weak head reducts are added.

- Case: $K = *$. Then by Lemma 47, $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T \equiv S :: *\}) = \mathit{reachable}[F^a_{\lambda-}](\{\Gamma \vdash T \equiv S :: *\})$, which is finite.

- Case: $K = K_1 {\times} K_2$. If T and S are not pairs, then Lemma 47 and Corollary 46 imply $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T \equiv S :: K_1 {\times} K_2\})$ is finite. Otherwise, when $T = \langle T_1, T_2\rangle$ and $S = \langle S_1, S_2\rangle$, by the inductive hypothesis, $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T_1 \equiv S_1 :: K_1\})$ and $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T_2 \equiv S_2 :: K_2\})$ are both finite.

- Case: $K = L_1 {\Rightarrow} K_2$. If T and S are not type abstractions, then again by Lemma 47 $\mathit{reachable}[F^a_\lambda](\{\Gamma \vdash T \equiv S :: L_1 {\Rightarrow} K_2\})$ is finite. Otherwise, if $T = \lambda X{::}L_1.T_2$ and $S = \lambda X{::}L_1.S_2$, then inductively $\mathit{reachable}[F^a_\lambda](\{\Gamma, X{::}L_1 \vdash T_2 \equiv S_2 :: K_2\})$ is finite.∎

Corollary 49
Membership in νF_λ is decidable.

6 Adding Extensionality

A further extension to the system is extensionality (η-equivalence) for pairs and functions: pairs with equivalent components are equivalent, and pointwise-equivalent functions are equivalent. No changes to the syntax, well-formedness, or reduction is necessary, so all such properties remain unaltered. The only change is to add two rules

$$
\frac{\begin{array}{c} T \text{ and } S \text{ are not both paths} \\ \Gamma \vdash \pi_1\, T \equiv \pi_1\, S :: K_1 \qquad \Gamma \vdash \pi_2\, T \equiv \pi_2\, S :: K_2 \end{array}}{\Gamma \vdash T \equiv S :: K_1 {\times} K_2} \tag{43}
$$

$$\frac{\begin{array}{c}T \text{ and } S \text{ are not both paths}\\ Z \notin FV(T) \cup FV(S)\\ \Gamma, Z{::}L_1 \vdash T\,Z \equiv S\,Z :: K_2\end{array}}{\Gamma \vdash T \equiv S :: L_1{\Rightarrow}K_2} \tag{44}$$

and to omit the (now admissible) structural congruence rules 17 and 34 for pairs and functions. The resulting generating function F_η for equivalence is shown in Figure 7.

Rules 43 and 44 require that at least one of T and S be a non-path. This is necessary for consistency, given that equivalence is defined coinductively. Let $\mathcal{J}$ be the set containing the following three judgments:

$$\begin{array}{c}X{::}{*}{\times}{*}, Y{::}{*}{\times}{*} \vdash X \equiv Y :: {*}{\times}{*}\\ X{::}{*}{\times}{*}, Y{::}{*}{\times}{*} \vdash \pi_1\,X \equiv \pi_1\,Y :: *\\ X{::}{*}{\times}{*}, Y{::}{*}{\times}{*} \vdash \pi_2\,X \equiv \pi_2\,Y :: *\end{array}$$

If we drop the non-path restriction then we would have $\mathcal{J} \subseteq F_\eta(\mathcal{J})$ and so by the Principle of Coinduction all three judgments would be in the greatest fixed point: the first judgment would produce the last two by Rule 37, and the last two would produce the first by Rule 43. None of the three should be provable in any consistent system. A similar problem would arise for functions if the path restriction were removed from Rule 44.

Proofs for most of the expected equational properties then follow straightforwardly, in most cases with few changes.

Proposition 50 (Reflexivity)
If $\Gamma \vdash T :: K$ *then* $\Gamma \vdash T \equiv T :: K$.

Proof: We must show that $I := \{(\Gamma \vdash T \equiv T :: K) \mid \Gamma \vdash T :: K\} \subseteq \nu F_\eta$. By Corollary 3, showing $I \subseteq F_\eta(I) \cup F_\eta(F_\eta(I))$ is sufficient. Let $(\Gamma \vdash T \equiv T :: K) \in I$ be given. We proceed by the possible cases, using Proposition 28:

- Case: $T = \texttt{int}$ and $K = *$. Then $(\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *) \in F_\eta(\emptyset) \subseteq F_\eta(I)$.

- Case: $T = X$ and $K = \Gamma(X)$. Then $(\Gamma \vdash X \equiv X :: K) \in F_\eta(\emptyset) \subseteq F_\eta(I)$.

- Case: $T = \pi_i\,P$, where $\Gamma \vdash P :: K_1{\times}K_2$ and $K = K_i$. Then $(\Gamma \vdash P \equiv P :: K_1{\times}K_2) \in I$, so $(\Gamma \vdash \pi_i\,P \equiv \pi_i\,P :: K) \in F_\eta(I)$.

- Case: $T = P\,S$, where $\Gamma \vdash P :: L{\Rightarrow}K$ and $\Gamma \vdash S :: L$. Then $(\Gamma \vdash P \equiv P :: L{\Rightarrow}K) \in I$ and $(\Gamma \vdash S \equiv S :: L) \in I$, so $(\Gamma \vdash P\,S \equiv P\,S :: K) \in F_\eta(I)$.

- Case: $T = T_1{\rightarrow}T_2$ and $K = *$. Then $\Gamma \vdash T_1 :: *$ and $\vdash T_2 :: *$, so $(\Gamma \vdash T_1 \equiv T_1 :: *) \in I$ and $(\Gamma \vdash T_2 \equiv T_2 :: *) \in I$. Thus $(\Gamma \vdash T_1{\rightarrow}T_2 \equiv T_1{\rightarrow}T_2 :: *) \in F_\eta(I)$.

- Case: $T = \langle T_1, T_2\rangle$ and $K = K_1{\times}K_2$. Then $\vdash T_1 :: K_1$ and $\vdash T_2 :: K_2$, so $(\Gamma \vdash \pi_1\,\langle T_1, T_2\rangle \equiv \pi_1\,\langle T_1, T_2\rangle :: K_1) \in I$ and $(\Gamma \vdash \pi_2\,\langle T_1, T_2\rangle \equiv \pi_2\,\langle T_1, T_2\rangle :: K_2) \in I$. Thus $(\Gamma \vdash \langle T_1, T_2\rangle \equiv \langle T_1, T_2\rangle :: K_1{\times}K_2) \in F_\eta(I)$.

- Case: $T \rightsquigarrow T'$ for some T'. Then by Proposition 30, $\vdash T' :: K$ as well, and so $(\Gamma \vdash T' \equiv T' :: K) \in I$. Thus $(\Gamma \vdash T \equiv T' :: K) \in F_\eta(I)$ so $(\Gamma \vdash T \equiv T :: K) \in F_\eta(F_\eta(I))$.

- Case: $T = \lambda X{::}L_1.T_2$. By inversion of Rule 30 we know that $K = L_1{\Rightarrow}K_2$, so $(\Gamma, Z{::}L_1 \vdash (\lambda X{::}L_1.T_2)\, Z \equiv (\lambda X{::}L_1.T_2)\, Z :: K_2) \in I$. Thus $(\Gamma \vdash \lambda X{::}L_1.T_2 \equiv \lambda X{::}L_1.T_2 :: L_1{\Rightarrow}K_2) \in F_\eta(I)$.

The fold/unfold rule still holds:

Corollary 51 (Fold/Unfold and β-Equivalence)
If $\Gamma \vdash T :: K$ and $T \rightsquigarrow T'$ then $\Gamma \vdash T \equiv T' :: K$.

Proof: Assume $\Gamma \vdash T :: K$ and $T \rightsquigarrow T'$. By Proposition 30, $\Gamma \vdash T' :: K$. By Proposition 50, $(\Gamma \vdash T' \equiv T' :: K) \in \nu F_\lambda$. Hence $(\Gamma \vdash T' \equiv T :: K) \in F_\eta(\nu F_\eta) = \nu F_\eta$. ∎

Proposition 52 (Symmetry)
If $\Gamma \vdash T \equiv S :: K$ then $\Gamma \vdash S \equiv T :: K$.

Proof: By coinduction we can show that $SY(\nu F_\eta) \subseteq F_\eta(SY(\nu F_\eta))$ where $SY(\cdot)$ is the symmetric closure operator. ∎

Proposition 53 (Weakening)
If $\Gamma_1, \Gamma_3 \vdash T \equiv S :: K$ and $dom(\Gamma_1, \Gamma_3) \cap dom(\Gamma_2) = \emptyset$ then $\Gamma_1, \Gamma_2, \Gamma_3 \vdash T \equiv S :: K$.

Proof: The argument for Proposition 35 still applies once we replace the final two cases:

- Case: $K = L_1{\Rightarrow}K_2$, T_1 and T_2 are not both paths, and $\Gamma_1, \Gamma_3, Z{::}L_1 \vdash T_1\, Z \equiv T_2\, Z :: K_2$ where $Z \notin FV(T_1) \cup FV(T_2)$. Without loss of generality we can assume $Z \notin \mathrm{dom}(\Gamma_2)$, in which case we have $(\Gamma_1, \Gamma_2, \Gamma_3, Z{::}L_1 \vdash T_1\, Z \equiv T_2\, Z :: K_2) \in W(\nu F_\eta)$, and so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 \equiv T_2 :: L_1{\Rightarrow}K_2) \in F_\eta(W(\nu F_\eta))$.
- Case: $K = K_1{\times}K_2$, T_1 and T_2 are not both paths, $\Gamma_1, \Gamma_3 \vdash \pi_1\, T_1 \equiv \pi_1\, T_2 :: K_1$, and $\Gamma_1, \Gamma_3 \vdash \pi_2\, T_1 \equiv \pi_2\, T_2 :: K_2$. Then $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash \pi_1\, T_1 \equiv \pi_1\, T_2 :: K_1) \in W(\nu F_\eta)$ and $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash \pi_2\, T_1 \equiv \pi_2\, T_2 :: K_2) \in W(\nu F_\eta)$, so $(\Gamma_1, \Gamma_2, \Gamma_3 \vdash T_1 \equiv T_2 :: K_1{\times}K_2) \in F_\eta(W(\nu F_\eta))$. ∎

Proposition 54 (Congruence for Projections)
If $\Gamma \vdash T \equiv S :: K_1{\times}K_2$ then $\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i$.

Proof: By induction on *height*(T) + *height*(S).

- Case: $(\Gamma \vdash T \equiv S :: K_1{\times}K_2) \in \nu F_\eta$ with $T = P_1$, $S = P_2$. Then immediately $(\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \in F_\eta(\nu F_\eta) = \nu F_\eta$.
- Case: $T \rightsquigarrow T'$, $(\Gamma \vdash T' \equiv S :: K_1{\times}K_2) \in \nu F_\eta$ and $\Gamma \vdash T :: K_1{\times}K_2$. By the inductive hypothesis, $(\Gamma \vdash \pi_i\, T' \equiv \pi_i\, S :: K_i) \in \nu F_\eta$. Then $\Gamma \vdash \pi_i\, T :: K_i$ and $\pi_i\, T \rightsquigarrow \pi_i\, T'$, so $(\Gamma \vdash \pi_i\, T \equiv \pi_i\, S :: K_i) \in F_\eta(\nu F_\eta) = \nu F_\eta$.
- Case: $S \rightsquigarrow S'$, $(\Gamma \vdash T \equiv S' :: K_1{\times}K_2) \in \nu F_\eta$ and $\Gamma \vdash S :: K_1{\times}K_2$. Similar to the previous case.
- Case: $\Gamma \vdash \pi_1\, T \equiv \pi_1\, S :: K_1$ and $\Gamma \vdash \pi_2\, T \equiv \pi_2\, S :: K_2$, with at least one of T or S being a non-path. Immediate. ∎

Proposition 55 (Congruence for Pairs)
If $\Gamma \vdash T_1 \equiv S_1 :: K_1$ *and* $\Gamma \vdash T_2 \equiv S_2 :: K_2$ *then* $\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1 \times K_2$.

Proof: $\Gamma \vdash \pi_i \, \langle T_1, T_2 \rangle :: K_i$ and $\Gamma \vdash \pi_i \, \langle S_1, S_2 \rangle :: K_i$, so $(\Gamma \vdash \pi_i \, \langle T_1, T_2 \rangle \equiv \pi_i \, \langle S_1, S_2 \rangle :: K_i) \in F_\eta(F_\eta(\nu F_\eta)) = \nu F_\eta$. Thus by extensionality $(\Gamma \vdash \langle T_1, T_2 \rangle \equiv \langle S_1, S_2 \rangle :: K_1 \times K_2) \in F_\eta(\nu F_\eta) = \nu F_\eta$. ∎

Proposition 56 (Congruence for Abstractions)
Assume $\Gamma, X{::}L_1 \vdash T \equiv S :: K_2$. *Then* $\Gamma \vdash \lambda X{::}L_1.T \equiv \lambda X{::}L_1.S :: L_1 \Rightarrow K_2$.

Proof: We have $\Gamma, X{::}L_1 \vdash (\lambda X{::}L_1.T) \, X :: K_2$, so by Rule 32, $\Gamma, X{::}L_1 \vdash (\lambda X{::}L_1.T) \, X \equiv S :: K_2$. By a similar application of Rule 33 we have $\Gamma, X{::}L_1 \vdash (\lambda X{::}L_1.T) \, X \equiv (\lambda X{::}L_1.S) \, X :: K_2$. By extensionality, $\Gamma \vdash \lambda X{::}L_1.T \equiv \lambda X{::}L_1.S :: L_1 \Rightarrow K_2$. ∎

Proposition 57 (Functionality)
If $\Gamma_1, Y{::}L, \Gamma_2 \vdash T_1 \equiv S_1 :: K$ *and* $\Gamma_1 \vdash T_2 \equiv S_2 :: L$ *then* $\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K$.

Proof: The same argument for Proposition 36 works, once we replace the last two cases of Part 2.

- Case: $K = K_1 \Rightarrow K_2$, T_1 and S_1 are not both paths, $\Gamma_1 \, Y{::}L, \Gamma_2 \vdash \pi_1 \, T_1 \equiv \pi_1 \, S_1 :: K_1$, and $\Gamma_1, Y{::}L, \Gamma_2 \vdash \pi_2 \, T_1 \equiv \pi_2 \, S_1 :: K_2$. Then we have $(\Gamma_1, \Gamma_2 \vdash \pi_1 \, (\{T_2/Y\}T_1) \equiv \pi_1 \, (\{S_2/Y\}S_1) :: K_1) \in H(\nu F_\eta)$ and $(\Gamma_1, \Gamma_2 \vdash \pi_2 \, (\{T_2/Y\}T_1) \equiv \pi_2 \, (\{S_2/Y\}S_1) :: K_2) \in H(\nu F_\eta)$, and substituting into a non-path always yields a non-path, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: K_1 \times K_2) \in F_\eta(H(\nu F_\eta))$.
- Case: $K = K_1 \Rightarrow L_2$, T_1 and S_1 are not both paths, and $\Gamma_1, Y{::}L, \Gamma_2, Z{::}L_1 \vdash T_1 \, Z \equiv S_1 \, Z :: K_2$, where $Z \notin FV(T_1) \cup FV(S_1)$. Then $(\Gamma_1, \Gamma_2, Z{::}L_1 \vdash (\{T_2/Y\}T_1) \, Z \equiv (\{S_2/Y\}S_1) \, Z :: K_2) \in H(\nu F_\eta)$, and substituting into a non-path always yields a non-path, so $(\Gamma_1, \Gamma_2 \vdash \{T_2/Y\}T_1 \equiv \{S_2/Y\}S_1 :: L_1 \Rightarrow K_2) \in F_\eta(H(\nu F_\eta))$. ∎

Proposition 58 (Congruence for Applications)
If $\Gamma \vdash T_1 \equiv S_1 :: L \Rightarrow K$ *and* $\Gamma \vdash T_2 \equiv S_2 :: L$ *then* $\Gamma \vdash T_1 \, T_2 \equiv S_1 \, S_2 :: K$.

Proof: By induction on $\mathit{height}(T_1 \, T_2) + \mathit{height}(S_1 \, S_2)$, and cases on the forms of T_1 and S_1. In all cases, $\Gamma \vdash T_1 \, T_2 :: K$ and $\Gamma \vdash S_1 \, S_2 :: K$.

- Case: $T_1 = P_1$ and $S_1 = P_2$. Then $(\Gamma \vdash P_1 \, S_1 \equiv P_2 \, S_2 :: K) \in F_\eta(\nu F_\lambda) = \nu F_\lambda$.
- Case: $T_1 \rightsquigarrow T_1'$ and $(\Gamma \vdash T_1' \equiv S_1 :: L \Rightarrow K) \in \nu F_\lambda$. Then $T_1 \, S_1 \rightsquigarrow T_1' \, S_1$ and inductively $(\Gamma \vdash T_1' \, S_1 \equiv T_2 \, S_2 :: K) \in \nu F_\lambda$, so $(\Gamma \vdash T_1 \, S_1 \equiv T_2 \, S_2 :: K) \in F_\eta(\nu F_\lambda) = \nu F_\lambda$.
- Case: $S_1 \rightsquigarrow S_1'$ and $(\Gamma \vdash T_1 \equiv S_1' :: L \Rightarrow K) \in \nu F_\lambda$. Similar.
- Case: $T_1 = \lambda X{::}L.T_1'$, $S_1 = \lambda X{::}L.S_1'$, and $(\Gamma, X{::}L \vdash T_1' \equiv S_1' :: K) \in \nu F_\lambda$. Then $(\Gamma \vdash \{T_2/X\}T_1' \equiv \{S_2/X\}S_1' :: K) \in \nu F_\lambda$ by Proposition 57. Thus $(\Gamma \vdash (\lambda X{::}L.T_1') \, T_2 \equiv (\lambda X{::}L.S_1') \, S_2 :: K) \in F_\eta(F_\eta(\nu F_\lambda)) = \nu F_\lambda$. ∎

Proposition 59 (Congruence for Recursive Types)
If $\Gamma, X{::}L \vdash S_1 \equiv S_2 :: L$ *then* $\Gamma \vdash \mu X{::}L.\, S_1 \equiv \mu X{::}L.\, S_2 :: L$.

Proof: The same argument for Proposition 38 applies, once we replace the pair and function cases in Part 1:

- Case: $K = K_1{\Rightarrow}K_2$, T_1 and T_2 are not both paths, $\Gamma_1\, Y{::}L, \Gamma_2 \vdash \pi_1\, T_1 \equiv \pi_1\, T_2 :: K_1$, and $\Gamma_1, Y{::}L, \Gamma_2 \vdash \pi_2\, T_1 \equiv \pi_2\, T_2 :: K_2$. Then $(\Gamma_1, \Gamma_2 \vdash \pi_1\,(\sigma_1 T_1) \equiv \pi_1\,(\sigma_2 T_2) :: K_1) \in C(\nu F_\eta)$ and $(\Gamma_1, \Gamma_2 \vdash \pi_2\,(\sigma_1 T_1) \equiv \pi_2\,(\sigma_2 T_2) :: K_2) \in C(\nu F_\eta)$. Substituting into a non-path always yields a non-path, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1 \equiv \sigma_2 T_2 :: K_1{\times}K_2) \in F_\eta(C(\nu F_\eta))$.

- Case: $K = K_1{\Rightarrow}L_2$, T_1 and T_2 are not both paths, and $\Gamma_1, Y{::}L, \Gamma_2, Z{::}L_1 \vdash T_1\, Z \equiv T_2\, Z :: K_2$. Then $(\Gamma_1, \Gamma_2, Z{::}L_1 \vdash (\sigma_1 T_1)\, Z \equiv (\sigma_2 T_2)\, Z :: K_2) \in H(\nu F_\eta)$, and substituting into a non-path always yields a non-path, so $(\Gamma_1, \Gamma_2 \vdash \sigma_1 T_1 \equiv \sigma_2 T_2 :: L_1{\Rightarrow}K_2) \in F_\eta(H(\nu F_\eta))$. ∎

The fact that equivalence is closed under reductions is less obvious in the presence of extensionality, but it follows easily once we have a strengthening property (i.e., that we can drop unused variables from the typing context). This is easy to show because all kinds are inhabited, and hence we can apply Proposition 57, where the substitutions have no effect for unused variables.

Proposition 60 (Inhabitation of Kinds)
Define $\overline{T}_K$ *by induction on kinds as follows:*

$$\begin{array}{lcl} \overline{T}_* & := & \texttt{int} \\ \overline{T}_{K_1 \times K_2} & := & \langle \overline{T}_{K_1}, \overline{T}_{K_2} \rangle \\ \overline{T}_{L_1 \Rightarrow K_2} & := & \lambda X{::}L_1.(\overline{T}_{K_2}) \end{array}$$

Then for every kind K *we have* $\vdash \overline{T}_K :: K$.

Proof: By induction on K. ∎

Corollary 61 (Strengthening)
If $\Gamma_1, X{::}L, \Gamma_2 \vdash T \equiv S :: K$ *and* $X \notin FV(T) \cup FV(S)$ *then* $\Gamma_1, \Gamma_2 \vdash T \equiv S :: K$.

Proof: Assume $\Gamma_1, X{::}L, \Gamma_2 \vdash T \equiv S :: K$ and $X \notin FV(T) \cup FV(S)$. By Propositions 60, 50, and 53, $\Gamma_1 \vdash \overline{T}_L \equiv \overline{T}_L :: L$. Thus by Proposition 57, $\Gamma_1, \Gamma_2 \vdash T \equiv S :: K$. ∎

Proposition 62

1. *If* $\Gamma \vdash T' \equiv S' :: K$, $T \rightsquigarrow^* T'$, $S \rightsquigarrow^* S'$, $\Gamma \vdash T :: K$, *and* $\Gamma \vdash S :: K$ *then* $\Gamma \vdash T \equiv S :: K$.

2. *If* $\Gamma \vdash T \equiv S :: K$, $T \rightsquigarrow^* T'$, *and* $S \rightsquigarrow^* S'$ *then* $\Gamma \vdash T' \equiv S' :: K$.

Proof:

1. Same argument as for Proposition 9

2. By induction on K and $height(T) + height(S)$ (ordered lexicographically), and cases on the justification for $(\Gamma \vdash T \equiv S :: K) \in \nu F_\eta = F_\eta(\nu F_\eta)$.

- Case: $T \not\rightsquigarrow$ and $S \not\rightsquigarrow$. Then $T = T'$ and $S = S'$, so the desired result is true by assumption.
- Case: $\Gamma \vdash T \equiv S :: K$ because $T \rightsquigarrow U$, $\Gamma \vdash T :: K$, and $\Gamma \vdash U \equiv S :: K$. If $U \rightsquigarrow^* T'$ then $\Gamma \vdash T' \equiv S' :: K$ follows inductively. Otherwise $T = T'$, and the inductive hypothesis yields $\Gamma \vdash U \equiv S' :: K$, so that $(\Gamma \vdash T' \equiv S' :: K) \in F_\eta(\nu F_\eta) = \nu F_\eta$.
- Case: $\Gamma \vdash T \equiv S :: K$ because $S \rightsquigarrow U$, $\Gamma \vdash S :: K$, and $\Gamma \vdash T \equiv U :: K$. Analogous to the previous case.
- Case: $\Gamma \vdash T \equiv S :: K_1{\times}K_2$ because $\Gamma \vdash \pi_1 T \equiv \pi_1 S :: K_1$, and $\Gamma \vdash \pi_2 T \equiv \pi_2 S :: K_2$. Then $\pi_i T \rightsquigarrow^* \pi_i T'$ and $\pi_i S \rightsquigarrow^* \pi_i S'$, so by the inductive hypothesis $\Gamma \vdash \pi_1 T' \equiv \pi_1 S' :: K_1$ and $\Gamma \vdash \pi_2 T' \equiv \pi_2 S' :: K_2$. If T' and S' are both paths then these equivalences can be in $\nu F_\eta = F_\eta(\nu F_\eta)$ only if $\Gamma \vdash T' \equiv S' :: K_1{\times}K_2$; otherwise, we can use extensionality to conclude that $\Gamma \vdash T' \equiv S' :: K_1{\times}K_2$.
- Case: $\Gamma \vdash T \equiv S :: L_1{\Rightarrow}K_2$ because $\Gamma, Z{::}L_1 \vdash T\,Z \equiv S\,Z :: K_2$, where $Z \notin FV(T) \cup FV(S)$. Then $T\,Z \rightsquigarrow^* T'\,Z$ and $S\,Z \rightsquigarrow^* S'\,Z$, so by the inductive hypothesis $\Gamma, Z{::}L_1 \vdash T'\,Z \equiv S'\,Z :: K_2$. If T' and S' are not both paths then $\Gamma \vdash T' \equiv S' :: L_1{\Rightarrow}K_2$ follows by extensionality. Otherwise it must be that $\Gamma, Z{::}L_1 \vdash T' \equiv S' :: L_1{\Rightarrow}K_2$. By Proposition 30 and Corollary 61, $\Gamma \vdash T' \equiv S' :: L_1{\Rightarrow}K_2$.∎

The greatest difficulty caused by extensionality is proving that equivalence remains transitive. Specifically, it might be that $\Gamma \vdash P_1 \equiv T_2 :: K$ and $\Gamma \vdash T_2 \equiv P_3 :: K$ hold only because of extensionality (where T_2 is not a path), but then the desired conclusion $\Gamma \vdash P_1 \equiv P_3 :: K$ does not itself follow directly from extensionality. We handle this case separately.

Lemma 63
Define $\overline{E}_K$ by induction on K as follows:

$$\begin{array}{lcl} \overline{E}_{*} & := & \bullet \\ \overline{E}_{K_1\times K_2} & := & \overline{E}_{K_1}[\pi_1 \bullet] \\ \overline{E}_{L_1\Rightarrow K_2} & := & \overline{E}_{K_2}[\bullet\ \overline{T}_{L_1}] \end{array}$$

Then for every $\Gamma \vdash T :: K$ we have $\Gamma \vdash \overline{E}_K[T] :: *$.

Proof: By induction on K. ∎

Lemma 64
Assume $\Gamma \vdash P_1 :: K$ and $\Gamma \vdash P_3 :: K$. If $\Gamma \vdash E[P_1] \equiv P_2 :: K'$ and $\Gamma \vdash P_2 \equiv E[P_3] :: K'$ then $(\Gamma \vdash P_1 \equiv P_3 :: K) \in F_\eta(TR(\nu F_\eta))$.

Proof: We proceed by induction on E:

- Case: $E = \bullet$. There are four subcases:
  - Subcase: $P_1 = P_2 = P_3 = \texttt{int}$. Trivial.
  - Subcase: $P_1 = P_2 = P_3 = X$. Trivial.
  - Subcase: $P_1 = \pi_i P_1'$, $P_2 = \pi_i P_2'$, and $P_3 = \pi_i P_3'$ where $\Gamma \vdash P_1' \equiv P_2' :: K_1{\times}K_2$, $\Gamma \vdash P_2' \equiv P_3' :: K_1{\times}K_2$, and $K = K_i$. Then $(\Gamma \vdash P_1' \equiv P_3' :: K_1{\times}K_2) \in TR(\nu F_\eta)$, so $(\Gamma \vdash P_1 \equiv P_2 :: K) \in F_\eta(TR(\nu F_\eta))$.

- Subcase: $P_1 = P_1' \, T_1$, $P_2 = P_2' \, T_2$, and $P_3 = P_3' \, T_3$ where $\Gamma \vdash P_1' \equiv P_2' :: L \Rightarrow K$, $\Gamma \vdash P_2' \equiv P_3' :: L \Rightarrow K$, $\Gamma \vdash T_1 \equiv T_2 :: L$, and $\Gamma \vdash T_2 \equiv T_3 :: L$. Then $(\Gamma \vdash P_1' \equiv P_3' :: L \Rightarrow K) \in \mathit{TR}(\nu F_\eta)$ and $(\Gamma \vdash T_1 \equiv T_3 :: L) \in \mathit{TR}(\nu F_\eta)$, so $(\Gamma \vdash P_1 \equiv P_2 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

- Case: $E = \pi_i \, E'$. Then by definition of F_η it must be that $P_2 = \pi_i \, P_2'$, $\Gamma \vdash E'[P_1] \equiv P_2' :: K_1 \times K_2$, $\Gamma \vdash P_2' \equiv E'[P_3] :: K_1 \times K_2$, and $K = K_i$. By the inductive hypothesis $(\Gamma \vdash P_1 \equiv P_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

- Case: $E = E' \, S$. Then by definition of F_η it must be that $P_2 = P_2' \, S_2$, $\Gamma \vdash E'[P_1] \equiv P_2' :: L \Rightarrow K$, and $\Gamma \vdash P_2' \equiv E'[P_3] :: L \Rightarrow K$. By the inductive hypothesis, $(\Gamma \vdash P_1 \equiv P_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.∎

Corollary 65
If $\Gamma \vdash P_1 \equiv T_2 :: K$ *and* $\Gamma \vdash T_2 \equiv P_3 :: K$ *then* $(\Gamma \vdash P_1 \equiv P_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

Proof: By Lemma 63 and Propositions 54 and 58, $\Gamma \vdash \overline{E}_K[P_1] \equiv \overline{E}_K[T_2] :: *$ and $\Gamma \vdash \overline{E}_K[T_2] \equiv \overline{E}_K[P_3] :: *$. Since $\overline{E}_K[P_1]$ and $\overline{E}_K[P_3]$ are paths, the definition of νF_η and determinacy of reduction ensure that $\overline{E}_K[T_2] \leadsto^* P_2$ with $\Gamma \vdash \overline{E}_K[P_1] \equiv P_2 :: *$ and $\Gamma \vdash P_2 \equiv \overline{E}_K[P_3] :: *$. By Lemma 64, $(\Gamma \vdash P_1 \equiv P_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$. ∎

Proposition 66 (Transitivity)
If $\Gamma \vdash T_1 \equiv T_2 :: K$ *and* $\Gamma \vdash T_2 \equiv T_3 :: K$ *then* $\Gamma \vdash T_1 \equiv T_3 :: K$.

Proof: We must show that νF_η is transitively closed, i.e.,$\mathit{TR}(\nu F_\eta) \subseteq \nu F_\eta$, where

$$\mathit{TR}(\mathcal{J}) := \{(\Gamma \vdash T_1 \equiv T_3 :: K) \mid \exists T_2. \, (\Gamma \vdash T_1 \equiv T_2 :: K), (\Gamma \vdash T_2 \equiv T_3 :: K) \in \mathcal{J}\}$$

is the transitive-closure operator. It suffices to show $\mathit{TR}(\nu F_\eta) \subseteq F_\eta(\mathit{TR}(\nu F_\eta))$. Assume $(\Gamma \vdash T_1 \equiv T_3 :: K) \in \mathit{TR}(\nu F_\eta)$ because $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\eta$ and $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\eta$. We proceed by induction on $\mathit{height}(T_2)$ and cases on the justifications for the two equivalences being assumed.

- Case: T_1 and T_3 are paths. By Corollary 65.

- Case: $T_1 = T_1' {\rightarrow} T_1''$, $T_2 = T_2' {\rightarrow} T_2''$, $T_3 = T_3' {\rightarrow} T_3''$, and $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\eta$ and $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\eta$ because $(\Gamma \vdash T_1' \equiv T_2' :: *)$, $(\Gamma \vdash T_1'' \equiv T_2'' :: *)$, $(\Gamma \vdash T_2' \equiv T_3' :: *)$, $(\Gamma \vdash T_2'' \equiv T_3'' :: *) \in \nu F_\eta$. Then $(\Gamma \vdash T_1' \equiv T_3' :: *) \in \mathit{TR}(\nu F_\eta)$ and $(\Gamma \vdash T_1'' \equiv T_3'' :: *) \in \mathit{TR}(\nu F_\eta)$, so $(\Gamma \vdash T_1' {\rightarrow} T_1'' \equiv T_3' {\rightarrow} T_3'' :: *) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

- Case: $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\eta$ because $(\Gamma \vdash T_1' \equiv T_2 :: K) \in \nu F_\eta$, $T_1 \leadsto T_1'$, and $\Gamma \vdash T_1 :: K$. Then $(\Gamma \vdash T_1' \equiv T_3 :: K) \in \mathit{TR}(\nu F_\eta)$, so $(\Gamma \vdash T_1 \equiv T_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

- Case: $(\Gamma \vdash T_2 \equiv T_3 :: K) \in \nu F_\eta$ because $(\Gamma \vdash T_2 \equiv T_3' :: K) \in \nu F_\eta$, $T_3 \leadsto T_3'$, and $\Gamma \vdash T_3 :: K$. Then $(\Gamma \vdash T_1 \equiv T_3' :: K) \in \mathit{TR}(\nu F_\eta)$, so $(\Gamma \vdash T_1 \equiv T_3' :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

- Case: $(\Gamma \vdash T_1 \equiv T_2 :: K) \in \nu F_\eta$ because $(\Gamma \vdash T_1 \equiv T_2' :: K) \in \nu F_\eta$, $T_2 \leadsto T_2'$, and $\Gamma \vdash T_2 :: K$. By Proposition 62, $(\Gamma \vdash T_2' \equiv T_3 :: K) \in \nu F_\eta$. Then $\mathit{height}(T_2) > \mathit{height}(T_2')$, so the induction hypothesis applies and $(\Gamma \vdash T_1 \equiv T_3 :: K) \in F_\eta(\mathit{TR}(\nu F_\eta))$.

$$
\begin{aligned}
F^a_{\eta-}(\mathcal{J}) := {} & \{\, (\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *) \mid \text{for all } \Gamma \} \\
& \cup \{\, (\Gamma \vdash X \equiv X :: K) \mid \Gamma \vdash X :: K \} \\
& \cup \{\, (\Gamma \vdash \pi_i\, P_1 \equiv \pi_i\, P_2 :: K_i) \mid (\Gamma \vdash P_1 \equiv P_2 :: K_1{\times}K_2) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash P_1\, T_1 \equiv P_2\, T_2 :: K) \mid \\
& \qquad\qquad (\Gamma \vdash P_1 \equiv P_2 :: L{\Rightarrow}K) \in \mathcal{J} \text{ and } (\Gamma \vdash T_1 \equiv T_2 :: L) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash T_1{\to}T_2 \equiv S_1{\to}S_2 :: *) \mid \\
& \qquad\qquad (\Gamma \vdash T_1 \equiv S_1 :: *) \in \mathcal{J} \text{ and } (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J} \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: *) \mid T \rightsquigarrow T' \text{ and } (\Gamma \vdash T' \equiv S :: *) \in \mathcal{J} \text{ and } \Gamma \vdash T :: * \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: *) \mid T \not\rightsquigarrow,\ S \rightsquigarrow S',\ (\Gamma \vdash T \equiv S' :: *) \in \mathcal{J},\ \text{and } \Gamma \vdash S :: * \} \\
& \cup \{\, (\Gamma \vdash T \equiv S :: K_1{\times}K_2) \mid \text{at least one of } T \text{ and } S \text{ is not a path,} \\
& \qquad\qquad (\Gamma \vdash \pi_1\, T \equiv \pi_1\, S :: K_1) \in \mathcal{J},\ \text{and } (\Gamma \vdash \pi_2\, T \equiv \pi_2\, S :: K_2) \in \mathcal{J} \} \\[1ex]
F^a_{\eta}(\mathcal{J}) := {} & F^a_{\eta-}(\mathcal{J}) \\
& \cup \{\, (\Gamma \vdash T \equiv S :: L_1{\Rightarrow}K_2) \mid \text{at least one of } T \text{ and } S \text{ is not a path,} \\
& \qquad\qquad Z \notin FV(T){\cup}FV(S) \text{ and } (\Gamma, Z{::}L_1 \vdash T\, Z \equiv S\, Z :: K_2) \in \mathcal{J} \}
\end{aligned}
$$

Figure 8: Algorithmic Generating Function for Equivalence with Extensionality

- Case: $K = K_1{\times}K_2$ where $\Gamma \vdash \pi_1\, T_1 \equiv \pi_1\, T_2 :: K_1$, $\Gamma \vdash \pi_2\, T_1 \equiv \pi_1\, T_2 :: K_2$, $\Gamma \vdash \pi_1\, T_2 \equiv \pi_1\, T_3 :: K_1$, and $\Gamma \vdash \pi_2\, T_2 \equiv \pi_2\, T_3 :: K_2$, where T_1 and T_3 are not both paths. Then $\Gamma \vdash \pi_1\, T_2 \equiv \pi_1\, T_3 :: K_1$ and $\Gamma \vdash \pi_2\, T_2 \equiv \pi_2\, T_3 :: K_2$ by Proposition 54, so $(\Gamma \vdash \pi_1\, T_1 \equiv \pi_1\, T_3 :: K_1) \in TR(\nu F_\eta)$ and $(\Gamma \vdash \pi_2\, T_1 \equiv \pi_2\, T_3 :: K_2) \in TR(\nu F_\eta)$. By extensionality, $(\Gamma \vdash T_1 \equiv T_3 :: K_2) \in F_\eta(TR(\nu F_\eta))$.

- Case: $K = L_1{\Rightarrow}K_2$, where $\Gamma, Z{::}L_1 \vdash T_1\, Z \equiv T_2\, Z :: K_2$, $Z \notin FV(T_1) \cup FV(T_2)$, and T_1 and T_3 are not both paths. Then $\Gamma, Z{::}L_1 \vdash T_2\, Z \equiv T_3\, Z :: K_2$ by Propositions 53 and 54, so $(\Gamma, Z{::}L_1 \vdash T_1\, Z \equiv T_3\, Z :: K_2) \in TR(\nu F_\eta)$. By extensionality, $(\Gamma \vdash T_1 \equiv T_3 :: K_2) \in F_\eta(TR(\nu F_\eta))$.

6.1 Decidability

The corresponding algorithmic (invertible) generating function for equivalence is shown in Figure 8. The function F^a_η differs from F_η in only two respects: the usual asymmetric restriction for the reduction cases, and the restriction of the reduction cases to proper types of kind $*$. Since application and projection commute with reduction, we can choose to apply equal projections or applications to non-paths and then reduce only once we reach kind $*$.

As for F_π and F^a_π before, the two functions have the same fixed point:

Proposition 67
$\nu F^a_\eta = \nu F_\eta$.

Proof: Since pointwise $F^a_\eta \subseteq F_\eta$, we immediately have $\nu F^a_\eta \subseteq \nu F_\eta$. To show that $\nu F_\eta \subseteq \nu F^a_\eta$ it suffices to show that $\nu F_\eta \subseteq F^a_\eta(\nu F_\eta)$. Assume $\Gamma \vdash T \equiv S :: K \in \nu F_\eta$. We proceed by induction on K;

- Case: $(\Gamma \vdash T \equiv S :: K) \in \nu F_\eta$ because $\Gamma \vdash T :: K$, $T \rightsquigarrow T'$, and $(\Gamma \vdash T' \equiv S :: K) \in \nu F_\eta$. There are three subcases:
  - Subcase: $K = *$. Then $(\Gamma \vdash T \equiv S :: K) \in F^a_\eta(\nu F_\eta)$.
  - Subcase: $K = K_1{\times}K_2$. Then $\Gamma \vdash \pi_1\, T \equiv \pi_1\, S :: K_1$ and $\Gamma \vdash \pi_2\, T \equiv \pi_2\, S :: K_2$ by Proposition 54. Since T can be reduced it is not a path, so $(\Gamma \vdash T \equiv S :: K_1{\times}K_2) \in F^a_\eta(\nu F_\eta)$.
  - Subcase: $K = L_1{\Rightarrow}K_2$. Let $Z \notin FV(T) \cup FV(S)$. Then $\Gamma, Z{::}L_1 \vdash T\, Z \equiv S\, Z :: K_2$ by Propositions 53 and 58. Since T can be reduced it is not a path, so $(\Gamma \vdash T \equiv S :: L_1{\Rightarrow}K_2) \in F^a_\eta(\nu F_\eta)$.
- Case: $(\Gamma \vdash T \equiv S :: K) \in \nu F_\eta$ because $\Gamma \vdash S :: K$, $S \rightsquigarrow S'$, and $(\Gamma \vdash T \equiv S' :: K) \in \nu F_\eta$. There are two subcases:
  - Subcase: $T \rightsquigarrow T'$. Then $\Gamma \vdash T' \equiv S :: K$ by Proposition 62, so $(\Gamma \vdash T \equiv S :: K) \in F^a_\eta(\nu F_\eta)$.
  - Subcase: $T \not\rightsquigarrow$. The same general argument for the previous part applies.
- All other cases follow directly by induction.

Again F^a_η is invertible but not immediately finite-state because predecessor judgments may again have different contexts. Therefore, we consider $F^a_{\eta-}$, which never changes typing contexts.

Because of extensionality for pairs, predecessor types are not longer guaranteed to be top-down subterms of the original pair of types; for example, the predecessor of $(X{::}{*}{\times}{*} \vdash X \equiv \langle \pi_1\, X, \pi_2\, X\rangle :: {*}{\times}{*})$ contains $(X{::}{*}{\times}{*} \vdash \pi_1\, X \equiv \pi_1\, \langle \pi_1\, X, \pi_2\, X\rangle :: *)$. However, we can show that any non-subterm types are created by projections from top-down subterms.

Definition 68
We say that $T \mathrel{{}_\pi{\sqsubseteq}} S$ if $T = E[T']$ and $T' \sqsubseteq S$ for some T and E where E contains only projections, i.e., if T is a projection from a top-down subterm of S.

Lemma 69

1. *If $U \sqsubseteq E[T]$ and $T \sqsubseteq S$ where E contains only projections, then $U = E'[U']$ for some $U' \sqsubseteq S$ where E' contains only projections.*
2. *If $U \sqsubseteq T_1$ and $T_1 \mathrel{{}_\pi{\sqsubseteq}} S$ then $U \mathrel{{}_\pi{\sqsubseteq}} S$.*
3. *If $U \mathrel{{}_\pi{\sqsubseteq}} T_1$ and $T_1 \mathrel{{}_\pi{\sqsubseteq}} S$ then $U \mathrel{{}_\pi{\sqsubseteq}} S$.*

Proof:

1. By induction on $U \sqsubseteq E[T]$.
   - Case: $U = E[T]$. Take $E' = E$ and $U' = T$.
   - Case: $E = \bullet$ and $U \sqsubseteq T$. Take $E' = E$ and $U' = T$.
   - Case: $E = \pi_i\, E_1$ and $U \sqsubseteq E_1[T]$. By the inductive hypothesis.
   - Case: $E = E_1[\pi_i\, \bullet]$, $T = \langle T_1, T_2\rangle$, and $U \sqsubseteq E_1[T_i]$. By the inductive hypothesis, since by transitivity any top-down subterm of T_i is a top-down subterm of S.

- Case: $U \sqsubseteq E[T']$ where $T \leadsto T'$. By the inductive hypothesis (since $T' \sqsubseteq T \sqsubseteq S$).

2. This is a rewording of Part 1.

3. By Part 2, since adding more projections to a projection doesn't matter. ∎

Proposition 70

1. *If* $(\Gamma' \vdash T' \equiv S' :: K') \in \mathit{pred}[F^a_{\eta-}](\Gamma \vdash T \equiv S :: K)$ *then* $\Gamma = \Gamma'$ *and* $T' \ {}_{\pi}\!\sqsubseteq T$ *and* $S' \ {}_{\pi}\!\sqsubseteq S$.

2. *If* $(\Gamma'' \vdash T'' \equiv S'' :: K'') \in \mathit{reachable}[F^a_{\eta-}](\{\Gamma \vdash T \equiv S :: K\})$ *then* $\Gamma = \Gamma''$ *and* $T'' \ {}_{\pi}\!\sqsubseteq T$ *and* $S'' \ {}_{\pi}\!\sqsubseteq S$.

Proof:

1. By definition of $F^a_{\eta-}$.

2. By induction, using Part 1 and Lemma 69.∎

Corollary 71
$F^a_{\eta-}$ *is finite-state.*

Proof: There are only finitely many top-down subterms, and for each there are only finitely many well-formed projections possible. ∎

Proposition 72
F^a_{η} *is finite-state.*

Proof: Exactly analogous to the proof for F^a_{λ}. ∎

Corollary 73
Membership in νF_η *is decidable.*

7 Unfold/Unfold

Both the TILT and FLINT compilers for Standard ML have used an alternate definition of equivalence modifying the standard fold/unfold rule [LS98, VDP+03]. This corresponds to replacing Rules 3 and 4 by the single rule

$$\frac{\Gamma \vdash \{\mu X{::}K.\,T/T\}X \equiv \{\mu X{::}K.\,S/S\}X :: K}{\Gamma \vdash \mu X{::}K.\,T \equiv \mu X{::}K.\,S :: K} \tag{45}$$

That is, two recursive types are equal if and only if their unfoldings are equal. Further, this change means that recursive types are equal only to other recursive types. (Hence, such a system generally requires a more limited set of `fold` and `unfold` explicit coercions witnessing an isomorphism between a recursive type and unfoldings (non-recursive). This rule comes about because a recursive `datatype` specification in Standard ML actually gives the unfolding of a recursive type, rather than the recursive type itself [VDP+03]. If the actual definition of a recursive type is to be predictable

$$\begin{array}{rl}
F_{uu}(\mathcal{J}) := & \{ (\Gamma \vdash \texttt{int} \equiv \texttt{int} :: *) \mid \text{for all } \Gamma\} \\
\cup & \{ (\Gamma \vdash P \equiv P :: K) \mid \Gamma \vdash P :: K\} \\
\cup & \{ (\Gamma \vdash T_1{\rightarrow}T_2 \equiv S_1{\rightarrow}S_2 :: *) \mid (\Gamma \vdash T_1 \equiv S_1 :: *), (\Gamma \vdash T_2 \equiv S_2 :: *) \in \mathcal{J}\} \\
\cup & \{ (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow_\pi T' \text{ and } (\Gamma \vdash T' \equiv S :: K) \in \mathcal{J} \text{ and } \Gamma \vdash T :: K\} \\
\cup & \{ (\Gamma \vdash T \equiv S :: K) \mid S \rightsquigarrow_\pi S' \text{ and } (\Gamma \vdash T \equiv S' :: K) \in \mathcal{J} \text{ and } \Gamma \vdash S :: K\} \\
\cup & \{ (\Gamma \vdash T \equiv S :: K) \mid T \rightsquigarrow_\mu T' \text{ and } S \rightsquigarrow_\mu S' \text{ and } (\Gamma \vdash T' \equiv S' :: K) \in \mathcal{J} \text{ and } \Gamma \vdash T :: K\} \\
\cup & \{ (\Gamma \vdash \langle T_1, T_2\rangle \equiv \langle S_1, S_2\rangle :: K_1{\times}K_2) \mid (\Gamma \vdash T_1 \equiv S_1 :: K_1), (\Gamma \vdash T_2 \equiv S_2 :: K_2) \in \mathcal{J}\}
\end{array}$$

Figure 9: Generating Function for Unfold/Unfold

(up to equivalence) knowing only its unfolding, it suffices to require that types are equal whenever their unfoldings are.

The corresponding generating function F_{uu} is shown in Figure **??**, where $T \rightsquigarrow_\pi S$ is reduction restricted to projections from pairs, and $T \rightsquigarrow_\mu S$ is reduction restricted to only unfolding steps. This definition is finite-state for exactly the same reason as F_π, and can be made invertible in the same fashion.

The coinductive equivalence νF_{uu}, can be defined in terms of equivalence of the (potentially) infinite trees resulting from unfolding types replacing the rule $\mu X{::}K.\, T = \{\mu X{::}K.\, T/X\}T$, by the rule $\mu X{::}K.\, T = \mu \_{::}K.\, \{\mu X{::}K.\, T/X\}T$. where $\_$ represents a bound type variable that is never referenced. Thus, we have

$$\begin{array}{l}
\mu X{::}*.\, \texttt{int}{\rightarrow}X \\
= \mu\_{::}*.\, \texttt{int}{\rightarrow}\mu X{::}*.\, \texttt{int}{\rightarrow}X \\
= \mu\_{::}*.\, \texttt{int}{\rightarrow}\mu\_{::}*.\, \texttt{int}{\rightarrow}\mu X{::}*.\, \texttt{int}{\rightarrow}X \\
= \cdots
\end{array}$$

whose limit is an alternating sequence of $\mu\_$ and $\texttt{int}{\rightarrow}$. Also,

$$\begin{array}{l}
\mu X{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}X \\
= \mu\_{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}\mu X{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}X \\
= \mu\_{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}\mu\_{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}\mu X{::}*.\, \texttt{int}{\rightarrow}\texttt{int}{\rightarrow}X \\
= \cdots
\end{array}$$

whose limit has a μ after every second $\texttt{int}{\rightarrow}$. Hence, under the more restrictive unfold/unfold equivalence, these two types are not equal, even coinductively.

There still exist non-trivial equivalences generated by the unfold/unfold rule. For example, the type $\mu X_1{::}*.\, \mu X_2{::}*.\, X_1{\rightarrow}X_1$ is coinductively equal to the type $\mu X_1{::}*.\, \mu X_2{::}*.\, X_1{\rightarrow}(\mu Y{::}*.\, X_2)$.

8 Related Work

The most interesting previous extensions of coinductive equivalence or subtyping with fold/unfold rules are those involving type isomorphisms [PZ00, DPR05]. These are motivated by the problem of searching libraries of code (specifically class interfaces, which can be self-referential) ignoring the order or currying/uncurrying of arguments.

Studies of *inductive* equivalence with β-equivalence and fold/unfold are more common than the coinductive case. For example, Bruce's [Bru02] target for object encodings includes the fold/unfold rule in an inductively-defined type equivalence relation. Statman [Sta02] has described a decidability proof for an *inductive* definition of equivalence for a simply-typed lambda calculus with $\beta\eta$ for functions and a fixed-point combinator $Y : (0 \to 0) \to 0$. The proof does not directly apply to pairs encoded as functions, because the Y combinator cannot be used to recursively define functions.

9 Conclusion and Future Work

We have defined some interesting extensions of the usual coinductive theory for recursive types, up to $\beta\eta$-equivalence with first-order type operations and recursively-defined pairs. In all cases we have shown that equivalence is well-behaved and decidable. The results presented here are enough to allow a straightforward proof of type soundness ("well-typed programs don't go wrong") for a term language with this type system.

Though we have studied type equivalence, we conjecture that the ideas in this paper should be directly applicable to subtyping as well.

The issue with bound variables that caused us to consider only first-order type operators is exactly the same issue that Colazzo and Ghelli [CG99] encountered in combining recursive types with bounded polymorphism in Kernel Fun. They showed decidability of equivalence and subtyping for simple recursive types extended with bounded universal quantifiers. More recently the work of Gauthier and Pottier [GP04] showed that through a transformation analogous to DeBruijn numbering, universally-bound variables could be eliminated while preserving equivalence of recursive types; perhaps these ideas could be adapted here to remove the restriction of type operators to first-order.

Finally, this paper studies only a syntactic equational theory. We have not provided a semantic model for our types, or even formally related our types to μ-free infinite trees. An open question is how best to do so because a reduction relation on infinite trees may be necessary. If so, the restriction to contractive types, guaranteeing termination of weak head reduction, could make the trees involved easier to work with than arbitrary infinite lambda terms.

References

[AC93] Roberto M. Amadio and Luca Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4), September 1993.

[AF96] Martín Abadi and Marcelo P. Fiore. Syntactic considerations on recursive types. In *IEEE Symp. on Logic in Computer Science (LICS'96)*, pages 242–252, 1996.

[BCP99] Kim B. Bruce, Luca Cardelli, and Benjamin C. Pierce. Comparing object encodings. *Information and Computation*, (155):108–133, 1999.

[BH97] Michael Brandt and Fritz Henglein. Coinductive axiomatization of recursive type equality and subtyping. In *Third International Conf. on Typed Lambda Calculi and Applications (TLCA '97)*, volume 1210, pages 63–81, 1997.

[Bru02] Kim B. Bruce. *Foundations of Object-Oriented Languages.* MIT Press, 2002.

[CG99] Dario Colazzo and Giorgio Ghelli. Subtyping recursive types in Kernel Fun. In *IEEE Symp. on Logic in Computer Science (LICS '99)*, pages 137–146, 1999.

[CHP99] Karl Crary, Robert Harper, and Sidd Puri. What is a recursive module? In *ACM SIGPLAN '99 Conference on Programming Language Design and Implementation (PLDI '99)*, pages 50–63, 1999.

[CS02] Gregory D. Collins and Zhong Shao. Intensional analysis of higher-kinded recursive types. Technical Report YALEU/DCS/TR-1240, Department of Computer Science, Yale University, 2002.

[DPR05] Roberto Di Cosmo, François Pottier, and Didier Rémy. Subtyping recursive types modulo associative commutative products. In *Seventh International Conference on Typed Lambda Calculi and Applications (TLCA '05)*, 2005.

[Fio04] Marcelo Fiore. Isomorphisms of generic recursive polynomial types. In *ACM Symposium on Principles of Programming Languages (POPL '04)*, pages 77–88, 2004.

[GLP02] Vladimir Gapeyev, Michael Levin, and Benjamin Pierce. Recursive subtyping revealed. *Journal of Functional Programming*, 12(6):511–548, 2002.

[GP04] Nadji Gauthier and François Pottier. Numbering matters: First-order canonical forms for second-order recursive types. In *Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming (ICFP '04)*, pages 150–161, 2004.

[HS97a] Robert Harper and Christopher Stone. An interpretation of Standard ML in type theory. Technical Report CMU-CS-97-147, School of Computer Science, Carnegie Mellon University, 1997.

[HS97b] Robert Harper and Christopher Stone. A type-theoretic account of Standard ML 1996 (version 2). Technical Report CMU-CS-96-136R, School of Computer Science, Carnegie Mellon University, 1997.

[LS98] Christopher League and Zhong Shao. Formal semantics of the FLINT intermediate language. Technical Report Yale-CS-TR 1171, Department of Computer Science, Yale University, 1998.

[PZ00] Jens Palsberg and Tian Zhao. Efficient and flexible matching of recursive types. In *IEEE Symp. on Logic in Computer Science (LICS '00)*, pages 388–398, 2000.

[Sta02] Rick Statman. On the Lambda Y calculus. In *IEEE Symp. on Logic in Computer Science (LICS '02)*, pages 159–166, 2002.

[VDP$^+$03] Joseph C. Vanderwaart, Derek R. Dreyer, Leaf Petersen, Karl Crary, and Robert Harper. Typed compilation of recursive datatypes. In *International Workshop on Types in Language Design and Implementation (TLDI '03)*, pages 98–108, 2003.

[Win93] Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction.* MIT Press, 1993.

A Generalizing Amadio-Cardelli

Gapeyev et al. [GLP02] define several algorithms for determining membership in greatest fixed points of finite-state functions. As an exercise in using properties of coinduction, we show one more algorithm. This variant was discussed in the specific case of subtyping recursive types, but not generalized (because there are similar but more efficient possibilities).

Definition 74
The generalized Amadio-Cardelli algorithm *is defined by*

$$\begin{array}{rl} \mathit{gfp}^{ac}[F](\mathcal{A}, x) := & \textit{if } x \in \mathcal{A} \textit{ then true} \\ & \textit{else if support}[F](x) = \uparrow \textit{ then false} \\ & \textit{else } \bigwedge_{y \in \mathcal{A}'} \mathit{gfp}^{ac}[F](\mathcal{A} \cup \{x\}, y) \\ & \qquad\quad \textit{where } \mathcal{A}' := \mathit{support}[F](x) \end{array}$$

The following lemma appears in Gapeyev et al. [GLP02]:

Lemma 75
Assume $F : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ is invertible. Then $\mathcal{A} \subseteq F(\mathcal{B})$ if and only if $\mathit{support}[F](\mathcal{A}) \subseteq \mathcal{B}$.

Then,

Proposition 76
If $F : U_{eq} \to U_{eq}$ is finite-state and $\mathit{gfp}^{ac}[F](\mathcal{A}, x) = \mathit{true}$ then $x \in \nu F^{+\mathcal{A}}$.

Proof: By induction on the execution of the algorithm. By inspection, there are two cases in which the algorithm can return *true*:

- Case: $x \in \mathcal{A}$. Then $x \in \mathcal{A} \subseteq \mathcal{A} \cup F(\nu F^{+\mathcal{A}}) = F^{+\mathcal{A}}(\nu F^{+\mathcal{A}}) = \nu F^{+\mathcal{A}}$, which exists since $F^{+\mathcal{A}}$ is monotone.

- Case: $\bigwedge_{y \in \mathit{support}[F](x)} \mathit{gfp}^{ac}[F](\mathcal{A} \cup \{x\}, y)$. By the inductive hypothesis, for all $y \in \mathit{support}[F](x)$ we have $y \in \nu F^{+\mathcal{A} \cup \{x\}}$. That is, $\mathit{support}[F](x) \subseteq \nu F^{+\mathcal{A} \cup \{x\}}$. By Lemma 75, $x \in F(\nu F^{+\mathcal{A} \cup \{x\}}) \subseteq F^{+\mathcal{A}}(\nu F^{+\mathcal{A} \cup \{x\}})$. Therefore $\nu F^{+\mathcal{A} \cup \{x\}} = F^{+\mathcal{A} \cup \{x\}}(\nu F^{+\mathcal{A} \cup \{x\}}) = F^{+\mathcal{A}}(\nu F^{+\mathcal{A} \cup \{x\}}) \cup \{x\} = F^{+\mathcal{A}}(\nu F^{+\mathcal{A} \cup \{x\}})$. By Coinduction we have $\nu F^{+\mathcal{A} \cup \{x\}} \subseteq \nu G$, and therefore $x \in F^{+\mathcal{A}}(\nu F^{+\mathcal{A} \cup \{x\}}) \subseteq F^{+\mathcal{A}}(\nu G) = \nu G$. ∎

Proposition 77
Assume $F : U_{eq} \to U_{eq}$ is invertible. If $\mathit{gfp}^{ac}[F](\mathcal{A}, x) = \mathit{false}$ then $x \notin \nu F^{+\mathcal{A}}$.

Proof: By induction on the execution of the algorithm. Again there are two cases in which the algorithm might yield the answer *false*.

- Case: $x \notin \mathcal{A}$ and $\mathit{support}[F](x) = \uparrow$. Since F is invertible, there is no X such that $x \in F(X)$. Thus $x \notin F(\nu F^{+\mathcal{A}})$ and so $x \notin \mathcal{A} \cup F(\nu F^{+\mathcal{A}}) = \nu F^{+\mathcal{A}}$.

- Case: $x \notin \mathcal{A}$ and $\mathit{gfp}^{ac}[F](\mathcal{A} \cup \{x\}, y) = \mathit{false}$ for some $y \in \mathit{support}[F](x)$. By the inductive hypothesis, $y \notin \nu F^{+\mathcal{A} \cup \{x\}}$. That is, $\mathit{support}[F](x) \not\subseteq \nu F^{+\mathcal{A} \cup \{x\}}$. By Lemma 75, $x \notin F(\nu F^{+\mathcal{A} \cup \{x\}})$ and so using monotonicity, $\nu F^{+\mathcal{A}} \subseteq \nu F^{+\mathcal{A} \cup \{x\}}$ implies $x \notin \mathcal{A} \cup F(\nu F^{+\mathcal{A}}) = \nu F^{+\mathcal{A}}$. ∎

Proposition 78
If $F : U_{eq} \to U_{eq}$ is finite-state then $gfp^{ac}[F](\mathcal{A}, x)$ terminates for all $\mathcal{A} \subseteq U_{eq}$ and $x \in U_{eq}$.

Proof: At each recursive call the set $\mathcal{A}$ strictly increases, but only by an element of $reachable[F](x)$, and the latter set is finite by assumption. Further, the branching factor is finite since the conjunction over the support set again is checking only reachable elements. ∎

Corollary 79
If $F : U_{eq} \to U_{eq}$ is finite-state then

1. $gfp^{ac}[F](\emptyset, x) = true$ *iff* $x \in \nu F$
2. $gfp^{ac}[F](\emptyset, x) = false$ *iff* $x \notin \nu F$