MPRI 2.4, Functional programming and type systems
Metatheory of System F

Didier Rémy
Plan of the course

Simply typed lambda-calculus

Metatheory of System F

ADTs, Recursive types, Existential types, GATDs

Going higher order with $F^\omega$!

Logical relations

Side effects, References, Value restriction

Type reconstruction

Overloading
Logical relations and parametricity
Contents

- Introduction
- Normalization of $\lambda_{st}$
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
What are logical relations?

So far, most proofs involving terms have proceeded by induction on the structure of *terms* (or, equivalently, on *typing derivations*).

Logical relations are relations between well-typed terms defined inductively on the structure of *types*. They allow proofs between terms by induction on the structure of *types*.

**Unary relations**

- Unary relations are predicates on expressions (or sets of expressions)
- They can be used to prove type safety and strong normalization

**Binary relations**

- Binary relations relate pairs of expressions of related types
- They can be used to prove equivalence of programs and non-interference properties.

*Logical relations are a common proof method for programming languages.*
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

What can do a term of type $\forall \alpha. \alpha \rightarrow \text{int}$?
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

What can do a term of type $\forall \alpha. \alpha \to \text{int}$?

- the function cannot examine its argument

so?
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

What can do a term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

- the function cannot examine its argument
- it always returns the same integer

for example?
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

What can do a term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

- the function cannot examine its argument
- it always returns the same integer
- $\lambda x. n$,
  $\lambda x. (\lambda y. y) \ n$,
  $\lambda x. (\lambda y. n) \ x$.
  etc.

What do they all have in common?
In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

What can do a term of type $\forall \alpha. \alpha \to int$?

- the function cannot examine its argument
- it always returns the same integer
- $\lambda x. n$,
  $\lambda x. (\lambda y. y) \ n$,
  $\lambda x. (\lambda y. n) \ x$.
  etc.
- they are all $\beta\eta$-equivalent to the term $\lambda x. n$
Parametricity?  Inhabitants of polymorphic types

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \to \text{int}$?

▷ behaves as $\lambda x. n$
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

▷ behaves as $\lambda x. n$

A term $a$ of type $\forall \alpha. \alpha \rightarrow \alpha$?
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

▷ behaves as $\lambda x. n$

A term $a$ of type $\forall \alpha. \alpha \rightarrow \alpha$?

▷ behaves as $\lambda x. x$
In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

▷ behaves as $\lambda x. n$

A term $a$ of type $\forall \alpha. \alpha \rightarrow \alpha$?

▷ behaves as $\lambda x. x$

A term type $\forall \alpha \beta. \alpha \rightarrow \beta \rightarrow \alpha$?
Parametricity? Inhabitants of polymorphic types

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \rightarrow \text{int}$?

- behaves as $\lambda x. n$

A term $a$ of type $\forall \alpha. \alpha \rightarrow \alpha$?

- behaves as $\lambda x. x$

A term type $\forall \alpha \beta. \alpha \rightarrow \beta \rightarrow \alpha$?

- behaves as $\lambda x. \lambda y. x$

A term type $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$?

?
Parametricity?

In the presence of polymorphism (and in the absence of effects), a type can reveal a lot of information about the terms that inhabit it.

A term of type $\forall \alpha. \alpha \rightarrow \text{int}$?
- behaves as $\lambda x. n$

A term $a$ of type $\forall \alpha. \alpha \rightarrow \alpha$?
- behaves as $\lambda x. x$

A term type $\forall \alpha \beta. \alpha \rightarrow \beta \rightarrow \alpha$?
- behaves as $\lambda x. \lambda y. x$

A term type $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$?
- behaves either as $\lambda x. \lambda y. x$ or $\lambda x. \lambda y. y$
Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

\[ \text{whoami} : \forall \alpha. \text{list}\, \alpha \rightarrow \text{list}\, \alpha \]
Pametricity

Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

\[
\text{whoami} : \forall \alpha. \text{list } \alpha \rightarrow \text{list } \alpha
\]

▷ The length of the result depends only on the length of the argument
Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

```
whoami : ∀α. list α → list α
```

▷ The length of the result depends only on the length of the argument
▷ All elements of the results are elements of the argument
Similarly, the type of a polymorphic function may also reveal a "free theorem" about its behavior!

What properties may we learn from a function

\[ \text{whoami} : \forall \alpha. \text{list} \alpha \to \text{list} \alpha \]

- The length of the result depends only on the length of the argument
- All elements of the results are elements of the argument
- The choice \((i, j)\) of pairs such that \(i\)-th element of the result is the \(j\)-th element of the argument does not depend on the element itself.
Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

\[
\text{whoami} : \forall \alpha. \text{list } \alpha \rightarrow \text{list } \alpha
\]

- The length of the result depends only on the length of the argument
- All elements of the results are elements of the argument
- The choice \((i, j)\) of pairs such that \(i\)-th element of the result is the \(j\)-th element of the argument does not depend on the element itself.
- The function is preserved by a transformation of its argument that preserves the shape of the argument

\[
\forall f, x, \quad \text{whoami} \,(\text{map } f \, x) = \text{map } f \,(\text{whoami } x)
\]
Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

\[
\text{whoami} : \forall \alpha. \text{list}\ \alpha \to \text{list}\ \alpha
\]

What property may we learn for the list sorting function?

\[
\text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list}\ \alpha \to \text{list}\ \alpha
\]
Pametricity

Similarly, the type of a polymorphic function may also reveal a "free theorem" about its behavior!

What properties may we learn from a function

$$\text{whoami} : \forall \alpha. \text{list } \alpha \to \text{list } \alpha$$

What property may we learn for the list sorting function?

$$\text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha$$

If $f$ is order-preserving, then sorting commutes with $\text{map } f$

$$(\forall x, y, \text{ cmp } (f \ x) \ (f \ y) = \text{ cmp } \ x \ y) \implies$$

$$\forall \ell, \text{ sort } \text{ cmp } (\text{map } f \ \ell) = \text{map } f \ (\text{sort } \text{ cmp } \ell)$$
Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

$$\text{whoami} : \forall \alpha. \text{list } \alpha \to \text{list } \alpha$$

What property may we learn for the list sorting function?

$$\text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha$$

If $f$ is order-preserving, then sorting commutes with $\text{map } f$

$$(\forall x, y, \text{ cmp}_2 (f x) (f y) = \text{ cmp}_1 x y) \implies$$

$$\forall \ell, \text{ sort } \text{ cmp}_2 (\text{map } f \ell) = \text{map } f (\text{sort } \text{ cmp}_1 \ell)$$
Similarly, the type of a polymorphic function may also reveal a "free theorem" about its behavior!

What properties may we learn from a function

$$\text{whoami} : \forall \alpha. \text{list } \alpha \rightarrow \text{list } \alpha$$

What property may we learn for the list sorting function?

$$\text{sort} : \forall \alpha. (\alpha \rightarrow \alpha \rightarrow \text{bool}) \rightarrow \text{list } \alpha \rightarrow \text{list } \alpha$$

If $f$ is order-preserving, then sorting commutes with $\text{map } f$

$$(\forall x, y, \text{cmp}_2 (f x) (f y) = \text{cmp}_1 x y) \implies$$

$$\forall \ell, \text{sort } \text{cmp}_2 (\text{map } f \ell) = \text{map } f (\text{sort } \text{cmp}_1 \ell)$$

Application:

- If $\text{sort}$ is correct on lists of integers, then it is correct on any list
- May be useful to reduce testing.
Similarly, the type of a polymorphic function may also reveal a "free theorem" about its behavior!

What properties may we learn from a function

\[ \text{whoami} : \forall \alpha. \text{list} \alpha \rightarrow \text{list} \alpha \]

What property may we learn for the list sorting function?

\[ \text{sort} : \forall \alpha. (\alpha \rightarrow \alpha \rightarrow \text{bool}) \rightarrow \text{list} \alpha \rightarrow \text{list} \alpha \]

If \( f \) is order-preserving, then sorting commutes with \( \text{map} f \)

\[ \forall x, y, \text{cmp}_2 (f x) (f y) = \text{cmp}_1 x y \implies \forall \ell, \text{sort} \text{cmp}_2 (\text{map} f \ell) = \text{map} f (\text{sort} \text{cmp}_1 \ell) \]

Note that there are many other inhabitants of this type, but they all satisfy this free theorem.

**Can you give a few?**
Pametricity

Similarly, the type of a polymorphic function may also reveal a “free theorem” about its behavior!

What properties may we learn from a function

\[ \text{whoami} : \forall \alpha. \text{list } \alpha \to \text{list } \alpha \]

What property may we learn for the list sorting function?

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha \]

If \( f \) is order-preserving, then sorting commutes with \( \text{map } f \)

\[
(\forall x, y, \ cmp_2 (f \ x) (f \ y) = cmp_1 \ x \ y) \implies \forall \ell, \ \text{sort } \ cmp_2 (\text{map } f \ \ell) = \text{map } f (\text{sort } \ cmp_1 \ \ell)
\]

Note that there are many other inhabitants of this type, but they all satisfy this free theorem. (e.g., a function that sorts in reverse order, or a function that removes (or adds) duplicates).
Parametricity

This phenomenon was studied by Reynolds [1983] and by Wadler [1989; 2007], among others. Wadler’s paper contains the ‘free theorem’ about the list sorting function.

An account based on an operational semantics is offered by Pitts [2000]. Bernardy et al. [2010] generalize the idea of testing polymorphic functions to arbitrary polymorphic types and show how testing any function can be restricted to testing it on (possibly infinitely many) particular values at some particular types.
Contents

- Introduction
- **Normalization of $\lambda_{st}$**
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
Normalization of simply-typed $\lambda$-calculus

Types usually ensure termination of programs—as long as neither types nor terms contain any form of recursion.
Normalization of simply-typed $\lambda$-calculus

Types usually ensure termination of programs—as long as neither types nor terms contain any form of recursion.

Even if one wishes to add recursion explicitly later on, it is an important property of the design that non-termination is originating from the constructions introduced especially for recursion and could not occur without them.
Normalization of simply-typed $\lambda$-calculus

Types usually ensure termination of programs—as long as neither types nor terms contain any form of recursion.

Even if one wishes to add recursion explicitly later on, it is an important property of the design that non-termination is originating from the constructions introduced especially for recursion and could not occur without them.

The simply-typed $\lambda$-calculus is also lifted at the level of types in richer type systems such as $F^\omega$; then, the decidability of type-equality depends on the termination of the reduction at the type level.
Normalization of simply-typed \( \lambda \)-calculus

Types usually ensure termination of programs—as long as neither types nor terms contain any form of recursion.

Even if one wishes to add recursion explicitly later on, it is an important property of the design that non-termination is originating from the constructions introduced especially for recursion and could not occur without them.

The simply-typed \( \lambda \)-calculus is also lifted at the level of types in richer type systems such as \( F^\omega \); then, the decidability of type-equality depends on the termination of the reduction at the type level.

The proof of termination for the simply-typed \( \lambda \)-calculus is a simple and illustrative use of logical relations.

*Notice however, that our simply-typed \( \lambda \)-calculus is equipped with a call-by-value semantics. Proofs of termination are usually done with a strong evaluation strategy where reduction can occur in any context.*
Normalization

Proving termination of reduction in fragments of the $\lambda$-calculus is often a difficult task because reduction may create new redexes or duplicate existing ones.

Hence the size of terms may grow (much) larger during reduction. The difficulty is to find some underlying structure that decreases.

We follow the proof schema of Pierce [2002], which is a modern presentation in a call-by-value setting of an older proof by Hindley and Seldin [1986]. The proof method is due to [Tait, 1967].
Calculus

Take the call-by-value $\lambda_{st}$ with primitive booleans and conditional. Write $B$ the type of booleans and $tt$ and $ff$ for true and false.

We define $V[\tau]$ and $E[\tau]$ the subsets of closed values and closed expressions of (ground) type $\tau$ by induction on types as follows:

$$V[B] \triangleq \{ tt, ff \}$$

$$V[\tau_1 \rightarrow \tau_2] \triangleq \{ \lambda x: \tau_1 . M | \lambda x: \tau_1 . M : \tau_1 \rightarrow \tau_2 \} \land \forall V \in V[\tau_1], (\lambda x: \tau_1 . M) \downarrow V \in E[\tau_2] \}$$

$$E[\tau] \triangleq \{ M | M : \tau \land \exists V \in V[\tau], M \downarrow V \}$$

We write $M \downarrow N$ for $M \rightarrow^* N$.

The goal is to show that any closed expression of type $\tau$ is in $E[\tau]$.

Remarks

Although usual with logical relations, well-typedness is actually not required here and omitted: otherwise, we would have to carry unnecessary type-preservation proof obligations.
Calculus

Take the call-by-value $\lambda_{st}$ with primitive booleans and conditional.

Write $B$ the type of booleans and $tt$ and $ff$ for $true$ and $false$.

We define $V[\tau]$ and $E[\tau]$ the subsets of closed values and closed expressions of (ground) type $\tau$ by induction on types as follows:

$$V[B] \overset{\Delta}{=} \{tt, ff\}$$
$$V[\tau_1 \to \tau_2] \overset{\Delta}{=} \{\lambda x: \tau_1. M \mid \forall V \in V[\tau_1], (\lambda x: \tau_1. M) V \in E[\tau_2]\}$$
$$E[\tau] \overset{\Delta}{=} \{M \mid \exists V \in V[\tau], M \Downarrow V\}$$

We write $M \Downarrow N$ for $M \longrightarrow^* N$.

The goal is to show that any closed expression of type $\tau$ is in $E[\tau]$.

Remarks

$V[\tau] \subseteq E[\tau]$—by definition.

$E[\tau]$ is closed by inverse reduction—by definition, i.e.

If $M \Downarrow N$ and $N \in E[\tau]$ then $M \in E[\tau]$. 

\[ \text{12(2)} \]
Problem

We wish to show that every closed term of type $\tau$ is in $E[\tau]$.

- Proof by induction on the typing derivation.
- Problem with abstraction: the premise is not closed.

We need to strengthen the hypothesis, i.e. also give a semantics to open terms.

- The semantics of open terms can be given by abstracting over the semantics of their free variables.
Generalize the definition to open terms

We define a *semantic judgment* for open terms $\Gamma \vdash M : \tau$ so that $\Gamma \vdash M : \tau$ implies $\Gamma \models M : \tau$ and $\emptyset \vdash M : \tau$ means $M \in E[\tau]$.

We interpret free term variables of type $\tau$ as *closed values* in $V[\tau]$.

We interpret environments $\Gamma$ as *closing substitutions* $\gamma$, i.e. mappings from term variables to *closed values*:

We write $\gamma \in G[\Gamma]$ to mean $\text{dom}(\gamma) = \text{dom}(\Gamma)$ and $\gamma(x) \in V[\tau]$ for all $x : \tau \in \Gamma$.

$$\Gamma \vdash M : \tau \iff \forall \gamma \in G[\Gamma], \quad \gamma(M) \in E[\tau]$$
Fundamental Lemma

**Theorem (fundamental lemma)**
If $\Gamma \vdash M : \tau$ then $\Gamma \models M : \tau$.

**Corollary (termination of well-typed terms):**
If $\emptyset \vdash M : \tau$ then $M \in \mathcal{E}[\tau]$.

That is, closed well-typed terms of type $\tau$ evaluate to values of type $\tau$. 
Proof by induction on the typing derivation

Routine cases

**Case** $\Gamma \vdash \texttt{tt} : B$ or $\Gamma \vdash \texttt{ff} : B$: by definition, $\texttt{tt}, \texttt{ff} \in \mathcal{V}[B]$ and $\mathcal{V}[B] \subseteq \mathcal{E}[B]$.

**Case** $\Gamma \vdash x : \tau$: $\gamma \in \mathcal{G}[\Gamma]$, thus $\gamma(x) \in \mathcal{V}[\tau] \subseteq \mathcal{E}[\tau]$

**Case** $\Gamma \vdash M_1 M_2 : \tau$:

By inversion, $\Gamma \vdash M_1 : \tau_2 \rightarrow \tau$ and $\Gamma \vdash M_2 : \tau_2$.

Let $\gamma \in \mathcal{G}[\Gamma]$. We have $\gamma(M_1 M_2) = (\gamma M_1) (\gamma M_2)$.

By IH, we have $\Gamma \vdash M_1 : \tau_2 \rightarrow \tau$ and $\Gamma \vdash M_2 : \tau_2$.

Thus $\gamma M_1 \in \mathcal{E}[\tau_2 \rightarrow \tau]$ (1) and $\gamma M_2 \in \mathcal{E}[\tau_2]$ (2).

By (2), there exists $V \in \mathcal{V}[\tau_2]$ such that $\gamma M_2 \Downarrow V$.

Thus $(\gamma M_1) (\gamma M_2) \Downarrow (\gamma M_1) V \in \mathcal{E}[\tau]$ by (1).

Then, $(\gamma M_1) (\gamma M_2) \in \mathcal{E}[\tau]$, by closure by inverse reduction.

**Case** $\Gamma \vdash \text{if } M \text{ then } M_1 \text{ else } M_2 : \tau$: By cases on the evaluation of $\gamma M$. 
Proof by induction on the typing derivation (key case)

The interesting case

Case \( \Gamma \vdash \lambda x : \tau_1. M : \tau_1 \rightarrow \tau \):

Assume \( \gamma \in \mathcal{G}[\Gamma] \).
We must show that \( \gamma(\lambda x : \tau_1. M) \in \mathcal{E}[\tau_1 \rightarrow \tau] \) (1)

That is, \( \lambda x : \tau_1. \gamma M \in \mathcal{V}[\tau_1 \rightarrow \tau] \) (we may assume \( x \notin \text{dom}(\gamma) \) w.l.o.g.)

Let \( V \in \mathcal{V}[\tau_1] \), it suffices to show \( \lambda x : \tau_1. \gamma M \in \mathcal{V}[\tau] \) (2).

We have \( \lambda x : \tau_1. \gamma M \rightarrow (\gamma M)[x \mapsto V] = \gamma' M \) where \( \gamma' \) is \( \gamma[x \mapsto V] \in \mathcal{G}[\Gamma, x : \tau_1] \) (3)

Since \( \Gamma, x : \tau_1 \vdash M : \tau \), we have \( \Gamma, x : \tau_1 \vdash M : \tau \) by IH on \( M \). Therefore by (3), we have \( \gamma' M \in \mathcal{E}[\tau] \). Since \( \mathcal{E}[\tau] \) is closed by inverse reduction, this proves (2) which finishes the proof of (1).
Variations

We have shown both *termination* and *type soundness*, simultaneously.

Termination would not hold if we had a fix point.
But type soundness would still hold.

The proof may be modified by choosing:

\[
\mathcal{E}[\tau] = \{ M : \tau \mid \forall N, M \Downarrow N \implies (N \in \mathcal{V}[\tau] \lor \exists N', N \rightarrow N') \}
\]

Compare with

\[
\mathcal{E}[\tau] = \{ M : \tau \mid \exists V \in \mathcal{V}[\tau], M \Downarrow V \}
\]

**Exercise**

*Show type soundness with this semantics.*
Contents

- Introduction
- Normalization of $\lambda_{st}$
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
(Bibliography)

Mostly following Bob Harper’s course notes *Practical foundations for programming languages* [Harper, 2012].

See also

- *Types, Abstraction and Parametric Polymorphism* [Reynolds, 1983]
- *Parametric Polymorphism and Operational Equivalence* [Pitts, 2000].
- *Theorems for free!* [Wadler, 1989].
- Course notes taken by Lau Skorstengaard on Amal Ahmed’s OPLSS lectures.

We assume a call-by-value operational semantics instead of call-by-name in [Harper, 2012].
When are two programs equivalent?
When are two programs equivalent

\[ M \Downarrow N ? \]
When are two programs equivalent

\[ M \downarrow N \ ? \]

\[ M \downarrow V \text{ and } N \downarrow V ? \]
When are two programs equivalent

\[ M \downarrow N \ ? \]
\[ M \downarrow V \text{ and } N \downarrow V ? \]

But what if \( M \) and \( N \) are functions?

Aren’t \( \lambda x. (x + x) \) and \( \lambda x. 2 \times x \) equivalent?

Idea

?
When are two programs equivalent

\[ M \downarrow N \ ? \]
\[ M \downarrow V \text{ and } N \downarrow V ? \]

But what if \( M \) and \( N \) are functions?

Aren’t \( \lambda x. (x + x) \) and \( \lambda x. 2 \times x \) equivalent?

**Idea** two functions are observationally equivalent if when applied to *equivalent arguments*, they lead to observationally *equivalent results*.

Are we general enough?
Observational equivalence

We can only *observe* the behavior of full *programs*, *i.e.* closed terms of some computation type, such as $B$ (the only one so far).

If $M : B$ and $N : B$, then $M \simeq N$ iff there exists $V$ such that $M \Downarrow V$ and $N \Downarrow V$. (Call $M \simeq N$ *behavioral equivalence*.)

To compare programs at other types, we
Observational equivalence

We can only observe the behavior of full programs, i.e. closed terms of some computation type, such as \( B \) (the only one so far).

If \( M : B \) and \( N : B \), then \( M \simeq N \) iff there exists \( V \) such that \( M \Downarrow V \) and \( N \Downarrow V \). (Call \( M \simeq N \) behavioral equivalence.)

To compare programs at other types, we place them in arbitrary closing contexts.

**Definition (observational equivalence)**

\[ \Gamma \vdash M \simeq N : \tau \overset{\Delta}{=} \forall C : (\Gamma \triangleright \tau) \leadsto (\emptyset \triangleright B), \; C[M] \simeq C[N] \]

**Typing of contexts**

\[ C : (\Gamma \triangleright \tau) \leadsto (\Delta \triangleright \sigma) \iff (\forall M, \; \Gamma \vdash M : \tau \implies \Delta \vdash C[M] : \sigma) \]

There is an equivalent definition given by a set of typing rules. This is needed to prove some properties by induction on the typing derivations.

We write \( M \preceq \tau N \) for \( \emptyset \vdash M \preceq N : \tau \).
Observational equivalence

Observational equivalence is the coarsiest consistent congruence, where:

\[ \equiv \] is consistent if \( \emptyset \vdash M \equiv N : B \) implies \( M \simeq N \).
\[ \equiv \] is a congruence if it is an equivalence and is closed by context, i.e.
\[ \Gamma \vdash M \equiv N : \tau \land C : (\Gamma \triangleright \tau) \leadsto (\Delta \triangleright \sigma) \quad \implies \quad \Delta \vdash C[M] \equiv C[N] : \sigma \]

**Consistent**: by definition, using the empty context.

**Congruence**: by compositionality of contexts.

**Coarsiest**: Assume \( \equiv \) is a consistent congruence.

We assume \( \Gamma \vdash M \equiv N : \tau \) (1) and show \( \Gamma \vdash M \simeq N : \tau \).

Let \( C : (\Gamma \triangleright \tau) \leadsto (\emptyset \triangleright B) \) (2). We must show that \( C[M] \simeq C[N] \).
This follows by consistency applied to \( \Gamma \vdash C[M] \equiv C[N] : B \) which itself follows by congruence from (1) and (2).
Problem with Observational Equivalence

Problems

- Observational equivalence is too difficult to test.
- Because of quantification over all contexts (too many for testing).
- But many contexts will do the same experiment.

Solution

We take advantage of types to reduce the number of experiments.

- Defining/testing the equivalence on base types.
- Propagating the definition mechanically at other types.

Logical relations provide the infrastructure for conducting such proofs.
Contents

- Introduction
- Normalization of $\lambda_{st}$
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
Logical equivalence for closed terms

Unary logical relations interpret types by predicates on (i.e. sets of) closed values of that type.

Binary relations interpret types by binary relations on closed values of that type, i.e. sets of pairs of related values of that type.

That is $\mathcal{V}[\tau] \subseteq \text{Val}(\tau) \times \text{Val}(\tau)$.

Then, $\mathcal{E}[\tau]$ is the closure of $\mathcal{V}[\tau]$ by inverse reduction.

We have $\mathcal{V}[\tau] \subseteq \mathcal{E}[\tau] \subseteq \text{Exp}(\tau) \times \text{Exp}(\tau)$. 
Logical equivalence for closed terms

We recursively define two relations $\mathcal{V}[\tau]$ and $\mathcal{E}[\tau]$ between values of type $\tau$ and expressions of type $\tau$ by

\[
\mathcal{V}[\text{B}] \triangleq \{(\text{tt}, \text{tt}), (\text{ff}, \text{ff})\}
\]

\[
\mathcal{V}[\tau \to \sigma] \triangleq \{(V_1, V_2) \mid V_1, V_2 \vdash \tau \to \sigma \land \\
\forall (W_1, W_2) \in \mathcal{V}[\tau], (V_1 W_1, V_2 W_2) \in \mathcal{E}[\sigma]\}
\]

\[
\mathcal{E}[\tau] \triangleq \{(M_1, M_2) \mid M_1, M_2 : \tau \land \\
\exists (V_1, V_2) \in \mathcal{V}[\tau], M_1 \Downarrow V_1 \land M_2 \Downarrow V_2\}
\]

In the following we will leave the typing constraint in gray implicit (as a global condition for sets $\mathcal{V}[\cdot]$ and $\mathcal{E}[\cdot]$).

We also write

\[
M_1 \sim_\tau M_2 \text{ for } (M_1, M_2) \in \mathcal{E}[\tau] \text{ and } \\
V_1 \approx_\tau V_2 \text{ for } (V_1, V_2) \in \mathcal{V}[\tau].
\]
Logical equivalence for closed terms

We recursively define two relations $\mathcal{V}[\tau]$ and $\mathcal{E}[\tau]$ between values of type $\tau$ and expressions of type $\tau$ by

\[
\mathcal{V}[B] \triangleq \{(tt, tt), (ff, ff)\}
\]

\[
\mathcal{V}[\tau \rightarrow \sigma] \triangleq \{(V_1, V_2) \mid V_1, V_2 \vdash \tau \rightarrow \sigma \land \\
\forall (W_1, W_2) \in \mathcal{V}[\tau], (V_1 W_1, V_2 W_2) \in \mathcal{E}[\sigma]\}
\]

\[
\mathcal{E}[\tau] \triangleq \{(M_1, M_2) \mid M_1, M_2 : \tau \land \\
\downarrow (M_1, M_2) \in \mathcal{V}[\tau]\}
\]

where $\downarrow (M_1, M_2)$ means $\{(V_1, V_2) \mid M_i \downarrow V_i\}$

In the following we will leave the typing constraint in gray implicit (as a global condition for sets $\mathcal{V}[\cdot]$ and $\mathcal{E}[\cdot]$).

We also write

\[
M_1 \sim_{\tau} M_2 \text{ for } (M_1, M_2) \in \mathcal{E}[\tau] \text{ and}
\]

\[
V_1 \approx_{\tau} V_2 \text{ for } (V_1, V_2) \in \mathcal{V}[\tau].
\]
Logical equivalence for closed terms (variant)

In a language with non-termination

We change the definition of $\mathcal{E}[\tau]$ to

$$\mathcal{E}[\tau] \triangleq \{ (M_1, M_2) \mid M_1, M_2 : \tau \land$$

$$\forall V_1, M_1 \downarrow V_1 \implies \exists V_2, M_2 \downarrow V_2 \land (V_1, V_2) \in \mathcal{V}[\tau] \}$$

Notice

$$\mathcal{V}[\tau \to \sigma] \triangleq \{ (V_1, V_2) \mid V_1, V_2 \vdash \tau \to \sigma \land$$

$$\forall (W_1, W_2) \in \mathcal{V}[\tau], (V_1 W_1, V_2 W_2) \in \mathcal{E}[\sigma] \}$$

$$= \{ ((\lambda x : \tau. M_1), (\lambda x : \tau. M_2)) \mid (\lambda x : \tau. M_1), (\lambda x : \tau. M_2) \vdash \tau \to \sigma \land$$

$$\forall (W_1, W_2) \in \mathcal{V}[\tau], ((\lambda x : \tau. M_1) W_1, (\lambda x : \tau. M_2) W_2) \in \mathcal{E}[\sigma] \}$$
Properties of logical equivalence for closed terms

Closure by reduction

By definition, since reduction is deterministic: Assume $M_1 \Downarrow N_1$ and $M_2 \Downarrow N_2$ and $(M_1, M_2) \in \mathcal{E}[\tau]$, i.e. there exists $(V_1, V_2) \in \mathcal{V}[\tau]$ (1) such that $M_i \Downarrow V_i$. Since reduction is deterministic, we must have $M_i \Downarrow N_i \Downarrow V_i$. This, together with (1), implies $(N_1, M_2) \in \mathcal{E}[\tau]$.

Closure by inverse reduction

Immediate, by construction of $\mathcal{E}[\tau]$.

Corollaries

- If $(M_1, M_2) \in \mathcal{E}[\tau \rightarrow \sigma]$ and $(N_1, N_2) \in \mathcal{E}[\tau]$, then $(M_1 \ N_1, M_2 \ N_2) \in \mathcal{E}[\sigma]$.
- To prove $(M_1, M_2) \in \mathcal{E}[\tau \rightarrow \sigma]$, it suffices to show $(M_1 \ V_1, M_2 \ V_2) \in \mathcal{E}[\sigma]$ for all $(V_1, V_2) \in \mathcal{V}[\tau]$. 
Properties of logical equivalence for closed terms

**Consistency** $(\sim_B) \subseteq (\simeq)$

Immediate, by definition of $\mathcal{E}[B]$ and $\mathcal{V}[B] \subseteq (\simeq)$.

**Lemma**

Logical equivalence is symmetric and transitive (at any given type).

*Note:* Reflexivity is not at all obvious.

**Proof**

We show it simultaneously for $\sim_\tau$ and $\simeq_\tau$ by induction on type $\tau$. 
Properties of logical equivalence for closed terms (proof)

For $\sim_\tau$, the proof is immediate by transitivity and symmetry of $\approx_\tau$.

For $\approx_\tau$, it goes as follows.

*Case $\tau$ is B for values*: the result is immediate.

*Case $\tau$ is $\tau \rightarrow \sigma$:

By IH, symmetry and transitivity hold at types $\tau$ and $\sigma$.

For symmetry, assume $V_1 \approx_{\tau \rightarrow \sigma} V_2 \ (H)$, we must show $V_2 \approx_{\tau \rightarrow \sigma} V_1$.

Assume $W_1 \approx_\tau W_2$. We must show $V_2 \ W_1 \sim_\sigma V_1 \ W_2 \ (C)$. We have $W_2 \approx_\tau W_1$ by symmetry at type $\tau$. By (H), we have $V_2 \ W_2 \sim_\sigma V_1 \ W_1$ and (C) follows by symmetry of $\sim$ at type $\sigma$.

For transitivity, assume $V_1 \approx_{\tau \rightarrow \sigma} V_2 \ (H1)$ and $V_2 \approx_{\tau \rightarrow \sigma} V_3 \ (H2)$. To show $V_1 \approx_{\tau \rightarrow \sigma} V_3$, we assume $W_1 \approx_\tau W_3$ and show $V_1 \ W_1 \sim_\sigma V_2 \ W_3 \ (C)$.

By (H1), we have $V_1 \ W_1 \sim_\sigma V_2 \ W_3 \ (C1)$. By **symmetry and transitivity of $\approx_\tau$ (IH)**, we get $W_3 \approx_\tau W_3$. **It’s not reflexivity!**

By (H2), we have $V_2 \ W_3 \sim_\sigma V_3 \ W_3 \ (C2)$. (C) follows by transitivity of $\sim_\sigma$ applied to (C1) and (C2).
Logical equivalence for open terms

When $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$, we wish to define a judgment $\Gamma \vdash M_1 \sim M_2 : \tau$ to mean that the open terms $M_1$ and $M_2$ are equivalent at type $\tau$.

The solution is to interpret program variables of $\text{dom}(\Gamma)$ by pairs of related values and typing contexts $\Gamma$ by a set of (closing) bisubstitutions $\gamma$ mapping variable type assignments to pairs of related values.

$$G[\emptyset] \triangleq \{ \emptyset \}$$
$$G[\Gamma, x : \tau] \triangleq \{ \gamma, x \mapsto (V_1, V_2) \mid \gamma \in G[\Gamma] \land (V_1, V_2) \in V[\tau] \}$$

Given a bisubstitution $\gamma$, we write $\gamma_i$ for the substitution that maps $x$ to $V_i$ whenever $\gamma$ maps $x$ to $(V_1, V_2)$.

**Definition**

$$\Gamma \vdash M_1 \sim M_2 : \tau \iff \forall \gamma \in G[\Gamma], \ (\gamma_1 M_1, \gamma_2 M_2) \in E[\tau]$$

We also write $\vdash M_1 \sim M_2 : \tau$ or $M_1 \sim \tau M_2$ for $\emptyset \vdash M_1 \sim M_2 : \tau$. 
Properties of logical equivalence for open terms

Immediate properties

Open logical equivalence is symmetric and transitive.

(Proof is immediate by the definition and the symmetry and transitivity of closed logical equivalence.)
Fundamental lemma of logical equivalence

**Theorem (Reflexivity)** (also called the fundamental lemma)

If $\Gamma \vdash M : \tau$, then $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Hypothesis</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-True</td>
<td>$\Gamma \vdash \text{tt} : \text{bool}$</td>
<td>$\Gamma \vdash \text{tt} : \text{bool}$</td>
</tr>
<tr>
<td>C-False</td>
<td>$\Gamma \vdash \text{ff} : \text{bool}$</td>
<td>$\Gamma \vdash \text{ff} : \text{bool}$</td>
</tr>
<tr>
<td>C-Var</td>
<td>$x : \tau \in \Gamma$</td>
<td>$\Gamma \vdash x : \tau$</td>
</tr>
<tr>
<td>C-Abs</td>
<td>$\Gamma, x : \tau \vdash M_1 : \sigma$</td>
<td>$\Gamma \vdash \lambda x : \tau. M_1 : \tau \rightarrow \sigma$</td>
</tr>
<tr>
<td>C-App</td>
<td>$\Gamma \vdash M_1 : \tau \rightarrow \sigma$</td>
<td>$\Gamma \vdash N_1 : \tau$</td>
</tr>
<tr>
<td>C-If</td>
<td>$\Gamma \vdash M_1 : B$</td>
<td>$\Gamma \vdash N_1 : \tau$</td>
</tr>
</tbody>
</table>
Fundamental lemma of logical equivalence

**Theorem (Reflexivity) (also called the fundamental lemma)**

If $\Gamma \vdash M : \tau$, then $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

- C-True
  
  $\Gamma \vdash \text{tt} : \text{bool}$

- C-False
  
  $\Gamma \vdash \text{ff} : \text{bool}$

- C-Var
  
  $x : \tau \in \Gamma \\
  \Gamma \vdash x : \tau$

- C-Abs
  
  $\Gamma, x : \tau \vdash M_1 : \sigma \\
  \Gamma \vdash \lambda x : \tau. M_1 : \tau \rightarrow \sigma$

- C-App
  
  $\Gamma \vdash M_1 : \tau \rightarrow \sigma \\
  \Gamma \vdash N_1 : \tau \\
  \Gamma \vdash M_1 N_1 : \sigma$

- C-If
  
  $\Gamma \vdash M_1 : \text{B} \\
  \Gamma \vdash N_1 : \tau \\
  \Gamma \vdash N_1' : \tau \\
  \Gamma \vdash \text{if } M_1 \text{ then } N_1 \text{ else } N_1' : \tau$
Fundamental lemma of logical equivalence

**Theorem (Reflexivity) (also called the fundamental lemma))**

If $\Gamma \vdash M : \tau$, then $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-True</td>
<td>$\Gamma \vdash tt : bool$</td>
</tr>
<tr>
<td>C-False</td>
<td>$\Gamma \vdash ff : bool$</td>
</tr>
<tr>
<td>C-VAR</td>
<td>$x : \tau \in \Gamma \quad \Gamma \vdash x : \tau$</td>
</tr>
<tr>
<td>C-Abs</td>
<td>$\Gamma, x : \tau \vdash M_1 : \sigma \quad \Gamma \vdash \lambda x : \tau. M_1 : \tau \rightarrow \sigma$</td>
</tr>
<tr>
<td>C-App</td>
<td>$\Gamma \vdash M_1 : \tau \rightarrow \sigma \quad \Gamma \vdash N_1 : \tau \quad \Gamma \vdash M_1 N_1 : \sigma$</td>
</tr>
<tr>
<td>C-IF</td>
<td>$\Gamma \vdash M_1 : B \quad \Gamma \vdash N_1 : \tau \quad \Gamma \vdash N_1' : \tau \quad \Gamma \vdash \text{if } M_1 \text{ then } N_1 \text{ else } N_1' : \tau$</td>
</tr>
</tbody>
</table>
Fundamental lemma of logical equivalence

**Theorem (Reflexivity)** *(also called the fundamental lemma))*

If $\Gamma \vdash M : \tau$, then $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

**C-True**

$\Gamma \vdash \text{tt} \sim \text{tt} : bool$

**C-False**

$\Gamma \vdash \text{ff} \sim \text{ff} : bool$

**C-VAR**

$x : \tau \in \Gamma$

$\Gamma \vdash x \sim x : \tau$

**C-Abs**

$\Gamma, x : \tau \vdash M_1 \sim M_2 : \sigma$

$\Gamma \vdash \lambda x : \tau. M_1 \sim \lambda x : \tau. M_2 : \tau \rightarrow \sigma$

**C-App**

$\Gamma \vdash M_1 \sim M_2 : \tau \rightarrow \sigma$

$\Gamma \vdash N_1 \sim N_2 : \tau$

$\Gamma \vdash M_1 N_1 \sim M_2 N_2 : \sigma$

**C-If**

$\Gamma \vdash M_1 \sim M_2 : \text{B}$

$\Gamma \vdash N_1 \sim N_2 : \tau$

$\Gamma \vdash N_1' \sim N_2' : \tau$

$\Gamma \vdash \text{if } M_1 \text{ then } N_1 \text{ else } N_1' \sim \text{if } M_2 \text{ then } N_2 \text{ else } N_2' : \tau$
Fundamental lemma of logical equivalence

**Theorem (Reflexivity)** *(also called the fundamental lemma))*

*If* $\Gamma \vdash M : \tau$, *then* $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

- **C-True**
  $\Gamma \vdash tt \sim tt : bool$

- **C-False**
  $\Gamma \vdash ff \sim ff : bool$

- **C-VAR**
  $x : \tau \in \Gamma$
  $\Gamma \vdash x \sim x : \tau$

- **C-Abs**
  $\Gamma, x : \tau \vdash M_1 \sim M_2 : \sigma$
  $\Gamma \vdash \lambda x : \tau. M_1 \sim \lambda x : \tau. M_2 : \tau \rightarrow \sigma$

- **C-APP**
  $\Gamma \vdash M_1 \sim M_2 : \tau \rightarrow \sigma$
  $\Gamma \vdash N_1 \sim N_2 : \tau$
  $\Gamma \vdash M_1 N_1 \sim M_2 N_2 : \sigma$

- **C-IF**
  $\Gamma \vdash M_1 \sim M_2 : B$
  $\Gamma \vdash N_1 \sim N_2 : \tau$
  $\Gamma \vdash N_1' \sim N_2' : \tau$
  $\Gamma \vdash \text{if } M_1 \text{ then } N_1 \text{ else } N_1' \sim \text{if } M_2 \text{ then } N_2 \text{ else } N_2' : \tau$
Fundamental lemma of logical equivalence

**Theorem (Reflexivity)** *(also called the fundamental lemma)*

If $\Gamma \vdash M : \tau$, then $\Gamma \vdash M \sim M : \tau$.

**Proof** By induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-True</td>
<td>$\Gamma \vdash tt \sim tt : bool$</td>
</tr>
<tr>
<td>C-False</td>
<td>$\Gamma \vdash ff \sim ff : bool$</td>
</tr>
</tbody>
</table>
| C-VAR | $x : \tau \in \Gamma$  
  $\Gamma \vdash x \sim x : \tau$ |
| C-ABS | $\Gamma, x : \tau \vdash M_1 \sim M_2 : \sigma$  
  $\Gamma \vdash \lambda x : \tau. M_1 \sim \lambda x : \tau. M_2 : \tau \rightarrow \sigma$ |
| C-APP | $\Gamma \vdash M_1 \sim M_2 : \tau \rightarrow \sigma$  
  $\Gamma \vdash N_1 \sim N_2 : \tau$  
  $\Gamma \vdash M_1 N_1 \sim M_2 N_2 : \sigma$ |
| C-IF | $\Gamma \vdash M_1 \sim M_2 : B$  
  $\Gamma \vdash N_1 \sim N_2 : \tau$  
  $\Gamma \vdash N_1' \sim N_2' : \tau$  
  $\Gamma \vdash \text{if } M_1 \text{ then } N_1 \text{ else } N_1' \sim \text{if } M_2 \text{ then } N_2 \text{ else } N_2' : \tau$ |
Proof of compatibility lemmas

Each case can be shown independently.

**Rule C-Abs:** Assume $\Gamma, x : \tau \vdash M_1 \sim M_2 : \sigma$ (1)
We show $\Gamma \vdash \lambda x : \tau. M_1 \sim \lambda x : \tau. M_2 : \tau \rightarrow \sigma$. Let $\gamma \in \mathcal{G}[\Gamma]$.
We show $(\gamma_1 (\lambda x : \tau. M_1), \gamma_2 (\lambda x : \tau. M_2)) \in \mathcal{V}[\tau \rightarrow \sigma]$. Let $(V_1, V_2)$ be in $\mathcal{V}[\tau]$.
We show $(\gamma_1 (\lambda x : \tau. M_1) V_1, \gamma_2 (\lambda x : \tau. M_2) V_2) \in \mathcal{E}[\sigma]$ (2).

Since $\gamma_i (\lambda x : \tau. M_i) V_i \Downarrow (\gamma_i, x \mapsto V_i) M_i \overset{\Delta}{=} \gamma'_i M_i$, by inverse reduction, it suffices to show $(\gamma'_1 M_1, \gamma'_2 M_2) \in \mathcal{E}[\sigma]$. This follows from (1) since $\gamma' \in \mathcal{G}[\Gamma, x : \tau]$.

**Rule C-App (and C-If):** By induction hypothesis and the fact that substitution distributes over applications (and conditional).
We must show $\Gamma \vdash M_1 N_1 \sim M_2 M_2 : \sigma$ (1). Let $\gamma \in \mathcal{G}[\Gamma]$. From the premises $\Gamma \vdash M_1 \sim M_2 : \tau \rightarrow \sigma$ and $\Gamma \vdash N_1 \sim N_2 : \tau$, we have $(\gamma_1 M_1, \gamma_2 M_2) \in \mathcal{E}[\tau \rightarrow \sigma]$ and $(\gamma_1 N_1, \gamma_2 N_2) \in \mathcal{E}[\tau]$. Therefore $(\gamma_1 M_1 \gamma_1 N_1, \gamma_2 M_2 \gamma_2 N_2) \in \mathcal{E}[\sigma]$. That is $(\gamma_1 (M_1 N_1), \gamma_2 (M_2 N_2)) \in \mathcal{E}[\sigma]$, which proves (1).

**Rule C-True, C-False, and C-Var:** are immediate
Proof of compatibility lemmas (cont.)

**Rule C-IF**: We show $\Gamma \vdash$ if $M_1$ then $N_1$ else $N_1'$ $\sim$ if $M_2$ then $N_2$ else $N_2'$ : $\tau$.

Assume $\gamma \in G[\llbracket \gamma \rrbracket]$.

We show $(\gamma_1 (\text{if } M_1 \text{ then } N_1 \text{ else } N_1'), \gamma_2 (\text{if } M_2 \text{ then } N_2 \text{ else } N_2')) \in E[\llbracket \tau \rrbracket]$, That is $(\text{if } \gamma_1 M_1 \text{ then } \gamma_1 N_1 \text{ else } \gamma_1 N_1', \text{if } \gamma_2 M_2 \text{ then } \gamma_2 N_2 \text{ else } \gamma_2 N_2') \in E[\llbracket \tau \rrbracket] \quad (1)$.

From the premise $\Gamma \vdash M_1 \sim M_2 : B$, we have $(\gamma_1 M_1, \gamma_2 M_2) \in E[\llbracket B \rrbracket]$. Therefore $M_1 \Downarrow V$ and $M_2 \Downarrow V$ where $V$ is either tt or ff:

- **Case $V$ is tt**: Then, $(\text{if } \gamma_i M_i \text{ then } \gamma_i N_i \text{ else } \gamma_i N_i') \Downarrow \gamma_i N_i$, i.e. $\gamma_i (\text{if } M_i \text{ then } N_i \text{ else } N_i') \Downarrow \gamma_i N_i$. From the premise $\Gamma \vdash N_1 \sim N_2 : \tau$, we have $(\gamma_1 N_1, \gamma_2 N_2) \in E[\llbracket \tau \rrbracket]$ and (1) follows by closer by inverse reduction.

- **Case $V$ is ff**: similar.
Proof of reflexivity

By induction on the derivation of $\Gamma \vdash M : \tau$.
We must show $\Gamma \vdash M \sim M : \tau$:

All cases immediately follow from compatibility lemmas.

*Case $M$ is $tt$ or $ff$: Immediate by Rule $C$-**True** or Rule $C$-**False**

*Case $M$ is $x$: Immediate by Rule $C$-**Var**.

*Case $M$ is $M' N$: By inversion of the typing rule $\text{App}$, induction hypothesis, and Rule $C$-**App**.

*Case $M$ is $\lambda \tau : N$: By inversion of the typing rule $\text{Abs}$, induction hypothesis, and Rule $C$-**Abs**.
Properties of logical relations

**Corollary (equivalence)** Open logical relation is an equivalence relation

**Logical equivalence is a congruence**
If $\Gamma \vdash M \sim M' : \tau$ and $C : (\Gamma \triangleright \tau) \sim (\Delta \triangleright \sigma)$, then $\Delta \vdash C[M] \sim C[M'] : \sigma$.

**Proof** By induction on the proof of $C : (\Gamma \triangleright \tau) \sim (\Delta \triangleright \sigma)$.

Similar to the proof of reflexivity—but *we need a syntactic definition of context-typing derivations* (which we have omitted) to be able to reason by induction on the context-typing derivation.

**Soundness of logical equivalence**
Logical equivalence implies observational equivalence.
If $\Gamma \vdash M \sim M' : \tau$ then $\Gamma \vdash M \equiv M' : \tau$.

Proof: Logical equivalence is a consistent congruence, hence included in observational equivalence which is the coarsiest such relation.
Properties of logical equivalence

Completeness of logical equivalence
Observational equivalence of closed terms implies logical equivalence.
That is \((\simeq_\tau) \subseteq (\sim_\tau)\).

Proof by induction on \(\tau\).

Case \(\mathcal{B}\): In the empty context, by consistency \(\simeq_\mathcal{B}\) is a subrelation of \(\approx_\mathcal{B}\) which coincides with \(\sim_\mathcal{B}\).

Case \(\tau \rightarrow \sigma\): By congruence of observational equivalence!

By hypothesis, we have \(M_1 \simeq_{\tau \rightarrow \sigma} M_2\) (1). To show \(M_1 \sim_{\tau \rightarrow \sigma} M_2\), we assume \(V_1 \simeq_\tau V_2\) (2) and show \(M_1 V_1 \sim_\sigma M_2 V_2\) (3).

By soundness applied to (2), we have \(V_1 \simeq_\tau V_2\) from (2). By congruence with (1), we have \(M_1 V_1 \simeq_\sigma M_2 V_2\), which implies (3) by IH at type \(\sigma\).
**Logical equivalence: example of application**

**Fact:** Assume \( \text{not} \triangleq \lambda x: B. \text{if } x \text{ then } \text{ff} \text{ else } \text{tt} \)

and \( M \triangleq \lambda x: B. \lambda y: \tau. \lambda z: \tau. \text{if } \text{not } x \text{ then } y \text{ else } z \)

and \( M' \triangleq \lambda x: B. \lambda y: \tau. \lambda z: \tau. \text{if } x \text{ then } z \text{ else } y \).

Show that \( M \cong_{B \rightarrow \tau \rightarrow \tau \rightarrow \tau} M' \).
Logical equivalence: example of application

**Fact:** Assume $\text{not} \triangleq \lambda x : B. \text{if } x \text{ then } \text{ff} \text{ else } \text{tt}$ and $M \triangleq \lambda x : B. \lambda y : \tau. \lambda z : \tau. \text{if } \text{not } x \text{ then } y \text{ else } z$
and $M' \triangleq \lambda x : B. \lambda y : \tau. \lambda z : \tau. \text{if } x \text{ then } z \text{ else } y$.

Show that $M \simeq_{B \rightarrow \tau \rightarrow \tau \rightarrow \tau} M'$.

**Proof**

It suffices to show $M \; V_0 \; V_1 \; V_2 \sim_{\tau} M' \; V_0' \; V_1' \; V_2'$ whenever $V_0 \approx_B V_0'$ (1) and $V_1 \approx_{\tau} V_1'$ (2) and $V_2 \approx_{\tau} V_2'$ (3).
Logical equivalence: example of application

Fact: Assume $\text{not} \triangleq \lambda x: B. \text{if } x \text{ then } \text{ff} \text{ else } \text{tt}$
and $M \triangleq \lambda x: B. \lambda y: \tau. \lambda z: \tau. \text{if } \text{not } x \text{ then } y \text{ else } z$
and $M' \triangleq \lambda x: B. \lambda y: \tau. \lambda z: \tau. \text{if } x \text{ then } z \text{ else } y$.

Show that $M \cong_{B \to \tau \to \tau \to \tau} M'$.

Proof

It suffices to show $M \ V_0 \ V_1 \ V_2 \sim_{\tau} M' \ V_0' \ V_1' \ V_2'$ whenever $V_0 \cong_B V_0'$ (1)
and $V_1 \cong_{\tau} V_1'$ (2) and $V_2 \cong_{\tau} V_2'$ (3). By inverse reduction, it suffices to show:
if $\text{not } V_0$ then $V_1$ else $V_2 \sim_{\tau}$ if $V_0'$ then $V_2'$ else $V_1'$ (4).

?
Logical equivalence: example of application

**Fact:** Assume \( \text{not} \overset{\Delta}{=} \lambda x : B. \text{if } x \text{ then } \text{ff} \text{ else } \text{tt} \)
and \( M \overset{\Delta}{=} \lambda x : B. \lambda y : \tau. \lambda z : \tau. \text{if } \text{not } x \text{ then } y \text{ else } z \)
and \( M' \overset{\Delta}{=} \lambda x : B. \lambda y : \tau. \lambda z : \tau. \text{if } x \text{ then } z \text{ else } y \).

Show that \( M \cong_{B \rightarrow \tau \rightarrow \tau \rightarrow \tau} M' \).

**Proof**

It suffices to show \( M \ V_0 \ V_1 \ V_2 \sim_{\tau} M' \ V'_0 \ V'_1 \ V'_2 \) whenever \( V_0 \cong_{B} V'_0 \) (1) and \( V_1 \cong_{\tau} V'_1 \) (2) and \( V_2 \cong_{\tau} V'_2 \) (3). By inverse reduction, it suffices to show:

- If \( \text{not } V_0 \) then \( V_1 \) else \( V_2 \) \( \sim_{\tau} \) if \( V'_0 \) then \( V'_2 \) else \( V'_1 \) (4).

It follows from (1) that we have only two cases:

**Case** \( V_0 = V'_0 = \text{tt} \): Then \( \text{not } V_0 \downarrow \text{ff} \) and thus \( M \downarrow V_2 \) while \( M' \downarrow V_2 \).
Then (4) follows by inverse reduction and (3).

**Case** \( V_0 = V'_0 = \text{ff} \): is symmetric.
Introduction
Normalization of $\lambda_{st}$
Observational equivalence in $\lambda_{st}$
Logical relations in stlc
Logical relations in $F$
Applications
Extensions
Observational equivalence

We now extend the notion of logical equivalence to System F.

\[ \tau ::= \ldots | \alpha | \forall \alpha. \tau \]
\[ M ::= \ldots | \Lambda \alpha. M | M \tau \]

We write typing contexts \( \Delta; \Gamma \) where \( \Delta \) binds variables and \( \Gamma \) binds program variables.

Typing of contexts becomes \( C : (\Delta; \Gamma \triangleright \tau) \leadsto (\Delta'; \Gamma' \triangleright \tau') \).

Observational equivalence

We (re)defined \( \Delta; \Gamma \vdash M \cong M' : \tau \) as

\[ \forall C : (\Delta; \Gamma \triangleright \tau) \leadsto (\emptyset; \emptyset \triangleright B), \quad C[M] \simeq C[M'] \]

As before, write \( M \cong_\tau N \) for \( \emptyset; \emptyset \vdash M \cong N : \tau \) (in particular, \( \tau \) is closed).
Logical equivalence

For closed terms (no free program variables)

- We need to give the semantics of polymorphic types \( \forall \alpha. \tau \)
- **Problem:** We cannot do it in terms of the semantics of instances \( \tau[\alpha \mapsto \sigma] \) since the semantics is defined by induction on types.
- **Solution:** we give the semantics of terms with open types—in some suitable environment that interprets type variables by logical relations (sets of pairs of related values) of closed types \( \rho_1 \) and \( \rho_2 \)

Let \( R(\rho_1, \rho_2) \) be the set of relations on values of closed types \( \rho_1 \) and \( \rho_2 \), that is \( \mathcal{P}(\text{Val}(\rho_1) \times \text{Val}(\rho_2)) \). We optionally restrict to **admissible** relations, i.e. relations that are closed by observational equivalence:

\[
R \in R^\#(\tau_1, \tau_2) \implies \\
\forall (V_1, V_2) \in R, \forall W_1, W_2, W_1 \equiv V_1 \land W_2 \equiv V_2 \implies (W_1, W_2) \in R
\]

The restriction to **admissible relations** is required for **completeness** of logical equivalence with respect to observational equivalence but **not for soundness**.
Example of admissible relations

For example, both

\[ R_1 \triangleq \{ (tt, 0), (ff, 1) \} \]
\[ R_2 \triangleq \{ (tt, 0) \} \cup \{ (ff, n) \mid n \in \mathbb{Z}^* \} \]

are admissible relations in \( \mathcal{R}^\#(B, int) \).

But

\[ R_3 \triangleq \{ (tt, \lambda x : \tau. 0), (ff, \lambda x : \tau. 1) \} \]

although in \( \mathcal{R}(B, \tau \rightarrow int) \), is not admissible.

Why?
Example of admissible relations

For example, both

\[
R_1 \triangleq \{(tt, 0), (ff, 1)\}
\]

\[
R_2 \triangleq \{(tt, 0)\} \cup \{(ff, n) \mid n \in \mathbb{Z}^*\}
\]

are admissible relations in \(\mathcal{R}^\#(B, \text{int})\).

But

\[
R_3 \triangleq \{(tt, \lambda x: \tau. 0), (ff, \lambda x: \tau. 1)\}
\]

although in \(\mathcal{R}(B, \tau \rightarrow \text{int})\), is not admissible.

Taking \(M_0 \triangleq \lambda x: \tau. (\lambda z: \text{int}. z) \ 0\), we have \(M \simeq_{\tau \rightarrow \text{int}} \lambda x: \tau. 0\) but \((tt, M)\) is not in \(R_3\).
Example of admissible relations

For example, both

\[ R_1 \Delta = \{(tt, 0), (ff, 1)\} \]
\[ R_2 \Delta = \{(tt, 0)\} \cup \{(ff, n) \mid n \in \mathbb{Z}^*\} \]

are admissible relations in \( \mathcal{R}^\#(B, \text{int}) \).

But

\[ R_3 \Delta = \{(tt, \lambda x: \tau. 0), (ff, \lambda x: \tau. 1)\} \]

although in \( \mathcal{R}(B, \tau \rightarrow \text{int}) \), is not admissible.

**Note** A relation \( R \) in \( \mathcal{R}(\tau_1, \tau_2) \) can always be turned into an admissible relation \( R^\# \) in \( \mathcal{R}^\#(\tau_1, \tau_2) \) by closing \( R \) by observational equivalence.

**Note** It is a key that such relations can relate values at different types.
Interpretation of type environments

Interpretation of type variables

We write $\eta$ for mappings $\alpha \mapsto (\rho_1, \rho_2, R)$ where $R \in R(\rho_1, \rho_2)$.

We write $\eta_i$ (resp. $\eta_R$) for the type (resp. relational) substitution that maps $\alpha$ to $\rho_i$ (resp. $R$) whenever $\eta$ maps $\alpha$ to $(\rho_1, \rho_2, R)$.

We define

\[
\mathcal{V}[\alpha]_\eta \triangleq \eta_R(\alpha)
\]

\[
\mathcal{V}[\forall \alpha. \tau]_\eta \triangleq \{(V_1, V_2) \mid V_1 : \eta_1(\forall \alpha. \tau) \land V_2 : \eta_2(\forall \alpha. \tau) \land \forall \rho_1, \rho_2, \forall R \in R(\rho_1, \rho_2), (V_1 \rho_1, V_2 \rho_2) \in \mathcal{E}[\tau]_{\eta, \alpha \mapsto (\rho_1, \rho_2, R)}\}
\]
Logical equivalence for closed terms with open types

We redefine

\[ \mathcal{V}[B]_{\eta} \triangleq \{(tt, tt), (ff, ff)\} \]

\[ \mathcal{V}[\tau \rightarrow \sigma]_{\eta} \triangleq \{(V_1, V_2) \mid V_1 \vdash \eta_1(\tau \rightarrow \sigma) \land V_2 \vdash \eta_2(\tau \rightarrow \sigma) \land \\
\forall (W_1, W_2) \in \mathcal{V}[\tau]_{\eta}, (V_1 W_1, V_2 W_2) \in \mathcal{E}[\sigma]_{\eta}\} \]

\[ \mathcal{E}[\tau]_{\eta} \triangleq \{(M_1, M_2) \mid M_1 : \eta_1 \tau \land M_2 : \eta_2 \tau \land \\
\exists (V_1, V_2) \in \mathcal{V}[\tau]_{\eta}, M_1 \downarrow V_1 \land M_2 \downarrow V_2\} \]

\[ \mathcal{G}[\emptyset]_{\eta} \triangleq \{\emptyset\} \]

\[ \mathcal{G}[\Gamma, x : \tau]_{\eta} \triangleq \{\gamma, x \mapsto (V_1, V_2) \mid \gamma \in \mathcal{G}[\Gamma]_{\eta} \land (V_1, V_2) \in \mathcal{V}[\tau]_{\eta}\} \]

and define

\[ \mathcal{D}[\emptyset] \triangleq \{\emptyset\} \]

\[ \mathcal{D}[\Delta, \alpha] \triangleq \{\eta, \alpha \mapsto (\rho_1, \rho_2, \mathcal{R}) \mid \eta \in \mathcal{D}[\Delta] \land \mathcal{R} \in \mathcal{R}(\rho_1, \rho_2)\}\]
Logical equivalence for open terms

Definition We define $\Delta; \Gamma \vdash M \sim M' : \tau$ as

$$\wedge \left\{ \begin{array}{l}
\Delta; \Gamma \vdash M, M' : \tau \\
\forall \eta \in D[\Delta], \forall \gamma \in G[\Gamma] \eta, (\eta_1(\gamma_1 M_1), \eta_2(\gamma_2 M_2)) \in E[\tau] \eta
\end{array} \right\}$$

(Notations are a bit heavy, but intuitions should remain simple.)

Notation

We also write $M_1 \sim_{\tau} M_2$ for $\vdash M_1 \sim M_2 : \tau$ (i.e. $\emptyset; \emptyset \vdash M_1 \sim M_2 : \tau$).

In this case, $\tau$ is a closed type and $M_1$ and $M_2$ are closed terms of type $\tau$; hence, this coincides with the previous definition $(M_1, M_2)$ in $E[\tau]$, which may still be used as a shorthand for $E[\tau] \emptyset$. 
Properties

Respect for observational equivalence

If \((M_1, M_2) \in \mathcal{E}[\tau]_\eta^\#\) and \(N_1 \simeq_{\eta_1(\tau)} M_1\) and \(N_2 \simeq_{\eta_2(\tau)} M_2\) then \((N_1, N_2) \in \mathcal{E}[\tau]_\eta^\#\).

*(We use \(^\#\) to indicate that admissibility is required in the definition of \(\mathcal{R}^\#\))

Proof. By induction on \(\tau\).

Assume \((M_1, M_2) \in \mathcal{E}[\tau]_\eta\) (1) and \(N_1 \simeq_{\eta_1(\tau)} M_1\) (2). We show \((N_1, M_2) \in \mathcal{E}[\tau]_\eta\).

*Case \(\tau\) is \(\forall \alpha. \sigma\): Assume \(R \in \mathcal{R}^\#(\rho_1, \rho_2)\). Let \(\eta_\alpha\) be \(\eta, \alpha \mapsto (\rho_1, \rho_2, R)\). We have \((M_1 \rho_1, M_2 \rho_2) \in \mathcal{E}[\sigma]_{\eta_\alpha}\), from (1).

By congruence from (2), we have \(N_1 \rho_1 \simeq_{\delta(\tau)} M_1 \rho_1\).

Hence, by induction hypothesis, \((M_1 \rho_1, M_2 \rho_2) \in \mathcal{E}[\sigma]_{\eta_\alpha}\), as expected.

*Case \(\tau\) is \(\alpha\): Relies on admissibility, indeed.

*Other cases:* the proof is similar to the case of the simply-typed \(\lambda\)-calculus.
Properties

Respect for observational equivalence

If \((M_1, M_2) \in \mathcal{E}[\tau]_\eta^\#\) and \(N_1 \equiv_{\eta_1(\tau)} M_1\) and \(N_2 \equiv_{\eta_2(\tau)} M_2\) then
\((N_1, N_2) \in \mathcal{E}[\tau]_\eta^\#\).

(We use \(^\#\) to indicate that admissibility is required in the definition of \(\mathcal{R}^\#\))

Proof. By induction on \(\tau\).

Corollary

The relation \(\mathcal{V}[\tau]_\eta^\#\) is an admissible relation in \(\mathcal{R}^\#(\eta_1 \tau, \eta_2 \tau)\).

Application: we may take this relation when admissibility is required.
Properties

**Lemma (Closure under observational equivalence)**

If $\Delta; \Gamma \vdash M_1 \sim^\# M_2 : \tau$ and $\Delta; \Gamma \vdash M_1 \cong N_1 : \tau$ and $\Delta; \Gamma \vdash M_2 \cong N_2 : \tau$, then $\Delta; \Gamma \vdash N_1 \sim^\# N_2 : \tau$

Requires admissibility

**Lemma (Compositionality)**

Assume $\Delta \vdash \sigma$ and $\Delta, \alpha \vdash \tau$ and $\eta \in \mathcal{D}[\Delta]$. Then,

$$\forall[\tau[\alpha \mapsto \sigma]]_\eta = \forall[\tau]_{\eta, \alpha \mapsto (\eta_1 \sigma, \eta_2 \sigma, \forall[\sigma]_\eta)}$$

Proof by induction on $\tau$. 

Key lemma
Parametricity

**Theorem (Reflexivity) (also called the fundamental lemma)**

If $\Delta; \Gamma \vdash M : \tau$ then $\Delta; \Gamma \vdash M \sim M : \tau$.

**Notice:** Admissibility is not required for the fundamental lemma

**Proof** by induction on the typing derivation, using compatibility lemmas.

**Compatibility lemmas**

We redefine the lemmas to work in a typing context of the form $\Delta, \Gamma$ instead of $\Gamma$ and add two new lemmas:

**C-TABS**

\[
\Delta, \alpha; \Gamma \vdash M_1 \sim M_2 : \tau \\
\Delta; \Gamma \vdash \Lambda \alpha. M_1 \sim \Lambda \alpha. M_2 : \forall \alpha. \tau
\]

**C-TAPP**

\[
\Delta; \Gamma \vdash M_1 \sim M_2 : \forall \alpha. \tau \\
\Delta \vdash \sigma \\
\Delta; \Gamma \vdash M_1 \sigma \sim M_2 \sigma : \tau[\alpha \mapsto \sigma]
\]
Properties

**Soundness of logical equivalence**
Logical equivalence implies observational equivalence. If \( \Delta; \Gamma \vdash M_1 \sim M_2 : \tau \) then \( \Delta; \Gamma \vdash M_1 \cong M_2 : \tau \).

**Completeness of logical equivalence**
Observational equivalence implies logical equivalence with admissibility. If \( \Delta; \Gamma \vdash M_1 \cong M_2 : \tau \) then \( \Delta; \Gamma \vdash M_1 \sim \# M_2 : \tau \).

As a particular case, \( M_1 \cong_{\tau} M_2 \) iff \( M_1 \sim_{\#} M_2 \).

**Note:** Admissibility is not required for soundness—only for completeness.

That is, proofs that some observational equivalence hold do not usually require admissibility.
Properties

Extensionality (A fact, hence does not depend on admissibility)

\[ M_1 \simeq_{\tau \rightarrow \sigma} M_2 \text{ iff } \forall (V : \tau), M_1 V \simeq_{\sigma} M_2 V \text{ iff } \forall (N : \tau), M_1 N \simeq_{\sigma} M_2 N \]

\[ M_1 \simeq_{\forall \alpha. \tau} M_2 \text{ iff for all closed type } \rho, M_1 \rho \simeq_{\tau[\alpha \mapsto \rho]} M_2 \rho. \]

Proof. Forward direction is immediate as \( \simeq \) is a congruence. Backward direction uses logical relations and admissibility, but the exported statement does not.

Case Value abstraction: It suffices to show \( M_1 \sim_{\tau \rightarrow \sigma} M_2 \). That is, assuming \( N_1 \sim_{\tau} N_2 \) (1), we show \( M_1 N_1 \sim_{\sigma} M_2 N_2 \) (2). By assumption, we have \( M_1 N_1 \simeq_{\sigma} M_2 N_1 \) (3). By the fundamental lemma, we have \( M_2 \sim_{\tau \rightarrow \sigma} M_2 \).

Hence, from (1), we must have \( M_2 N_1 \sim_{\sigma} M_2 N_2 \). We conclude (2) by respect for observational equivalence with (3)—which requires admissibility.

Case Type abstraction: It suffices to show \( M_1 \sim_{\forall \alpha. \tau} M_2 \). That is, given \( R \in \mathcal{R}(\rho_1, \rho_2) \), we show \( (M_1 \rho_1, M_2 \rho_2) \in \mathcal{E}[\tau]_{\alpha \mapsto (\rho_1, \rho_2, R)} \) (4).

By assumption, we have \( M_1 \rho_1 \simeq_{\tau[\alpha \mapsto \rho_1]} M_2 \rho_1 \) (5).

By the fundamental lemma, we have \( M_2 \sim_{\forall \alpha. \tau} M_2 \).

Hence, we have \( (M_2 \rho_1, M_2 \rho_2) \in \mathcal{E}[\tau]_{\alpha \mapsto (\rho_1, \rho_2, R)} \)

We conclude (4) by respect for observational equivalence with (5).
Contents

- Introduction
- Normalization of $\lambda_{st}$
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha$

**Fact** If $M : \forall \alpha. \alpha \rightarrow \alpha$, then $M \cong_{\forall \alpha. \alpha \rightarrow \alpha} id$ where $id = \Lambda \alpha. \lambda x : \alpha. x$.

**Proof** By *extensionality*, it suffices to show that for any $\rho$ and $V : \rho$ we have $M \rho V \cong_{\rho} id_{\rho} V$. In fact, by closure by inverse reduction, it suffices to show $M \rho V \cong_{\rho} V$ (1).

By parametricity, we have $M \sim_{\forall \alpha. \alpha \rightarrow \alpha} M$ (2).

Consider $R$ in $R(\rho, \rho)$ equal to $\{(V, V)\}$ and $\eta$ be $[\alpha \mapsto (\rho, \rho, R)]$. (3)

By construction, we have $(V, V) \in \mathcal{V}[\alpha]_{\eta}$.

Hence, from (2), we have $(M \rho V, M \rho V) \in \mathcal{E}[\alpha]_{\eta}$, which means that the pair $(M \rho V, M \rho V)$ reduces to a pair of values in (the singleton) $R$. This implies that $M \rho V$ reduces to $V$, which in turn, implies (1).

(3) Admissibility is not needed
Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$

Fact  Let $\sigma$ be $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$. If $M : \sigma$, then either
$M \cong_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \cong_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

Proof  By extensionality, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \cong_\sigma V_i$ (1).
Applications

Inhabitants of $\forall \alpha. \alpha \to \alpha \to \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either

$M \cong_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \cong_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \cong_\sigma V_i \ (1)$.

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $\mathcal{R}(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$.
Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$

Fact  Let $\sigma$ be $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$. If $M : \sigma$, then either

$M \equiv_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \equiv_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

Proof  By extensionality, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \equiv_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \equiv_\sigma V_i \ (1)$.

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $R(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in V[\alpha]_\eta$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in V[\alpha]_\eta$.

We have $(M, M) \in E[\sigma]$ by parametricity.
Applications

Fact Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either
\[ M \simeq_{\sigma} W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1 \] or
\[ M \simeq_{\sigma} W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2 \]

Proof By extensionality, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \simeq_{\rho} W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \simeq_{\sigma} V_i \ (1)$.

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $R(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in V[\alpha]_{\eta}$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in V[\alpha]_{\eta}$.

We have $(M, M) \in E[\sigma]$ by parametricity. Hence, $(M B tt ff, M \rho V_1 V_2)$ is in $V[\alpha]_{\eta}$, which means that $(M B tt ff, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

\[
\bigvee \begin{cases} 
M B tt ff \simeq_{B} tt & \land & M \rho V_1 V_2 \simeq_{\rho} V_1 \\
M B tt ff \simeq_{B} ff & \land & M \rho V_1 V_2 \simeq_{\rho} V_2
\end{cases}
\]

Next ?
Applications

Inhabitants of $\forall \alpha. \alpha \to \alpha \to \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either

$M \cong_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \cong_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \cong_\sigma V_i$ (1).

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $R(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in \mathcal{V}[\alpha]_\eta$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in \mathcal{V}[\alpha]_\eta$.

We have $(M, M) \in \mathcal{E}[\sigma]$ by parametricity. Hence, $(M \ B \ tt \ ff, M \rho V_1 V_2)$ is in $\mathcal{V}[\alpha]_\eta$, which means that $(M \ B \ tt \ ff, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$\forall \rho, V_1, V_2, \quad \lor \begin{cases} M \ B \ tt \ ff \cong_B tt \land M \rho V_1 V_2 \cong_\rho V_1 \\ M \ B \ tt \ ff \cong_B ff \land M \rho V_1 V_2 \cong_\rho V_2 \end{cases}$$

Since, $M \ B \ tt \ ff$ is independent of $\rho$, $V_1$, and $V_2$, this actually shows (1).
Applications

Inhabitants of $\forall \alpha. \alpha \to \alpha \to \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either
$M \cong_\sigma W_1 \overset{\Delta}{=} \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \cong_\sigma W_2 \overset{\Delta}{=} \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \cong_\sigma V_i \ (1)$.

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $\mathcal{R}(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in \mathcal{V}[\alpha]_\eta$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in \mathcal{V}[\alpha]_\eta$.

We have $(M, M) \in \mathcal{E}[\sigma]$ by parametricity. Hence, $(M \ B tt ff, M \rho V_1 V_2)$ is in $\mathcal{V}[\alpha]_\eta$, which means that $(M \ B tt ff, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$\bigvee \left\{ \begin{align*}
\forall \rho, V_1, V_2, & \quad M \ B \ tt \ ff \cong_B tt \land M \rho V_1 V_2 \cong_\rho V_1 \\
\forall \rho, V_1, V_2, & \quad M \ B \ tt \ ff \cong_B ff \land M \rho V_1 V_2 \cong_\rho V_2
\end{align*} \right\}$$

Since, $M \ B \ tt \ ff \text{ is independent of } \rho, V_1, \text{ and } V_2$, this actually shows $(1)$. 

Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$. If $M : \sigma$, then either

$M \simeq_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_1$ or $M \simeq_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1 : \alpha. \lambda x_2 : \alpha. x_2$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \simeq_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \simeq_\sigma V_i$ (1).

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(\text{tt}, V_1), (\text{ff}, V_2)\}$ in $\mathcal{R}(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(\text{tt}, V_1) \in \mathcal{V}[\alpha]_\eta$ since $R(\text{tt}, V_1)$ and, similarly, $(\text{ff}, V_2) \in \mathcal{V}[\alpha]_\eta$.

We have $(M, M) \in \mathcal{E}[\sigma]$ by parametricity. Hence, $(M B \text{tt ff}, M \rho V_1 V_2)$ is in $\mathcal{V}[\alpha]_\eta$, which means that $(M B \text{tt ff}, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$\bigvee\left\{ M B \text{tt ff} \simeq_B \text{tt} \land M \rho V_1 V_2 \simeq_\rho V_1 \right\}$$

$$\quad \bigvee\left\{ M B \text{tt ff} \simeq_B \text{ff} \land M \rho V_1 V_2 \simeq_\rho V_2 \right\}$$

*Since, $M B \text{tt ff}$ is independent of $\rho$, $V_1$, and $V_2$, this actually shows (1).*
Applications

Inhabitants of $\forall \alpha. \alpha \to \alpha \to \alpha$

Fact Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either

$M \cong_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_1$ or $M \cong_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_2$

Proof By extensionality, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \cong_\sigma V_i$ (1).

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $R(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in V[\alpha]_\eta$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in V[\alpha]_\eta$.

We have $(M, M) \in E[\sigma]$ by parametricity. Hence, $(M B \quad tt, ff, M \rho V_1 V_2)$ is in $V[\alpha]_\eta$, which means that $(M B \quad tt, ff, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$\lor \begin{cases} 
M B \quad tt, ff \cong_B tt \land M \rho V_1 V_2 \cong_\rho V_1 \\
M B \quad tt, ff \cong_B ff \land M \rho V_1 V_2 \cong_\rho V_2
\end{cases}$$

Since, $M B \quad tt, ff$ is independent of $\rho, V_1$, and $V_2$, this actually shows (1).
Applications

Inhabitants of \( \forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha \)

**Fact**  Let \( \sigma \) be \( \forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha \). If \( M : \sigma \), then either
\[ M \approx_{\sigma} W_1 \overset{\Delta}{=} \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_1 \quad \text{or} \quad M \approx_{\sigma} W_2 \overset{\Delta}{=} \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_2 \]

**Proof**  By *extensionality*, it suffices to show that for either \( i = 1 \) or \( i = 2 \), for any closed type \( \rho \) and \( V_1, V_2 : \rho \), we have \( M \rho V_1 V_2 \approx_{\rho} W_i \rho V_1 V_2 \), or just
\[ M \rho V_1 V_2 \approx_{\sigma} V_i \ (1) \]

Let \( \rho \) and \( V_1, V_2 : \rho \) be fixed. Consider \( R \) equal to \{ \( (0, V_1) \), \( (1, V_2) \) \} in \( R(\mathbb{N}, \rho) \) and \( \eta \) be \( \alpha \mapsto (\mathbb{N}, \rho, R) \). We have \( (0, V_1) \in \mathcal{V}[\alpha]_{\eta} \) since \( R(0, V_1) \) and, similarly, \( (1, V_2) \in \mathcal{V}[\alpha]_{\eta} \).

We have \( (M, M) \in \mathcal{E}[\sigma] \) by parametricity. Hence, \( (M \mathbb{N} \ 0 \ 1, M \rho V_1 V_2) \) is in \( \mathcal{V}[\alpha]_{\eta} \), which means that \( (M \mathbb{N} \ 0 \ 1, M \rho V_1 V_2) \) reduces to a pair of values in \( R \), which implies:

\[
\bigvee \left\{ M \mathbb{N} \ 0 \ 1 \overset{\mathbb{N}}{=} 0 \wedge M \rho V_1 V_2 \overset{\rho}{=} V_1 \\
M \mathbb{N} \ 0 \ 1 \overset{\mathbb{N}}{=} 1 \wedge M \rho V_1 V_2 \overset{\rho}{=} V_2 \right\}
\]

Since, \( M \mathbb{N} \ 0 \ 1 \) is independent of \( \rho \), \( V_1 \), and \( V_2 \), this actually shows \((1)\).
Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$. If $M : \sigma$, then either $M \simeq_{\sigma} W_1 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_1$ or $M \simeq_{\sigma} W_2 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_2$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \simeq_{\rho} W_i \rho V_1 V_2$, or just $M \rho V_1 V_2 \simeq_{\sigma} V_i \ (1)$.

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(W_1, V_1), (W_2, V_2)\}$ in $R(\sigma, \rho)$ and $\eta$ be $\alpha \mapsto (\sigma, \rho, R)$. We have $(W_1, V_1) \in \mathcal{V}[\alpha]_\eta$ since $R(W_1, V_1)$ and, similarly, $(W_2, V_2) \in \mathcal{V}[\alpha]_\eta$.

We have $(M, M) \in \mathcal{E}[\sigma]$ by parametricity. Hence, $(M \sigma \ W_1 W_2, M \rho V_1 V_2)$ is in $\mathcal{V}[\alpha]_\eta$, which means that $(M \sigma \ W_1 W_2, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$\bigvee \left\{ \begin{array}{l} M \sigma \ W_1 W_2 \simeq_{\sigma} W_1 \land M \rho V_1 V_2 \simeq_{\rho} V_1 \\
M \sigma \ W_1 W_2 \simeq_{\sigma} W_2 \land M \rho V_1 V_2 \simeq_{\rho} V_2 \end{array} \right. $$

Since, $M \sigma \ W_1 W_2$ is independent of $\rho, V_1,$ and $V_2$, this actually shows $(1)$.
Applications

Inhabitants of $\forall \alpha. \alpha \to \alpha \to \alpha$

**Fact** Let $\sigma$ be $\forall \alpha. \alpha \to \alpha \to \alpha$. If $M : \sigma$, then either

$$ M \cong_\sigma W_1 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_1 $$

or

$$ M \cong_\sigma W_2 \triangleq \Lambda \alpha. \lambda x_1: \alpha. \lambda x_2: \alpha. x_2 $$

**Proof** By *extensionality*, it suffices to show that for either $i = 1$ or $i = 2$, for any closed type $\rho$ and $V_1, V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho W_i \rho V_1 V_2$, or just

$$ M \rho V_1 V_2 \cong_\sigma V_i \ (1) $$

Let $\rho$ and $V_1, V_2 : \rho$ be fixed. Consider $R$ equal to $\{(tt, V_1), (ff, V_2)\}$ in $R(B, \rho)$ and $\eta$ be $\alpha \mapsto (B, \rho, R)$. We have $(tt, V_1) \in V[\alpha]_\eta$ since $R(tt, V_1)$ and, similarly, $(ff, V_2) \in V[\alpha]_\eta$.

We have $(M, M) \in E[\sigma]$ by parametricity. Hence, $(M B \ tt \ ff, M \rho V_1 V_2)$ is in $V[\alpha]_\eta$, which means that $(M B \ tt \ ff, M \rho V_1 V_2)$ reduces to a pair of values in $R$, which implies:

$$ \lor \begin{cases} M B \ tt \ ff \cong_B tt \land M \rho V_1 V_2 \cong_\rho V_1 \ \\ M B \ tt \ ff \cong_B ff \land M \rho V_1 V_2 \cong_\rho V_2 \end{cases} $$

*Since, $M B \ tt \ ff$ is independent of $\rho$, $V_1$, and $V_2$, this actually shows (1).*
Exercise

Inhabitants of $\forall \alpha. \alpha \rightarrow \alpha$

Redo the proof that all inhabitants of $\forall \alpha. \alpha \rightarrow \alpha$ are observationally equivalent to the identity, following the schema that we used for booleans.
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

**Fact** Let $\text{nat}$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : \text{nat}$, then $M \cong_{\text{nat}} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f^n x$. 

?
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

**Fact** Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

That is, the inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$ are the Church naturals.
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

**Fact** Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \equiv_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

**Proof**

?
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

Fact Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \overset{\Delta}{=} \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

Proof By extensionality, it suffices to show that there exists $n$ such that for any closed type $\rho$ and closed values $V_1 : \rho \to \rho$ and $V_2 : \rho$, we have $M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2$, or, by closure by inverse reduction and replacing observational by logical equivalence, $M \rho V_1 V_2 \sim_{\rho} V_1^n V_2$ (1), since $N_n \rho V_1 V_2$ reduces to $V_1^n V_2$. ?
**Applications**

**Inhabitants of** \( \forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha \)

**Fact** Let \( \text{nat} \) be \( \forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha \). If \( M : \text{nat} \), then \( M \cong_{\text{nat}} N_n \) for some integer \( n \), where \( N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x \).

**Proof** By *extensionality*, it suffices to show that there exists \( n \) such that for any closed type \( \rho \) and closed values \( V_1 : \rho \to \rho \) and \( V_2 : \rho \), we have \( M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2 \), or, by closure by inverse reduction and replacing observational by logical equivalence, \( M \rho V_1 V_2 \sim_{\rho} V_1^n V_2 \) (1), since \( N_n \rho V_1 V_2 \) reduces to \( V_1^n V_2 \). Let \( \rho \) and \( V_1 : \rho \to \rho \) and \( V_2 : \rho \) be fixed.

Let \( Z \) be \( N_0 \text{nat} \) and \( S \) be \( N_1 \text{nat} \). Let \( R \) in \( \mathcal{R}(\text{nat}, \rho) \) be \( \{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\} \) and \( \eta \) be \( \alpha \mapsto (\text{nat}, \rho, R) \).

We have \( (Z, V_2) \in \mathcal{V}[\alpha]_\eta \).
We also have \( (S, V_1) \in \mathcal{V}[\alpha \to \alpha]_\eta \).
Applications

Inhabitants of \( \forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha \)

**Fact** Let \( \text{nat} \) be \( \forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha \). If \( M : \text{nat} \), then \( M \simeq_{\text{nat}} N_n \) for some integer \( n \), where \( N_n \triangleq \Lambda \alpha. \lambda f : \alpha \rightarrow \alpha. \lambda x : \alpha. f^n x \).

**Proof** By *extensionality*, it suffices to show that there exists \( n \) such that for any closed type \( \rho \) and closed values \( V_1 : \rho \rightarrow \rho \) and \( V_2 : \rho \), we have \( M \rho V_1 V_2 \simeq_{\rho} N_n \rho V_1 V_2 \), or, by closure by inverse reduction and replacing observational by logical equivalence, \( M \rho V_1 V_2 \sim_{\rho} V_1^n V_2 \) (1), since \( N_n \rho V_1 V_2 \) reduces to \( V_1^n V_2 \). Let \( \rho \) and \( V_1 : \rho \rightarrow \rho \) and \( V_2 : \rho \) be fixed.

Let \( Z \) be \( N_0 \text{nat} \) and \( S \) be \( N_1 \text{nat} \). Let \( R \) in \( \mathcal{R}(\text{nat}, \rho) \) be \( \{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\} \) and \( \eta \) be \( \alpha \mapsto (\text{nat}, \rho, R) \).

We have \( (Z, V_2) \in \mathcal{V}[\alpha]_{\eta} \).

We also have \( (S, V_1) \in \mathcal{V}[\alpha \rightarrow \alpha]_{\eta} \). (A key to the proof.)

Indeed,

\[ ? \]
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

**Fact** Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

**Proof** By *extensionality*, it suffices to show that there exists $n$ such that for any closed type $\rho$ and closed values $V_1 : \rho \to \rho$ and $V_2 : \rho$, we have $M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2$, or, by closure by inverse reduction and replacing observational by logical equivalence, $M \rho V_1 V_2 \sim_{\rho} V^n_1 V_2$ (1), since $N_n \rho V_1 V_2$ reduces to $V^n_1 V_2$. Let $\rho$ and $V_1 : \rho \to \rho$ and $V_2 : \rho$ be fixed.

Let $Z$ be $N_0 nat$ and $S$ be $N_1 nat$. Let $R$ in $\mathcal{R}(nat, \rho)$ be $\{(S^k Z, V^n_1 V_2) | k \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (nat, \rho, R)$.

We have $(Z, V_2) \in \mathcal{V}[\alpha]_{\eta}$.
We also have $(S, V_1) \in \mathcal{V}[\alpha \to \alpha]_{\eta}$.

(A key to the proof.)

Indeed, assume $(W_1, W_2)$ in $\mathcal{V}[\alpha]_{\eta}$. There exists $k$ such that $W_1 = S^k Z$ and $W_2 = V^n_1 V_2$. Thus, $(S W_1, V_1 W_2)$ equal to $(S^{k+1} Z, V^{k+1}_1 V_2)$ is in $\mathcal{E}[\alpha]_{\eta}$. 
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

Fact Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

Proof By extensionality, it suffices to show that there exists $n$ such that for any closed type $\rho$ and closed values $V_1 : \rho \to \rho$ and $V_2 : \rho$, we have $M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2$, or, by closure by inverse reduction and replacing observational by logical equivalence, $M \rho V_1 V_2 \sim_{\rho} V_1^n V_2$ (1), since $N_n \rho V_1 V_2$ reduces to $V_1^n V_2$. Let $\rho$ and $V_1 : \rho \to \rho$ and $V_2 : \rho$ be fixed.

Let $Z$ be $N_0 nat$ and $S$ be $N_1 nat$. Let $R$ in $\mathcal{R}(nat, \rho)$ be $\{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (nat, \rho, R)$.

We have $(Z, V_2) \in \mathcal{V}[\alpha]_\eta$.
We also have $(S, V_1) \in \mathcal{V}[\alpha \to \alpha]_\eta$. (A key to the proof.)
Applications

Inhabitants of $\forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$

**Fact** Let $nat$ be $\forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \rightarrow \alpha. \lambda x : \alpha. f^n x$.

**Proof** By *extensionality*, it suffices to show that there exists $n$ such that for any closed type $\rho$ and closed values $V_1 : \rho \rightarrow \rho$ and $V_2 : \rho$, we have $M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2$, or, by closure by inverse reduction and replacing observational by logical equivalence, $M \rho V_1 V_2 \sim_{\rho} V_1^n V_2$ (1), since $N_n \rho V_1 V_2$ reduces to $V_1^n V_2$. Let $\rho$ and $V_1 : \rho \rightarrow \rho$ and $V_2 : \rho$ be fixed.

Let $Z$ be $N_0 nat$ and $S$ be $N_1 nat$. Let $R$ in $R(nat, \rho)$ be $\{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (nat, \rho, R)$.

We have $(Z, V_2) \in V[\alpha]_\eta$.
We also have $(S, V_1) \in V[\alpha \rightarrow \alpha]_\eta$. (A key to the proof.)

By parametricity, we have $M \sim_{nat} M$.  

?
Applications

**Fact** Let \( \text{nat} \) be \( \forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha \). If \( M : \text{nat} \), then \( M \cong_{\text{nat}} N_n \) for some integer \( n \), where \( N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x \).

**Proof** By *extensionality*, it suffices to show that there exists \( n \) such that for any closed type \( \rho \) and closed values \( V_1 : \rho \to \rho \) and \( V_2 : \rho \), we have
\[
M \rho V_1 V_2 \cong_{\rho} N_n \rho V_1 V_2,
\]
or, by closure by inverse reduction and replacing observational by logical equivalence, \( M \rho V_1 V_2 \sim_{\rho} V_1^n V_2 \) (1), since \( N_n \rho V_1 V_2 \) reduces to \( V_1^n V_2 \). Let \( \rho \) and \( V_1 : \rho \to \rho \) and \( V_2 : \rho \) be fixed.

Let \( Z \) be \( N_0 \text{nat} \) and \( S \) be \( N_1 \text{nat} \). Let \( R \) in \( \mathcal{R}(\text{nat}, \rho) \) be
\[
\{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\}
\]
and \( \eta \) be \( \alpha \mapsto (\text{nat}, \rho, R) \).

We have \( (Z, V_2) \in V[\alpha]_{\eta} \).

We also have \( (S, V_1) \in V[\alpha \to \alpha]_{\eta} \). (A key to the proof.)

By parametricity, we have \( M \sim_{\text{nat}} M \). Hence, \( (M \text{nat } S Z, M \rho V_1 V_2) \in \mathcal{E}[\alpha]_{\eta} \).

Thus, there exists \( n \) such that \( M \text{ nat } S Z \cong_{\text{nat}} S^n Z \) and \( M \rho V_1 V_2 \cong_{\rho} V_1^n V_2 \).
Applications

Inhabitants of $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$

Fact. Let $nat$ be $\forall \alpha. (\alpha \to \alpha) \to \alpha \to \alpha$. If $M : nat$, then $M \cong_{nat} N_n$ for some integer $n$, where $N_n \triangleq \Lambda \alpha. \lambda f : \alpha \to \alpha. \lambda x : \alpha. f^n x$.

Proof. By extensionality, it suffices to show that there exists $n$ such that for any closed type $\rho$ and closed values $V_1 : \rho \to \rho$ and $V_2 : \rho$, we have $M \rho V_1 V_2 \cong_\rho N_n \rho V_1 V_2$, or, by closure by inverse reduction and replacing observational by logical equivalence, $M \rho V_1 V_2 \sim_\rho V_1^n V_2$ (1), since $N_n \rho V_1 V_2$ reduces to $V_1^n V_2$. Let $\rho$ and $V_1 : \rho \to \rho$ and $V_2 : \rho$ be fixed.

Let $Z$ be $N_0 nat$ and $S$ be $N_1 nat$. Let $R$ in $R(nat, \rho)$ be $\{(S^k Z, V_1^k V_2) \mid k \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (nat, \rho, R)$.

We have $(Z, V_2) \in \mathcal{V}[\alpha]_\eta$.
We also have $(S, V_1) \in \mathcal{V}[\alpha \to \alpha]_\eta$. (A key to the proof.)

By parametricity, we have $M \sim_{nat} M$. Hence, $(M nat S Z, M \rho V_1 V_2) \in \mathcal{E}[\alpha]_\eta$. Thus, there exists $n$ such that $M nat S Z \cong_{nat} S^n Z$ and $M \rho V_1 V_2 \cong_\rho V_1^n V_2$.

Since, $M nat S Z$ is independent of $n$, we may conclude (1), provided the $S^n Z$ are all in different observational equivalence classes (easy to check).
Applications

Inhabitants of $\forall \alpha. \alpha \rightarrow (\tau \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha$

▷ Left as an exercise...
Applications

\[ \forall \alpha. \alpha \rightarrow (\tau \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \]

**Fact** Let \( \tau \) be closed and \textit{list} be \( \forall \alpha. \alpha \rightarrow (\tau \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \). Let \( C \) be \( \lambda H : \tau. \lambda T : \text{list}. \Lambda \alpha. \lambda n : \alpha. \lambda c : \tau \rightarrow \alpha \rightarrow \alpha. c \ H \ (T \ \alpha \ n \ c) \) and \( N \) be \( \Lambda \alpha. \lambda n : \alpha. \lambda c : \tau \rightarrow \alpha \rightarrow \alpha. n \). If \( M : \text{list} \), then \( M \cong_{\text{list}} N_n \) for some \( N_n \) in \( \mathcal{L}_n \) where \( \mathcal{L}_k \) is defined inductively by

\[
\mathcal{L}_0 \triangleq \{ N \} \quad \text{and} \quad \mathcal{L}_{k+1} \triangleq \{ C \ W_k \ N_k \mid W_k \in \text{Val}(\tau) \land N_k \in \mathcal{L}_k \}
\]

**Proof**

?
Applications

\(\forall \alpha. \alpha \to (\tau \to \alpha \to \alpha) \to \alpha\)

**Fact** Let \(\tau\) be closed and list be \(\forall \alpha. \alpha \to (\tau \to \alpha \to \alpha) \to \alpha\). Let \(C\) be \(\lambda H: \tau. \lambda T: \text{list}. \Lambda \alpha. \lambda n: \alpha. \lambda c: \tau \to \alpha \to \alpha. c \ H \ (T \ \alpha \ n \ c)\) and \(N\) be \(\Lambda \alpha. \lambda n: \alpha. \lambda c: \tau \to \alpha \to \alpha. n\). If \(M: \text{list}\), then \(M \equiv_{\text{list}} N_n\) for some \(N_n\) in \(L_n\) where \(L_k\) is defined inductively by

\[L_0 \triangleq \{N\} \quad \text{and} \quad L_{k+1} \triangleq \{C \ W_k \ N_k \mid W_k \in \text{Val}(\tau) \land N_k \in L_k\}\]

**Proof** By extensionality, it suffices to show that there exists \(n\) and \(N_n \in L_n\) such that for any closed type \(\rho\) and closed values \(V_1: \tau \to \rho \to \rho\) and \(V_2: \rho\), we have \(M \ \rho \ V_1 \ V_2 \sim_{\rho} N_n \ \rho \ V_1 \ V_2\), or, by closure by inverse reduction and replacing observational by logical equivalence, \(C \ W_n (\ldots (C \ W_1 \ N) \ldots)\) (1), since \(N_n \ \rho \ V_1 \ V_2\) reduces to \(C \ W_n (\ldots (C \ W_1 \ N) \ldots)\) where all \(W_k\) are in \(\text{Val}(\tau)\).

Let \(\rho\) and \(V_1: \alpha \to \rho \to \rho\) and \(V_2: \rho\) be fixed.

Let \(R\) in \(\mathcal{R}(\text{list}, \rho)\) be defined inductively as \(\bigcup R_n\) where \(R_{k+1}\) is

\[\{\downarrow (C \ G \ T, V_2 \ H \ U) \mid (G, H) \in \mathcal{V}[\tau]_\eta \land (T, U) \in R_k\}\]

and \(R_0\) is \(\{(N, V_1)\}\).

We have \((N, V_1) \in R_0 \subseteq \mathcal{V}[\alpha]_\eta\).

We also have \((C, V_2) \in \mathcal{V}[\tau \to \alpha \to \alpha]_\eta\).
Applications

∀α. α → (τ → α → α) → α

**Fact** Let τ be closed and list be ∀α. α → (τ → α → α) → α. If M : list, then M ≡list Nn for some Nn in Łn where Łk is defined inductively by Ł0 △ = {N} and Łk+1 △ = {CWk Nk | Wk ∈ Val(τ) ∧ Nk ∈ Łk}.

**Proof** By *extensionality*, it suffices to show that there exists n and Nn ∈ Łn such that for any closed type ρ and closed values V1 : τ → ρ → ρ and V2 : ρ, we have CWn (⋯(CW1 N)⋯) (1).

Let ρ and V1 : α → ρ → ρ and V2 : ρ be fixed. Let R in R(list, ρ) be defined inductively as ∪Rn where Rk+1 is

{⇓(CGT, V2HU) | (G, H) ∈ V[τ]η ∧ (T, U) ∈ Rk} and R0 is {(N, V1)}.

We have (N, V1) ∈ R0 ⊆ V[α]η.

We also have (C, V2) ∈ V[τ → α → α]η. (A key to the proof)

Indeed,
Applications

∀α. α → (τ → α → α) → α

Fact  Let τ be closed and list be ∀α. α → (τ → α → α) → α. If M : list, then M ≡_{list} N_n for some N_n in L_n where L_k is defined inductively by L_0 ≡ {N} and L_{k+1} ≡ {C \ W_k \ N_k | W_k ∈ Val(τ) ∧ N_k ∈ L_k}.

Proof  By extensionality, it suffices to show that there exists n and N_n ∈ L_n such that for any closed type ρ and closed values V_1 : τ → ρ → ρ and V_2 : ρ, we have C \ W_n (… (C \ W_1 \ N)…) (1).

Let ρ and V_1 : α → ρ → ρ and V_2 : ρ be fixed. Let R in R(list, ρ) be defined inductively as ∪ R_n where R_{k+1} is

\{ \downarrow (C \ G \ T, V_2 \ H \ U) | (G, H) ∈ \mathcal{V}[τ]_η ∧ (T, U) ∈ R_k \} and R_0 is \{ (N, V_1) \}.

We have (N, V_1) ∈ R_0 ⊆ \mathcal{V}[α]_η.

We also have (C, V_2) ∈ \mathcal{V}[τ → α → α]_η. (A key to the proof)

Indeed, assume (G, H) in \mathcal{V}[τ]_η and (T, U) in \mathcal{V}[α]_η, i.e. in R_k for some k. Then, \downarrow (C \ G \ T, V_2 \ H \ U) is in R^{k+1} ⊆ \mathcal{V}[α]_η. Hence, (C \ G \ T, V_2 \ H \ U) ∈ \mathcal{E}[α]_η, as expected.
Applications

\[ \forall \alpha. \alpha \to (\tau \to \alpha \to \alpha) \to \alpha \]

**Fact** Let \( \tau \) be closed and \textit{list} be \( \forall \alpha. \alpha \to (\tau \to \alpha \to \alpha) \to \alpha \). If \( M : \text{list} \), then \( M \cong_{\text{list}} N_n \) for some \( N_n \) in \( \mathcal{L}_n \) where \( \mathcal{L}_k \) is defined inductively by 
\[ \mathcal{L}_0 \overset{\Delta}{=} \{ N \} \] 
and 
\[ \mathcal{L}_{k+1} \overset{\Delta}{=} \{ CW_k N_k \mid W_k \in \text{Val}(\tau) \land N_k \in \mathcal{L}_k \} \].

**Proof** By \textit{extensionality}, it suffices to show that there exists \( n \) and \( N_n \in \mathcal{L}_n \) such that for any closed type \( \rho \) and closed values \( V_1 : \tau \to \rho \to \rho \) and \( V_2 : \rho \), we have \( CW_n (\ldots (CW_1 N) \ldots) \) (1).

Let \( \rho \) and \( V_1 : \alpha \to \rho \to \rho \) and \( V_2 : \rho \) be fixed. Let \( R \) in \( \mathcal{R}(\text{list}, \rho) \) be defined inductively as \( \bigcup R_n \) where \( R_{k+1} \) is 
\[ \{ \downarrow (CGT, V_2 H U) \mid (G, H) \in \mathcal{V}[\tau]_\eta \land (T, U) \in R_k \} \] 
and \( R_0 \) is \( \{ (N, V_1) \} \).

We have \( (N, V_1) \in R_0 \subseteq \mathcal{V}[\alpha]_\eta \). We also have \( (C, V_2) \in \mathcal{V}[\tau \to \alpha \to \alpha]_\eta \).

\[ ? \]
Applications

∀α. α → (τ → α → α) → α

Fact  Let τ be closed and list be ∀α. α → (τ → α → α) → α. If M : list, then M ≃_{list} N_n for some N_n in L_n where L_k is defined inductively by L_0 △= {N} and L_{k+1} △= {CW_k N_k | W_k ∈ Val(τ) ∧ N_k ∈ L_k}.

Proof By extensionality, it suffices to show that there exists n and N_n ∈ L_n such that for any closed type ρ and closed values V_1 : τ → ρ → ρ and V_2 : ρ, we have CW_n (... (CW_1 N) ...) (1).

Let ρ and V_1 : α → ρ → ρ and V_2 : ρ be fixed. Let R in R(list, ρ) be defined inductively as ∪ R_n where R_{k+1} is

{↓ (CG T, V_2 H U) | (G, H) ∈ ∀τη ∧ (T, U) ∈ R_k} and R_0 is {(N, V_1)}.

We have (N, V_1) ∈ R_0 ⊆ ∀αη. We also have (C, V_2) ∈ ∀τ→α→αη.

By parametricity, we have M ∼_{list} M.
Applications

∀α. α → (τ → α → α) → α

Fact  Let τ be closed and list be ∀α. α → (τ → α → α) → α. If M : list, then M ≈_{list} N_n for some N_n in L_n where L_k is defined inductively by L_0 \overset{\wedge}{=} \{ N \} and L_{k+1} \overset{\wedge}{=} \{ C W_k N_k \mid W_k \in \text{Val(τ)} \land N_k \in L_k \}.

Proof  By extensionality, it suffices to show that there exists n and N_n ∈ L_n such that for any closed type ρ and closed values V_1 : τ → ρ → ρ and V_2 : ρ, we have C W_n (\ldots (C W_1 N) \ldots) (1).

Let ρ and V_1 : α → ρ → ρ and V_2 : ρ be fixed. Let R in R(list, ρ) be defined inductively as \bigcup R_n where R_{k+1} is

\{ \llbracket (C G T, V_2 H U) \rrbracket \mid (G, H) \in \mathcal{V}[[tau]]_\eta \land (T, U) \in R_k \} and R_0 is \{ (N, V_1) \}.

We have (N, V_1) ∈ R_0 ≤ \mathcal{V}[[alpha]]_\eta. We also have (C, V_2) ∈ \mathcal{V}[[tau → alpha → alpha]]_\eta.

By parametricity, we have M \sim_{list} M. Hence, (M list C N, M \rho V_1 V_2) ∈ \mathcal{E}[[alpha]]_\eta.

Thus, there exists n such that M list C N ≈_{list} C W_n (\ldots (C W_1 N) \ldots) and M \rho V_1 V_2 ≈_{\rho} V_2 W_n (\ldots (V_2 W_1 V_1) \ldots).
Applications

\[ \forall \alpha. \alpha \rightarrow (\tau \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \]

Fact Let \( \tau \) be closed and \( \text{list} \) be \( \forall \alpha. \alpha \rightarrow (\tau \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \). If \( M : \text{list} \), then \( M \equiv_{\text{list}} N_n \) for some \( N_n \) in \( \mathcal{L}_n \) where \( \mathcal{L}_k \) is defined inductively by
\[ \mathcal{L}_0 \overset{\Delta}{=} \{ N \} \quad \text{and} \quad \mathcal{L}_{k+1} \overset{\Delta}{=} \{ CW_k, N_k \mid W_k \in \text{Val}(\tau) \land N_k \in \mathcal{L}_k \} \].

Proof By extensionality, it suffices to show that there exists \( n \) and \( N_n \in \mathcal{L}_n \) such that for any closed type \( \rho \) and closed values \( V_1 : \tau \rightarrow \rho \rightarrow \rho \) and \( V_2 : \rho \), we have \( CW_n (\ldots (CW_1 N) \ldots) \) (1).

Let \( \rho \) and \( V_1 : \alpha \rightarrow \rho \rightarrow \rho \) and \( V_2 : \rho \) be fixed. Let \( R \) in \( \mathcal{R}(\text{list}, \rho) \) be defined inductively as \( \bigcup R_n \) where \( R_{k+1} \) is
\[ \{ \Downarrow (CGT, V_2 HU) \mid (G, H) \in \mathcal{V}[\tau]_{\eta} \land (T, U) \in R_k \} \] and \( R_0 \) is \( \{(N, V_1)\} \).

We have \( (N, V_1) \in R_0 \subseteq \mathcal{V}[\alpha]_{\eta} \). We also have \( (C, V_2) \in \mathcal{V}[\tau \rightarrow \alpha \rightarrow \alpha]_{\eta} \).

By parametricity, we have \( M \sim_{\text{list}} M \). Hence, \( (M \text{ list} C N, M \rho V_1 V_2) \in \mathcal{E}[\alpha]_{\eta} \). Thus, there exists \( n \) such that \( M \text{ list} C N \equiv_{\text{list}} C W_n (\ldots (CW_1 N) \ldots) \) and \( M \rho V_1 V_2 \equiv_{\rho} V_2 W_n (\ldots (V_2 W_1 V_1) \ldots) \).

Since, \( M \text{ list} C N \) is independent of \( n \) and \( (W_k)_{k \in 1..n} \), we may conclude (1).

(This uses that \( \mathcal{R}_k \) are all in different observational equivalence classes, which is easy to check, as a length function would return different integers.)
Contents

- Introduction
- Normalization of $\lambda_{st}$
- Observational equivalence in $\lambda_{st}$
- Logical relations in stlc
- Logical relations in F
- Applications
- Extensions
Encodable features

We have shown that all expressions of type \( nat \) behave as natural numbers. Hence, natural numbers are definable.

Still, we could also provide a type \( nat \) of natural numbers as primitive.

Then, we may extend

- **behavioral equivalence**: if \( M_1 : nat \) and \( M_2 : nat \), we have \( M_1 \simeq_{nat} M_2 \) iff there exists \( n : nat \) such that \( M_1 \Downarrow n \) and \( M_2 \Downarrow n \).

- **logical equivalence**: \( \forall [nat] \triangleq \{(n, n) \mid n \in \mathbb{N}\} \)

All properties are preserved.
Encodable features

Given closed types $\tau_1$ and $\tau_2$, we defined

$$
\tau_1 \times \tau_2 \overset{\Delta}{=} \forall \alpha. (\tau_1 \to \tau_2 \to \alpha) \to \alpha
$$

$$(M_1, M_2) \overset{\Delta}{=} \Lambda\alpha. \lambda x: \tau_1 \to \tau_2 \to \alpha. x \ M_1 \ M_2$$

$$M.i \overset{\Delta}{=} M (\lambda x_1: \tau_1. \lambda x_2: \tau_2. x_i)$$

Facts

If $M : \tau_1 \times \tau_2$, then $M \cong_{\tau_1 \times \tau_2} (M_1, M_2)$ for some $M_1 : \tau_1$ and $M_2 : \tau_2$.

If $M : \tau_1 \times \tau_2$ and $M.1 \cong_{\tau_1} M_1$ and $M.2 \cong_{\tau_2} M_2$, then $M \cong_{\tau_1 \times \tau_2} (M_1, M_2)$

Primitive pairs

We may instead extend the language with \textit{primitive} pairs. Then,

$$\mathcal{V}[\tau \times \sigma]_\eta \overset{\Delta}{=} \{(V_1, W_1), (V_2, W_2)\}$$

$$\text{where } V_1, V_2 \in \mathcal{V}[\tau]_\eta \land (W_1, W_2) \in \mathcal{V}[\sigma]_\eta$$
Sums

We define:

\[ \mathcal{V}[\tau + \sigma]_\eta = \{(\text{inj}_1 V_1, \text{inj}_1 V_2) \mid (V_1, V_2) \in \mathcal{V}[\tau]_\eta\} \cup \{(\text{inj}_2 W_1, \text{inj}_2 W_2) \mid (W_1, W_2) \in \mathcal{V}[\sigma]_\eta\} \]

Notice that sums, as all datatypes, can also be encoded in System F.
Primitive Lists

We recursively define

\[ \mathcal{V}[\text{list } \tau]_\eta \triangleq \bigcup_k \mathcal{W}^k_\eta \]

where

\[ \mathcal{W}^0_\eta = \{ (\text{Nil, Nil}) \} \]

\[ \mathcal{W}^{k+1}_\eta = \{ (\text{Cons } H_1 T_1, \text{Cons } H_2 T_2) \mid (H_1, H_2) \in \mathcal{V}[\tau]_\eta \land (T_1, T_2) \in \mathcal{W}^k_\eta \} \]

\[ 1\text{This definition is well-founded.} \]
We recursively define

\[ \mathcal{V}[\text{list } \tau]_\eta \triangleq \bigcup_k \mathcal{W}^k_\eta \]

where

\[ \mathcal{W}^0_\eta = \{ (\text{Nil, Nil}) \} \]

\[ \mathcal{W}^{k+1}_\eta = \{ (\text{Cons } H_1 T_1, \text{Cons } H_2 T_2) \mid (H_1, H_2) \in \mathcal{V}[\tau]_\eta \land (T_1, T_2) \in \mathcal{W}^k_\eta \} \]

Assume that \((\alpha \mapsto \rho_1, \rho_2, R) \in \eta\) where \(R\) in \(R(\rho_1, \rho_2)\) is the graph \(\langle g \rangle\) of a function \(g\), i.e. equal to \(\{(V_1, V_2) \mid g V_1 \Downarrow V_2\}\). Then, we have:

\[ \mathcal{V}[\text{list } \alpha]_\eta(W_1, W_2) \]

\[ \iff \exists k, \forall \left\{ \begin{array}{l}
W_1 = \text{Nil} \land W_2 = \text{Nil} \\
W_1 = \text{Cons } H_1 T_1 \land W_2 = \text{Cons } H_2 T_2 \land g H_1 \Downarrow H_2 \\
\land (T_1, T_2) \in \mathcal{W}^k_\eta
\end{array} \right. \]

---

1This definition is well-founded.
We \textit{recursively}\footnote{This definition is well-founded.} define

$$
\mathcal{V}[\text{list } \tau]_{\eta} \triangleq \bigcup_k \mathcal{W}_{\eta}^k
$$

where

$$
\begin{align*}
\mathcal{W}_{\eta}^0 &= \{(\text{Nil}, \text{Nil})\} \\
\mathcal{W}_{\eta}^{k+1} &= \{(\text{Cons } H_1 T_1, \text{Cons } H_2 T_2) \\
&\quad \mid (H_1, H_2) \in \mathcal{V}[\tau]_{\eta} \land (T_1, T_2) \in \mathcal{W}_{\eta}^k\}
\end{align*}
$$

Assume that \((\alpha \mapsto \rho_1, \rho_2, R) \in \eta\) where \(R\) in \(\mathcal{R}(\rho_1, \rho_2)\) is the graph \(\langle g \rangle\) of a function \(g\), \textit{i.e.} equal to \(\{(V_1, V_2) \mid g V_1 \Downarrow V_2\}\). Then, we have:

$$
\mathcal{V}[\text{list } \alpha]_{\eta}(W_1, W_2)
\iff \exists k, \forall \left\{
\begin{array}{l}
W_1 = \text{Nil} \land W_2 = \text{Nil} \\
W_1 = \text{Cons } H_1 T_1 \land W_2 \Downarrow \text{Cons } (g H_1) T_2 \\
\land (T_1, T_2) \in \mathcal{W}_{\eta}^k
\end{array}\right.
$$
Primitve Lists

We recursively\(^1\) define

\[
\mathcal{V}[\text{list } \tau]_\eta \triangleq \bigcup_k \mathcal{W}^k_\eta
\]

where

\[
\begin{align*}
\mathcal{W}^0_\eta &= \{(\text{Nil}, \text{Nil})\} \\
\mathcal{W}^{k+1}_\eta &= \{(\text{Cons } H_1 T_1, \text{Cons } H_2 T_2) \\
&\quad \mid (H_1, H_2) \in \mathcal{V}[\tau]_\eta \land (T_1, T_2) \in \mathcal{W}^k_\eta\}
\end{align*}
\]

Assume that \((\alpha \mapsto \rho_1, \rho_2, R) \in \eta\) where \(R\) in \(\mathcal{R}(\rho_1, \rho_2)\) is the graph \(\langle g \rangle\) of a function \(g\), i.e. equal to \(\{(V_1, V_2) \mid g \downarrow V_1 \uparrow V_2\}\). Then, we have:

\[
\mathcal{V}[\text{list } \alpha]_\eta(W_1, W_2)
\]

\[
\iff \exists k, \forall
\begin{cases}
W_1 = \text{Nil} \land W_2 = \text{Nil} \\
W_1 = \text{Cons } H_1 T_1 \land W_2 \downarrow \text{Cons } (g H_1) T_2 \\
\land (T_1, T_2) \in \mathcal{W}^k_\eta
\end{cases}
\]

\[
\iff \text{map } \rho_1 \rho_2 g \downarrow W_1 \uparrow W_2
\]

---

\(^1\)This definition is well-founded.
Primitive Lists

We \textit{recursively}\footnote{This definition is well-founded.} define

\[ \mathcal{V}[\text{list } \tau]_\eta \triangleq \bigcup_k \mathcal{W}_\eta^k \]

where

\[ \mathcal{W}_\eta^0 = \{ (\text{Nil}, \text{Nil}) \} \]

\[ \mathcal{W}_\eta^{k+1} = \{ (\text{Cons } H_1 T_1, \text{Cons } H_2 T_2) \mid (H_1, H_2) \in \mathcal{V}[\tau]_\eta \land (T_1, T_2) \in \mathcal{W}_\eta^k \} \]

Assume that \((\alpha \mapsto \rho_1, \rho_2, R) \in \eta\) where \(R\) in \(\mathcal{R}(\rho_1, \rho_2)\) is the graph \(\langle g \rangle\) of a function \(g\), \textit{i.e.} equal to \(\{(V_1, V_2) \mid g V_1 \downarrow V_2\}\). Then, we have:

\[ \mathcal{V}[\text{list } \alpha]_\eta = \langle \text{map } \rho_1 \rho_2 \ g \rangle \]
Fact: Assume \( \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \) (1). Then

\[
(\forall x, y, \text{cmp}_2 (f x) (f y) = \text{cmp}_1 x y) \implies \forall \ell, \text{sort} \text{cmp}_2 (\text{map} f \ell) = \text{map} f (\text{sort} \text{cmp}_1 \ell)
\]
Applications

\( \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \)

**Fact:** Assume \( \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \) (1). Then

\[
(\forall x, y, \ cmp_2 (f x) (f y) = cmp_1 x y) \implies \\
\forall \ell, \ \text{sort} \ cmp_2 (\text{map} f \ \ell) = \text{map} f (\text{sort} \ cmp_1 \ \ell)
\]
Applications

\[ \text{sort} : \forall \alpha. (\alpha \rightarrow \alpha \rightarrow \text{bool}) \rightarrow \text{list } \alpha \]

**Fact:** Assume \( \text{sort} : \forall \alpha. (\alpha \rightarrow \alpha \rightarrow \text{bool}) \rightarrow \text{list } \alpha \rightarrow \text{list } \alpha \) (1). Then

\[
(\forall x, y, \ cmp_2 (f \ x) (f \ y) \cong \ cmp_1 \ x \ y) \implies \\
\forall \ell, \ \text{sort} \ cmp_2 (\text{map} \ f \ \ell) \cong \text{map} \ f \ (\text{sort} \ cmp_1 \ \ell)
\]
Applications

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list}\,\alpha \]

**Proof:** Assume \( \forall x, y, \ cp\ (f\ x)\ (f\ y) \cong cp\ x\ y \) (H).

We have \( \text{sort} \sim_\sigma \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list}\,\alpha \to \text{list}\,\alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{bool}]_{\eta},
\forall (V_1, V_2) \in \mathcal{V}[\text{list}\,\alpha]_{\eta},
(\text{sort}\ \rho_1\ cp_1\ V_1, \text{sort}\ \rho_2\ cp_2\ V_2) \in \mathcal{E}[\text{list}\,\alpha]_{\eta})
\]

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \).
Applications

**sort**: \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \)

**Proof:** Assume \( \forall x, y, \ cp (f x) (f y) \cong cp x y \) (H).

We have \( sort \sim_{\sigma} sort \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to B]_{\eta},
\forall (V_1, V_2) \in \mathcal{V}[\text{list} \alpha]_{\eta}, (\text{sort} \ \rho_1 \ \cp_1 \ \V_1, \ \text{sort} \ \rho_2 \ \cp_2 \ \V_2) \in \mathcal{E}[\text{list} \alpha]_{\eta}) \tag{1}
\]

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f V_1 \Downarrow V_1 \) and \( f W_1 \Downarrow W_1 \), hence \( cp_2 (f V_1)(f W_1) \Downarrow cp_1 V_2 W_2 \). Thus \( cp_2 (f V_1)(f W_1) \cong cp_1 V_2 W_2 \). With (H), this implies \( cp_2 V_1 W_1 \cong cp_1 V_2 W_2 \), i.e. \( cp_2 V_1 W_1 \sim cp_1 V_2 W_2 \) since we are at type B, as expected.
Applications

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \]

Proof: Assume \( \forall x, y, \ cp(f \ x) (f \ y) \equiv cp \ x \ y \) (H).

We have \( sort \sim_\sigma sort \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[ \forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{B}]_\eta, \]
\[ \forall (V_1, V_2) \in \mathcal{V}[\text{list } \alpha]_\eta, \ (\text{sort } \rho_1 \ cp_1 \ V_1, \text{sort } \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[\text{list } \alpha]_\eta) \]  \( (1) \)

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \downarrow V_1 \) and \( f \ W_1 \downarrow W_1 \), hence \( cp_2 \ (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 \ (f \ V_1)(f \ W_1) \equiv cp_1 \ V_2 W_2 \). With (H), this implies \( cp_2 \ V_1 W_1 \equiv cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since

\[ \mathcal{V}[\text{list } \alpha]_\eta \triangleq \langle \text{map } \rho_1 \rho_2 f \rangle \subseteq \mathcal{V}[\rho_1] \times \mathcal{V}[\rho_2] \]
Applications

\[ sort : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \]

**Proof:** Assume \( \forall x, y, \ cp(f \ x) (f \ y) \cong cp \ x \ y \) (H).

We have \( sort \sim_\sigma sort \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[[\alpha \to \alpha \to \text{B}]]_{\eta}, \\
\forall (V_1, V_2) \in \mathcal{V}[[\text{list } \alpha]]_{\eta}, (\text{sort } \rho_1 \ cp_1 \ V_1, \text{sort } \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[[\text{list } \alpha]]_{\eta})
\]

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \downarrow V_1 \) and \( f \ W_1 \downarrow W_1 \), hence \( cp_2 (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 (f \ V_1)(f \ W_1) \cong cp_1 \ V_2 W_2 \). With (H), this implies \( cp_2 \ V_1 W_1 \cong cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since

\[
\mathcal{V}[[\text{list } \alpha]]_{\eta} \triangleq \langle \text{map } \rho_1 \rho_2 \ f \rangle \subseteq \mathcal{V}[[\rho_1]] \times \mathcal{V}[[\rho_2]]
\]

(2) reads

\[
\forall V_1 : \text{list } \rho_1, V_2 :: \text{list } \rho_2, \\
\text{map } \rho_1 \rho_2 \ f \ V_1 \downarrow V_2 \implies \exists W_1, W_2, \\
\left\{ \begin{array}{l}
\text{map } \rho_1 \rho_2 \ f \ W_1 \downarrow W_2 \\
\text{sort } \rho_1 \ cp_1 \ V_1 \downarrow W_1 \\
\text{sort } \rho_2 \ cp_2 \ V_2 \downarrow W_2
\end{array} \right.
\]

68(4) 76
Applications

\[
\text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha
\]

**Proof:** Assume \(\forall x, y, \ cp (f \ x) (f \ y) \equiv cp \ x \ y\) (H).

We have \(\text{sort} \sim_\sigma \text{sort}\) where \(\sigma = \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha\).

Thus, for all \(\rho_1, \rho_2\), and relations \(R\) in \(R(\rho_1, \rho_2)\),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{B}]_{\eta}, \quad \forall (V_1, V_2) \in \mathcal{V}[\text{list} \alpha]_{\eta}, \quad (\text{sort} \ \rho_1 \ cp_1 \ V_1, \text{sort} \ \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[\text{list} \ \alpha]_{\eta})
\]

where \(\eta = \alpha \mapsto (\rho_1, \rho_2, R)\). We may choose \(R\) to be \(\langle f \rangle\) for some \(f\).

We have (1). Indeed, for all \((V_1, V_2)\) and \((W_1, W_2)\) in \(\langle f \rangle\), we have \(f \ V_1 \downarrow V_1\) and \(f \ W_1 \downarrow W_1\), hence \(cp_2 (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2\). Thus \(cp_2 (f \ V_1)(f \ W_1) \equiv cp_1 \ V_2 W_2\). With (H), this implies \(cp_2 \ V_1 W_1 \equiv cp_1 \ V_2 W_2\), i.e. \(cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2\) since we are at type B, as expected. Hence (2) holds.

Since

\[
\mathcal{V}[\text{list} \ \alpha]_{\eta} \triangleq \langle \text{map} \ \rho_1 \ \rho_2 \ f \rangle \subseteq \mathcal{V}[\rho_1] \times \mathcal{V}[\rho_2]
\]

(2) implies

\[
\forall V_1 : \text{list} \ \rho_1, \ V_2 :: \text{list} \ \rho_2, \quad \text{map} \ \rho_1 \ \rho_2 \ f \ V_1 \downarrow V_2 \implies \exists W_1, W_2,
\]

\[
\begin{cases}
\text{map} \ \rho_1 \ \rho_2 \ f \ W_1 \downarrow W_2 \\
\text{sort} \ \rho_1 \ cp_1 \ V_1 \downarrow W_1 \\
\text{sort} \ \rho_2 \ cp_2 \ V_2 \downarrow W_2
\end{cases}
\]
Applications

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \]

Proof: Assume \( \forall x, y, \ cp (f x) (f y) \equiv cp \ x \ y \) (H).

We have \( \text{sort} \sim_{\sigma} \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[ \forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{B}]_{\eta}, \]
\[ \forall (V_1, V_2) \in \mathcal{V}[\text{list} \alpha]_{\eta}, \ (\text{sort} \ \rho_1 \ cp_1 \ V_1, \ \text{sort} \ \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[\text{list} \ \alpha]_{\eta} \] (1)

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \downarrow V_1 \) and \( f \ W_1 \downarrow W_1 \), hence \( cp_2 \ (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 \ (f \ V_1)(f \ W_1) \equiv cp_1 \ V_2 W_2 \). With (H), this implies \( cp_2 \ V_1 W_1 \equiv cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since

\[ \mathcal{V}[\text{list} \ \alpha]_{\eta} \triangleq \langle \text{map} \ \rho_1 \ \rho_2 \ f \rangle \subseteq \mathcal{V}[\rho_1] \times \mathcal{V}[\rho_2] \]

(2) implies

\[ \forall V_1 : \text{list} \ \rho_1, \exists W_1, W_2, \left\{ \begin{array}{l} \text{map} \ \rho_1 \ \rho_2 \ f \ W_1 \downarrow W_2 \\
\text{sort} \ \rho_1 \ cp_1 \ V_1 \ \downarrow \ W_1 \\
\text{sort} \ \rho_2 \ cp_2 \ (\text{map} \ \rho_1 \ \rho_2 \ f \ V_1) \ \downarrow \ W_2 \end{array} \right\} \]
Applications

\[ \text{sort : } \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \]

**Proof:** Assume \( \forall x, y, \ cp (f \ x) (f \ y) \equiv cp \ x \ y \) (H).

We have \( \text{sort} \sim_\sigma \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{B}]_\eta, \\
\forall (V_1, V_2) \in \mathcal{V}[\text{list } \alpha]_\eta, \ (\text{sort } \rho_1 \ cp_1 \ V_1, \text{sort } \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[\text{list } \alpha]_\eta \tag{1}
\]

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \downarrow V_1 \) and \( f \ W_1 \downarrow W_1 \), hence \( cp_2 \ (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 \ (f \ V_1)(f \ W_1) \equiv cp_1 \ V_2 W_2 \). With (H), this implies \( cp_2 \ V_1 W_1 \equiv cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since \( \forall \ V_1 : \text{list } \rho_1 \),

\[
\exists W_1, W_2, \ \begin{cases} 
map \rho_1 \rho_2 f \ W_1 \downarrow W_2 \\
\text{sort } \rho_1 \ cp_1 \ V_1 \downarrow W_1 \\
\text{sort } \rho_2 \ cp_2 (\map \rho_1 \rho_2 f \ V_1) \downarrow W_2 
\end{cases}
\]

(2) implies
Applications

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \]

Proof: Assume \( \forall x, y, \ cp (f x) (f y) \equiv cp \ x \ y \) \((H)\).

We have \( \text{sort} \sim_\sigma \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list } \alpha \to \text{list } \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[ \forall (cp_1, cp_2) \in \mathcal{V}[[\alpha \to \alpha \to B]]_\eta, \]

\[ \forall (V_1, V_2) \in \mathcal{V}[[\text{list } \alpha]]_\eta, \ (\text{sort } \rho_1 \ cp_1 \ V_1, \text{sort } \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[[\text{list } \alpha]]_\eta \] \( \tag{1} \)

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \Downarrow V_1 \) and \( f \ W_1 \Downarrow W_1 \), hence \( cp_2 \ (f \ V_1)(f \ W_1) \Downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 \ (f \ V_1)(f \ W_1) \equiv cp_1 \ V_2 W_2 \). With \( (H) \), this implies \( cp_2 \ V_1 W_1 \equiv cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type \( B \), as expected. Hence (2) holds.

Since

\[ \mathcal{V}[[\text{list } \alpha]]_\eta \triangleq \langle \text{map } \rho_1 \ \rho_2 \ f \rangle \subseteq \mathcal{V}[[\rho_1]] \times \mathcal{V}[[\rho_2]] \]

(2) implies

\[ \forall V_1 : \text{list } \rho_1 \]

\[ \exists W_2, \left\{ \begin{array}{ll}
\text{map } \rho_1 \ \rho_2 \ f \ (\text{sort } \rho_1 \ cp_1 \ V_1) \Downarrow W_2
\\
\text{sort } \rho_2 \ cp_2 \ (\text{map } \rho_1 \ \rho_2 \ f \ V_1) \Downarrow W_2
\end{array} \right. \]
Applications

\[
\text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha
\]

Proof: Assume \( \forall x, y, \ cp (f \ x) (f \ y) \cong cp \ x \ y \) (H).

We have \( \text{sort} \sim_{\sigma} \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[
\forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to B]_{\eta},
\quad \forall (V_1, V_2) \in \mathcal{V}[\text{list} \alpha]_{\eta}, \ (\text{sort} \ \rho_1 \ cp_1 \ V_1, \text{sort} \ \rho_2 \ cp_2 \ V_2) \in \mathcal{E}[\text{list} \ alpha]_{\eta})
\]

(1)

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \( (V_1, V_2) \) and \( (W_1, W_2) \) in \( \langle f \rangle \), we have \( f \ V_1 \downarrow V_1 \) and \( f \ W_1 \downarrow W_1 \), hence \( cp_2 \ (f \ V_1)(f \ W_1) \downarrow cp_1 \ V_2 W_2 \). Thus \( cp_2 \ (f \ V_1)(f \ W_1) \cong cp_1 \ V_2 W_2 \). With (H), this implies \( cp_2 \ V_1 W_1 \cong cp_1 \ V_2 W_2 \), i.e. \( cp_2 \ V_1 W_1 \sim cp_1 \ V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since

\[
\mathcal{V}[\text{list} \ \alpha]_{\eta} \triangleq \langle \text{map} \ \rho_1 \ \rho_2 \ f \rangle \subseteq \mathcal{V}[\rho_1] \times \mathcal{V}[\rho_2]
\]

(2) implies

\[
\forall V_1 : \text{list} \ \rho_1, \quad \text{map} \ \rho_1 \ \rho_2 \ f \ (\text{sort} \ \rho_1 \ cp_1 \ V_1) \cong \text{sort} \ \rho_2 \ cp_2 \ (\text{map} \ \rho_1 \ \rho_2 \ f \ V_1)
\]
Applications

\[ \text{sort} : \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \]

Proof: Assume \( \forall x, y, \ cp (f x) (f y) \cong cp x y \) (H).

We have \( \text{sort} \sim_{\sigma} \text{sort} \) where \( \sigma \) is \( \forall \alpha. (\alpha \to \alpha \to \text{bool}) \to \text{list} \alpha \to \text{list} \alpha \).

Thus, for all \( \rho_1, \rho_2 \), and relations \( R \) in \( \mathcal{R}(\rho_1, \rho_2) \),

\[ \forall (cp_1, cp_2) \in \mathcal{V}[\alpha \to \alpha \to \text{B}]_{\eta}, \]
\[ \forall (V_1, V_2) \in \mathcal{V}[\text{list} \alpha]_{\eta}, \ (\text{sort} \rho_1 cp_1 V_1, \text{sort} \rho_2 cp_2 V_2) \in \mathcal{E}[\text{list} \alpha]_{\eta} \] \hspace{1cm} (1)

where \( \eta \) is \( \alpha \mapsto (\rho_1, \rho_2, R) \). We may choose \( R \) to be \( \langle f \rangle \) for some \( f \).

We have (1). Indeed, for all \((V_1, V_2)\) and \((W_1, W_2)\) in \( \langle f \rangle \), we have \( f V_1 \downarrow V_1 \) and \( f W_1 \downarrow W_1 \), hence \( cp_2 (f V_1)(f W_1) \downarrow cp_1 V_2 W_2 \). Thus \( cp_2 (f V_1)(f W_1) \cong cp_1 V_2 W_2 \). With (H), this implies \( cp_2 V_1 W_1 \cong cp_1 V_2 W_2 \), i.e. \( cp_2 V_1 W_1 \sim cp_1 V_2 W_2 \) since we are at type B, as expected. Hence (2) holds.

Since
\[ \mathcal{V}[\text{list} \alpha]_{\eta} \triangleq \langle \text{map} \rho_1 \rho_2 f \rangle \subseteq \mathcal{V}[\rho_1] \times \mathcal{V}[\rho_2] \]

(2) implies
\[ \forall V : \text{list} \rho_1, \ \text{map} \rho_1 \rho_2 f (\text{sort} \rho_1 cp_1 V) \cong \text{sort} \rho_2 cp_2 (\text{map} \rho_1 \rho_2 f V) \]
Applications

whoami : ∀α. list α → list α

Left as an exercise...
Existential types

We define:

\[ \mathcal{V}[\exists \alpha. \tau]_\eta \triangleq \{ \text{pack } V_1, \rho_1 \text{ as } \exists \alpha. \tau, \text{pack } V_2, \rho_2 \text{ as } \exists \alpha. \tau \mid \exists \rho_1, \rho_2, R \in \mathcal{R}(\rho_1, \rho_2), (V_1, V_2) \in \mathcal{E}[\tau]_{\eta, \alpha \mapsto (\rho_1, \rho_2, R)} \} \]

Compare with

\[ \mathcal{V}[\forall \alpha. \tau]_\eta = \{ (\Lambda \alpha. M_1, \Lambda \alpha. M_2) \mid \forall \rho_1, \rho_2, R \in \mathcal{R}(\rho_1, \rho_2), ((\Lambda \alpha. M_1) \rho_1, (\Lambda \alpha. M_2) \rho_2) \in \mathcal{E}[\tau]_{\eta, \alpha \mapsto (\rho_1, \rho_2, R)} \} \]
Existential types

Consider $V_1 \triangleq (not, tt)$, and $V_2 \triangleq (succ, 0)$ and $\sigma \triangleq (\alpha \to \alpha) \times \alpha$.

Let $R \in \mathcal{R}(bool, nat)$ be $\{(tt, 2n), (ff, 2n + 1) \mid n \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (bool, nat, R)$.

We have $(V_1, V_2) \in \mathcal{V}[\sigma]_\eta$.

Hence, $(\text{pack } V_1, bool \text{ as } \exists \alpha. \sigma, \text{ pack } V_2, nat \text{ as } \exists \alpha. \sigma) \in \mathcal{V}[\exists \alpha. \sigma]$. 
Existential types

Consider $V_1 \triangleq (\text{not}, \text{tt})$, and $V_2 \triangleq (\text{succ}, 0)$ and $\sigma \triangleq (\alpha \to \alpha) \times \alpha$.

Let $R \in \mathcal{R}(\text{bool}, \text{nat})$ be \{(tt, $2n$), (ff, $2n + 1$) | $n \in \mathbb{N}$\} and $\eta$ be $\alpha \mapsto (\text{bool}, \text{nat}, R)$.

We have $(V_1, V_2) \in \mathcal{V}[\sigma]_\eta$.

Hence, $(\text{pack } V_1, \text{bool as } \exists \alpha. \sigma, \text{pack } V_2, \text{nat as } \exists \alpha. \sigma) \in \mathcal{V}[\exists \alpha. \sigma]$.

Proof of $((\text{not}, \text{tt}), (\text{succ}, 0)) \in \mathcal{V}[\!(\alpha \to \alpha) \times \alpha\!]_\eta$ (1)
Existential types

Consider $V_1 \triangleq (\text{not}, \text{tt})$, and $V_2 \triangleq (\text{succ}, 0)$ and $\sigma \triangleq (\alpha \to \alpha) \times \alpha$.

Let $R \in R(\text{bool}, \text{nat})$ be $\{(\text{tt}, 2n), (\text{ff}, 2n + 1) \mid n \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (\text{bool}, \text{nat}, R)$.

We have $(V_1, V_2) \in \mathcal{V}[\sigma]_\eta$.

Hence, $(\text{pack } V_1, \text{bool as } \exists \alpha. \sigma, \text{pack } V_2, \text{nat as } \exists \alpha. \sigma) \in \mathcal{V}[(\exists \alpha. \sigma)]$.

**Proof** of $((\text{not}, \text{tt}), (\text{succ}, 0)) \in \mathcal{V}[(\alpha \to \alpha) \times \alpha]_\eta$ (1)

We have $(\text{tt}, 0) \in \mathcal{V}[\alpha]_\eta$, since $(\text{tt}, 0) \in R$.

We also have $(\text{not}, \text{succ}) \in \mathcal{V}[\alpha \to \alpha]_\eta$, which proves (1).
Existential types

Consider $V_1 \triangleq (\text{not}, \text{tt})$, and $V_2 \triangleq (\text{succ}, 0)$ and $\sigma \triangleq (\alpha \to \alpha) \times \alpha$. Let $R \in \mathcal{R}(\text{bool}, \text{nat})$ be $\{(\text{tt}, 2n), (\text{ff}, 2n + 1) \mid n \in \mathbb{N}\}$ and $\eta$ be $\alpha \mapsto (\text{bool}, \text{nat}, R)$.

We have $(V_1, V_2) \in \mathcal{V}[\sigma]_\eta$.

Hence, $(\text{pack } V_1, \text{bool as } \exists \alpha. \sigma, \text{pack } V_2, \text{nat as } \exists \alpha. \sigma) \in \mathcal{V}[\exists \alpha. \sigma]$.

**Proof** of $((\text{not}, \text{tt}), (\text{succ}, 0)) \in \mathcal{V}[{(\alpha \to \alpha) \times \alpha}]_\eta$ (1)

We have $(\text{tt}, 0) \in \mathcal{V}[\alpha]_\eta$, since $(\text{tt}, 0) \in R$.

We also have $(\text{not}, \text{succ}) \in \mathcal{V}[\alpha \to \alpha]_\eta$, which proves (1).

Indeed, assume $(W_1, W_2) \in \mathcal{V}[\alpha]_\eta$. Then $(W_1, W_2)$ is either of the form

- $(\text{tt}, 2n)$ and $(\text{not } W_1, \text{succ } W_2)$ reduces to $(\text{ff}, 2n + 1)$, or
- $(\text{ff}, 2n + 1)$ and $(\text{not } W_1, \text{succ } W_2)$ reduces to $(\text{tt}, 2n + 2)$.

In both cases, $(\text{not } W_1, \text{succ } W_2)$ reduces to a pair in $R$.

Hence, $(\text{not } W_1, \text{succ } W_2) \in \mathcal{E}[\alpha]_\eta$. 

Example
Representation independence

A client of an existential type $\exists \alpha. \tau$ should not see the difference between two implementations $N_1$ and $N_2$ of $\exists \alpha. \tau$ with witness types $\rho_1$ and $\rho_2$.

A client $M$ has type $\forall \alpha. \tau \to \sigma$ with $\alpha \notin \text{fv}(\sigma)$; it must use the argument parametrically, and the result is independent of the witness type.

Assume that $\rho_1$ and $\rho_2$ are two closed representation types and $R$ is in $\mathcal{R}(\rho_1, \rho_2)$. Let $\eta$ be $\alpha \mapsto (\rho_1, \rho_2, R)$.

Suppose that $N_1 : \tau[\alpha \mapsto \rho_1]$ and $N_2 : \tau[\alpha \mapsto \rho_2]$ are two equivalent implementations of the operations, i.e. such that $(N_1, N_2) \in \mathcal{E}[\tau]_\eta$.

A client $M$ satisfies $(M, M) \in \mathcal{E}[\forall \alpha. \tau \to \sigma]_\eta$. Thus $(M \rho_1 N_1, M \rho_2 N_2)$ is in $\mathcal{E}[\sigma]$ (as $\alpha$ is not free in $\sigma$).

That is, $M \rho_1 N_1 \simeq_\sigma M \rho_2 N_2$: the behavior with the implementation $N_1$ with representation type $\rho_1$ is indistinguishable from the behavior with the implementation $N_2$ with representation type $\rho_2$. 
How do we deal with recursive types?

Assume that we allow equi-recursive types.

\[ \tau ::= \ldots | \mu \alpha.\tau \]

A naive definition would be

\[ \mathcal{V}[\mu \alpha.\tau]_\eta = \mathcal{V}[\alpha \mapsto \mu \alpha.\tau]_\eta \]

But this is ill-founded.

The solution is to use indexed-logical relations.

We use a sequence of decreasing relations indexed by integers (fuel), which is consumed during unfolding of recursive types.
Step-indexed logical relations

(a taste)

We define a sequence $\mathcal{V}_k[\tau]_\eta$ indexed by natural numbers $n \in \mathbb{N}$ that relates values of type $\tau$ up to $n$ reduction steps. Omitting typing clauses:

$$
\begin{align*}
\mathcal{V}_k[B]_\eta &= \{(tt, tt), (ff, ff)\} \\
\mathcal{V}_k[\tau \to \sigma]_\eta &= \{(V_1, V_2) \mid \forall j < k, \forall (W_1, W_2) \in \mathcal{V}_j[\tau]_\eta, (V_1 \, W_1, V_2 \, W_2) \in \mathcal{E}_j[\sigma]_\eta\} \\
\mathcal{V}_k[\alpha]_\eta &= \eta_R(\alpha).k \\
\mathcal{V}_k[\forall \alpha. \tau]_\eta &= \{(V_1, V_2) \mid \forall \rho_1, \rho_2, R \in \mathcal{R}^k(\rho_1, \rho_2), \forall j < k, (V_1 \, \rho_1, V_2 \, \rho_2) \in \mathcal{V}_j[\tau]_\eta, \alpha \mapsto (\rho_1, \rho_2, R)\} \\
\mathcal{V}_k[\mu \alpha. \tau]_\eta &= \mathcal{V}_{k-1}[\alpha \mapsto \mu \alpha. \tau][\tau]_\eta \\
\mathcal{E}_k[\tau]_\eta &= \{(M_1, M_2) \mid \forall j < k, M_1 \Downarrow_j V_1 \\
&\quad \quad \implies \exists V_2, M_2 \Downarrow V_2 \land (V_1, V_2) \in \mathcal{V}_{k-j}[\tau]_\eta\}
\end{align*}
$$

By $\Downarrow_j$ means reduces in $j$-steps.

$\mathcal{R}^j(\rho_1, \rho_2)$ is composed of sequences of decreasing relations between closed values of closed types $\rho_1$ and $\rho_2$ of length (at least) $j$. 
Introduction

Observational equivalence

Logical rel in $\lambda_{st}$

Logical rel. in $F$

Applications

Extensions

Step-indexed logical relations

(a taste)

The relation is asymmetric.

If $\Delta; \Gamma \vdash M_1, M_2 : \tau$ we define $\Delta; \Gamma \vdash M_1 \preceq M_2 : \tau$ as

$$\forall \eta \in R^k_\Delta(\delta_1, \delta_2), \forall (\gamma_1, \gamma_2) \in G_k[\Gamma], \ (\gamma_1(\delta_1(M_1)), \gamma_2(\delta_2(M_2)) \in E_k[\tau]_\eta$$

and

$$\Delta; \Gamma \vdash M_1 \sim M_2 : \tau \triangleq \bigwedge \left\{ \begin{array}{l} \Delta; \Gamma \vdash M_1 \preceq M_2 : \tau \\ \Delta; \Gamma \vdash M_2 \preceq M_1 : \tau \end{array} \right\}$$

Notations and proofs get a bit involved...

Notations may be simplified by introducing a *later* guard $\triangleright$ to capture incrementation of the index and avoid the explicit manipulation of integers (but the meaning remains the same).
Logical relations for $F^\omega$?

Logical relations can be generalized to work for $F^\omega$, indeed.

There is a slight complication though in the interpretation of type functions.

This is out of this course scope, but one may, for instance, read [Atkey, 2012].


