Yale University
Department of Computer Science

Factoring Report

Adam Poswolsky
Yale University

Advisor: Carsten Schürmann
Yale University

YALEU/DCS/TR-1256
September 29, 2003 – revised: November 19, 2003

Submitted in partial fulfillment of the requirements for CS-690

Abstract

Factoring is the process of removing deep backtracking from a logical program. This work focuses on a factoring algorithm designed to take a logic program in Twelf and attempt to completely factor (remove all backtracking) it into the functional language Delphin. We discuss the design and implementation and formally prove that a factored program has the same semantics as the original program.

1 Introduction

Logic programs are known to incur a lot of backtracking during program execution. Often, however, backtracking is unnecessary and should be avoided for the sake of efficient execution and for the benefit of static analysis. For a subset of logic programs, which we call *well-moded* logic programs (whose arguments are assigned a fixed input/output role), backtracking can indeed be removed. If this removal is manual then it is both more prone to error and usually less elegant than if it is done automatically. We have designed and implemented an automatic technique called "factoring" that we describe in this paper.

Factoring proves to be an invaluable tool. Factoring removes all backtracking and is semantics preserving – that is, the computational behavior of a logic program and its factored cousin are indistinguishable. With respect to static analysis, factoring assists in coverage checking, which decides if a set of patterns covers all cases and relies therefore on the ability to locate sets of applicable cases at any point in the abstract execution of a logic program. Thus, the main applications of factoring include compilation of *well-moded* logic programs into low level languages via a cascade of intermediate languages and coverage checking. Also, thanks to factoring, logic programs can be directly analyzed for output coverage (i.e., all cases of pattern-matching against return arguments of subgoals are covered) without awkwardly rewriting the logic program.

In this paper we discuss the design and implementation of a *factoring* algorithm in the setting of Delphin [11]. Delphin is a pure functional programming language incorporating a two-level design that distinguishes cleanly between the tasks of representing data and programming with data. One level is the logical framework LF, serving as Delphin's data representation language. The other level is a type theory designed to support programming using pattern matching and recursion. Delphin has the look and feel of a regular functional language (such as SML), but also allows us to program with higher-order, dependently-typed data structures such as proofs and typing derivations in a natural and intuitive way. Delphin programs can be created in two different ways: programs can be parsed (as a normal functional language) or they can be automatically generated to solve a logic program represented in Twelf (the LF level of Delphin). Our factoring algorithm applies to programs created via the latter method.

Intuitively, it should be clear that not every logic program can be converted into an observational equivalent one that is deterministic. In fact, this problem is undecidable as shown by Sawamur and Takeshima [7]. However, it is decidable for a smaller subclass. The Mercury Project [4], for example, provides two different ways of running logic programs; it provides both a fast deterministic and a somewhat slower nondeterministic algorithm. The Mercury programmer can signal the logic programming engine by a pragma, which algorithm

to use. Mercury subsequently checks, if the given logic program is really deterministic before it grants the request.

In comparison to the Mercury project, our goals and contributions are different. While Mercury implements a checker for determinacy, our criterion can handle and execute a much larger class of programs, namely those that can be easily converted into deterministic programs using program transformation (i.e. *factoring*).

Factoring a *well-moded* logic program proceeds in two steps. In a first straightforward step, a program is transformed into a functional language $\mathcal{N}_\omega$ whose operational semantics reflects that of the logic programming language directly. Programs in $\mathcal{N}_\omega$, for example, can fail and subsequently cause backtracking. However, this step does eliminate the need for logic variables. The correspondence between backtracking and what appears to be *redundant cases* in $\mathcal{N}_\omega$ will be illustrated.

We do not focus on the first straightforward step except to present the semantics for $\mathcal{N}_\omega$. In the second step, $\mathcal{N}_\omega$ programs are transformed into Delphin ($\mathcal{T}_\omega$) whose operational semantics is deterministic. The main result of this paper is that factoring provides a decidable syntactic criterion for deterministic logic programs.

This paper is organized as follows. In Section 2 we discuss a basic example which will be used to motivate our work. In Section 3 we discuss logical frameworks in general, and how it is used in Delphin. The Delphin language itself is described in Section 4. In Section 5 we discuss both $\mathcal{N}_\omega$ (Section 5.2) and Delphin ($\mathcal{T}_\omega$) (Section 5.3). Then, in section 6, we will discuss factoring. We will then show in Section 7 interesting Meta theoretical properties. We will discuss some implementation issues in Section 8 and conclude the paper in Section 9.

2 Qualifying example

Throughout this paper we will focus on one qualifying example and show how it extends to all logical programs which have the input/output behavior necessary to facilitate its conversion into a functional paradigm. From this point we will refer to such programs as *deterministic*. By deterministic, we mean that by the nature of the logic program, there exists a way to represent it without using backtracking.

We will focus on a subset of ML which we call Mini-ML. To conserve space we only discuss the fragment containing natural numbers, functions, and recursion, but the example scales to other constructs as well.

$$
\begin{array}{lrcl}
\text{Types} & \tau & ::= & \tau_1 \Rightarrow \tau_2 \\
\text{Expressions} & e & ::= & x \mid e_1 @ e_2 \mid \mathbf{fn}\ x : \tau . e \mid \mathbf{z} \mid \mathbf{s}\ e \\
 & & & \mid (\mathbf{case}\ e\ \mathbf{of}\ \mathbf{z} \Rightarrow\ e_1 \mid \mathbf{s}\ x \Rightarrow\ e_2) \\
 & & & \mid \mathbf{rec}\ x . e
\end{array}
$$

Note that we will use Delphin to evaluate Mini-ML expressions. The evaluation rules for this Mini-ML are in Figure 1.

Expressions and inference rules have very natural encodings in the LF logical framework. LF extends the simply-typed λ-calculus with dependent types. We write $\Gamma \vdash_\Sigma M : A$ for the LF typing judgment, where Γ is the LF context, M is an object and A is the object's type.

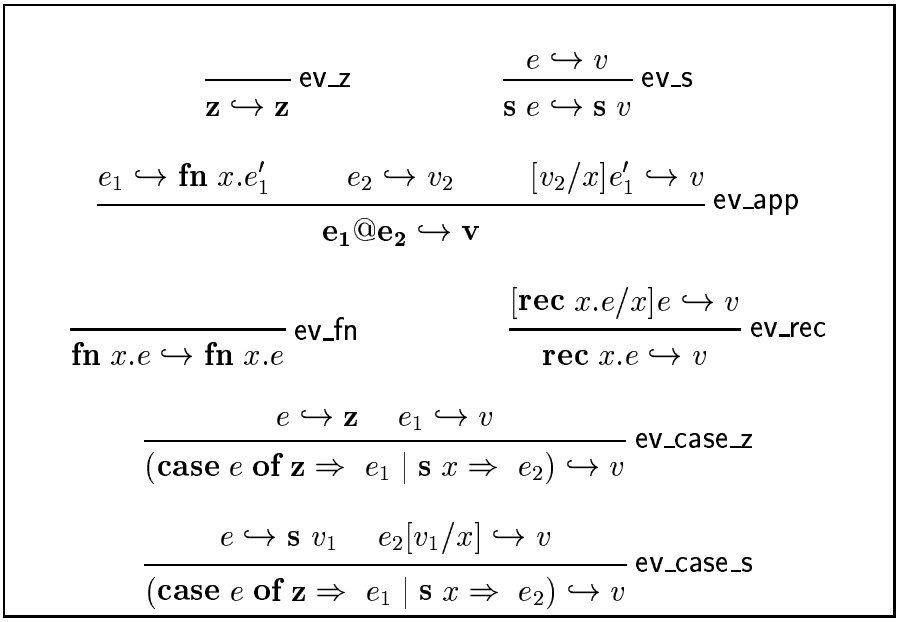

Figure 1: Evaluation rules for Mini-ML

In the LF logical framework we represent judgments as types and derivations as objects. For example, the encoding of a derivation of $\mathcal{D}$ of typing judgment $e \hookrightarrow v$ is captured by the definition of the type family "of":

$$\ulcorner \mathcal{D} :: e \hookrightarrow v \urcorner = \ulcorner \cdot \urcorner \vdash_{\Sigma} \ulcorner \mathcal{D} \urcorner \Uparrow \text{eval}\ \ulcorner e \urcorner\ \ulcorner v \urcorner \tag{1}$$

where we write $\ulcorner \cdot \urcorner$ for the representation function, and Σ for the LF signature that captures the representation of each Mini-ML type, each language construct and each individual rule.

We will focus on programs, such as Mini-ML, which can easily be represented in LF, and explain how to convert it to Delphin (which requires factoring) whenever possible.

Further encoding examples of Mini-ML will be presented in subsequent sections.

3 Logical Frameworks

As already mentioned in the previous section, we will focus on logic programs that have been written in a Logical Framework and attempt to convert them to Delphin.

A logical framework must guarantee that the underlying concepts being represented in the framework are represented *adequately*. For example, the correctness of a Delphin program that infers types and typing derivations of expressions of some calculus rests in the first place on an adequate encoding of expressions and typing derivations.

Of the many available logical frameworks, we have chosen to focus on converting logic programs represented in LF [3]. This is most convenient since LF already serves as Delphin's data representation language [11]. LF is expressive enough to represent many concepts in computer science, logic, and formal methods elegantly, and the conciseness, adequacy, and efficiency of LF representations make them superior to standard datatypes.

The type-theoretic aspect of this paper revolves around LF and a brief introduction is warranted here. In addition to the standard syntactic categories for objects, types, and kinds, we will also use substitutions in a critical way throughout this paper so we briefly introduce them here. (see also, for example, [1]).

$$
\begin{array}{lrcl}
\textit{Kinds} & K & ::= & \mathrm{type} \mid \Pi x{:}A.\, K \\
\textit{Atomic Types} & B & ::= & a \mid B\ M \\
\textit{Types} & A & ::= & B \mid \Pi x{:}A_1.\, A_2 \\
\textit{Objects} & M & ::= & x \mid c \mid M_1\ M_2 \mid \lambda x{:}A.\, M \\
\textit{Signatures} & \Sigma & ::= & \cdot \mid \Sigma, a : K \mid \Sigma, c : A \\
\textit{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x : A \\
\textit{Substitutions} & \sigma & ::= & \cdot \mid \sigma, M/x
\end{array}
$$

A *term* may come form any of the syntactic levels. As usual, we identify α-equivalent terms. We take $\beta\eta$-conversion as the notion of definitional equality [3, 1]. Substitutions are capture-avoiding and written as $U[\sigma]$ or $V[\sigma]$ with the special form $U[M/x]$ and $V[M/x]$.

Signatures, contexts, and substitutions may not declare a variable or constant more than once, and renaming of bound variables may be applied tacitly to ensure that condition. Besides equality, the main judgment is typing $\Gamma \vdash U : V$, suppressing the fixed signature Σ. We always assume our signatures, contexts and types to be valid.

Example 1 (Propositional Hilbert Calculus) *An LF signature that encodes the Hilbert calculus in LF is given in Figure 2. Terms are captured by type* i*, formulas by* o*, and proofs in the Hilbert calculus by the type family* hil *:* o $\rightarrow$ `type`*. As usual we omit the leading* Π*-binders of variables whose types are inferable, denoted above by uppercase variable names.* □

```
imp   : o → o → o.
forall : (i → o) → o.
k     : hil (imp A (imp B A)).
s     : hil (imp (imp A (imp B C)) (imp (imp A B) (imp A C))).
mp    : hil (imp A B) → hil A → hil B.
```

Figure 2: The Propositional Hilbert Calculus

Type-checking and definitional equality on well-typed terms for LF are decidable. Every term is equal to a unique β-normal η-long form which we call *canonical form*. In this paper we assume that all terms are in canonical form, because it simplifies the presentation significantly. In the implementation this is achieved incrementally, first by an initial conversion of input terms to η-long form and later by successive weak-head normalization as terms are traversed.

Since it is perhaps not so well-known, we will give only the typing rules for substitutions, which are used pervasively in this paper.

$$\frac{}{\Gamma' \vdash \cdot : \cdot} \qquad \frac{\Gamma' \vdash \sigma : \Gamma \qquad \Gamma' \vdash M : A[\sigma]}{\Gamma' \vdash (\sigma, M/x) : (\Gamma, x{:}A)}$$

For a context $\Gamma = (x_1{:}A_1, \ldots, x_n{:}A_n)$, we define $\mathsf{id}_\Gamma = (x_1/x_1, \ldots, x_n/x_n)$ so that $\Gamma \vdash \mathsf{id}_\Gamma : \Gamma$.

Composition of substitutions is defined by $(\cdot) \circ \sigma = (\cdot)$ and $(M/x, \theta) \circ \sigma = (M[\sigma]/x, \theta \circ \sigma)$. We will only apply a substitution $\Gamma' \vdash \sigma : \Gamma$ to a term $\Gamma \vdash U : V$ or a substitution $\Gamma \vdash \theta : \Gamma''$ resulting in $\Gamma' \vdash U[\sigma] : V[\sigma]$ and $\Gamma' \vdash \theta \circ \sigma : \Gamma''$, respectively.

3.1 Representation of Mini-ML Deductive System

Figure 3 illustrates how we encode Mini-ML in LF. As usual we omit the leading Π-binders of variables whose types are inferable. Note that the "function" eval is represented as a type family.

```
exp : type.
z   : exp.
s   : exp -> exp.
app : exp -> exp -> exp.
fn  : (exp -> exp) -> exp.
fix : (exp -> exp) -> exp.
eval : exp -> exp -> type.
%mode eval +E -V.
```

Figure 3: Mini-ML setup

3.2 Encoding Mini-ML function as relations

From a functional programming perspective the LF type families "eval" and "exp" can be regarded as datatype declarations. Each constant can be seen as a constructor of the type family that is named in the head of the constant's type. For better readability, the constants in the Mini-ML encoding above are grouped in a way that clarifies this view. We can see the Mini-ML "eval" function represented in LF in Figure 4.

4 Delphin

Delphin is a functional programming language which is designed to allow programming with datatypes that consist of a fixed set of constructors along with dynamic extensions of these datatypes valid in some world. The core language that is presented in this paper has been implemented and can be accessed through [11]. Delphin's syntax is inspired by that of Standard ML of New Jersey. Delphin permits function definition by pattern matching and recursion.

```
ev_z: eval z z.
ev_s: eval (s E) (s V)
       <- eval E V.
ev_fn: eval (fn E) (fn E).
ev_rec: eval (rec E) V
       <- eval (E (rec E)) V.
ev_app: eval (app E1 E2) V.
       <- eval E1 (fn E1')
       <- eval E2 V2
       <- eval (E1' V2) V.
ev_case_z : eval (case E1 E2 E3) V'
             <- eval E1 z
             <- eval E2 (V').
ev_case_s : eval (case E1 E2 E3) V'
             <- eval E1 (s V1')
             <- eval (E3 V1') V'.
```

Figure 4: Mini-ML operational semantics

Its datatypes are essentially LF types, and the objects manipulated by Delphin programs are LF objects. Delphin's implementation also includes a type checker and an interpreter. Delphin's type system is deceptively simple, since it only provides type constructors for function and product spaces. Although these constructors provide dependent types, there is no mechanism that would allow programmers to define their own Delphin types. The current design does not allow programs to be polymorphic, but we plan to investigate the issue of polymorphism in future work.

4.1 Datatypes

Delphin's datatype declarations are LF signatures, which include declarations of constructors for type families and worlds. The Twelf system [6] is an implementation of the logical framework LF which is designed to facilitate developing, implementing, experimenting with, and verifying properties about deductive systems, such as the Mini-ML type system in the example given above. In fact, Twelf is an extraordinarily useful and effective tool for engineering, developing, and debugging representations of data, and we have accordingly chosen to use and parse Twelf signatures as Delphin datatypes.

Programmers are free to extend datatypes dynamically during evaluation as long as these extensions conform to the rules stipulated by the world in which a function is defined. We say that *a function cannot leave the world in which it lives during evaluation.* The idea of worlds is not new; it was introduced in [8], studied as a means of defining recursive functions in [9], and applied to reasoning by induction in [10]. Worlds have also been implemented in the Twelf system [6], and are instrumental for the problem of coverage checking.

4.2 Language Features and Converter

Delphin programs consist of variable declarations, value definitions, and function definitions. Local function definitions are also possible.

There is a built-in *converter* that will convert a logic program written in Twelf to some corresponding $\mathcal{N}_\omega$ code by taking each judgment and making a new case out of it. For example, the *converter* will translate the code in Figure 4 to what we see in Figure 5.

This is the expected result of converting the type family eval : exp → exp → type into eval : $\forall e$:exp $\exists v$:exp.T

```
eval :: all {e:exp} exists {v:exp} true

fun eval z = <z, <>>
  | eval (s E) =
    let
      val <V, <>> = eval E
    in
      (s V)
    end
  | eval (fn T E) = <fn T E, <>>
  | eval (app E1 E2) =
    let
      val <fn T E1', <>> = eval E1
      val <V2, <>> = eval E2
    in
      eval (E1' V2)
    end
  | eval (fix E) =
      eval (E (fix E))
  | eval (case E1 E2 ([x:exp] E3 x)) = let
                                          val <z, <>> =  eval E1
                                          val <V, <>> =  eval E2
                                       in
                                         <V, <>>
                                       end
  | eval (case E1 E2 ([x:exp] E3 x)) = let
                                          val <s V1, <>> =  eval E1
                                          val <V, <>> =  eval (E3 V1)
                                       in
                                         <V, <>>
                                       end
```

Figure 5: A Mini-ML evaluator in $\mathcal{N}_\omega$ (result of converter from LF)

The first instruction declares the variable `eval` to be of type $\forall e : \text{exp}.\, \exists v : \text{exp}.\, \top$. In the text we write types in mathematical notation; in Delphin source code types are given in corresponding ASCII notation. $\forall$ stands for the Delphin-level dependent function space, and $\exists$ stands for the Delphin-level dependent product space; these should not be confused with the LF-level Π. $\top$ corresponds to the `unit` type of ML.

The next instruction defines the program `eval` by pattern matching; it closely resembles an ML function declaration. `<z, <>>` in the first case of `eval` illustrates the syntax for pairs. `<>` is `()` in ML, and has type $\top$. When programming an evaluator for Mini-ML programs there is no need to define a notation for substitution or environments. These concepts are provided by LF implicitly; the programmer can take advantage of them by simply applying $\texttt{E}_3$ to $\texttt{V}_1$ in the second case of `case`, $\texttt{E}'_1$ to $\texttt{V}_2$ in the `app` case, and `E` to `(fix E)` in the `fix` case. Therefore, juxtaposition can have one of two meanings. Depending on where it occurs, it is either an LF-level application or a Delphin-level application.

Note that the converter results in code that makes sense in $\mathcal{N}_\omega$ It is not yet fit for execution in a functional paradigm, since there is this question over which case is to be used. This refers to the *match-redundant* problem we see with `case`. Note that the process of *factoring* is used to merge redundant cases together, whenever possible. If an $\mathcal{N}_\omega$ program can be factored, the result is guaranteed not to have redundant cases, and it can then be executed using a deterministic operational semantics ($\mathcal{T}_\omega$) that we would intuitively expect.

5 Semantics

5.1 Notation

We introduce the following notational conventions.

5.1.1 Variable Names

We use the following convention for naming meta-variables.

P	=	Delphin Programs
L	=	LF Terms and Block Variables
F	=	Result of Factoring (See Section 6)
V	=	A subset of Delphin Programs which constitute values (See [11])
A	=	Stands for a variable of any of the above (All meta-variables above also match with this)
M	=	Stands for objects on the SC continuation stack (Section 5.2)
$\bot$	=	Described in Section 5.1.3

Factoring only has an effect on Delphin programs (**P**). As we will see, LF and Block variables are left untouched by this process.

Note that the introduction of **A** is just used to simplify the subsequent inference rules. In many cases there would be multiple versions to deal with application and with pairs, but this formalization allows us to eliminate those extra rules.

5.1.2 Case Statements

Delphin case statements are defined as a list Ω such that

$$\Omega ::= \cdot \quad | \quad \Omega, (\Psi \triangleright \psi \mapsto P)$$

5.1.3 Introduction to $\perp$

There are many instances of programs which do not evaluate to any result. The result of evaluating such a program will be to return $\perp$. Note that in this report, we do not consider $\perp$ a value (it is separate and will have a separate semantics)

Note that there is a difference between a program which does not terminate and a program which returns $\perp$. A non-terminating program also does not evaluate to a a value, but should not be confused with thinking it returns $\perp$. The only way a program will return $\perp$ is if it is trying to match with a case statement and cannot find any that match. This is analogous to a *non-exhaustive match failure* in ML.

5.2 $\mathcal{N}_\omega$ Operational Semantics

Recall that to convert a logic program to Delphin's operational semantics, we proceed in two steps. We first do a conversion from Twelf into $\mathcal{N}_\omega$ and then we show how to get from $\mathcal{N}_\omega$ to Delphin ($\mathcal{T}_\omega$). In this section we discuss $\mathcal{N}_\omega$ which is allowed to have redundant cases. We define an operational semantics, using continuations, which we claim (without proof) has the same power as Twelf's logic programming engine. More specifically, the result of running an $\mathcal{N}_\omega$ program is exactly the same as running it (pre-converted) in Twelf's logic engine.

We will introduce two *continuation* stacks. We introduce a SC (success continuation) and a FC (failure continuation) which are defined in Figure 6. Notice that both continuation stacks contain closures. The reason for this is that we are putting aside information on what needs to be executed and we need to include the environment in which it was to be executed.

$$\begin{array}{rcl} SC & = & \cdot \quad | \quad SC, (\Gamma; \eta; M) \\ M & = & \lambda z.\mathbf{let}\ \mathbf{x} = V \text{ in } P \quad | \quad \lambda z.\mathbf{pair1}\langle V; P\rangle \quad | \quad \lambda z.\mathbf{pair2}\langle V_1; V_2\rangle \\ & & | \quad \lambda z.\mathbf{app1} P\ A \quad | \quad \lambda z.\mathbf{app2} P\ V \\ FC & = & \cdot \quad | \quad FC, (\Gamma; \eta; SC; \mathbf{case}\Omega) \end{array}$$

Figure 6: Continuation Stacks used in $\mathcal{N}_\omega$ Operational Semantics

In order to capture the deep backtracking effect that Twelf's logical engine exhibits we introduce both continuations to mirror how Twelf would execute the program. The success continuation stack (SC) is used to store what should be done next if the current operation results in a value. The failure continuation stack (FC) holds what to do when the computation fails to return in a value. As we see, the only place where there is some nondeterminism is in the case statements. Therefore, the FC only holds case statements so we know what to continue trying in case it fails.

We now focus on describing the rules relating to the stack (Figure 7). The operators are of the form

$$FC; SC \vdash V \overset{SC}{\hookrightarrow} V'$$

and

$$FC \overset{FC}{\hookrightarrow} V$$

The first set of rules, $\overset{SC}{\hookrightarrow}$, refers to a situation when we have finished a partial step which resulted in V and now we look into SC to continue the desired computation. Note that we must include FC since if it fails in this step, it still needs to go into the FC stack.

The second set of rules, $\overset{FC}{\hookrightarrow}$, refers to a situation when we have hit a point when the evaluation has failed so we need to go back and continue with what is on the FC stack since that contains the parts in the execution where we had possibly more than one choice of what to do.

$$\frac{}{FC; \cdot \vdash V \overset{SC}{\hookrightarrow} V}\ \textsf{ev\_SC\_empty}$$

$$\frac{\Gamma; \eta; FC; SC \vdash M[V'/z] \hookrightarrow V}{FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash V' \overset{SC}{\hookrightarrow} V}\ \textsf{ev\_SC\_nonempty} \qquad \frac{\Gamma; \eta; FC; SC \vdash M[V'/z] \hookrightarrow \bot}{FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash V' \overset{SC}{\hookrightarrow} \bot}\ \textsf{ev\_SC\_nonempty}\bot$$

$$\frac{\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V}{(FC, (\Gamma; \eta; SC; \textbf{case}\ \Omega)) \overset{FC}{\hookrightarrow} V}\ \textsf{ev\_FC\_nonempty} \qquad \frac{\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot}{(FC, (\Gamma; \eta; SC; \textbf{case}\ \Omega)) \overset{FC}{\hookrightarrow} \bot}\ \textsf{ev\_FC\_nonempty}\bot$$

$$\frac{}{\cdot \overset{FC}{\hookrightarrow} \bot}\ \textsf{ev\_FC\_empty}$$

Figure 7: Continuation Stacks Rules resulting in Value (on left) and in $\bot$ (on right)

With the notion of the stacks set up, we now introduce the actual rules for $\mathcal{N}_\omega$. Since we introduce this concept of evaluating to $\bot$ we have two sets of rules (Figure 8 and Figure 9)

Most of the rules are defined as we would expect, and the interested reader is directed to [11] for a further in-depth explanation. We will describe here some of the basic differences in these semantics which make them unique.

$$\frac{FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V}{\Gamma; \eta; FC; SC \vdash L \hookrightarrow V}\ \mathsf{ev\_LF}$$

$$\frac{\Gamma; \eta; FC; SC \vdash \eta(\mathbf{x}) = V' \quad FC; SC \vdash V' \overset{SC}{\hookrightarrow} V}{\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow V}\ \mathsf{ev\_var} \qquad \frac{FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V}{\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow V}\ \mathsf{ev\_unit}$$

$$\frac{FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V}{\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow V}\ \mathsf{ev\_\Lambda} \qquad \frac{\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow V}\ \mathsf{ev\_rec}$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{let}\ \mathbf{x} = z \text{ in } P_2)) \vdash P_1 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V}\ \mathsf{ev\_let} \qquad \frac{\Gamma; (\eta, V'/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mathbf{let}\ \mathbf{x} = V' \text{ in } P_2 \hookrightarrow V}\ \mathsf{ev\_let\_SC}$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow V}\ \mathsf{ev\_app}$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\mathbf{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mathbf{app1}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V}\ \mathsf{ev\_app\_SC1} \qquad \frac{\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mathbf{app2}\ P_1'\ V_2 \hookrightarrow V}\ \mathsf{ev\_app\_SC2}$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{pair1}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \langle A_1; P_2\rangle \hookrightarrow V}\ \mathsf{ev\_pair}$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{pair2}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mathbf{pair1}\langle V_1; P_2\rangle \hookrightarrow V}\ \mathsf{ev\_pair\_SC1} \qquad \frac{FC; SC \vdash \langle V_1; V_2\rangle \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \mathbf{pair2}\langle V_1; V_2\rangle \hookrightarrow V}\ \mathsf{ev\_pair\_SC2}$$

$$\frac{\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V}{\Gamma; \eta; FC; SC \vdash \text{case } \Omega \hookrightarrow V}\ \mathsf{ev\_case}$$

$$\frac{\psi \circ \eta' = \eta \quad \Gamma; \eta'; (FC, (\Gamma; \eta; SC; \mathbf{case}\ \Omega)); SC \vdash P \hookrightarrow V}{\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V}\ \mathsf{ev\_yes} \qquad \frac{\psi \circ \eta' \neq \eta \quad \Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V}{\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V}\ \mathsf{ev\_no}$$

$$\frac{FC \overset{FC}{\hookrightarrow} V}{\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow V}\ \mathsf{ev\_nil}$$

Figure 8: $\mathcal{N}_\omega$ Operational Semantics resulting in a value

The first important rule to mention is ev_LF. As mentioned earlier, $\mathbf{L}$ matches both LF and Block objects. We just define them to evaluate to themselves. Therefore, we see when it gets to such a variable, it simply goes into the SC stack to see what to compute next. We also see that this extra rule allows us to simply ev_app and ev_pair to avoid multiple versions of the same inference rule.

The ev_let rule determines how to evaluate *let* statements in Delphin. When we have a program of the form "let $\mathbf{x} = P_1$ in P_2" we want to first evaluate P_1 and then take the result and continue to evaluate P_2 with $\mathbf{x}$ mapped to the result of the evaluation of P_1. If you were to trace out the execution of ev_let you will find that it evaluates P_1 and then continues to evaluate P_2 in ev_let_SC. Notice that if it were to fail in the evaluation of P_1 it can go into the FC and never evaluate P_2 which is the behavior we desire.

Note that the only rule which explicitly goes into the FC is ev_nil. This rule is important

because it handles the case when we have a case statement and all cases have been exhausted (either by failing to match or other failures along the way).

The analysis of ev_app and ev_pair are similar to that of ev_let and will therefore be omitted.

When we evaluate a case statement we check if there exists an η' such that $\psi \circ \eta' = \eta$ If this is the case, it means that the environment η matches the case described in $(\Psi \triangleright \psi \mapsto P)$ and we can try to execute that code that is relevant to this case (ev_yes). Note that the current environment and SC are stored with the rest of the cases in the FC, so that if this execution fails along the way it will try another case. Similarly, in the ev_no case we know that the top case does not apply, so we just repeat with the rest of the cases.

$$\frac{FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} \bot}{\Gamma; \eta; FC; SC \vdash L \hookrightarrow \bot}\ \mathsf{ev\_LF}\bot$$

$$\frac{\Gamma; \eta; FC; SC \vdash \eta(\mathbf{x}) = V' \quad FC; SC \vdash V' \overset{SC}{\hookrightarrow} \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow \bot}\ \mathsf{ev\_var}\bot \qquad \frac{FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} \bot}{\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow \bot}\ \mathsf{ev\_unit}\bot$$

$$\frac{FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} \bot}{\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow \bot}\ \mathsf{ev\_\Lambda}\bot \qquad \frac{\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow \bot}\ \mathsf{ev\_rec}\bot$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{let}\ \mathbf{x} = z \text{ in } P_2)) \vdash P_1 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow \bot}\ \mathsf{ev\_let}\bot \qquad \frac{\Gamma; (\eta, V'/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{let}\ \mathbf{x} = V' \text{ in } P_2 \hookrightarrow \bot}\ \mathsf{ev\_let\_SC}\bot$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow \bot}\ \mathsf{ev\_app}\bot \qquad \frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\mathbf{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{app1}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow \bot}\ \mathsf{ev\_app\_SC1}\bot$$

$$\frac{\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{app2}\ P_1'\ V_2 \hookrightarrow \bot}\ \mathsf{ev\_app\_SC2}\bot$$

$$\frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{pair1}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \langle A_1; P_2\rangle \hookrightarrow \bot}\ \mathsf{ev\_pair}\bot \qquad \frac{\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{pair2}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{pair1}\langle V_1; P_2\rangle \hookrightarrow \bot}\ \mathsf{ev\_pair\_SC1}\bot$$

$$\frac{FC; SC \vdash \langle V_1; V_2\rangle \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \mathbf{pair2}\langle V_1; V_2\rangle \hookrightarrow \bot}\ \mathsf{ev\_pair\_SC2}\bot$$

$$\frac{\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot}{\Gamma; \eta; FC; SC \vdash \text{case } \Omega \hookrightarrow \bot}\ \mathsf{ev\_case}\bot$$

$$\frac{\psi \circ \eta' = \eta \quad \Gamma; \eta'; (FC, (\Gamma; \eta; SC; \mathbf{case}\ \Omega)); SC \vdash P \hookrightarrow \bot}{\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot}\ \mathsf{ev\_yes}\bot \qquad \frac{\psi \circ \eta' \neq \eta \quad \Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot}{\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot}\ \mathsf{ev\_no}\bot$$

$$\frac{FC \overset{FC}{\hookrightarrow} \bot}{\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow \bot}\ \mathsf{ev\_nil}\bot$$

Figure 9: $\mathcal{N}_\omega$ Operational Semantics resulting in $\bot$

The rules for evaluation to $\bot$ are a straightforward adjustment from the rules that evaluate to a value. After careful analysis of the rules, the reader should notice that any program that evaluates to $\bot$ must be the result of a *non-exhaustive match failure* and its derivation will include a top reference to ev_FC_empty (Figure 7).

5.3 Delphin's Operational Semantics ($\mathcal{T}_\omega$)

The $\mathcal{N}_\omega$ operational semantics, although valid, do not make much sense in a *functional* language. Notice in Figure 5 that when one looks at the code we have a *match redundant* situation which is not allowed in any other functional programming language, and also shall not be allowed in Delphin. The implementation presented for $\mathcal{N}_\omega$ allow for this deep backtracking which is what factoring strives to remove.

One possible solution is to require the logic programmer to enforce that no two judgments have the same head. Therefore, the logic programmer would have to rewrite the logic program in Figure 4. One possible conversion is to remove the ev_case_z and ev_case_s with the code shown in Figure 10.

```
eval : exp -> exp -> type.  %name eval D.
%mode eval +E -V.
eval' : exp -> exp -> (exp -> exp) -> exp -> type.
%mode eval' +E1 +E2 +E3 -V.
ev_case : eval (case E1 E2 E3) V'
            <- eval E1 V
            <- eval' V E2 E3 V'.
eval'_z : eval' z E2 E3 V'
          <- eval E2 V'.
eval'_s : eval' (s V) E2 E3 V'
          <- eval (E3 V) V'.
```

Figure 10: Part of a new clean version of Mini-ML

Notice that although this is perfectly valid, it is quite cumbersome on the programmer and forcing the programmer to add additional judgments and type families simply to avoid this issue should be avoided whenever possible. This is the motivation for factoring. We want to show that if a program is factored, then we can run it with a simpler operational semantics which follow a *functional* paradigm. Since the $\mathcal{N}_\omega$ operational semantics mirrors Twelf's logic programming engine, we know that that we will get the same result as if we executed the logic program in Twelf.

Note that the result of *factoring* removes all need of backtracking. This already suggests that the class of logic programs that can be factored is limited to those that are "deterministic" ([12] [4]). This is discussed more in Section 6.2.

We now present Delphin's operational semantics. Again the reader is directed to [11] for a more detailed analysis. The rules presented here have the same semantics but differ slightly in the presentation. A brief overview of the rules (see Figure 11 and Figure 12) follow.

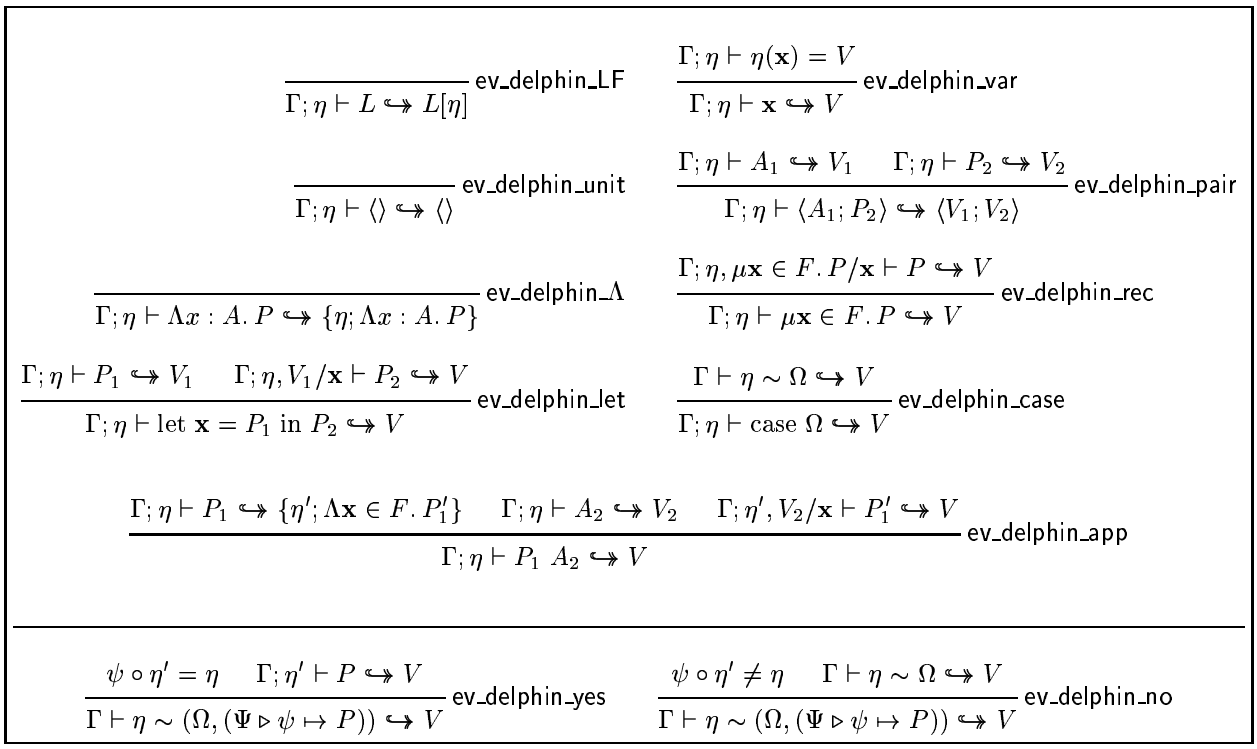

Figure 11: Delphin ($\mathcal{T}_\omega$) Operational Semantics resulting in a value

An important rule again is ev_delphin_LF which handles what we do when we reach a LF or Block variable. Notice that any such variable evaluates to itself.

Notice that the A variable is used in ev_delphin_app and ev_delphin_pair to avoid the multiple definition of the rule we see in [11].

The most significant part of this simpler operational semantics, are the rules to handle Delphin case statements. We see if we find a matching case, it uses ev_delphin_yes, which just executes that one case. Therefore, if there are *redundant cases*, those other cases will never be considered. However, we will show that if a program is *factored*, it will never have these redundant cases, and the final result would match.

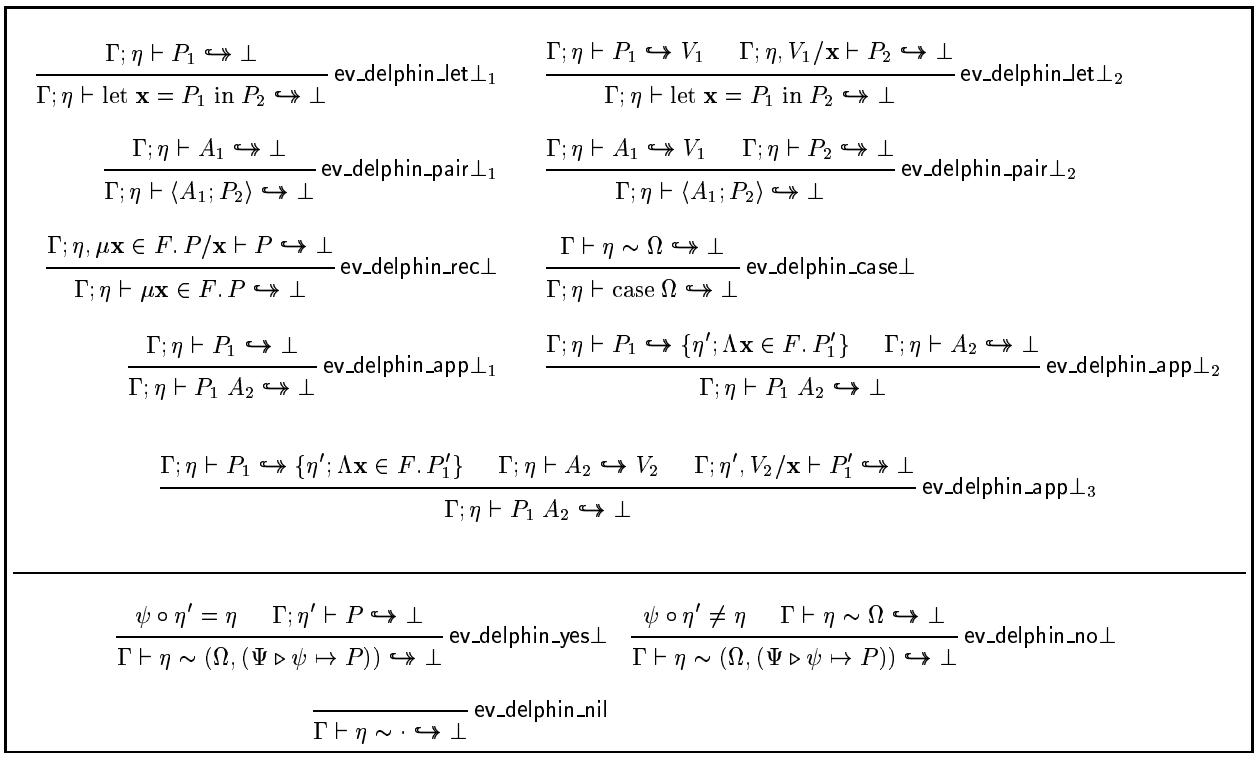

Figure 12: Delphin ($\mathcal{T}_\omega$) Operational Semantics resulting in a $\bot$

The rules for $\hookrightarrow$ that result in $\bot$ are a straight-forward definition based on Figure 11. Notice that rules with multiple premises, such as ev_delphin_let need to have multiple rules in the $\bot$ case (ev_delphin_let$\bot_1$ and ev_delphin_let$\bot_2$) to capture that either one of the premises could fail. Note that in the $\mathcal{N}_\omega$ operational semantics, this issue did not come up since we followed a continuation based approach and the inference rules had single premises.

6 Factoring

6.1 Introduction

The central insight and main contribution of this work is *factoring*, which converts logical programs that run in $\mathcal{N}_\omega$ into programs that run on Delphin's real operational semantics. Factoring attempts to *merge* redundant cases together. Note that the merging of two cases not only facilitates the use of a functional paradigm, but also eliminates what would otherwise be duplicate work. If there are two cases which share some of the same premises, when one fails, it would be nice to have saved the result of the premises that we would have to repeat. However, if it is merged together, we know longer have to worry about this repeated work. Our qualifying example, Mini-ML, also illustrates this behavior. As depicted in Figure 5, we see that both "case" statements do eval E_1.

Therefore, factoring can be defined as a first attempt to remove deep backtracking and convert programs into a functional program. The benefits of running factored programs are immense:

1. Deterministic at each step (see the operational semantics)

2. Replace Higher-Order Unification (substitutions) with Matching. Since Matching is faster than Higher-Order Unification, this yields faster evaluations.

3. It can be compiled. This is the smallest hurdle, but best illustrates the overall motivation for this work. Besides the enhances in representational power by using Delphin (with respect to the execution of logic programs), representing it in this purely functional paradigm allows us to go one step further – we can compile it into machine code. Once a program compiles into Delphin, we know it is type correct, and then we can simply build our own compiler, or output the equivalent SML code.

   There are some issues over which would be the better approach. Although we can output it as SML code, since we are doing higher-order pattern matching, some adjustments would also have to be made. However, since it is not very difficult to convert this to machine code, we can achieve an unprecedented increase in the speed of evaluating logic programs.

For inspiration, before we give the rules, we will show the results of factoring our Mini-ML example. Recall that the conversion from LF to Delphin resulted in the evaluator in Figure 5. The factored version of this program is depicted in Figure 13.

```
eval :: all {e:exp} exists {v:exp} true

fun eval ...
⋮
  | eval (case E1 E2 ([x:exp] E3 x)) =
     case ( eval E1 )
     of <z, <>> -->  let
                        val <V', <>> =  eval E2
                     in
                       <V', <>>
                     end
      | <s V1', <>> -->  let
                            val <V', <>> =  eval (E3 V1')
                         in
                           <V', <>>
                         end
⋮
```

Figure 13: A Mini-ML evaluator (converted to Delphin from LF and Factored)

6.2 Requirements

As already alluded to, not all logic programs can be *factored*. First, the Input/Output behavior of the program must be well-defined (*well-moded*). Recall in our Mini-ML example

we had "%mode eval +E -V". This states that we know all calls to the eval program will specify the first variable as input and the second variable as output. If we are dealing with a logic program which switches what it expects as Input/Output then it cannot be encoded in a functional paradigm.

The class of logic programs which can be automatically factored require that the preceding list of subgoals are the same (even in the same order) for each case that is collapsed into one case.

Although it is possible to widen the definition of factorable programs to allow subgoal reordering, we have chosen not to adopt such a definition. In particular, due to dependencies, it is not easy to reorder subgoals even if those reorders are semantic preserving. We therefore argue that our factoring algorithm is the most sensible to be used in practice, and this is the one we have implemented.

It is important to also repeat that it has been proven that it is not possible to detect all logical programs that *can* be factored [7].

Note: We must also stipulate that there does not exist one case which matches a *proper* subset of another case (there can be no partial overlap). This is formalized in the Appendix (Definition 2). Most logic program that we are concerned about fall naturally into this class, and it is often possible to convert other logical programs that do not meet this requirement into a semantically equivalent one that does. For example, we will define a constructor S to stand for the standard successor function, and the constant z to stand for zero. Assume we have three cases which differ in that they match against:

(1) s s x $\rightarrow$ do P_1

(2) s z $\rightarrow$ do P_2

(3) s y $\rightarrow$ do P_3

If we have as input "s z", it can match both (2) and (3). (In this case if we pick (2), then we run P_1. Note that the details are omitted for presentation purposes.) Similarly if we have as input "s s z" it can match both (1) and (3). To remedy this situation all we have to do is create four cases out of the three that are:

(A) s s x $\rightarrow$ do P_1

(B) s z $\rightarrow$ do P_2

(C) s z $\rightarrow$ do $P_3[z/y]$

(D) s s x $\rightarrow$ do $P_3[(sx)/y]$

Now it is clear these cases have the same semantic meaning, and now fit the requirement to be factored.

6.3 Factoring Algorithm

We now describe the factoring algorithm. This is broken up into four different judgments, which are all given and explained below. We will not give a definition of factored programs, but instead employ the idempotence property of factoring to declare a program F to be *factored* if and only if factoring F results in F (Section 7).

6.3.1 Main Judgment ($\overset{fact}{\rightsquigarrow}$)

We write $P \overset{fact}{\rightsquigarrow} F$ to express that F is the result of factoring P. The rules in Figure 14 traverse P until a case statement is found, which is subsequently passed to $\overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow}$.

$$\frac{}{L \overset{fact}{\rightsquigarrow} L}\ \textsf{fact\_LF}$$

$$\frac{}{\mathbf{x} \overset{fact}{\rightsquigarrow} \mathbf{x}}\ \textsf{fact\_var} \qquad \frac{}{\langle\rangle \overset{fact}{\rightsquigarrow} \langle\rangle}\ \textsf{fact\_unit}$$

$$\frac{P \overset{fact}{\rightsquigarrow} F}{\lambda x : D.P \overset{fact}{\rightsquigarrow} \lambda x : D.F}\ \textsf{fact\_}\Lambda \qquad \frac{P \overset{fact}{\rightsquigarrow} F}{\mu\mathbf{x} \in F_*.\, P \overset{fact}{\rightsquigarrow} \mu\mathbf{x} \in F_*.\, F}\ \textsf{fact\_rec}$$

$$\frac{A_1 \overset{fact}{\rightsquigarrow} F_{A_1} \quad P_2 \overset{fact}{\rightsquigarrow} F_{P_2}}{\langle A_1; P_2\rangle \overset{fact}{\rightsquigarrow} \langle F_{A_1}; F_{P_2}\rangle}\ \textsf{fact\_pair} \qquad \frac{P \overset{fact}{\rightsquigarrow} F \quad A \overset{fact}{\rightsquigarrow} F_A}{P\ A \overset{fact}{\rightsquigarrow} F\ F_A}\ \textsf{fact\_app}$$

$$\frac{P_1 \overset{fact}{\rightsquigarrow} F_1 \quad P_2 \overset{fact}{\rightsquigarrow} F_2}{\text{let } \mathbf{x} = P_1 \text{ in } P_2 \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} = F_1 \text{ in } F_2}\ \textsf{fact\_let} \qquad \frac{\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega'}{\text{case } \Omega \overset{fact}{\rightsquigarrow} \text{case } \Omega'}\ \textsf{fact\_case}$$

Figure 14: Rules for $\overset{fact}{\rightsquigarrow}$

6.3.2 Factor Case Judgment $\overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow}$

We write $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ if a list of cases Ω factors into Ω'. A list of cases is factored left to right, by folding in case by case to the already factored list. The three rules defining the traversal are shown in Figure 15.

$$\frac{}{\cdot \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \cdot}\ \textsf{case\_empty}$$

$$\frac{\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega' \quad \Omega' \oplus (\Psi \triangleright \psi \mapsto P) = \Omega''}{\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega''}\ \textsf{case\_nonempty}$$

$$\frac{\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega' \quad \Omega' \oplus (\Psi \triangleright \psi \mapsto P) \uparrow \quad P \overset{fact}{\rightsquigarrow} F}{\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega', (\Psi \triangleright \psi \mapsto F)}\ \textsf{case\_new}$$

Figure 15: Rules for $\overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow}$

6.3.3 Folding Cases ($\oplus$ Judgment)

We write $\Omega \oplus (\Psi triangleright \psi \mapsto P) = \Omega'$ to fold a case into Ω, which is assumed to be factored already. The rules are given in Figure 16 and compare the case $(\Psi \triangleright \psi \mapsto P)$ to every single case in Ω. Since Ω is already factored, there is at most one case with pattern ψ in Ω. Two patterns should be considered equal modulo reordering of variables declared in their co-domains, which is established by a renaming substitution t. Thus, the t used in the premise of `fold_yes` and `fold_no` satisfies $\Psi_1 \vdash t : \Psi_2$. While traversing Ω, the merging operation is guarded by the condition that ψ_1 and ψ_2 are equal modulo renaming. If it cannot find any such ψ_1 and ψ_2, then we cannot attempt to fold it in with any of the cases present in Ω and we write $\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$, whose rules are also in Figure 16.

Note that t refers to *only* renaming substitutions.

$$\frac{\psi_1 = \psi_2 \circ t \quad P_1 \overset{fact}{\rightsquigarrow} F_1 \quad P_2 \overset{fact}{\rightsquigarrow} F_2 \quad F_1 \star F_2[t] = P_3 \quad P_3 \overset{fact}{\rightsquigarrow} F_3}{\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)}\ \textsf{fold\_yes}$$

$$\frac{\psi_1 \neq \psi_2 \circ t \quad \Omega \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) = \Omega' \quad P_1 \overset{fact}{\rightsquigarrow} F_1}{\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto F_1)}\ \textsf{fold\_no}$$

$$\frac{}{\cdot \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) \uparrow}\ \textsf{fold}\uparrow\textsf{\_empty}$$

$$\frac{\psi_1 \neq \psi_2 \circ t \quad \Omega \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) \uparrow}{\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi_2 \triangleright \psi_2 \mapsto P_2) \uparrow}\ \textsf{fold}\uparrow\textsf{\_nonempty}$$

Figure 16: Rules for $\oplus$

6.3.4 Merge two Delphin Programs together ($\star$ Judgment)

We write $P_1 \star P_2 = P_3$ to express that P_3 is result of merging P_1 with P_2. The rules are shown in Figure 17. A necessary precondition for merging two programs is that both programs compute exactly the same values up to the point when another case statement is encountered. Because of the way logic programs are translated into $\mathcal{N}_\omega$, only a few cases actually do something interesting, all others act as the identity. The rule `merge_let`, for example, enforces that all subgoals are the same. The rules `merge_fold` and `merge_new` deserve special mention as they both try to fold the case $(\Psi \triangleright \psi \mapsto P)$ into Ω. The former succeeds, the later appends it to Ω, after factoring its body.

$$\frac{}{\langle\rangle \star \langle\rangle = \langle\rangle}\ \text{merge\_unit} \qquad \frac{}{L \star L = L}\ \text{merge\_LF}$$

$$\frac{}{(\lambda x : D.P) \star (\lambda x : D.P) = \lambda x : D.P}\ \text{merge\_}\Lambda \qquad \frac{}{(\mu \mathbf{x} \in F.\, P) \star (\mu \mathbf{x} \in F.\, P) = \mu \mathbf{x} \in F.\, P}\ \text{merge\_rec}$$

$$\frac{}{(P\ A) \star (P\ A) = P\ A}\ \text{merge\_app} \qquad \frac{P_{1_B} \star P_{2_B} = P_{3_B}}{\langle A_{1_A}; P_{1_B}\rangle \star \langle A_{1_A}; P_{2_B}\rangle = \langle A_{1_A}; P_{3_B}\rangle}\ \text{merge\_pair}$$

$$\frac{\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'}{\text{case}\ \Omega \star \text{case}\ [(\Psi \triangleright \psi \mapsto P)] = \text{case}\ \Omega'}\ \text{merge\_fold} \qquad \frac{\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow \quad P \overset{fact}{\rightsquigarrow} F}{\text{case}\ \Omega \star \text{case}\ [(\Psi \triangleright \psi \mapsto P)] = \text{case}\ (\Omega, (\Psi \triangleright \psi \mapsto F))}\ \text{merge\_new}$$

$$\frac{P_{1_B} \star P_{2_B} = P_{3_B}}{(\text{let}\ \mathbf{x} : D = P_{1_A}\ \text{in}\ P_{1_B}) \star (\text{let}\ \mathbf{x} : D = P_{1_A}\ \text{in}\ P_{2_B}) = \text{let}\ \mathbf{x} : D_1 = P_{1_A}\ \text{in}\ P_{3_B}}\ \text{merge\_let}$$

Figure 17: Rules for $\star$

To give some justification as to why we chose the rules as they are, lets see why we can't change merge_pair to try to merge both $\langle A_{1_A}; P_{1_B}\rangle$ (as right operand) and $\langle A_{2_A}; P_{2_B}\rangle$ (as left operand) where $(A_{1_A} \neq A_{2_A})$ and $(P_{1_B} \neq P_{2_B})$. Clearly, if such a merge did take place, the result must be some pair of the form $\langle A_{3_A}; P_{3_B}\rangle$. Let us assume we had the case that A_{1_A} evaluates to V_1 and P_{1_B} evaluates to $\bot$ and A_{2_A} evaluates to V_2 and P_{2_B} evaluates to V_3. Clearly the desired result would be $\langle V_2; V_3\rangle$, but based on our semantics we would get the result $\langle V_1; V_3\rangle$ which does not make sense since neither of the two programs being merged would result in that value.

Similar arguments can be made for the definition of merge_app and merge_let.

The most important rule is merge_fold, which uses the $\oplus$ judgment to add $(\Psi \triangleright \psi \mapsto P)$ to the list of cases in Ω. Note that we are assuming that the operand on the right is a list containing *exactly* one case. The definition could easily be extended to allow for a list of cases. However, this is designed to factor *Twelf* logical programs, which satisfy this invariant.

7 Meta Theoretical Properties

Here we would like to prove that Delphin's ($\mathcal{T}_\omega$) operational semantics is semantically the same as $\mathcal{N}_\omega$'s operational semantics with respect to factored programs. In other words, we would like to show that if we can factor a program, then the result of using the unfactored version in $\mathcal{N}_\omega$ is the same as the result of using the factored version on Delphin.

Note that in this section we just highlight the proof. A detailed proof can be found in the Appendix (Section 10).

First we remark, that the following meta theoretical investigation cannot be carried out without the proper definition of observational equivalence modulo factoring between values and environments, which is denoted as $\overset{*}{\Rightarrow}$ (See Appendix, Definition 1).

$\overset{*}{\Rightarrow}$ is defined as

- $V \overset{*}{\Rightarrow} W$ is defined recursively as:
  - If $V = \{\eta; \Lambda x : D.P\}$ and $W = \{\eta'; \Lambda x : D.F\}$ then
    - $\eta \overset{*}{\Rightarrow} \eta'$
    - and $P \overset{fact}{\rightsquigarrow} F$
  - If $V = \langle V_1; V_2\rangle$ and $W = \langle W_1; W_2\rangle$ then $V_1 \overset{*}{\Rightarrow} W_1$ and $V_2 \overset{*}{\Rightarrow} W_2$
  - Otherwise, $V = W$
- $\eta \overset{*}{\Rightarrow} \eta_1$ is defined recursively as:
  - $(\eta, \mu\mathbf{x} \in F^*.\, P/\mathbf{x}) \overset{*}{\Rightarrow} (\eta_1, \mu\mathbf{x} \in F^*.\, F/\mathbf{x})$ if the following is true:
    - $\eta \overset{*}{\Rightarrow} \eta_1$
    - $P \overset{fact}{\rightsquigarrow} F$
  - $(\eta, V/\mathbf{x}) \overset{*}{\Rightarrow} (\eta, W/\mathbf{x})$ if and only if $V \overset{*}{\Rightarrow} W$
  - $\cdot \overset{*}{\Rightarrow} \cdot$

It follows immediately from this definition that for Λ-free terms, $\overset{*}{\Rightarrow}$ implies $=$ (syntactical equality), a property that is essential in the proof of the main theorem.

The proof requires many Lemmas which are detailed in the Appendix. Here we just outline it.

(10.1) We first formalize basic definitions such as $\overset{*}{\Rightarrow}$ discussed above.

(10.2) We then prove some interesting properties of factored programs which are used in the proofs of the other Lemmas. Namely we prove that once we factor a program, if we factor it again we get exactly the same result (Idempotence Property).

(10.3) We then prove a very important property about renaming substitutions t. Namely, $\Gamma; \eta' \vdash F \hookrightarrow V$ if and only if $\Gamma; (t^{-1} \circ \eta') \vdash F[t] \hookrightarrow V$ (and it is true for $\perp$ as well).

– Note that the concept of applying a renaming substitution to a list of cases is defined in Definition 4. Namely, we apply the renaming substitution, t, to each case such that

$$(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$$

(10.4) We then prove important stack properties for $\overset{SC}{\hookrightarrow}$. Namely, $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ if and only if either $\cdot; SC \vdash W \overset{SC}{\hookrightarrow} V$ or $(\cdot; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ and $FC \overset{FC}{\hookrightarrow} V)$

(10.5) We now prove properties relating to $\star$ and $\oplus$ judgment. Namely, we prove that:

– If we are merging F_1 and F_2 (If $F_1 \star F_2 = P_3$), then

1. $P_3 \overset{fact}{\rightsquigarrow} F_3$
2. If $\Gamma; \eta \vdash F_2 \hookrightarrow\!\!\!\!\rightarrow V$ then
   (a) $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$
   (b) Either $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow V$ or $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow \bot$
3. If $\Gamma; \eta \vdash F_2 \hookrightarrow\!\!\!\!\rightarrow \bot$ then
   (a) $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow V$ if and only if $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$
   (b) $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ if and only if $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow \bot$

– If $\Omega \oplus (Psi \triangleright \psi \mapsto P) = \Omega'$ and $P \overset{fact}{\rightsquigarrow} F$ then

* If the case $(\Psi \triangleright \psi \mapsto P)$ applies and F evaluates to a value, then Ω' evaluates to the same value. Also, Ω evaluates to the same value or to $\bot$.
* If the case $(\Psi \triangleright \psi \mapsto P)$ applies and F evaluates to $\bot$, then Ω' behaves the same way as Ω
* If the case $(\Psi \triangleright \psi \mapsto P)$ does not apply, then Ω' behaves the same way as Ω

(10.6) Here we have the main Lemma (Embedding) of the paper. We show that

– If $\Gamma; \eta; FC; SC \vdash P \hookrightarrow V$ and $P \overset{fact}{\rightsquigarrow} F$ and $\eta \overset{*}{\Rightarrow} \eta_1$ then:

* Either $\Gamma; \eta_1 \vdash F \hookrightarrow\!\!\!\!\rightarrow W'$ and
  1. There exists a W such that $W \overset{*}{\Rightarrow} W'$
  2. $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
* or both $\Gamma; \eta_1 \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ and $FC \overset{FC}{\hookrightarrow} V$.

(10.7) We finally bring it all together to get our main result (Theorem Main) which states if $P \overset{fact}{\rightsquigarrow} F$ and $\Gamma; \cdot; \cdot; \cdot \vdash P \hookrightarrow V$ and V is Λ-free, then $\Gamma; \cdot \vdash F \hookrightarrow\!\!\!\!\rightarrow V$.

8 Implementation

The aforementioned factoring algorithm has been implemented in Twelf. Although Delphin is a stand-alone functional programming language, its underlying logical framework is *Twelf*,

and one may use Delphin to solve queries in Twelf, of restricted forms (Section 6.2, using Delphin.

To do this, we introduce a new command in Twelf, known as *%fquery*. This stands for *functional query*, and follows a syntax similar to *%query*. It proceeds following the steps outlined in this paper. First, a type family is converted into an $\mathcal{N}_\omega$ function with possibly redundant cases. We then factor the resulting function (an error is raised if it cannot be factored). Once factoring is successful, the function is executed using Delphin's operational semantics, and the result displayed to the screen. The difference between *%fquery* and *%query* is that it converts problems of backtracking to problems of unification which can be solved much easier.

Currently we have implemented factoring using *unification* to match the cases. When a query is executed through Delphin under this model it performs just as well as the execution of *%query*. However, since we can replace *unification* with *matching*, there is great room for improvement. Also, since we will eventually be compiling these patterns, the performance will most likely set a new precedent for future logic programming systems. Furthermore, this representation of Twelf logic programs can then further be compiled into a lower-level language (possibly directly to byte code or to Flint), which we leave to future work.

The use of factoring extends far past the small Mini-ML example given in Section 2. For example, TALT ([2]) is a typed assembly language built on top of Twelf. There are many type families (whose name is appended with "!") which is the result of another type family being manually factored (similar to the method we alluded to in Section 5.3). This was needed since it was necessary to assert that the resulting function was "total". Now with factoring built in, all of these extra type families can go away since the coverage checker will be able to tell it is total since it runs after factoring.

9 Conclusion

We have shown the need for *factoring* by a simple example of Mini-ML. Further we have argued that when we factor a program we can use a simpler operational semantics (Section 5.3) and have proven that it would return the same result as Twelf's logic engine.

We have also implemented the features mentioned in this paper, and the performance results are promising. In future research we intend to further enhance the performance of evaluating logic programs as well as extend the definition of factoring to work on user-entered Delphin programs (that do not originate in Twelf).

References

[1] T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

[2] K. Crary. Toward a foundational typed assembly language, 2002.

[3] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, Jan. 1993.

[4] F. Henderson, Z. Somogyi, and T. Conway. Determinism analysis in the mercury compiler, 1996.

[5] F. Pfenning. Elf: A language for logic definition and verified meta-programming. In *Fourth Annual Symposium on Logic in Computer Science*, pages 313–322, Pacific Grove, California, June 1989. IEEE Computer Society Press.

[6] F. Pfenning and C. Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.

[7] H. Sawamura and T. Takeshima. Recursive unsolvability of determinacy, solvable cases of determinacy and their applications to prolog optimization., 1985.

[8] C. Schürmann. *Automating the Meta-Theory of Deductive Systems.* PhD thesis, Carnegie Mellon University, 2000. CMU-CS-00-146.

[9] C. Schürmann. Recursion for higher-order encodings. In L. Fribourg, editor, *Proceedings of the Conference on Computer Science Logic (CSL 2001)*, pages 585–599, Paris, France, August 2001. Springer Verlag LNCS 2142.

[10] C. Schürmann. A type-theoretic approach to induction with higher-order encodings. In *Proceedings of the Conference on Logic for Programming, Artificial Intelligence and Reasoning(LPAR 2001)*, pages 266–281, Havana, Cuba, 2001. Springer Verlag LNAI 2250.

[11] C. Schürmann. The Delphin website: `http://www.cs.yale.edu/~carsten/delphin`, 2002.

[12] F. H. Zoltan Somogyu and T. Conway. Mercury: an efficient purely declarative logic programming language, 1995.

10 Appendix – Proofs for Section 7

10.1 Basic Definitions and Properties

Definition 1 (New Value Equality) $\stackrel{*}{\Rightarrow}$ *is defined as*

- $V \stackrel{*}{\Rightarrow} W$ *is defined recursively as:*
  - *If* $V = \{\eta; \Lambda x : D.P\}$ *and* $W = \{\eta'; \Lambda x : D.F\}$ *then*
    - $\eta \stackrel{*}{\Rightarrow} \eta'$
    - *and* $P \stackrel{fact}{\rightsquigarrow} F$
  - *If* $V = \langle V_1; V_2 \rangle$ *and* $W = \langle W_1; W_2 \rangle$ *then* $V_1 \stackrel{*}{\Rightarrow} W_1$ *and* $V_2 \stackrel{*}{\Rightarrow} W_2$
  - *Otherwise,* $V = W$
- $\eta \stackrel{*}{\Rightarrow} \eta_1$ *is defined recursively as:*
  - $(\eta, \mu \mathbf{x} \in F^*.P/\mathbf{x}) \stackrel{*}{\Rightarrow} (\eta_1, \mu \mathbf{x} \in F^*.F/\mathbf{x})$ *if the following is true:*
    - $\eta \stackrel{*}{\Rightarrow} \eta_1$
    - $P \stackrel{fact}{\rightsquigarrow} F$
  - $(\eta, V/\mathbf{x}) \stackrel{*}{\Rightarrow} (\eta, W/\mathbf{x})$ *if and only if* $V \stackrel{*}{\Rightarrow} W$
  - $\cdot \stackrel{*}{\Rightarrow} \cdot$

One property of this relation which is used, is that If V_1 *has no* Λ *terms, then*

$$\textit{If } V_1 \stackrel{*}{\Rightarrow} V_2 \textit{ then } V_1 = V_2$$

Definition 2 (Property of Cases)
For all pairs of cases $(\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2)$*, one of the following holds*

1. $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \hat{=} (\Psi_2 \triangleright \psi_2 \mapsto P_2)$*: There exists a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$*. Note that this also implies that* $\Psi_1 \vdash t : \Psi_2$
2. $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \not\hat{=} (\Psi_2 \triangleright \psi_2 \mapsto P_2)$*: There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$*. Note that this is where the* non-overlapping *property comes into play. We now also require that there are no instances of* η *such that we can find an* η' *and* η'' *such that* $\psi_1 \circ \eta' = \eta$ *and* $\psi_2 \circ \eta'' = \eta$

Definition 3 (Renaming Substitution Properties) *Note that in this paper we discuss renaming substitutions* t *very frequently. The one important fact of a renaming substitution is that since it is just a renaming substitution we know that the inverse* t^{-1} *always exists, and this is necessary for many of the proofs to follow.*

Definition 4 (Case applied to a renaming substitution) *We add one more property of substitutions which apply to when a specific case has a substitution applied to it.*

If t is a renaming substitution, then

$$(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$$

Similarly, we define what it means to apply a renaming substitution to a list of cases Ω

- $\cdot[t] = \cdot$
- $(\Gamma, (\Psi \triangleright \psi \mapsto P))[t] = \Gamma[t], (\Psi \triangleright \psi \mapsto P)[t]$

Note that the following proofs use this definition and everything is sound. However, we also present an informal rationale *as why this definition makes sense:*

Assume $\Psi_1 \vdash t : \Psi_2$. So we have that $(\Psi \triangleright \psi \mapsto P)$ makes sense in Ψ_2 but we want it to make sense in Ψ_1 instead. Therefore, $\Psi_2 \vdash t^{-1} : \Psi_1$. Assume $\Psi \vdash \psi : \Psi_2$. Therefore, $\Psi \vdash (t^{-1} \circ \psi) : \Psi_1$

Lemma 1 (Case Definition on $\stackrel{*}{\Rightarrow}$ 1)
If

- $(\Psi \triangleright \psi \mapsto P) \in \Omega$
- *and* $\eta \stackrel{*}{\Rightarrow} \eta_1$
- *and* $\psi \circ \eta' = \eta$

Then

- *There exists an η'' such that $\psi \circ \eta'' = \eta_1$*
- *and $\eta' \stackrel{*}{\Rightarrow} \eta''$*

Proof. *By Definition of $\stackrel{*}{\Rightarrow}$ and properties of case $(\Psi \triangleright \psi \mapsto P)$ ([11])*

Lemma 2 (Case Definition on $\stackrel{*}{\Rightarrow}$ 2)
If

- $(\Psi \triangleright \psi \mapsto P) \in \Omega$
- *and* $\eta \stackrel{*}{\Rightarrow} \eta_1$
- *and there does not exist an η' such that $\psi \circ \eta' = \eta$*

Then

- *There does not exist an η' such that $\psi \circ \eta' = \eta_1$*

Proof. *By Definition of $\stackrel{*}{\Rightarrow}$ and properties of case $(\Psi \triangleright \psi \mapsto P)$ ([11])*

10.2 Properties of Factoring

Lemma 3 *This is to show that when we factor an already factored program, we get the same result. (The main result is Part 1)*

1. *If* $P \overset{fact}{\rightsquigarrow} F$ *then* $F \overset{fact}{\rightsquigarrow} F$

2. *If* $\Omega_S \subseteq \Omega'$ *and there exists and* Ω *such that* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$, *then* $\Omega_S \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega_S$

3. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$ *(in other words, if* Ω' *is factored), then there does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega'$

4. *If* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and there does not exist a renaming substitution* t *such that*

   - $\psi_1 = \psi_2 \circ t$ *and*
   - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega$ *and*
   - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega$

   Then there does not exist a renaming substitution t *such that*

   - $\psi_1 = \psi_2 \circ t$ *and*
   - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *and*
   - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega'$

   (Note that this is saying that if we have an Ω *such that no two cases match, then in* Ω' *we also have the property that no two cases match)*

5. *If we are given a* ψ^* *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and for all* $(\Psi_1, \psi_1 \triangleright P_1) \in \Omega$ *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *and there does not exist a renaming substitution* t' *such that* $\psi^* = \psi \circ t'$ *then*

   For all $(\Psi_1, \psi_1 \triangleright P_1) \in \Omega'$, *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$

6. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$, *then for all* $(\Psi \triangleright \psi \mapsto F) \in \Omega'$, *there exists some* P *such that* $P \overset{fact}{\rightsquigarrow} F$. *(This states that all cases in* Ω' *have been factored)*

7. *If* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and for all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega$, *there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$, *then for all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega'$, *there exists a* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$

8. *If* $P \overset{fact}{\rightsquigarrow} F$ *and for all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$ *there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$, *then*
   $$\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$$

9. *If* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$, *then for all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$ *there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$

PROOF.

Part 1 – Proof over $\overset{fact}{\rightsquigarrow}$

- fact_LF
  1. $L \overset{fact}{\rightsquigarrow} L$ *(By Assumption)*
  2. $L \overset{fact}{\rightsquigarrow} L$ *(Trivial from Above)*
- fact_var
  1. $\mathbf{x} \overset{fact}{\rightsquigarrow} \mathbf{x}$ *(By Assumption)*
  2. $\mathbf{x} \overset{fact}{\rightsquigarrow} \mathbf{x}$ *(Trivial from Above)*
- fact_Unit
  1. $\langle\rangle \overset{fact}{\rightsquigarrow} \langle\rangle$ *(By Assumption)*
  2. $\langle\rangle \overset{fact}{\rightsquigarrow} \langle\rangle$ *(Trivial from Above)*
- fact_Λ
  1. $\lambda x : D.P \overset{fact}{\rightsquigarrow} \lambda x : D.F$ *(By Assumption)*
  2. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* fact_Λ*)*
  3. $F \overset{fact}{\rightsquigarrow} F$ *(By Induction Hypothesis)*
  4. $\lambda x : D.F \overset{fact}{\rightsquigarrow} \lambda x : D.F$ *(By* fact_Λ*)*
- fact_rec
  1. $\mu\mathbf{x} \in F_*.P \overset{fact}{\rightsquigarrow} \mu\mathbf{x} \in F_*.F$ *(By Assumption)*
  2. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* fact_rec*)*
  3. $F \overset{fact}{\rightsquigarrow} F$ *(By Induction Hypothesis)*
  4. $\mu\mathbf{x} \in F_*.F \overset{fact}{\rightsquigarrow} \mu\mathbf{x} \in F_*.F$ *(By* fact_rec*)*
- fact_pair
  1. $\langle A_1; P_2\rangle \overset{fact}{\rightsquigarrow} \langle F_{A_1}; F_{P_2}\rangle$ *(By Assumption)*
  2. $A_1 \overset{fact}{\rightsquigarrow} F_{A_1}$ *(By Definition of* fact_pair*)*
  3. $P_2 \overset{fact}{\rightsquigarrow} F_{P_2}$ *(By Definition of* fact_pair*)*
  4. $F_{A_1} \overset{fact}{\rightsquigarrow} F_{A_1}$ *(By Induction Hypothesis)*
  5. $F_{P_2} \overset{fact}{\rightsquigarrow} F_{P_2}$ *(By Induction Hypothesis)*
  6. $\langle F_{A_1}; F_{P_2}\rangle \overset{fact}{\rightsquigarrow} \langle F_{A_1}; F_{P_2}\rangle$ *(By* fact_pair*)*

- fact_app

  1. $P\ A \overset{fact}{\rightsquigarrow} F\ F_A$ *(By Assumption)*
  2. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* fact_app*)*
  3. $A \overset{fact}{\rightsquigarrow} F_A$ *(By Definition of* fact_app*)*
  4. $F \overset{fact}{\rightsquigarrow} F$ *(By Induction Hypothesis)*
  5. $F_A \overset{fact}{\rightsquigarrow} F_A$ *(By Induction Hypothesis)*
  6. $F\ F_A \overset{fact}{\rightsquigarrow} F\ F_A$ *(By* fact_app*)*

- fact_let

  1. $\text{let } \mathbf{x} = P_1 \text{ in } P_2 \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} = F_1 \text{ in } F_2$ *(By Assumption)*
  2. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Definition of* fact_let*)*
  3. $P_2 \overset{fact}{\rightsquigarrow} F_2$ *(By Definition of* fact_let*)*
  4. $F_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Induction Hypothesis)*
  5. $F_2 \overset{fact}{\rightsquigarrow} F_2$ *(By Induction Hypothesis)*
  6. $\text{let } \mathbf{x} = F_1 \text{ in } F_2 \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} = F_1 \text{ in } F_2$ *(By* fact_let*)*

- fact_case

  1. $\text{case } \Omega \overset{fact}{\rightsquigarrow} \text{case } \Omega'$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega'$ *(By Definition of* fact_case*)*
  3. $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega'$ *(By this Lemma Part 2)*
  4. $\text{case } \Omega' \overset{fact}{\rightsquigarrow} \text{case } \Omega'$ *(By* fact_case*)*

Part 2 – Proof over structure of Ω_S

We now do induction on the structure of Ω_S (so we assume it is a subset of something already factored). In other words, we need to show that if $\Omega_S \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'_S$ then $\Omega_S = \Omega'_S$.

- $\cdot$
  1. $\cdot \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \cdot$ *(By* case_empty*)*
  2. $\cdot = \cdot$ *(Trivial)*
- $\Omega, (\Psi \triangleright \psi \mapsto P)$
  1. $\Omega, (\Psi \triangleright \psi \mapsto P)$ *is a subset of something that has already been factored. (By Assumption)*
  2. *There exists some P' such that $P' \overset{fact}{\rightsquigarrow} P$ (from this Lemma Part 6)*
  3. $P \overset{fact}{\rightsquigarrow} P$ *(from this Lemma Part 1)*
  4. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Induction Hypothesis on first line)*
  5. *There does not exist a renaming substitution t such that $\psi_1 = \psi_2 \circ t$ and $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi \triangleright \psi \mapsto P)$ and $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi \triangleright \psi \mapsto P)$(from this Lemma Part 3)*
  6. *For all $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$, there does not exist a renaming substitution t such that $\psi = \psi_A \circ t$ (By Above)*
  7. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By Part 8 of this Lemma)*
  8. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi \triangleright \psi \mapsto P)$ *(By* case_new*)*

Part 3 – Proof over $\overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow}$

- case_empty

  1. $\cdot \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \cdot$ *(By Assumption)*
  2. *Since* $\cdot$ *has no cases in it, it is trivially true*

- case_nonempty

  1. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega''$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Definition of* case_nonempty*)*
  3. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega'$ *(By Induction Hypothesis)*
  4. $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) = \Omega''$ *(By Definition of* case_nonempty*)*
  5. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega''$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega''$ *(By this Lemma Part 4)*

- case_new

  1. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega', (\Psi \triangleright \psi \mapsto F)$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Definition of* case_new*)*
  3. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega'$ *(By Induction Hypothesis)*
  4. $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By Definition of* case_new*)*
  5. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* case_new*)*
  6. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega'$*), there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By this Lemma Part 9)*
  7. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega', (\Psi \triangleright \psi \mapsto F)$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega', (\Psi \triangleright \psi \mapsto F)$ *(By Above)*

Part 4 – Proof over $\oplus$

- fold_no

  1. $\Omega, (\Psi_A \triangleright \psi_A \mapsto P_A) \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega', (\Psi_A \triangleright \psi_A \mapsto F_A)$ *(By Assumption)*
  2. *By Assumption, there does not exist a renaming substitution* t *such that*
     - $\psi_1 = \psi_2 \circ t$ *and*
     - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$ *and*
     - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$
  3. *By Above, there does not exist a renaming substitution* t *such that*
     - $\psi_1 = \psi_2 \circ t$ *and*
     - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega$ *and*
     - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega$
  4. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega$*, there does not exist a renaming substitution* t *such that* $\psi_A = \psi_1 \circ t$ *(From Above)*
  5. *There does not exist a renaming substitution* t *such that* $\psi_A = \psi_B \circ t$ *(By Definition of* fold_no*)*
  6. $\Omega \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega'$ *(By Definition of* fold_no*)*
  7. *By the Induction Hypothesis, there does not exist a renaming substitution* t *such that*
     - $\psi_1 = \psi_2 \circ t$ *and*
     - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *and*
     - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega'$
  8. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega'$ *there does not exist a renaming substitution* t *such that* $\psi_A = \psi_1 \circ t$ *(By Part 5 of this Lemma)*
  9. *Therefore, we combine the last two lines to get what we need: There does not exist a renaming substitution* t *such that*
     - $\psi_1 = \psi_2 \circ t$ *and*
     - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega', (\Psi_A \triangleright \psi_A \mapsto F_A)$ *and*
     - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega', (\Psi_A \triangleright \psi_A \mapsto F_A)$

- fold_yes

  1. $\Omega, (\Psi_A \triangleright \psi_A \mapsto P_A) \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega, (\Psi_A \triangleright \psi_A \mapsto F)$ *(By Assumption)*
  2. *By Assumption, there does not exist a renaming substitution* t *such that*
     - $\psi_1 = \psi_2 \circ t$ *and*
     - $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$ *and*
     - $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$
  3. *By Above, there does not exist a renaming substitution* t *such that*

- $\psi_1 = \psi_2 \circ t$ *and*
- $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto F)$ *and*
- $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto F)$

Part 5 – Proof over $\oplus$

- fold_no

  1. $\Omega, (\Psi_A \rhd \psi_A \mapsto P_A) \oplus (\Psi_B \rhd \psi_B \mapsto P_B) = \Omega', (\Psi_A \rhd \psi_A \mapsto F_A)$ *(By Assumption)*
  2. *We are given a* ψ^* *such that for all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \rhd \psi_A \mapsto P_A)$ *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(By Assumption)*
  3. *For all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega$ *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(By Above)*
  4. *There does not exist a renaming substitution* t *such that* $\psi^* = \psi_A \circ t$ *(From Above)*
  5. *There does not exist a renaming substitution* t *such that* $\psi^* = \psi_B \circ t$ *(By Assumption)*
  6. $\Omega \oplus (\Psi_B \rhd \psi_B \mapsto P_B) = \Omega'$ *(By Definition of* fold_no*)*
  7. *For all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega'$*, there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(Induction Hypothesis)*
  8. *For all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega', (\Psi_A \rhd \psi_A \mapsto F_A)$*, there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(From Above)*

- fold_yes

  1. $\Omega, (\Psi_A \rhd \psi_A \mapsto P_A) \oplus (\Psi_B \rhd \psi_B \mapsto P_B) = \Omega, (\Psi_A \rhd \psi_A \mapsto F)$ *(By Assumption)*
  2. *We are given a* ψ^* *such that for all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \rhd \psi_A \mapsto P_A)$ *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(By Assumption)*
  3. *For all* $(\Psi_1 \rhd \psi_1 \mapsto P_1) \in \Omega, (\Psi_A \rhd \psi_A \mapsto F)$ *there does not exist a renaming substitution* t *such that* $\psi^* = \psi_1 \circ t$ *(By Above)*

Part 6 – Proof over $\overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow}$

- case_empty

  1. $\cdot \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \cdot$ *(By Assumption)*
  2. *Since it is empty, the result is trivial*

- case_nonempty

  1. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega''$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega'$ *(By Definition of* case_nonempty*)*
  3. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega'$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Induction Hypothesis)*
  4. $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) = \Omega''$ *(By Definition of* case_nonempty*)*
  5. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega''$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By this Lemma Part 7)*

- case_new

  1. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega', (\Psi \triangleright \psi \mapsto F)$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\rightarrow} \Omega'$ *(By Definition of* case_new*)*
  3. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega'$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Induction Hypothesis)*
  4. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* case_new*)*
  5. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega', (\Psi \triangleright \psi \mapsto F)$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Above)*

Part 7 – Proof over $\oplus$

- fold_no

  1. $\Omega, (\Psi_A \triangleright \psi_A \mapsto P_A) \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega', (\Psi_A \triangleright \psi_A \mapsto F_A)$ *(By Assumption)*
  2. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Assumption)*
  3. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Above)*
  4. $\Omega \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega'$ *(By Definition of* fold_no*)*
  5. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega'$*, there exists a* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Induction Hypothesis)*
  6. $P_A \overset{fact}{\rightsquigarrow} F_A$ *(By Definition of* fold_no*)*
  7. *Combining the last two lines, we get that for all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega', (\Psi_A \triangleright \psi_A \mapsto F_A)$*, there exists a* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$

- fold_yes

  1. $\Omega, (\Psi_A \triangleright \psi_A \mapsto P_A) \oplus (\Psi_B \triangleright \psi_B \mapsto P_B) = \Omega, (\Psi_A \triangleright \psi_A \mapsto F)$ *(By Assumption)*
  2. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto P_A)$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Assumption)*
  3. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega$*, there exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Above)*
  4. *There exists some* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F$ *(By Definition of* fold_yes*)*
  5. *For all* $(\Psi_1 \triangleright \psi_1 \mapsto F_1) \in \Omega, (\Psi_A \triangleright \psi_A \mapsto F)$*, there exists a* P_1 *such that* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Induction Hypothesis)*

Part 8 – Proof over structure of Ω

- $\cdot$

  1. $\cdot \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By* fold↑_empty*)*

- $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$

  1. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  2. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Assumption)*
  3. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Above)*
  4. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By Induction Hypothesis)*
  5. *There is no renaming substitution* t *such that* $\psi = \psi_1 \circ t$ *(By Above)*
  6. *Since* t *is a renaming substitution, there is also no* t *such that* $\psi_1 = \psi \circ t$
  7. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By* fold↑_nonempty*)*

Part 9 – Proof over $\oplus$

- fold↑_empty

  1. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  2. $\cdot \oplus (\Psi \triangleright \psi \mapsto P)\uparrow$ *(By Assumption)*
  3. *Trivially (since* $\cdot$ *denotes empty* Ω*), for all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \cdot$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$

- fold↑_nonempty

  1. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  2. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P)\uparrow$ *(By Assumption)*
  3. $\Omega \oplus (\Psi \triangleright \psi \mapsto P)\uparrow$ *(By Definition of* fold↑_nonempty*)*
  4. *There is no renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold↑_nonempty*)*
  5. *Since* t *is a renaming substitution, there is also no* t *such that* $\psi = \psi_1 \circ t$
  6. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Induction Hypothesis)*
  7. *We just combine the last two steps to see that for all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$*.*

Lemma 4 *If* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *then there exists an* F *such that* $P \overset{fact}{\rightsquigarrow} F$

PROOF. *By Induction over the* $\oplus$ *judgment*

- fold_no
  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto F_1)$ *(By Assumption)*
  2. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *(By Definition of* fold_no*)*
  3. $P \overset{fact}{\rightsquigarrow} F$ *(By The Induction Hypothesis)*
- fold_yes
  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(By Assumption)*
  2. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* fold_yes*)*

10.3 Renaming Substitution Property

Lemma 5 *If t is a renaming substitution and $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$, then*

$$\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow V \text{ if and only if } \Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow V$$

PROOF. *Proceed by induction on $\hookrightarrow\!\!\!\!\!\rightarrow$. Recall that $\Omega[t]$ is defined in Definition 4 and that the inverse of t, denoted as t^{-1}, always exists (since it is a renaming substitution).*

- ev_delphin_no

  1. *t is a renaming substitution (By Assumption)*
  2. *$(\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} (\Omega, (\Psi \triangleright \psi \mapsto P))$ (By Assumption)*
  3. *There exists some P' such that $P' \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.6)*
  4. *$P \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.1)*
  5. *$\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ (By Lemma 3.2)*
  6. *We must show both:*
     - *(a) If $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow V$ then $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow V$*
       - *i. $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Assumption)*
       - *ii. For all η'', $\psi \circ \eta'' \neq \eta'$. (By Definition of* ev_delphin_no*)*
       - *iii. $\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Definition of* ev_delphin_no*)*
       - *iv. $\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Induction Hypothesis)*
       - *v. $(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$ by Definition 4*
       - *vi. There does not exist an η''' such that $(t^{-1} \circ \psi) \circ \eta''' = (t^{-1} \circ \eta')$ If such an η''' did exist, then $\psi \circ \eta''' = \eta'$ contradicting our above assumption*
       - *vii. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright \psi \mapsto P)[t]) \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Application of* ev_delphin_no*)*
       - *viii. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Substitution Properties)*
     - *(b) If $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow V$ then $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow V$*
       - *i. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Assumption)*
       - *ii. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow V$ by Definition 4 and Substitution Properties*
       - *iii. For all η'', $(t^{-1} \circ \psi) \circ \eta'' \neq (t^{-1} \circ \eta')$. (By Definition of* ev_delphin_no*)*
       - *iv. $\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Definition of* ev_delphin_no*)*
       - *v. $\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow V$ (By Induction Hypothesis)*
       - *vi. There does not exist an η''' such that $\psi \circ \eta''' = \eta'$. If such an η''' did exist, then $(t^{-1} \circ \psi) \circ \eta''' = (t^{-1} \circ \eta')$ contradicting our above assumption*
       - *vii. $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow V$ by Application of* ev_delphin_no

- ev_delphin_yes

  1. *t is a renaming substitution (By Assumption)*
  2. *$(\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} (\Omega, (\Psi \triangleright \psi \mapsto P))$ (By Assumption)*
  3. *There exists some P' such that $P' \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.6)*
  4. *$P \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.1)*
  5. *$\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ (By Lemma 3.2)*
  6. *We must show both:*
     (a) *If $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\rightarrow V$*
     then $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\rightarrow V$
        i. *$\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\rightarrow V$ (By Assumption)*
        ii. *$\psi \circ \eta'' = \eta'$. (By Definition of* ev_delphin_yes*)*
        iii. *$(t^{-1} \circ \psi) \circ \eta'' = (t^{-1} \circ \eta')$ (By Substitution Properties)*
        iv. *$\Gamma; \eta'' \vdash P \hookrightarrow\!\!\!\!\rightarrow V$ (By Definition of* ev_delphin_yes*)*
        v. *$(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$ by Definition 4*
        vi. *So we need to evaluate $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P))$. By Inversion (using* ev_delphin_yes*), and Line 6(a)iii and Line 6(a)iv, we see that this also $\hookrightarrow\!\!\!\!\rightarrow V$*
     (b) *If $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\rightarrow V$*
     then $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\rightarrow V$
        i. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\rightarrow V$ (By Assumption)*
        ii. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)) \hookrightarrow\!\!\!\!\rightarrow V$ (By Definition 4 and Substitution Properties)*
        iii. *$(t^{-1} \circ \psi) \circ \eta'' = (t^{-1} \circ \eta')$. (By Definition of* ev_delphin_yes*)*
        iv. *$\psi \circ \eta'' = \eta'$ (By Substitution Properties)*
        v. *$\Gamma; \eta'' \vdash P \hookrightarrow\!\!\!\!\rightarrow V$ (By Definition of* ev_delphin_yes*)*
        vi. *So we need to evaluate $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P))$. By Inversion (using* ev_delphin_yes*), and Line 6(b)iv and Line 6(b)v, we see that this also $\hookrightarrow\!\!\!\!\rightarrow V$*

Lemma 6 *If t is a renaming substitution and $F \overset{fact}{\rightsquigarrow} F$, then*

$$\Gamma; \eta' \vdash F \hookrightarrow V \text{ if and only if } \Gamma; (t^{-1} \circ \eta') \vdash F[t] \hookrightarrow V$$

PROOF. *Proceed by induction on $\hookrightarrow$. Recall that the inverse of t, denoted as t^{-1}, always exists (since it is a renaming substitution). Note that some qualifying cases are shown, and all other cases trivially follow the same model.*

- ev_delphin_var
  1. *t is a renaming substitution (By Assumption)*
  2. *We must show both:*
     (a) *If $\Gamma; \eta' \vdash \mathbf{x} \hookrightarrow V$ then $\Gamma; (t^{-1} \circ \eta') \vdash \mathbf{x}[t] \hookrightarrow V$*
        i. $\Gamma; \eta' \vdash \mathbf{x} \hookrightarrow V$ *(By Assumption)*
        ii. $\Gamma; \eta' \vdash \eta'(\mathbf{x}) = V$ *(By Definition of* ev_delphin_var*)*
        iii. *By property of substitution* $\eta'(\mathbf{x}) = ((t \circ t^{-1}) \circ \eta')(\mathbf{x}) = (t^{-1} \circ \eta')(\mathbf{x}[t])$
        iv. $\Gamma; (t^{-1} \circ \eta') \vdash \mathbf{x}[t] \hookrightarrow V$ *by application of* ev_delphin_var
     (b) *If $\Gamma; (t^{-1} \circ \eta') \vdash \mathbf{x}[t] \hookrightarrow V$ then $\Gamma; \eta' \vdash \mathbf{x} \hookrightarrow V$*
        i. $\Gamma; (t^{-1} \circ \eta') \vdash \mathbf{x}[t] \hookrightarrow V$ *(By Assumption)*
        ii. $\Gamma; (t^{-1} \circ \eta') \vdash (t^{-1} \circ \eta')(\mathbf{x}[t]) = V$ *(By Definition of* ev_delphin_var*)*
        iii. *By property of substitution* $(t^{-1} \circ \eta')(\mathbf{x}[t]) = ((t \circ t^{-1}) \circ \eta')(\mathbf{x}) = \eta'(\mathbf{x})$
        iv. $\Gamma; \eta' \vdash \mathbf{x} \hookrightarrow V$ *by application of* ev_delphin_var
- ev_delphin_pair
  1. *t is a renaming substitution (By Assumption)*
  2. $\langle A_1; P_2 \rangle \overset{fact}{\rightsquigarrow} \langle A_1; P_2 \rangle$ *(By Assumption)*
  3. $A_1 \overset{fact}{\rightsquigarrow} A_1$ *(By Inversion)*
  4. $P_2 \overset{fact}{\rightsquigarrow} P_2$ *(By Inversion)*
  5. *We must show both:*
     (a) *If $\Gamma; \eta' \vdash \langle A_1; P_2 \rangle \hookrightarrow \langle V_1; V_2 \rangle$ then $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2 \rangle)[t] \hookrightarrow \langle V_1; V_2 \rangle$*
        i. $\Gamma; \eta' \vdash \langle A_1; P_2 \rangle \hookrightarrow \langle V_1; V_2 \rangle$ *(By Assumption)*
        ii. $\Gamma; \eta' \vdash A_1 \hookrightarrow V_1$ *(By Definition of* ev_delphin_pair*)*
        iii. $\Gamma; (t^{-1} \circ \eta') \vdash A_1[t] \hookrightarrow V_1$ *(By Induction Hypothesis)*
        iv. $\Gamma; \eta' \vdash P_2 \hookrightarrow V_2$ *(By Definition of* ev_delphin_pair*)*
        v. $\Gamma; (t^{-1} \circ \eta') \vdash P_2[t] \hookrightarrow V_2$ *(By Induction Hypothesis)*
        vi. $\Gamma; (t^{-1} \circ \eta') \vdash \langle A_1[t]; P_2[t] \rangle \hookrightarrow \langle V_1; V_2 \rangle$ *By Application of* ev_delphin_pair
        vii. $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2 \rangle)[t] \hookrightarrow \langle V_1; V_2 \rangle$ *By Property of Substitutions*

(b) If $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2 \rangle)[t] \hookrightarrow\!\!\!\!\rightarrow \langle V_1; V_2 \rangle$ *then* $\Gamma; \eta' \vdash \langle A_1; P_2 \rangle \hookrightarrow\!\!\!\!\rightarrow \langle V_1; V_2 \rangle$

i. $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2 \rangle)[t] \hookrightarrow\!\!\!\!\rightarrow \langle V_1; V_2 \rangle$ *(By Assumption)*

ii. $\Gamma; (t^{-1} \circ \eta')\langle A_1[t]; P_2[t] \rangle \hookrightarrow\!\!\!\!\rightarrow \langle V_1; V_2 \rangle$ *(By Property of Substitutions)*

iii. $\Gamma; (t^{-1} \circ \eta') \vdash A_1[t] \hookrightarrow\!\!\!\!\rightarrow V_1$ *(By Definition of* ev_delphin_pair*)*

iv. $\Gamma; \eta' \vdash A_1 \hookrightarrow\!\!\!\!\rightarrow V_1$ *(By Induction Hypothesis)*

v. $\Gamma; (t^{-1} \circ \eta') \vdash P_2[t] \hookrightarrow\!\!\!\!\rightarrow V_2$ *(By Definition of* ev_delphin_pair*)*

vi. $\Gamma; \eta' \vdash P_2 \hookrightarrow\!\!\!\!\rightarrow V_2$ *(By Induction Hypothesis)*

vii. $\Gamma; \eta' \vdash \langle A_1; P_2 \rangle \hookrightarrow\!\!\!\!\rightarrow \langle V_1; V_2 \rangle$ *By Application of* ev_delphin_pair

- ev_delphin_case

1. $\text{case}\ \Omega \overset{fact}{\rightsquigarrow} \text{case}\ \Omega$ *(By Assumption)*
2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Inversion)*
3. *t is a renaming substitution (By Assumption)*
4. $\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\rightarrow V$ *(By Lemma 5)*
5. *Therefore,* $\Gamma; \eta' \vdash \text{case}\ \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma; (t^{-1} \circ \eta') \vdash \text{case}\ \Omega[t] \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_case*)*

Lemma 7 *If t is a renaming substitution and $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$, then*

$$\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot \text{ if and only if } \Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$$

PROOF. *Proceed by induction on $\hookrightarrow\!\!\!\!\!\rightarrow$. Recall that $\Omega[t]$ is defined in Definition 4 and that the inverse of t, denoted as t^{-1}, always exists (since it is a renaming substitution).*

- ev_delphin_nil

  1. *t is a renaming substitution (By Assumption)*
  2. *$\Gamma \vdash \eta' \sim \cdot \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Assumption)*
  3. *$\cdot[t] = \cdot$ (By Substitution Properties)*
  4. *$\Gamma \vdash (t^{-1} \circ \eta') \sim \cdot \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By* ev_delphin_nil*)*
  5. *Therefore, $\Gamma \vdash \eta' \sim \cdot \hookrightarrow\!\!\!\!\!\rightarrow \bot$ if and only if $\Gamma \vdash (t^{-1} \circ \eta') \sim \cdot[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$*

- ev_delphin_no$\bot$

  1. *t is a renaming substitution (By Assumption)*
  2. *$(\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} (\Omega, (\Psi \triangleright \psi \mapsto P))$ (By Assumption)*
  3. *There exists some P' such that $P' \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.6)*
  4. *$P \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.1)*
  5. *$\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ (By Lemma 3.2)*
  6. *We must show both:*
     (a) *If $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$*
     then $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. *$\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Assumption)*
        ii. *For all η'', $\psi \circ \eta'' \neq \eta'$. (By Definition of* ev_delphin_no$\bot$*)*
        iii. *$\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Definition of* ev_delphin_no$\bot$*)*
        iv. *$\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Induction Hypothesis)*
        v. *$(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$ by Definition 4*
        vi. *There does not exist an η''' such that $(t^{-1} \circ \psi) \circ \eta''' = (t^{-1} \circ \eta')$ If such an η''' did exist, then $\psi \circ \eta''' = \eta'$ contradicting our above assumption*
        vii. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright \psi \mapsto P)[t]) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Application of* ev_delphin_no$\bot$*)*
        viii. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Substitution Properties)*
     (b) *If $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$*
     then $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Assumption)*
        ii. *$\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ by Definition 4 and Substitution Properties*
        iii. *For all η'', $(t^{-1} \circ \psi) \circ \eta'' \neq (t^{-1} \circ \eta')$. (By Definition of* ev_delphin_no$\bot$*)*

iv. $\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Definition of* ev_delphin_no$\bot$*)*

v. $\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Induction Hypothesis)*

vi. There does not exist an η''' *such that* $\psi \circ \eta''' = \eta'$*. If such an* η''' *did exist, then* $(t^{-1} \circ \psi) \circ \eta''' = (t^{-1} \circ \eta')$ *contradicting our above assumption*

vii. $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *by Application of* ev_delphin_no$\bot$

- ev_delphin_yes$\bot$

1. *t is a renaming substitution (By Assumption)*
2. $(\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\!\rightarrow} (\Omega, (\Psi \triangleright \psi \mapsto P))$ *(By Assumption)*
3. *There exists some* P' *such that* $P' \overset{fact}{\rightsquigarrow} P$ *(By Lemma 3.6)*
4. $P \overset{fact}{\rightsquigarrow} P$ *(By Lemma 3.1)*
5. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
6. *We must show both:*

(a) If $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
then $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$

i. $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*

ii. $\psi \circ \eta'' = \eta'$*. (By Definition of* ev_delphin_yes$\bot$*)*

iii. $(t^{-1} \circ \psi) \circ \eta'' = (t^{-1} \circ \eta')$ *(By Substitution Properties)*

iv. $\Gamma; \eta'' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Definition of* ev_delphin_yes$\bot$*)*

v. $(\Psi \triangleright \psi \mapsto P)[t] = (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)$ *by Definition 4*

vi. So we need to evaluate $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P))$*. By Inversion (using* ev_delphin_yes$\bot$*), and Line 6(a)iii and Line 6(a)iv, we see that this also* $\hookrightarrow\!\!\!\!\!\rightarrow \bot$

(b) If $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$
then $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$

i. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega, (\Psi \triangleright \psi \mapsto P))[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*

ii. $\Gamma \vdash (t^{-1} \circ \eta') \sim (\Omega[t], (\Psi \triangleright (t^{-1} \circ \psi) \mapsto P)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Definition 4 and Substitution Properties)*

iii. $(t^{-1} \circ \psi) \circ \eta'' = (t^{-1} \circ \eta')$*. (By Definition of* ev_delphin_yes$\bot$*)*

iv. $\psi \circ \eta'' = \eta'$ *(By Substitution Properties)*

v. $\Gamma; \eta'' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Definition of* ev_delphin_yes$\bot$*)*

vi. So we need to evaluate $\Gamma \vdash \eta' \sim (\Omega, (\Psi \triangleright \psi \mapsto P))$*. By Inversion (using* ev_delphin_yes$\bot$*), and Line 6(b)iv and Line 6(b)v, we see that this also* $\hookrightarrow\!\!\!\!\!\rightarrow \bot$

Lemma 8 *If t is a renaming substitution and $F \overset{fact}{\rightsquigarrow} F$, then*

$$\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\!\rightarrow \bot \textit{ if and only if } \Gamma; (t^{-1} \circ \eta') \vdash F[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$$

PROOF. *Proceed by induction on $\hookrightarrow\!\!\!\!\!\rightarrow$. Recall that the inverse of t, denoted as t^{-1}, always exists (since it is a renaming substitution). Note that some qualifying cases are shown, and all other cases trivially follow the same model.*

- ev_delphin_pair$\bot_2$

  1. *t is a renaming substitution (By Assumption)*
  2. *$\langle A_1; P_2\rangle \overset{fact}{\rightsquigarrow} \langle A_1; P_2\rangle$ (By Assumption)*
  3. *$A_1 \overset{fact}{\rightsquigarrow} A_1$ (By Inversion)*
  4. *$P_2 \overset{fact}{\rightsquigarrow} P_2$ (By Inversion)*
  5. *We must show both:*
     (a) *If $\Gamma; \eta' \vdash \langle A_1; P_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ then $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2\rangle)[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$*
        i. *$\Gamma; \eta' \vdash \langle A_1; P_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Assumption)*
        ii. *$\Gamma; \eta' \vdash A_1 \hookrightarrow\!\!\!\!\!\rightarrow V_1$ (By Definition of* ev_delphin_pair$\bot_2$*)*
        iii. *$\Gamma; (t^{-1} \circ \eta') \vdash A_1[t] \hookrightarrow\!\!\!\!\!\rightarrow V_1$ (By Lemma 6)*
        iv. *$\Gamma; \eta' \vdash P_2 \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Definition of* ev_delphin_pair$\bot_2$*)*
        v. *$\Gamma; (t^{-1} \circ \eta') \vdash P_2[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Induction Hypothesis)*
        vi. *$\Gamma; (t^{-1} \circ \eta') \vdash \langle A_1[t]; P_2[t]\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Application of* ev_delphin_pair$\bot_2$*)*
        vii. *$\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2\rangle)[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Property of Substitutions)*
     (b) *If $\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2\rangle)[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ then $\Gamma; \eta' \vdash \langle A_1; P_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$*
        i. *$\Gamma; (t^{-1} \circ \eta') \vdash (\langle A_1; P_2\rangle)[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Assumption)*
        ii. *$\Gamma; (t^{-1} \circ \eta')\langle A_1[t]; P_2[t]\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Property of Substitutions)*
        iii. *$\Gamma; (t^{-1} \circ \eta') \vdash A_1[t] \hookrightarrow\!\!\!\!\!\rightarrow V_1$ (By Definition of* ev_delphin_pair$\bot_2$*)*
        iv. *$\Gamma; \eta' \vdash A_1 \hookrightarrow\!\!\!\!\!\rightarrow V_1$ (By Lemma 6)*
        v. *$\Gamma; (t^{-1} \circ \eta') \vdash P_2[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Definition of* ev_delphin_pair$\bot_2$*)*
        vi. *$\Gamma; \eta' \vdash P_2 \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Induction Hypothesis)*
        vii. *$\Gamma; \eta' \vdash \langle A_1; P_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Application of* ev_delphin_pair$\bot_2$*)*

- ev_delphin_case$\bot$

  1. *case $\Omega \overset{fact}{\rightsquigarrow}$ case Ω (By Assumption)*
  2. *$\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\!\rightarrow} \Omega$ (By Inversion)*
  3. *t is a renaming substitution (By Assumption)*
  4. *$\Gamma \vdash \eta' \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ if and only if $\Gamma \vdash (t^{-1} \circ \eta') \sim \Omega[t] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ (By Lemma 7)*

5. *Therefore,* $\Gamma; \eta' \vdash \text{case } \Omega \hookrightarrow \bot$ *if and only if* $\Gamma; (t^{-1} \circ \eta') \vdash \text{case } \Omega[t] \hookrightarrow \bot$ *(By* ev_delphin_case$\bot$*)*

10.4 Continuation Stack Properties

Here we prove important properties for $\overset{SC}{\hookrightarrow}$. Namely, $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ if and only if either $\cdot; SC \vdash W \overset{SC}{\hookrightarrow} V$ or $(\cdot; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ and $FC \overset{FC}{\hookrightarrow} V)$

Lemma 9

1. *If* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *then either*
   - $\cdot; SC \vdash W \overset{SC}{\hookrightarrow} V$
   - *or* $(\cdot; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V)$
2. *Let* S *represent either Delphin programs* P *or stack objects* M
   If $\Gamma; \eta; FC; SC \vdash S \hookrightarrow V$ *then either*
   - $\Gamma; \eta; \cdot; SC \vdash S \hookrightarrow V$
   - *or* $(\Gamma; \eta; \cdot; SC \vdash S \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V)$
3. *If* $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *then either*
   - $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow V$
   - *or* $(\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V)$

Proof.

Part 1 – Proof over $\overset{SC}{\hookrightarrow}$

- ev_SC_empty
  1. $FC; \cdot \vdash V \overset{SC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\cdot; \cdot \vdash V \overset{SC}{\hookrightarrow} V$ *(By* ev_SC_empty*)*
- ev_SC_nonempty
  1. $FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash M[W/z] \hookrightarrow V$ *(By Definition of* ev_SC_nonempty*)*
  3. *By Part 2 of this Lemma, either*
     - $\Gamma; \eta; \cdot; SC \vdash M[W/z] \hookrightarrow V$
       - $\cdot; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} V$ *(By Application of* ev_SC_nonempty*)*
     - *or* $\Gamma; \eta; \cdot; SC \vdash M[W/z] \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\cdot; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} \bot$ *(By Application of* ev_SC_nonempty$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

Part 2 – Proof over $\hookrightarrow$

- ev_LF
  1. $\Gamma; \eta; FC; SC \vdash L \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_LF*)*
  3. *By Part 1 of this Lemma, we know that either*
     - $\cdot; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash L \hookrightarrow V$ *(By* ev_LF*)*
     - *or* $\cdot; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash L \hookrightarrow \bot$ *(By* ev_LF$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_var
  1. $\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow V$ *(By Assumption)*
  2. $\eta(\mathbf{x}) = V'$ *(By Definition of* ev_var*)*
  3. $FC; SC \vdash V' \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_var*)*
  4. *By Part 1 of this Lemma, we know that either*
     - $\cdot; SC \vdash V' \overset{SC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash V' \hookrightarrow V$ *(By* ev_var*)*
     - *or* $\cdot; SC \vdash V' \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash L \hookrightarrow \bot$ *(By* ev_var$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_unit
  1. $\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_unit*)*
  3. *By Part 1 of this Lemma, we know that either*
     - $\cdot; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \langle\rangle \hookrightarrow V$ *(By* ev_unit*)*
     - *or* $\cdot; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \langle\rangle \hookrightarrow \bot$ *(By* ev_unit$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_let
  1. $\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By Assumption)*

2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_let*)*
3. *By the Induction Hypothesis, either:*
   - $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$
     - $\Gamma; \eta; \cdot; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By* ev_let*)*
   - *or* $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
     - $\Gamma; \eta; \cdot; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow \bot$ *(By* ev_let$\bot$*)*
     - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_let_SC
  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x}\ {=}W \text{ in } P_2 \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; (\eta, W/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow V$ *(By Definition of* ev_let_SC*)*
  3. *By The Induction Hypothesis, either:*
     - $\Gamma; (\eta, W\mathbf{x}); \cdot; SC \vdash P_2 \hookrightarrow V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x}\ {=}W \text{ in } P_2 \hookrightarrow V$ *(By* ev_let_SC*)*
     - *or* $\Gamma; (\eta, W\mathbf{x}); \cdot; SC \vdash P_2 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x}\ {=}W \text{ in } P_2 \hookrightarrow \bot$ *(By* ev_let_SC$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_Λ
  1. $\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_Λ*)*
  3. *By Part 1 of this Lemma, we know that either*
     - $\cdot; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By* ev_Λ*)*
     - *or* $\cdot; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \Lambda x : A.\, P \hookrightarrow \bot$ *(By* ev_$\Lambda\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_rec
  1. $\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow V$ *(By Definition of* ev_rec*)*
  3. *By the Induction Hypothesis, either:*
     - $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); \cdot; SC \vdash P \hookrightarrow V$
       - $\Gamma; \eta; \cdot; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow V$ *(By* ev_rec*)*
     - *or* $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); \cdot; SC \vdash P \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$

* $\Gamma; \eta; \cdot; SC \vdash \mu \mathbf{x} \in F.\, P \hookrightarrow V$ *(By* ev_rec$\perp$*)*
* $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_app

1. $\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_app*)*
3. *By the Induction Hypothesis, either:*
   - $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow V$
     * $\Gamma; \eta; \cdot; SC \vdash P_1\ A_2 \hookrightarrow V$ *(By* ev_app*)*
   - *or* $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow \perp$ *and* $FC \overset{FC}{\hookrightarrow} V$
     * $\Gamma; \eta; \cdot; SC \vdash P_1\ A_2 \hookrightarrow V$ *(By* ev_app$\perp$*)*
     * $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_app_SC1

1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{app1}}\ \{\eta'; \Lambda \mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$ *(By Definition of* ev_app_SC1*)*
3. *By the Induction Hypothesis, either:*
   - $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$
     * $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{app1}}\ \{\eta'; \Lambda \mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V$ *(By* ev_app_SC1*)*
   - *or* $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow \perp$ *and* $FC \overset{FC}{\hookrightarrow} V$
     * $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{app1}}\ \{\eta'; \Lambda \mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow \perp$ *(By* ev_app_SC1$\perp$*)*
     * $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_app_SC2

1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow V$ *(By Definition of* ev_app_SC2*)*
3. *By the Induction Hypothesis, either:*
   - $\Gamma; (\eta, V_2/\mathbf{x}); \cdot; SC \vdash P_1' \hookrightarrow V$
     * $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow V$ *(By* ev_app_SC2*)*
   - *or* $\Gamma; (\eta, V_2/\mathbf{x}); \cdot; SC \vdash P_1' \hookrightarrow \perp$ *and* $FC \overset{FC}{\hookrightarrow} V$
     * $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow \perp$ *(By* ev_app_SC2$\perp$*)*
     * $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_pair

1. $\Gamma; \eta; FC; SC \vdash \langle A_1; P_2 \rangle \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2 \rangle)) \vdash A_1 \hookrightarrow V$ *(By Definition of* ev_pair*)*

3. *By the Induction Hypothesis, either:*
   - $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow V$
     - $\Gamma; \eta; \cdot; SC \vdash \langle A_1; P_2\rangle \hookrightarrow V$ *(By* ev_pair*)*
   - *or* $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
     - $\Gamma; \eta; \cdot; SC \vdash \langle A_1; P_2\rangle \hookrightarrow \bot$ *(By* ev_pair$\bot$*)*
     - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_pair_SC1
  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V$ *(By Definition of* ev_pair_SC1*)*
  3. *By the Induction Hypothesis, either:*
     - $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow V$ *(By* ev_pair_SC1*)*
     - *or* $\Gamma; \eta; \cdot; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow \bot$ *(By* ev_pair_SC1$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_pair_SC2
  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash \langle V_1; V_2\rangle \hookrightarrow V$ *(By Definition of* ev_pair_SC2*)*
  3. *By Part 1 of this Lemma, we know that either*
     - $\cdot; SC \vdash \langle V_1; V_2\rangle \overset{SC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow V$ *(By* ev_pair_SC2*)*
     - *or* $\cdot; SC \vdash \langle V_1; V_2\rangle \overset{SC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow \bot$ *(By* ev_pair_SC2$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_case
  1. $\Gamma; \eta; FC; SC \vdash \text{case } \Omega \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_case*)*
  3. *By Part 3 of this Lemma, we know that either*
     - $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow V$
       - $\Gamma; \eta; \cdot; SC \vdash \text{case } \Omega \hookrightarrow V$ *(By* ev_case*)*
     - *or* $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \eta; \cdot; SC \vdash \text{case } \Omega \hookrightarrow \bot$ *(By* ev_case$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

Part 3 – Proof over $\hookrightarrow$

- ev_yes
  1. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
  2. $\psi \circ \eta' = \eta$ *(By Definition of* ev_yes*)*
  3. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Definition of* ev_yes*)*
  4. *By Part 2 of this Lemma, we know that either*
     - $\Gamma; \eta'; \cdot; SC \vdash P \hookrightarrow V$
       - $\Gamma; \eta'; (\cdot, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Lemma 10.2)*
       - $\Gamma; \cdot; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By* ev_yes*)*
     - *or* $\Gamma; \eta'; \cdot; SC \vdash P \hookrightarrow \bot$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$
       - $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$ *(copied from Above)*
       - $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Inversion using* ev_FC_nonempty*)*
       - *By the Induction Hypothesis, we know that either*
         - $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow V$

           (a) $(\cdot, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$ *(By* ev_FC_nonempty*)*

           (b) $\Gamma; \eta'; (\cdot, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Lemma 11.2)*

           (c) $\Gamma; \cdot; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By* ev_yes*)*
         - *or* $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$

           (a) $(\cdot, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} \bot$ *(By* ev_FC_nonempty$\bot$*)*

           (b) $\Gamma; \eta'; (\cdot, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow \bot$ *(By Lemma 12.2)*

           (c) $\Gamma; \cdot; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By* ev_yes$\bot$*)*

           (d) $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_no
  1. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
  2. *There does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *(By Definition of* ev_no*)*
  3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_no*)*
  4. *By the Induction Hypothesis, we know that either*
     - $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow V$
       - $\Gamma; \cdot; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By* ev_no*)*
     - *or* $\Gamma; \cdot; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
       - $\Gamma; \cdot; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By* ev_no$\bot$*)*
       - $FC \overset{FC}{\hookrightarrow} V$ *(copied from Above)*

- ev_nil

1. $\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow V$ *(By Assumption)*
2. $\cdot \overset{FC}{\hookrightarrow} \bot$ *(By* ev_FC_empty*)*
3. $\Gamma; \cdot; SC \vdash \eta \sim \cdot \hookrightarrow \bot$ *(By* ev_nil$\bot$*)*
4. $FC \overset{FC}{\hookrightarrow} V$ *(By Definition of* ev_nil*)*

Lemma 10

1. *If* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *then for all* FC^*, $(FC^*, FC); SC \vdash W \overset{SC}{\hookrightarrow} V$
2. *Let* S *represent either Delphin programs* P *or stack objects* M
   If $\Gamma; \eta; FC; SC \vdash S \hookrightarrow V$ *then* $\Gamma; \eta; (FC^*, FC); SC \vdash S \hookrightarrow V$
3. *If* $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *then* $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$
4. *If* $FC \overset{FC}{\hookrightarrow} V$ *then* $(FC^*, FC) \overset{FC}{\hookrightarrow} V$

Proof.

Part 1 – Proof over $\overset{SC}{\hookrightarrow}$

- ev_SC_empty

  1. $FC; \cdot \vdash V \overset{SC}{\hookrightarrow} V$ *(By Assumption)*
  2. $(FC^*, FC); \cdot \vdash V \overset{SC}{\hookrightarrow} V$ *(By* ev_SC_empty*)*

- ev_SC_nonempty

  1. $FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash M[W/z] \hookrightarrow V$ *(By Definition of* ev_SC_nonempty*)*
  3. $\Gamma; \eta; (FC^*, FC); SC \vdash M[W/z] \hookrightarrow V$ *(By Part 2 of this Lemma)*
  4. $(FC^*, FC); (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} V$ *(By Application of* ev_SC_nonempty*)*

Part 2 – Proof over $\hookrightarrow$

- ev_LF
  1. $\Gamma; \eta; FC; SC \vdash L \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_LF*)*
  3. $(FC^*, FC); SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash L \hookrightarrow V$ *(By Application of* ev_LF*)*
- ev_var
  1. $\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow V$ *(By Assumption)*
  2. $\eta(\mathbf{x}) = V'$ *(By Definition of* ev_var*)*
  3. $FC; SC \vdash V' \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_var*)*
  4. $(FC^*, FC); SC \vdash V' \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash \mathbf{x} \hookrightarrow V$ *(By Application of* ev_var*)*
- ev_unit
  1. $\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_unit*)*
  3. $(FC^*, FC); SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle\rangle \hookrightarrow V$ *(By Application of* ev_unit*)*
- ev_let
  1. $\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_let*)*
  3. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$ *(By the Induction Hypothesis)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By Application of* ev_let*)*
- ev_let_SC
  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x}\ {=}W \text{ in } P_2 \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; (\eta, W/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow V$ *(By Definition of* ev_let_SC*)*
  3. $\Gamma; (\eta, W/\mathbf{x}); (FC^*, FC); SC \vdash P_2 \hookrightarrow V$ *(By Induction Hypothesis)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{let}}\ \boldsymbol{x}\ {=}W \text{ in } P_2 \hookrightarrow V$ *(By Application of* ev_let_SC*)*
- ev_Λ

1. $\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By Assumption)*
2. $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_Λ*)*
3. $(FC^*, FC); SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By Application of* ev_Λ*)*

- ev_rec

1. $\Gamma; \eta; FC; SC \vdash \mu \mathbf{x} \in F.\, P \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; (\eta, \mu \mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow V$ *(By Definition of* ev_rec*)*
3. $\Gamma; (\eta, \mu \mathbf{x} \in F.\, P/\mathbf{x}); (FC^*, FC); SC \vdash P \hookrightarrow V$ *(By Induction Hypothesis)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash \mu \mathbf{x} \in F.\, P \hookrightarrow V$ *(By Application of* ev_rec*)*

- ev_app

1. $\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_app*)*
3. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\mathbf{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow V$ *(By Induction Hypothesis)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash P_1\ A_2 \hookrightarrow V$ *(By Application of* ev_app*)*

- ev_app_SC1

1. $\Gamma; \eta; FC; SC \vdash \mathbf{app1}\ \{\eta'; \Lambda \mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\mathbf{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$ *(By Definition of* ev_app_SC1*)*
3. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta'; \lambda z.\mathbf{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$ *(By Induction Hypothesis)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash \mathbf{app1}\ \{\eta'; \Lambda \mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V$ *(By Application of* ev_app_SC1*)*

- ev_app_SC2

1. $\Gamma; \eta; FC; SC \vdash \mathbf{app2}\ P_1'\ V_2 \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow V$ *(By Definition of* ev_app_SC2*)*
3. $\Gamma; (\eta, V_2/\mathbf{x}); (FC^*, FC); SC \vdash P_1' \hookrightarrow V$ *(By Induction Hypothesis)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash \mathbf{app2}\ P_1'\ V_2 \hookrightarrow V$ *(By Application of* ev_app_SC2*)*

- ev_pair

1. $\Gamma; \eta; FC; SC \vdash \langle A_1; P_2 \rangle \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\mathbf{pair1}\langle z; P_2 \rangle)) \vdash A_1 \hookrightarrow V$ *(By Definition of* ev_pair*)*
3. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\mathbf{pair1}\langle z; P_2 \rangle)) \vdash A_1 \hookrightarrow V$ *(By Induction Hypothesis)*
4. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle A_1; P_2 \rangle \hookrightarrow V$ *(By Application of* ev_pair*)*

- ev_pair_SC1

  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V$ *(By Definition of* ev_pair_SC1*)*
  3. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V$ *(By Induction Hypothesis)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow V$ *(By Application of* ev_pair_SC1*)*

- ev_pair_SC2

  1. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow V$ *(By Assumption)*
  2. $FC; SC \vdash \langle V_1; V_2\rangle \hookrightarrow V$ *(By Definition of* ev_pair_SC2*)*
  3. $(FC^*, FC); SC \vdash \langle V_1; V_2\rangle \hookrightarrow V$ *(By Part 1 of this Lemma)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow V$ *(By Application of* ev_pair_SC2*)*

- ev_case

  1. $\Gamma; \eta; FC; SC \vdash \text{case } \Omega \hookrightarrow V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_case*)*
  3. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Part 3 of this Lemma)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{case } \Omega \hookrightarrow V$ *(By Application of* ev_case*)*

Part 3 – Proof over $\hookrightarrow$

- ev_yes

  1. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
  2. $\psi \circ \eta' = \eta$ *(By Definition of* ev_yes*)*
  3. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Definition of* ev_yes*)*
  4. $\Gamma; \eta'; ((FC^*, FC), (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Part 2 of this Lemma)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Application of* ev_yes*)*

- ev_no

  1. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
  2. *There does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *(By Definition of* ev_no*)*
  3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_no*)*
  4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By the Induction Hypothesis)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Application of* ev_no*)*

- ev_nil

  1. $\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow V$ *(By Assumption)*
  2. $FC \overset{FC}{\hookrightarrow} V$ *(By Definition of* ev_nil*)*
  3. $(FC^*, FC) \overset{FC}{\hookrightarrow} V$ *(By Part 4 of this Lemma)*
  4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \cdot \hookrightarrow V$ *(By Application of* ev_nil*)*

Part 4 – Proof over $\overset{FC}{\hookrightarrow}$

- ev_FC_nonempty

  1. $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_FC_nonempty*)*
  3. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Part 3 of this Lemma)*
  4. $FC^*, (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$ *(By Application of* ev_FC_nonempty*)*

Lemma 11

1. *If* $FC^* \overset{FC}{\hookrightarrow} V$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ *then* $(FC^*, FC); SC \vdash W \overset{SC}{\hookrightarrow} V$
2. *Let* S *represent either Delphin programs* P *or stack objects* M
   If $FC^* \overset{FC}{\hookrightarrow} V$ *and* $\Gamma; \eta; FC; SC \vdash S \hookrightarrow \bot$ *then* $\Gamma; \eta; (FC^*, FC); SC \vdash S \hookrightarrow V$
3. *If* $FC^* \overset{FC}{\hookrightarrow} V$ *and* $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *then* $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$
4. *If* $FC^* \overset{FC}{\hookrightarrow} V$ *and* $FC \overset{FC}{\hookrightarrow} \bot$ *then* $(FC^*, FC) \overset{FC}{\hookrightarrow} V$

Proof.

Part 1 – Proof over $\overset{SC}{\hookrightarrow}$

- ev_SC_nonempty$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $\Gamma; \eta; FC; SC \vdash M[W/z] \hookrightarrow \bot$ *(By Definition of* ev_SC_nonempty$\bot$*)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash M[W/z] \hookrightarrow V$ *(By Part 2 of this Lemma)*
  5. $(FC^*, FC); (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} V$ *(By Application of* ev_SC_nonempty*)*

Part 2 – Proof over $\hookrightarrow$

- ev_LF$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash L \hookrightarrow \bot$ *(By Assumption)*
  3. $FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_LF$\bot$*)*
  4. $(FC^*, FC); SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash L \hookrightarrow V$ *(By Application of* ev_LF*)*
- ev_var$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow \bot$ *(By Assumption)*
  3. $\eta(\mathbf{x}) = V'$ *(By Definition of* ev_var$\bot$*)*
  4. $FC; SC \vdash V' \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_var$\bot$*)*
  5. $(FC^*, FC); SC \vdash V' \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  6. $\Gamma; \eta; (FC^*, FC); SC \vdash \mathbf{x} \hookrightarrow V$ *(By Application of* ev_var*)*
- ev_unit$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow \bot$ *(By Assumption)*
  3. $FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_unit$\bot$*)*
  4. $(FC^*, FC); SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle\rangle \hookrightarrow V$ *(By Application of* ev_unit*)*
- ev_let$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow \bot$ *(By Assumption)*
  3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow \bot$ *(By Definition of* ev_let$\bot$*)*
  4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$ *(By the Induction Hypothesis)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By Application of* ev_let*)*
- ev_let_SC$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*

2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x} = W \text{ in } P_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, W/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow \bot$ *(By Definition of* ev_let_SC$\bot$*)*
4. $\Gamma; (\eta, W/\mathbf{x}); (FC^*, FC); SC \vdash P_2 \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{let}}\ \boldsymbol{x} = W \text{ in } P_2 \hookrightarrow V$ *(By Application of* ev_let_SC*)*

- ev_$\Lambda\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow \bot$ *(By Assumption)*
3. $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_$\Lambda\bot$*)*
4. $(FC^*, FC); SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} V$ *(By Part 1 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By Application of* ev_Λ*)*

- ev_rec$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow \bot$ *(By Definition of* ev_rec$\bot$*)*
4. $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); (FC^*, FC); SC \vdash P \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow V$ *(By Application of* ev_rec*)*

- ev_app$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\boldsymbol{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow \bot$ *(By Definition of* ev_app$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\boldsymbol{app1}\ z\ A_2)) \vdash P_1 \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash P_1\ A_2 \hookrightarrow V$ *(By Application of* ev_app*)*

- ev_app_SC1$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \boldsymbol{app1}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\boldsymbol{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow \bot$ *(By Definition of* ev_app_SC1$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta'; \lambda z.\boldsymbol{app2}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \boldsymbol{app1}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow V$ *(By Application of* ev_app_SC1*)*

- ev_app_SC2$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow \bot$ *(By Definition of* ev_app_SC2$\bot$*)*
4. $\Gamma; (\eta, V_2/\mathbf{x}); (FC^*, FC); SC \vdash P_1' \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow V$ *(By Application of* ev_app_SC2*)*

- ev_pair$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \langle A_1; P_2 \rangle \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow \bot$ *(By Definition of* ev_pair$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle A_1; P_2 \rangle \hookrightarrow V$ *(By Application of* ev_pair*)*

- ev_pair_SC1$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2 \rangle \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow \bot$ *(By Definition of* ev_pair_SC1$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow V$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2 \rangle \hookrightarrow V$ *(By Application of* ev_pair_SC1*)*

- ev_pair_SC2$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2 \rangle \hookrightarrow \bot$ *(By Assumption)*
3. $FC; SC \vdash \langle V_1; V_2 \rangle \hookrightarrow \bot$ *(By Definition of* ev_pair_SC2$\bot$*)*
4. $(FC^*, FC); SC \vdash \langle V_1; V_2 \rangle \hookrightarrow V$ *(By Part 1 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2 \rangle \hookrightarrow V$ *(By Application of* ev_pair_SC2*)*

- ev_case$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \text{case}\ \Omega \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_case$\bot$*)*
4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Part 3 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{case}\ \Omega \hookrightarrow V$ *(By Application of* ev_case*)*

Part 3 – Proof over $\hookrightarrow$

- ev_yes$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Assumption)*
  3. $\psi \circ \eta' = \eta$ *(By Definition of* ev_yes$\bot$*)*
  4. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow \bot$ *(By Definition of* ev_yes$\bot$*)*
  5. $\Gamma; \eta'; ((FC^*, FC), (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Part 2 of this Lemma)*
  6. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Application of* ev_yes*)*

- ev_no$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Assumption)*
  3. *There does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *(By Definition of* ev_no$\bot$*)*
  4. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_no$\bot$*)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By the Induction Hypothesis)*
  6. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Application of* ev_no*)*

- ev_nil$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow \bot$ *(By Assumption)*
  3. $FC \overset{FC}{\hookrightarrow} \bot$ *(By Definition of* ev_nil*)*
  4. $(FC^*, FC) \overset{FC}{\hookrightarrow} V$ *(By Part 4 of this Lemma)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \cdot \hookrightarrow V$ *(By Application of* ev_nil*)*

Part 4 – Proof over $\overset{FC}{\hookrightarrow}$

- ev_empty

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $\cdot \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $(FC^*, \cdot) = FC^*$
  4. $(FC^*, \cdot) \overset{FC}{\hookrightarrow} V$ *(By Above)*

- ev_FC_nonempty$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} V$ *(By Assumption)*
  2. $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_FC_nonempty$\bot$*)*
  4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Part 3 of this Lemma)*
  5. $FC^*, (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$ *(By Application of* ev_FC_nonempty*)*

Lemma 12

1. *If* $FC^* \overset{FC}{\hookrightarrow} \bot$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ *then* $(FC^*, FC); SC \vdash W \overset{SC}{\hookrightarrow} \bot$
2. *Let* S *represent either Delphin programs* P *or stack objects* M
   If $FC^* \overset{FC}{\hookrightarrow} \bot$ *and* $\Gamma; \eta; FC; SC \vdash S \hookrightarrow \bot$ *then* $\Gamma; \eta; (FC^*, FC); SC \vdash S \hookrightarrow \bot$
3. *If* $FC^* \overset{FC}{\hookrightarrow} \bot$ *and* $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *then* $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow \bot$
4. *If* $FC^* \overset{FC}{\hookrightarrow} \bot$ *and* $FC \overset{FC}{\hookrightarrow} \bot$ *then* $(FC^*, FC) \overset{FC}{\hookrightarrow} \bot$

Proof.

Part 1 – Proof over $\overset{SC}{\hookrightarrow}$

- ev_SC_nonempty$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $FC; (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $\Gamma; \eta; FC; SC \vdash M[W/z] \hookrightarrow \bot$ *(By Definition of* ev_SC_nonempty$\bot$*)*
  4. $\Gamma; \eta; (FC^*, FC); SC \vdash M[W/z] \hookrightarrow \bot$ *(By Part 2 of this Lemma)*
  5. $(FC^*, FC); (SC, (\Gamma; \eta; \lambda z.M)) \vdash W \overset{SC}{\hookrightarrow} \bot$ *(By Application of* ev_SC_nonempty$\bot$*)*

Part 2 – Proof over $\hookrightarrow$

- ev_LF$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash L \hookrightarrow \bot$ *(By Assumption)*
  3. $FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_LF$\bot$*)*
  4. $(FC^*, FC); SC \vdash L[\eta] \overset{SC}{\hookrightarrow} \bot$ *(By Part 1 of this Lemma)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash L \hookrightarrow \bot$ *(By Application of* ev_LF$\bot$*)*
- ev_var$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow \bot$ *(By Assumption)*
  3. $\eta(\mathbf{x}) = V'$ *(By Definition of* ev_var$\bot$*)*
  4. $FC; SC \vdash V' \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_var$\bot$*)*
  5. $(FC^*, FC); SC \vdash V' \overset{SC}{\hookrightarrow} \bot$ *(By Part 1 of this Lemma)*
  6. $\Gamma; \eta; (FC^*, FC); SC \vdash \mathbf{x} \hookrightarrow \bot$ *(By Application of* ev_var$\bot$*)*
- ev_unit$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow \bot$ *(By Assumption)*
  3. $FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_unit$\bot$*)*
  4. $(FC^*, FC); SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} \bot$ *(By Part 1 of this Lemma)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle\rangle \hookrightarrow \bot$ *(By Application of* ev_unit$\bot$*)*
- ev_let$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow \bot$ *(By Assumption)*
  3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow \bot$ *(By Definition of* ev_let$\bot$*)*
  4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}{=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow \bot$ *(By the Induction Hypothesis)*
  5. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow \bot$ *(By Application of* ev_let$\bot$*)*
- ev_let_SC$\bot$
  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*

2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{let}}\ \boldsymbol{x} = W \text{ in } P_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, W/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow \bot$ *(By Definition of* ev_let_SC$\bot$*)*
4. $\Gamma; (\eta, W/\mathbf{x}); (FC^*, FC); SC \vdash P_2 \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{let}}\ \boldsymbol{x} = W \text{ in } P_2 \hookrightarrow \bot$ *(By Application of* ev_let_SC$\bot$*)*

- ev_$\Lambda\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow \bot$ *(By Assumption)*
3. $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} \bot$ *(By Definition of* ev_$\Lambda\bot$*)*
4. $(FC^*, FC); SC \vdash \{\eta; \Lambda x : A.\, P\} \overset{SC}{\hookrightarrow} \bot$ *(By Part 1 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \Lambda x : A.\, P \hookrightarrow \bot$ *(By Application of* ev_$\Lambda\bot$*)*

- ev_rec$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow \bot$ *(By Definition of* ev_rec$\bot$*)*
4. $\Gamma; (\eta, \mu\mathbf{x} \in F.\, P/\mathbf{x}); (FC^*, FC); SC \vdash P \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \mu\mathbf{x} \in F.\, P \hookrightarrow \bot$ *(By Application of* ev_rec$\bot$*)*

- ev_app$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow \bot$ *(By Definition of* ev_app$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash P_1\ A_2 \hookrightarrow \bot$ *(By Application of* ev_app$\bot$*)*

- ev_app_SC1$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{app1}}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow \bot$ *(By Definition of* ev_app_SC1$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{app1}}\ \{\eta'; \Lambda\mathbf{x} \in A.\, P_1'\}\ A_2 \hookrightarrow \bot$ *(By Application of* ev_app_SC1$\bot$*)*

- ev_app_SC2$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; (\eta, V_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow \bot$ *(By Definition of* ev_app_SC2$\bot$*)*
4. $\Gamma; (\eta, V_2/\mathbf{x}); (FC^*, FC); SC \vdash P_1' \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{app2}}\ P_1'\ V_2 \hookrightarrow \bot$ *(By Application of* ev_app_SC2$\bot$*)*

- ev_pair$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \langle A_1; P_2 \rangle \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow \bot$ *(By Definition of* ev_pair$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P_2\rangle)) \vdash A_1 \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \langle A_1; P_2 \rangle \hookrightarrow \bot$ *(By Application of* ev_pair$\bot$*)*

- ev_pair_SC1$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow \bot$ *(By Definition of* ev_pair_SC1$\bot$*)*
4. $\Gamma; \eta; (FC^*, FC); (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle V_1; z\rangle)) \vdash P_2 \hookrightarrow \bot$ *(By Induction Hypothesis)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair1}}\langle V_1; P_2\rangle \hookrightarrow \bot$ *(By Application of* ev_pair_SC1$\bot$*)*

- ev_pair_SC2$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow \bot$ *(By Assumption)*
3. $FC; SC \vdash \langle V_1; V_2 \rangle \hookrightarrow \bot$ *(By Definition of* ev_pair_SC2$\bot$*)*
4. $(FC^*, FC); SC \vdash \langle V_1; V_2 \rangle \hookrightarrow \bot$ *(By Part 1 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \textbf{\textit{pair2}}\langle V_1; V_2\rangle \hookrightarrow \bot$ *(By Application of* ev_pair_SC2$\bot$*)*

- ev_case$\bot$

1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
2. $\Gamma; \eta; FC; SC \vdash \text{case}\ \Omega \hookrightarrow \bot$ *(By Assumption)*
3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_case$\bot$*)*
4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Part 3 of this Lemma)*
5. $\Gamma; \eta; (FC^*, FC); SC \vdash \text{case}\ \Omega \hookrightarrow \bot$ *(By Application of* ev_case$\bot$*)*

Part 3 – Proof over $\hookrightarrow$

- ev_yes$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Assumption)*
  3. $\psi \circ \eta' = \eta$ *(By Definition of* ev_yes$\bot$*)*
  4. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow \bot$ *(By Definition of* ev_yes$\bot$*)*
  5. $\Gamma; \eta'; ((FC^*, FC), (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow \bot$ *(By Part 2 of this Lemma)*
  6. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Application of* ev_yes$\bot$*)*

- ev_no$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Assumption)*
  3. *There does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *(By Definition of* ev_no$\bot$*)*
  4. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_no$\bot$*)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By the Induction Hypothesis)*
  6. $\Gamma; (FC^*, FC); SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ *(By Application of* ev_no$\bot$*)*

- ev_nil$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow \bot$ *(By Assumption)*
  3. $FC \overset{FC}{\hookrightarrow} \bot$ *(By Definition of* ev_nil*)*
  4. $(FC^*, FC) \overset{FC}{\hookrightarrow} \bot$ *(By Part 4 of this Lemma)*
  5. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \cdot \hookrightarrow \bot$ *(By Application of* ev_nil$\bot$*)*

Part 4 – Proof over $\overset{FC}{\hookrightarrow}$

- ev_empty

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $\cdot \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $(FC^*, \cdot) = FC^*$
  4. $(FC^*, \cdot) \overset{FC}{\hookrightarrow} \bot$ *(By Above)*

- ev_FC_nonempty$\bot$

  1. $FC^* \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  2. $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} \bot$ *(By Assumption)*
  3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Definition of* ev_FC_nonempty$\bot$*)*
  4. $\Gamma; (FC^*, FC); SC \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Part 3 of this Lemma)*
  5. $FC^*, (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} \bot$ *(By Application of* ev_FC_nonempty$\bot$*)*

10.5 Properties of $\star$ and $\oplus$

Lemma 13 *This Lemma has multiple parts.*

1. *If* $P_1 \overset{fact}{\rightsquigarrow} F_1$ *and* $P_2 \overset{fact}{\rightsquigarrow} F_2$ *and* $F_1 \star F_2 = P_3$ *then*
   (a) *There exists a* F_3 *such that* $P_3 \overset{fact}{\rightsquigarrow} F_3$
   (b) *If* $\Gamma; \eta \vdash F_2 \hookrightarrow\!\!\!\!\rightarrow V$ *then*
      i. $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$
      ii. *Either* $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow V$ *or* $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow \bot$
   (c) *If* $\Gamma; \eta \vdash F_2 \hookrightarrow\!\!\!\!\rightarrow \bot$ *then*
      i. $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$
      ii. $\Gamma; \eta \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma; \eta \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow \bot$

2. *If* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and* $\psi \circ \eta' = \eta$ *and* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow V$, *then* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$

   Note: η' *refers to a matching of* η *with the case* $(\Psi \triangleright \psi \mapsto P)$

3. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and* $\psi \circ \eta' = \eta$ *and* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *then*

   $$\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V \text{ if and only if } \Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$$

   Note: This means that if Ω *is factored and you add a case which matches* η *but returns* $\bot$, *then the resulting* Ω', *would behave the same as* Ω. *(See also Part 4 of this Lemma)*

4. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and* $\psi \circ \eta' = \eta$ *and* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *then*

   $$\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow \bot \text{ if and only if } \Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$$

5. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and* $\psi \circ \eta' = \eta$ *and* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow V$, *then*
   - *Either* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$
   - *or* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow \bot$

6. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and for all* η', $\psi \circ \eta' \neq \eta$ *then*

   $$\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V \text{ if and only if } \Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$$

7. *If* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *and* $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *and for all* η', $\psi \circ \eta' \neq \eta$ *then*

   $$\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow \bot \text{ if and only if } \Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$$

Proof of Part 1

PROOF. *Proof proceeds by induction over* $\star$

- merge_unit *and* merge_LF *and* merge_Λ *and* merge_rec *and* merge_app
  1. *These are all of the form:* $P \star P = P$ *(By Assumption)*
  2. *Trivially all parts are satisfied:*
     (a) *Part 1a*
     - P *is the result of something being factored (By Assumption)*
     - $P \overset{fact}{\rightsquigarrow} P$ *(By Lemma 3.1)*

     (b) *Part 1b*
     - *If* $\Gamma; \eta \vdash P \hookrightarrow V$ *then* $\Gamma; \eta \vdash P \hookrightarrow V$ *(Trivial)*

     (c) *Part 1c*
     - $\Gamma; \eta \vdash P \hookrightarrow \bot$ *(By Assumption)*
     - $\Gamma; \eta \vdash P \hookrightarrow V$ *if and only if* $\Gamma; \eta \vdash P \hookrightarrow V$ *(Trivial)*
     - $\Gamma; \eta \vdash P \hookrightarrow \bot$ *if and only if* $\Gamma; \eta \vdash P \hookrightarrow \bot$ *(Trivial)*

- merge_pair
  1. $(\langle A_{1_A}; P_{1_B} \rangle) \star (\langle A_{1_A}; P_{2_B} \rangle) = \langle A_{1_A}; P_{3_B} \rangle$ *(By Assumption)*
  2. $P_{1_B} \star P_{2_B} = P_{3_B}$ *(By Definition of* merge_pair*)*
  3. *We have three parts to prove:*
     - *Part 1a*
       (a) $\langle A_{1_A}; P_{1_B} \rangle$ *is the result of something being factored. (By Assumption)*
       (b) $\langle A_{1_A}; P_{1_B} \rangle \overset{fact}{\rightsquigarrow} \langle A_{1_A}; P_{1_B} \rangle$ *(By Lemma 3.1)*
       (c) $A_{1_A} \overset{fact}{\rightsquigarrow} A_{1_A}$ *(By Inversion)*
       (d) $P_{1_B} \overset{fact}{\rightsquigarrow} P_{1_B}$ *(By Inversion)*
       (e) $P_{3_B} \overset{fact}{\rightsquigarrow} F_{3_B}$ *(By Induction Hypothesis on Line 2)*
       (f) $\langle A_{1_A}; P_{3_B} \rangle \overset{fact}{\rightsquigarrow} \langle A_{1_A}; F_{3_B} \rangle$ *(By* fact_pair*)*
     - *Part 1b*
       (a) $\Gamma; \eta \vdash (\langle A_{1_A}; P_{2_B} \rangle) \hookrightarrow \langle V_1; V_2 \rangle$ *(By Assumption)*
       (b) $\Gamma; \eta \vdash A_{1_A} \hookrightarrow V_1$ *(By Inversion)*
       (c) $\Gamma; \eta \vdash P_{2_B} \hookrightarrow V_2$ *(By Inversion)*
       (d) *Part 1(b)i*
       - $\Gamma; \eta \vdash F_{3_B} \hookrightarrow V_2$ *(By Induction Hypothesis on Line 2)*
       - $\Gamma; \eta \vdash \langle A_{1_A}; F_{3_B} \rangle \hookrightarrow \langle V_1; V_2 \rangle$ *(By* ev_delphin_pair*)*

       (e) *Part 1(b)ii*

* *Either* $\Gamma;\eta \vdash P_{1_B} \hookrightarrow V_2$ *or* $\Gamma;\eta \vdash P_{1_B} \hookrightarrow \bot$ *(By Induction Hypothesis on Line 2)*
* *Either* $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow \langle V_1; V_2\rangle$ *or* $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow \bot$ *(By* ev_delphin_pair *and* ev_delphin_pair$\bot_2$*)*

– *Part 1c*

(a) $\Gamma;\eta \vdash (\langle A_{1_A}; P_{2_B}\rangle) \hookrightarrow \bot$ *(By Assumption)*

(b) By Inversion, Either

* $\Gamma;\eta \vdash A_{1_A} \hookrightarrow \bot$
  · $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow \bot$ *(By* ev_delphin_pair$\bot_1$*)*
  · $\Gamma;\eta \vdash \langle A_{1_A}; F_{3_B}\rangle \hookrightarrow \bot$ *(By* ev_delphin_pair$\bot_1$*)*
  · *Therefore,* $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow V$ *if and only if* $\Gamma;\eta \vdash \langle A_{1_A}; F_{3_B}\rangle \hookrightarrow \hookrightarrow V$ *(since both sides are false)*
  · *And,* $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow \bot$ *if and only if* $\Gamma;\eta \vdash \langle A_{1_A}; F_{3_B}\rangle \hookrightarrow \bot$ *(since both sides are true)*
* *or* $\Gamma;\eta \vdash A_{1_A} \hookrightarrow V_1$ *and* $\Gamma;\eta \vdash P_{2_B} \hookrightarrow \bot$
  · *Part 1(c)i*
  · *(Start)*
  · $\Gamma;\eta \vdash P_{1_B} \hookrightarrow V_2$ *if and only if* $\Gamma;\eta \vdash F_{3_B} \hookrightarrow V_2$ *(By Induction Hypothesis on Line 2)*
  · $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow V$ *if and only if* $\Gamma;\eta \vdash \langle A_{1_A}; F_{3_B}\rangle \hookrightarrow V$ *(By* ev_delphin_pair*)*
  · *(End)*
  · *Part 1(c)ii*
  · *(Start)*
  · $\Gamma;\eta \vdash P_{1_B} \hookrightarrow \bot$ *if and only if* $\Gamma;\eta \vdash F_{3_B} \hookrightarrow \bot$ *(By Induction Hypothesis on Line 2)*
  · $\Gamma;\eta \vdash \langle A_{1_A}; P_{1_B}\rangle \hookrightarrow \bot$ *if and only if* $\Gamma;\eta \vdash \langle A_{1_A}; F_{3_B}\rangle \hookrightarrow \bot$ *(By* ev_delphin_pair$\bot_2$*)*
  · *(End)*

- merge_let

1. $(\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B}) \star (\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{2_B}) = \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{3_B}$ *(By Assumption)*
2. $P_{1_B} \star P_{2_B} = P_{3_B}$ *(By Definition of* merge_let*)*
3. *We have three parts to prove:*

– *Part 1a*

(a) $\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B}$ *is the result of something being factored (By Assumption)*

(b) $\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B}$ *(By Lemma 3.1)*

(c) $P_{1_A} \overset{fact}{\rightsquigarrow} P_{1_A}$ *(By Inversion)*

(d) $P_{3_B} \overset{fact}{\rightsquigarrow} F_{3_B}$ *(By Induction Hypothesis on Line 2)*

(e) $\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{3_B} \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B}$ *(By* fact_let*)*

– *Part 1b*

(a) $\Gamma; \eta \vdash (\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{2_B}) \hookrightarrow V$ *(By Assumption)*

(b) $\Gamma; \eta \vdash P_{1_A} \hookrightarrow V_1$ *(By Inversion)*

(c) $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{2_B} \hookrightarrow V$ *(By Inversion)*

(d) *Part 1(b)i*

* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash F_{3_B} \hookrightarrow V$ *(By Induction Hypothesis on Line 2)*
* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow V$ *(By* ev_delphin_let*)*

(e) *Part 1(b)ii*

* *Either* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{1_B} \hookrightarrow V$ *or* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{1_B} \hookrightarrow \bot$ *(By Induction Hypothesis on Line 2)*
* *Either* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow V$ *or* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow \bot$ *(By* ev_delphin_let *and* ev_delphin_let$\bot_2$*)*

– *Part 1c*

(a) $\Gamma; \eta \vdash (\text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{2_B}) \hookrightarrow \bot$ *(By Assumption)*

(b) *By Inversion, Either*

* $\Gamma; \eta \vdash P_{1_A} \hookrightarrow \bot$
  - $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow \bot$ *(By* ev_delphin_let$\bot_1$*)*
  - $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow \bot$ *(By* ev_delphin_let$\bot_1$*)*
  - *Therefore,* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow V$ *if and only if* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow V$ *(since both sides are false)*
  - *And,* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow \bot$ *if and only if* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow \bot$ *(since both sides are true)*
* *or* $\Gamma; \eta \vdash P_{1_A} \hookrightarrow V_1$ *and* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{2_B} \hookrightarrow \bot$
  - *Part 1(c)i*
  - *(Start)*
  - $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{1_B} \hookrightarrow V$ *if and only if* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash F_{3_B} \hookrightarrow V$ *(By Induction Hypothesis on Line 2)*
  - $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow V$ *if and only if* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow V$ *(By* ev_delphin_let*)*
  - *(End)*
  - *Part 1(c)ii*
  - *(Start)*
  - $\Gamma; (\eta, V_1/\mathbf{x}) \vdash P_{1_B} \hookrightarrow \bot$ *if and only if* $\Gamma; (\eta, V_1/\mathbf{x}) \vdash F_{3_B} \hookrightarrow \bot$ *(By Induction Hypothesis on Line 2)*
  - $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } P_{1_B} \hookrightarrow \bot$ *if and only if* $\Gamma; \eta \vdash \text{let } \mathbf{x} : D = P_{1_A} \text{ in } F_{3_B} \hookrightarrow \bot$ *(By* ev_delphin_let$\bot_2$*)*

· *(End)*

- merge_new

  1. $\text{case}\ \Omega \star (\text{case}\ [(\Psi \triangleright \psi \mapsto P)]) = \text{case}\ (\Omega, (\Psi \triangleright \psi \mapsto P))$ *(By Assumption)*
  2. $\text{case}\ \Omega$ *is the result of something being factored. (By Assumption)*
  3. $\text{case}\ \Omega \overset{fact}{\rightsquigarrow} \text{case}\ \Omega$ *(By Lemma 3.1)*
  4. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Inversion)*
  5. $\text{case}\ [(\Psi \triangleright \psi \mapsto P)]$ *is the result of something being factored. (By Assumption)*
  6. $\text{case}\ [(\Psi \triangleright \psi \mapsto P)] \overset{fact}{\rightsquigarrow} \text{case}\ [(\Psi \triangleright \psi \mapsto P)]$ *(By Lemma 3.1)*
  7. $[(\Psi \triangleright \psi \mapsto P)] \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} [(\Psi \triangleright \psi \mapsto P)]$ *(By Inversion)*
  8. *There exists some* P' *such that* $P' \overset{fact}{\rightsquigarrow} P$ *(By Lemma 3.4)*
  9. $P \overset{fact}{\rightsquigarrow} P$ *(By Lemma 3.1 or Definition of* merge_new*)*
  10. $\Gamma; \eta \vdash \text{case}\ [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By* ev_delphin_case*)*
  11. $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow V$ *if and only if (There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow V$*)*
  12. *From the last two steps, we have that* $\Gamma; \eta \vdash \text{case}\ [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow V$ *if and only if (There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow V$*)*
  13. $\Gamma; \eta \vdash \text{case}\ [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_case$\bot$*)*
  14. $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *if and only if Either there does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *or (there exists such an* η' *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow \bot$*)*
  15. *From the last two steps, we have that* $\Gamma; \eta \vdash \text{case}\ [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *if and only if Either there does not exist an* η' *such that* $\psi \circ \eta' = \eta$ *or (there exists such an* η' *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow \bot$*)*
  16. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By Definition of* merge_new*)*
  17. *We now address Parts 1a, 1b, and 1c:*
      (a) *For Part 1a we can show that* $\text{case}\ (\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact}{\rightsquigarrow} \text{case}\ (\Omega, (\Psi \triangleright \psi \mapsto P))$*. By* fact_case*, this reduces to showing that* $(\Omega, (\Psi \triangleright \psi \mapsto P)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} (\Omega, (\Psi \triangleright \psi \mapsto P))$
         - $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi \triangleright \psi \mapsto P)$ *(By* case_new *with Line 4 and Line 9 and Line 16)*

      (b) *For Part 1b we assume* $\Gamma; \eta \vdash (\text{case}\ [(\Psi \triangleright \psi \mapsto P)]) \hookrightarrow\!\!\!\!\!\rightarrow V$*. This is true if and only if there exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Line 12). We must now show:*

- *Part 1(b)i.* $\Gamma; \eta \vdash \text{case } (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$. *This reduces (By* ev_delphin_case*) to showing that* $\Gamma \vdash \eta \sim \Omega, (\Psi \triangleright \psi \mapsto P) \hookrightarrow V$ *(This follows directly by* ev_delphin_yes*)*
- *Part 1(b)ii. Here we just to show that Either* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow V$ *or* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow \bot$. *This reduces (By* ev_delphin_case*) to showing that Either* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *or* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$.
  * *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi \triangleright \psi \mapsto P)$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi \triangleright \psi \mapsto P)$*(By Lemma 3.3)*
  * *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Above)*
  * *By Inversion and Definition 2 (using* ev_delphin_no$\bot$ *and* ev_delphin_nil*), we know that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$

(c) For Part 1c we assume $\Gamma; \eta \vdash (\text{case } [(\Psi \triangleright \psi \mapsto P)]) \hookrightarrow \bot$. *This is true if and only if* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$ *(By* ev_delphin_case$\bot$*). By Line 15, this implies two possibilities: Either (1) There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow \bot$ *or (2) for all* η', $\psi \circ \eta' \neq \eta$.

- *Part 1(c)i. Here we need to show that* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow V$ *if and only if* $\Gamma; \eta \vdash \text{case } (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$. *This reduces (By* ev_delphin_case*) to showing that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \hookrightarrow V$. *Recall that there are two ways that* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$. *We do case analysis over both cases:*
  * *There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow \bot$.
    - *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi_2 \circ t$ *and* $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi \triangleright \psi \mapsto P)$ *and* $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi \triangleright \psi \mapsto P)$*(By Lemma 3.3)*
    - *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Above)*
    - *By Inversion and Definition 2 (using* ev_delphin_no$\bot$ *and* ev_delphin_nil*), we know that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$
    - *Therefore,* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(Since both sides are false)*
  * *For all* η', $\psi \circ \eta' \neq \eta$.
    - *Therefore,* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By* ev_delphin_no*)*
- *Part 1(c)ii. Here we need to show that* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow \bot$ *if and only if* $\Gamma; \eta \vdash \text{case } (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$. *This reduces (By* ev_delphin_case$\bot$*) to showing that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \hookrightarrow \bot$. *Recall that there are two ways that* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$. *We do case analysis over both cases:*
  * *There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow \bot$.

- *There does not exist a renaming substitution t such that $\psi_1 = \psi_2 \circ t$ and $(\Psi_1 \triangleright \psi_1 \mapsto P_1) \in \Omega, (\Psi \triangleright \psi \mapsto P)$ and $(\Psi_2 \triangleright \psi_2 \mapsto P_2) \in \Omega, (\Psi \triangleright \psi \mapsto P)$(By Lemma 3.3)*
- *For all $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega$, there does not exist a renaming substitution t such that $\psi = \psi_A \circ t$ (By Above)*
- *By Inversion and Definition 2 (using* ev_delphin_no$\bot$ *and* ev_delphin_nil*), we know that $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$*
- *Therefore, $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$ if and only if $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ (Since both sides are true)*

* *For all η', $\psi \circ \eta' \neq \eta$.*

- *Therefore, $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$ if and only if $\Gamma \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow \bot$ (By* ev_delphin_no$\bot$*)*

- merge_fold

1. case $\Omega \star$ (case $[(\Psi \triangleright \psi \mapsto P)]$) = case Ω' *(By Assumption)*
2. case Ω *is the result of something being factored. (By Assumption)*
3. case $\Omega \overset{fact}{\rightsquigarrow}$ case Ω *(By Lemma 3.1)*
4. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega$ *(By Inversion)*
5. case $[(\Psi \triangleright \psi \mapsto P)]$ *is the result of something being factored. (By Assumption)*
6. case $[(\Psi \triangleright \psi \mapsto P)] \overset{fact}{\rightsquigarrow}$ case $[(\Psi \triangleright \psi \mapsto P)]$ *(By Lemma 3.1)*
7. $[(\Psi \triangleright \psi \mapsto P)] \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} [(\Psi \triangleright \psi \mapsto P)]$ *(By Inversion)*
8. *There exists some P' such that $P' \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.4)*
9. *$P \overset{fact}{\rightsquigarrow} P$ (By Lemma 3.1)*
10. *$\Gamma; \eta \vdash$* case $[(\Psi \triangleright \psi \mapsto P)] \hookrightarrow V$ *if and only if $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow V$ (By* ev_delphin_case*)*
11. *$\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow V$ if and only if (There exists an η' such that $\psi \circ \eta' = \eta$ and $\Gamma; \eta' \vdash P \hookrightarrow V$)*
12. *From the last two steps, we have that $\Gamma; \eta \vdash$* case $[(\Psi \triangleright \psi \mapsto P)] \hookrightarrow V$ *if and only if (There exists an η' such that $\psi \circ \eta' = \eta$ and $\Gamma; \eta' \vdash P \hookrightarrow V$)*
13. *$\Gamma; \eta \vdash$* case $[(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$ *if and only if $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$ (By* ev_delphin_case$\bot$*)*
14. *$\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$ if and only if Either there does not exist an η' such that $\psi \circ \eta' = \eta$ or (there exists such an η' and $\Gamma; \eta' \vdash P \hookrightarrow \bot$)*
15. *From the last two steps, we have that $\Gamma; \eta \vdash$* case $[(\Psi \triangleright \psi \mapsto P)] \hookrightarrow \bot$ *if and only if Either there does not exist an η' such that $\psi \circ \eta' = \eta$ or (there exists such an η' and $\Gamma; \eta' \vdash P \hookrightarrow \bot$)*
16. *$\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ (By Definition of* merge_fold*)*

17. *We now address Parts 1a, 1b, and 1c:*

    (a) *For Part 1a we can show that* case $\Omega' \overset{fact}{\rightsquigarrow}$ case Ω'. *By* fact_case, *this reduces to showing that* $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$

    - $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$ *(By* case_nonempty *with Line 4 and Line 16)*
    - $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega'$ *(By Lemma 3.2)*

    (b) *For Part 1b we assume* $\Gamma; \eta \vdash (\text{case } [(\Psi \triangleright \psi \mapsto P)]) \hookrightarrow V$. *This is true if and only if there exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\rightarrow V$ *(By Line 12). We must now show:*

    - *Part 1(b)i.* $\Gamma; \eta \vdash \text{case } \Omega' \hookrightarrow\!\!\!\!\rightarrow V$. *This reduces (By* ev_delphin_case*) to showing that* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$ *(This follows directly by Part 2 of this Lemma with Line 16 and Line 9)*
    - *Part 1(b)ii. Here we just to show that Either* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *or* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow \bot$. *This reduces (By* ev_delphin_case*) to showing that Either* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *or* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow \bot$. *(This follows directly by Part 5 of this Lemma with Line 16 and Line 9)*

    (c) *For Part 1c we assume* $\Gamma; \eta \vdash (\text{case } [(\Psi \triangleright \psi \mapsto P)]) \hookrightarrow\!\!\!\!\rightarrow \bot$. *This is true if and only if* $\Gamma \vdash \eta \sim [(\Psi \triangleright \psi \mapsto P)] \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_case$\bot$*). By Line 15), this implies two possibilities: Either (1) There exists an* η' *such that* $\psi \circ \eta' = \eta$ *and* $\Gamma; \eta' \vdash P \hookrightarrow\!\!\!\!\rightarrow \bot$ *or (2) for all* η', $\psi \circ \eta' \neq \eta$.

    - *Part 1(c)i. Here we need to show that* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma; \eta \vdash \text{case } \Omega' \hookrightarrow\!\!\!\!\rightarrow V$. *This reduces (By* ev_delphin_case*) to showing that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$. *(This follows via Inversion by Part 3 and Part 6 of this Lemma with Line 4 and Line 16 and Line 9)*
    - *Part 1(c)ii. Here we need to show that* $\Gamma; \eta \vdash \text{case } \Omega \hookrightarrow \bot$ *if and only if* $\Gamma; \eta \vdash \text{case } \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$. *This reduces (By* ev_delphin_case$\bot$*) to showing that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$. *(This follows via Inversion by Part 4 and Part 7 of this Lemma with Line 4 and Line 16 and Line 9)*

Proof of Part 2

PROOF. *Proof proceeds by induction on derivations using the* $\oplus$ *judgment*

- fold_no

  1. $\Omega, (\Psi_1 \rhd \psi_1 \mapsto P_1) \oplus (\Psi \rhd \psi \mapsto P) = \Omega', (\Psi_1 \rhd \psi_1 \mapsto F_1)$ *(By Assumption)*
  2. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  3. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow V$ *(By Assumption)*
  4. $\psi \circ \eta' = \eta$ *(By Assumption)*
  5. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_no*)*
  6. *There does not exist an* η'' *such that* $\psi_1 \circ \eta'' = \eta$. *(By Definition 2)*
  7. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Definition of* fold_no*)*
  8. $\Omega \oplus (\Psi \rhd \psi \mapsto P) = \Omega'$ *(By Definition of* fold_no*)*
  9. $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$ *(By Induction Hypothesis)*
  10. $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \rhd \psi_1 \mapsto F_1) \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_no *with Line 6)*
  11. $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \rhd \psi_1 \mapsto F_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(By Above)*

- fold_yes

  1. $\Omega, (\Psi_1 \rhd \psi_1 \mapsto P_1) \oplus (\Psi \rhd \psi \mapsto P) = \Omega, (\Psi_1 \rhd \psi_1 \mapsto F_3)$ *(By Assumption)*
  2. $\psi \circ \eta' = \eta$ *(By Assumption)*
  3. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  4. $F \overset{fact}{\rightsquigarrow} F$ *(By Lemma 3.1)*
  5. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow V$ *(By Assumption)*
  6. *There exists a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_yes*)*
  7. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Definition of* fold_yes*)*
  8. $F_1 \star F[t] = P_3$ *(By Definition of* fold_yes*)*
  9. $P_3 \overset{fact}{\rightsquigarrow} F_3$ *(By Definition of* fold_yes*)*
  10. *Let* $\eta'' = t^{-1} \circ \eta'$. *Recall that* $\eta = \psi \circ \eta' = (\psi_1 \circ t^{-1}) \circ \eta' = \psi_1 \circ (t^{-1} \circ \eta')$
  11. *Therefore,* $\psi_1 \circ \eta'' = \eta$
  12. $\Gamma; \eta'' \vdash F[t] \hookrightarrow\!\!\!\!\rightarrow V$ *(By Lemma 6)*
  13. $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$ *(By Part 1 of this Lemma)*
  14. $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \rhd \psi_1 \mapsto F_3) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*

Proof of Part 3

PROOF. *Proof proceeds by induction on derivations using the* $\oplus$ *judgment*

- fold_no

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact}{\rightsquigarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
  3. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
  4. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
  5. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  6. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  7. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*
  8. $\psi \circ \eta' = \eta$ *(By Assumption)*
  9. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_no*)*
  10. *There does not exist an* η'' *such that* $\psi_1 \circ \eta'' = \eta$ *(By Definition 2)*
  11. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *(By Definition of* fold_no*)*
  12. *We must show both:*
      - *(a)* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow V$
        - *i.* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Assumption)*
        - *ii.* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Inversion using* ev_delphin_no *with Line 10)*
        - *iii.* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Induction Hypothesis)*
        - *iv.* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow V$ *by* ev_delphin_no *(see Line 10)*
      - *(b)* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow V$
        - *i.* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Assumption)*
        - *ii.* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Inversion using* ev_delphin_no *with Line 10)*
        - *iii.* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow V$ *(By Induction Hypothesis)*
        - *iv.* $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow V$ *by* ev_delphin_no *(see Line 10)*

- fold_yes

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(By Assumption)*
  2. $\psi \circ \eta' = \eta$ *(By Assumption)*
  3. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  4. $F \overset{fact}{\rightsquigarrow} F$ *(By Lemma 3.1)*
  5. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*

6. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
7. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
8. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
9. *There exists a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_yes*)*
10. $P_1 \star F[t] = P_3$ *(By Definition of* fold_yes*)*
11. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Assumption)*
12. *Let* $\eta'' = t^{-1} \circ \eta'$. *Recall that* $\eta = \psi \circ \eta' = (\psi_1 \circ t^{-1}) \circ \eta' = \psi_1 \circ (t^{-1} \circ \eta')$
13. *Therefore,* $\psi_1 \circ \eta'' = \eta$
14. $\Gamma; \eta'' \vdash F[t] \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Lemma 8)*
15. $P_3 \overset{fact}{\rightsquigarrow} F_3$ *(By Definition of* fold_yes*)*
16. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow V$ *if and only if* $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*
17. $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ *(By Part 1 of this Lemma)*
18. $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*
19. *We combine the last three steps to get* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow V$

Proof of Part 4

PROOF. *Proof proceeds by induction on derivations using the* $\oplus$ *judgment*

- fold_no

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact}{\rightsquigarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  2. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
  3. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
  4. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
  5. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  6. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  7. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*
  8. $\psi \circ \eta' = \eta$ *(By Assumption)*
  9. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_no*)*
  10. *There does not exist an* η'' *such that* $\psi_1 \circ \eta'' = \eta$ *(By Definition 2)*
  11. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *(By Definition of* fold_no*)*
  12. *We must show both:*
      - *(a)* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        - *i.* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*
        - *ii.* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no$\bot$ *with Line 10)*
        - *iii.* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Induction Hypothesis)*
        - *iv.* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *by* ev_delphin_no$\bot$ *(see Line 10)*
      - *(b)* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        - *i.* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*
        - *ii.* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no$\bot$ *with Line 10)*
        - *iii.* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Induction Hypothesis)*
        - *iv.* $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *by* ev_delphin_no$\bot$ *(see Line 10)*

- fold_yes

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(By Assumption)*
  2. $\psi \circ \eta' = \eta$ *(By Assumption)*
  3. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  4. $F \overset{fact}{\rightsquigarrow} F$ *(By Lemma 3.1)*
  5. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*

6. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
7. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
8. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
9. *There exists a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_yes*)*
10. $P_1 \star F[t] = P_3$ *(By Definition of* fold_yes*)*
11. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Assumption)*
12. *Let* $\eta'' = t^{-1} \circ \eta'$. *Recall that* $\eta = \psi \circ \eta' = (\psi_1 \circ t^{-1}) \circ \eta' = \psi_1 \circ (t^{-1} \circ \eta')$
13. *Therefore,* $\psi_1 \circ \eta'' = \eta$
14. $\Gamma; \eta'' \vdash F[t] \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Lemma 8)*
15. $P_3 \overset{fact}{\rightsquigarrow} F_3$ *(By Definition of* fold_yes*)*
16. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_yes$\bot$*)*
17. $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Part 1 of this Lemma)*
18. $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_yes$\bot$*)*
19. *We combine the last three steps to get* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow \bot$

Proof of Part 5

PROOF. *Proof proceeds by induction on derivations using the* $\oplus$ *judgment*

- fold_no

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto F_1)$ *(By Assumption)*
  2. $(\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\twoheadrightarrow} (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1))$ *(By Assumption)*
  3. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\twoheadrightarrow} \Omega$ *(By Lemma 3.2)*
  4. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  5. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\twoheadrightarrow V$ *(By Assumption)*
  6. $\psi \circ \eta' = \eta$ *(By Assumption)*
  7. *There does not exist a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_no*)*
  8. *There does not exist an* η'' *such that* $\psi_1 \circ \eta'' = \eta$ *(By Definition 2)*
  9. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Definition of* fold_no*)*
  10. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *(By Definition of* fold_no*)*
  11. *Either* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *or* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By Induction Hypothesis)*
  12. $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow V$ *if and only if* $\Gamma \vdash \eta \sim \Omega \hookrightarrow V$ *(By* ev_delphin_no *and Line 8)*
  13. $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow \bot$ *if and only if* $\Gamma \vdash \eta \sim \Omega \hookrightarrow \bot$ *(By* ev_delphin_no$\bot$ *and Line 8)*
  14. *Therefore, Either* $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow V$ *or* $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow \bot$ *(Combine last three steps)*

- fold_yes

  1. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(By Assumption)*
  2. $(\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \overset{fact-case}{\rightsquigarrow\!\!\!\!\twoheadrightarrow} (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1))$ *(By Assumption)*
  3. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
  4. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
  5. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\twoheadrightarrow} \Omega$ *(By Lemma 3.2)*
  6. $\psi \circ \eta' = \eta$ *(By Assumption)*
  7. $P \overset{fact}{\rightsquigarrow} F$ *(By Assumption)*
  8. $F \overset{fact}{\rightsquigarrow} F$ *(By Lemma 3.1)*
  9. $\Gamma; \eta' \vdash F \hookrightarrow\!\!\!\!\twoheadrightarrow V$ *(By Assumption)*

10. *There exists a renaming substitution t such that $\psi_1 = \psi \circ t$ (By Definition of* fold_yes*)*
11. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Definition of* fold_yes *and copied from Above)*
12. $P_1 \star F[t] = P_3$ *(By Definition of* fold_yes*)*
13. $P_3 \overset{fact}{\rightsquigarrow} F_3$ *(By Definition of* fold_yes*)*
14. *Let $\eta'' = t^{-1} \circ \eta'$. Recall that $\eta = \psi \circ \eta' = (\psi_1 \circ t^{-1}) \circ \eta' = \psi_1 \circ (t^{-1} \circ \eta')$*
15. *Therefore, $\psi_1 \circ \eta'' = \eta$*
16. $\Gamma; \eta'' \vdash F[t] \hookrightarrow\!\!\!\!\rightarrow V$ *(By Lemma 6)*
17. $\Gamma; \eta'' \vdash F_3 \hookrightarrow\!\!\!\!\rightarrow V$ *(By Part 1 of this Lemma)*
18. *Either $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ or $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ (By Part 1 of this Lemma with Line 12)*
19. $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*
20. $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow \bot$ *if and only if* $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_yes$\bot$*)*
21. *Therefore, Either $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ or $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow \bot$ (Combine last three steps)*

Proof of Part 6

PROOF. *Proof proceeds by induction on derivations using the* $\oplus$ *judgment*

- fold_no

  1. *For all* η', $\psi \circ \eta' \neq \eta$ *(by Assumption)*
  2. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  3. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightsquigarrow} \Omega$ *(By Lemma 3.2)*
  4. *There exists some* P_1' *such that* $P_1' \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.6)*
  5. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
  6. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(by Assumption)*
  7. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *by Definition of* fold_no
  8. *We must show both:*
     (a) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$ *(by Assumption)*
        ii. *By Inversion, we have two cases to handle:*
        ev_delphin_no
        - *There exists no* η'' *such that* $\psi_1 \circ \eta'' = \eta$ *(Given)*
        - $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$ *if and only if* $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_no*)*
        - $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *(Given)*
        - *By the Induction Hypothesis, we know that* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$
        - $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(From Above)*

        ev_delphin_yes
        - $\psi_1 \circ \eta'' = \eta$ *(Given)*
        - $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ *(Given)*
        - $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*

     (b) $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$
        i. $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(by Assumption)*
        ii. *By Inversion, we have two cases to handle:*
        ev_delphin_no
        - *There exists no* η'' *such that* $\psi_1 \circ \eta'' = \eta$ *(Given)*
        - $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow V$ *(Given)*
        - *By the Induction Hypothesis, we know that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$
        - $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_no*)*

        ev_delphin_yes
        - $\psi_1 \circ \eta'' = \eta$ *(Given)*

- $\Gamma; \eta'' \vdash P_1 \hookrightarrow\!\!\!\!\rightarrow V$ *(Given)*
- $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\rightarrow V$ *(By* ev_delphin_yes*)*

- fold_yes

  1. *For all* η', $\psi \circ \eta' \neq \eta$ *(by Assumption)*
  2. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(by Assumption)*
  3. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  4. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
  5. *There exists a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_yes*)*
  6. *For all* η'', $\psi_1 \circ \eta'' \neq \eta$.
     (Proof: If there was such an η''*, then* $(\psi \circ t) \circ \eta'' = \psi \circ (t \circ \eta'') = \eta$*, which contradicts our assumption that there is no* η' *such that* $\psi \circ \eta' = \eta$ *(Let* $\eta' = (t \circ \eta'')$*)*
  7. *We must show both:*
     (a) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow V$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$ *(by Assumption)*
        ii. $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *(By Inversion using* ev_delphin_no*)*
        iii. *Therefore,* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow V$ *(By application of* ev_delphin_no*)*
     (b) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow V \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\rightarrow V$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\rightarrow V$ *(by Assumption)*
        ii. $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\rightarrow V$ *(By Inversion using* ev_delphin_no*)*
        iii. *Therefore,* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_3)) \hookrightarrow\!\!\!\!\rightarrow V$ *(By application of* ev_delphin_no*)*

Proof of Part 7

PROOF. *Proof proceeds by induction on derivations using the $\oplus$ judgment*

- fold_no

  1. *For all η', $\psi \circ \eta' \neq \eta$ (by Assumption)*
  2. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  3. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
  4. *There exists some P_1' such that $P_1' \overset{fact}{\rightsquigarrow} P_1$ (By Lemma 3.6)*
  5. $P_1 \overset{fact}{\rightsquigarrow} P_1$ *(By Lemma 3.1)*
  6. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(by Assumption)*
  7. $\Omega \oplus (\Psi \triangleright \psi \mapsto P) = \Omega'$ *by Definition of* fold_no
  8. *We must show both:*
     (a) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(by Assumption)*
        ii. *By Inversion, we have two cases to handle:*
        ev_delphin_no$\bot$
        – *There exists no η'' such that $\psi_1 \circ \eta'' = \eta$ (Given)*
        – $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(Given)*
        – *By the Induction Hypothesis, we know that* $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        – $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Application of* ev_delphin_no$\bot$*)*
        ev_delphin_yes$\bot$
        – $\psi_1 \circ \eta'' = \eta$ *(Given)*
        – $\Gamma; \eta'' \vdash P_1 \hookrightarrow \bot$ *(Given)*
        – $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Application of* ev_delphin_yes$\bot$*)*
     (b) $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. $\Gamma \vdash \eta \sim \Omega', (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Assumption)*
        ii. *By Inversion, we have two cases to handle:*
        ev_delphin_no$\bot$
        – *There exists no η'' such that $\psi_1 \circ \eta'' = \eta$ (Given)*
        – $\Gamma \vdash \eta \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(Given)*
        – *By the Induction Hypothesis, we know that* $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        – $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Application of* ev_delphin_no$\bot$*)*
        ev_delphin_yes$\bot$
        – $\psi_1 \circ \eta'' = \eta$ *(Given)*
        – $\Gamma; \eta'' \vdash P_1 \hookrightarrow \bot$ *(Given)*
        – $\Gamma \vdash \eta \sim \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Application of* ev_delphin_yes$\bot$*)*

- fold_yes

  1. *For all* η', $\psi \circ \eta' \neq \eta$ *(by Assumption)*
  2. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \oplus (\Psi \triangleright \psi \mapsto P) = \Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)$ *(by Assumption)*
  3. $\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)$ *(By Assumption)*
  4. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega$ *(By Lemma 3.2)*
  5. *There exists a renaming substitution* t *such that* $\psi_1 = \psi \circ t$ *(By Definition of* fold_yes*)*
  6. *For all* η'', $\psi_1 \circ \eta'' \neq \eta$.
     (Proof: If there was such an η'', *then* $(\psi \circ t) \circ \eta'' = \psi \circ (t \circ \eta'') = \eta$, *which contradicts our assumption that there is no* η' *such that* $\psi \circ \eta' = \eta$ *(Let* $\eta' = (t \circ \eta'')$*)*
  7. *We must show both:*
     (a) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(by Assumption)*
        ii. $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no$\bot$*)*
        iii. *Therefore,* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By application of* ev_delphin_no$\bot$*)*
     (b) $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\!\rightarrow \bot \implies \Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$
        i. $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto F_3)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(by Assumption)*
        ii. $\Gamma \vdash \eta \sim \Omega \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no$\bot$*)*
        iii. *Therefore,* $\Gamma \vdash \eta \sim (\Omega, (\Psi_1 \triangleright \psi_1 \mapsto P_1)) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By application of* ev_delphin_no$\bot$*)*

10.6 Main Lemma

Recall that in Definition 1, we define a new sense of "equality" on values. The reason for this is that if the result contains a Λ expression, then the factored version may very likely be different than the unfactored version (since we are merging cases together). However, if the result is a value which does not have any Λ terms, then we should get the same result. Therefore,

Lemma 14 (Embedding)

1. *If* $\Gamma; \eta; FC; SC \vdash P \hookrightarrow V$ *and* $P \overset{fact}{\rightsquigarrow} F$ *and* $\eta \overset{*}{\Rightarrow} \eta_1$ *then:*
   - *Either* $\Gamma; \eta_1 \vdash F \hookrightarrow\!\!\!\!\rightarrow W'$ *and*
     - *There exists a* W *such that* $W \overset{*}{\Rightarrow} W'$
     - $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
   - *or both* $\Gamma; \eta_1 \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
2. *If* $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *and* $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *and* $\eta \overset{*}{\Rightarrow} \eta_1$*, then:*
   - *Either* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow W'$ *and*
     - *There exists a* W *such that* $W \overset{*}{\Rightarrow} W'$
     - $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
   - *or both* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$

PROOF. *We prove this by induction on* $\hookrightarrow$
Proof of Part 1

- ev_LF
  1. $\Gamma; \eta; FC; SC \vdash L \hookrightarrow V$ *(By Assumption)*
  2. $L \overset{fact}{\rightsquigarrow} L$ *(By* fact_LF*)*
  3. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*
  4. $\Gamma; \eta_1 \vdash L \hookrightarrow\!\!\!\!\rightarrow L[\eta_1]$ *(By* ev_delphin_LF*)*
  5. *Therefore, we must show that* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *where* $W \overset{*}{\Rightarrow} L[\eta_1]$
     (a) $L[\eta] \overset{*}{\Rightarrow} L[\eta_1]$ *(By Definition of 1)*
     (b) $FC; SC \vdash L[\eta] \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_LF*)*
     (c) *(Note that we have* $W = L[\eta]$*)*
- ev_var
  1. $\Gamma; \eta; FC; SC \vdash \mathbf{x} \hookrightarrow V$ *(By Assumption)*

2. $\mathbf{x} \overset{fact}{\rightsquigarrow} \mathbf{x}$ *(By* fact_var*)*

3. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*

4. $\eta(\mathbf{x}) = V'$ *(By Definition of* ev_var*)*

5. $\eta_1(\mathbf{x}) = W'$ *(By Definition 1)*

6. $\Gamma; \eta_1 \vdash \mathbf{x} \hookrightarrow W'$ *(By* ev_delphin_var*)*

7. *Therefore, we must show that* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *where* $W \overset{*}{\Rightarrow} w'$

   (a) $\Gamma; \eta \vdash \eta(\mathbf{x}) = W$ *such that* $W \overset{*}{\Rightarrow} W'$ *(By Definition 1)*

   (b) $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_var*)*

- ev_unit

1. $\Gamma; \eta; FC; SC \vdash \langle\rangle \hookrightarrow V$ *(By Assumption)*

2. $\langle\rangle \overset{fact}{\rightsquigarrow} \langle\rangle$ *(By* fact_unit*)*

3. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*

4. $\Gamma; \eta_1 \vdash \langle\rangle \hookrightarrow \langle\rangle$ *(By* ev_delphin_unit*)*

5. *Therefore, we must show that* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *where* $W \overset{*}{\Rightarrow} \langle\rangle$

   (a) $\langle\rangle \overset{*}{\Rightarrow} \langle\rangle$ *(By Definition 1)*

   (b) $FC; SC \vdash \langle\rangle \overset{SC}{\hookrightarrow} V$ *(By Definition of* ev_unit*)*

- ev_let

1. $\Gamma; \eta; FC; SC \vdash \text{let } \mathbf{x} = P_1 \text{ in } P_2 \hookrightarrow V$ *(By Assumption)*

2. $\text{let } \mathbf{x} = P_1 \text{ in } P_2 \overset{fact}{\rightsquigarrow} \text{let } \mathbf{x} = F_1 \text{ in } F_2$ *(By Assumption and* fact_let*)*

3. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Inversion using* fact_let*)*

4. $P_2 \overset{fact}{\rightsquigarrow} F_2$ *(By Inversion using* fact_let*)*

5. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}\ {=}z \text{ in } P_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_let*)*

6. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*

7. *By the Induction Hypothesis on Line 5, we know that:*

   (a) *Either* $\Gamma; \eta_1 \vdash F_1 \hookrightarrow W_1'$ *and there exists a* W_1 *such that* $W_1 \overset{*}{\Rightarrow} W_1'$ *and* $FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{let}}\ \boldsymbol{x}\ {=}z \text{ in } P_2)) \vdash W_1 \overset{SC}{\hookrightarrow} V$

      i. $\Gamma; (\eta, W_1/\mathbf{x}); FC; SC \vdash P_2 \hookrightarrow V$ *(By Inversion using* ev_SC *and* ev_let_SC*)*

      ii. $(\eta, W_1/\mathbf{x}) \overset{*}{\Rightarrow} (\eta_1, W_1'/\mathbf{x})$ *(By Definition 1)*

      iii. *By the Induction Hypothesis on Line 7(a)i, we know that:*

         A. *Either* $\Gamma; (\eta_1, W_1'/\mathbf{x}) \vdash F_2 \hookrightarrow W_2'$ *and there exists a* W_2 *such that* $W_2 \overset{*}{\Rightarrow} W_2'$ *and* $FC; SC \vdash W_2 \overset{SC}{\hookrightarrow} V$

- $\Gamma; \eta_1 \vdash \text{let } \mathbf{x} = F_1 \text{ in } F_2 \hookrightarrow\!\!\!\!\rightarrow W_2'$ *(By* ev_delphin_let*)*
- $W_2 \stackrel{*}{\Rightarrow} W_2'$ *(Given)*
- $FC; SC \vdash W_2 \stackrel{SC}{\hookrightarrow} V$ *(Given)*

B. or both $\Gamma; (\eta_1, W_1'/\mathbf{x}) \vdash F_2 \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $FC \stackrel{FC}{\hookrightarrow} V$

- $\Gamma; \eta_1 \vdash \text{let } \mathbf{x} = F_1 \text{ in } F_2 \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_let$\bot_2$*)*
- $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*

(b) or both $\Gamma; \eta_1 \vdash F_1 \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $FC \stackrel{FC}{\hookrightarrow} V$

- $\Gamma; \eta_1 \vdash \text{let } \mathbf{x} = F_1 \text{ in } F_2 \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_let$\bot_1$*)*
- $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*

- ev_Λ

1. $\Gamma; \eta; FC; SC \vdash \Lambda x : A.\, P \hookrightarrow V$ *(By Assumption)*
2. $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \stackrel{SC}{\hookrightarrow} V$ *(By Definition of* ev_Λ*)*
3. $\Lambda x : A.\, P \stackrel{fact}{\rightsquigarrow} \Lambda x : A.\, F$ *(By Assumption and Inversion on* fact_Λ*)*
4. $P \stackrel{fact}{\rightsquigarrow} F$ *(By Inversion using* fact_Λ*)*
5. $\eta \stackrel{*}{\Rightarrow} \eta_1$ *(By Assumption)*
6. $\Gamma; \eta_1 \vdash \Lambda x : A.\, F \hookrightarrow\!\!\!\!\rightarrow \{\eta_1; \Lambda x : A.\, F\}$ *(By* ev_delphin_Λ*)*
7. *Therefore, we must show that* $FC; SC \vdash W \stackrel{SC}{\hookrightarrow} V$ *where* $W \stackrel{*}{\Rightarrow} \{\eta_1; \Lambda x : A.\, F\}$
   (a) $\{\eta; \Lambda x : A.\, P\} \stackrel{*}{\Rightarrow} \{\eta_1; \Lambda x : A.\, F\}$ *(By Definition 1) (Recall that this is what makes the definition of* $\stackrel{*}{\Rightarrow}$ *different from regular equality)*
   (b) $FC; SC \vdash \{\eta; \Lambda x : A.\, P\} \stackrel{SC}{\hookrightarrow} V$ *(Copied from Line 2)*

- ev_rec

1. $\Gamma; \eta; FC; SC \vdash \mu\mathbf{x} \in F^*.\, P \hookrightarrow V$ *(By Assumption)*
2. $\Gamma; (\eta, \mu\mathbf{x} \in F^*.\, P/\mathbf{x}); FC; SC \vdash P \hookrightarrow V$ *(By Definition of* ev_rec*)*
3. $\mu\mathbf{x} \in F^*.\, P \stackrel{fact}{\rightsquigarrow} \mu\mathbf{x} \in F^*.\, F$ *(By Assumption and Inversion on* fact_rec*)*
4. $P \stackrel{fact}{\rightsquigarrow} F$ *(By Inversion using* fact_rec*)*
5. $\eta \stackrel{*}{\Rightarrow} \eta_1$ *(By Assumption)*
6. $(\eta, \mu\mathbf{x} \in F^*.\, P/\mathbf{x}) \stackrel{*}{\Rightarrow} (\eta_1, \mu\mathbf{x} \in F^*.\, F/\mathbf{x})$ *(By Definition 1)*
7. *By the Induction Hypothesis on Line 2, we know that:*
   (a) Either $\Gamma; (\eta_1, \mu\mathbf{x} \in F^*.\, F/\mathbf{x}) \vdash F \hookrightarrow\!\!\!\!\rightarrow W'$ *and there exists a* W *such that* $W \stackrel{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \stackrel{SC}{\hookrightarrow} V$
      i. $\Gamma; \eta_1 \vdash \mu\mathbf{x} \in F^*.\, F \hookrightarrow\!\!\!\!\rightarrow W'$ *(By* ev_delphin_rec*)*

ii. There exists a W such that $W \stackrel{*}{\Rightarrow} W'$ (Given)

iii. $FC; SC \vdash W \stackrel{SC}{\hookrightarrow} V$ (Given)

(b) or both $\Gamma; (\eta_1, \mu\mathbf{x} \in F^*.\, F/\mathbf{x}) \vdash F \hookrightarrow\!\!\!\!\!\rightarrow \bot$ and $FC \stackrel{FC}{\hookrightarrow} V$

- $\Gamma; \eta_1 \vdash \mu\mathbf{x} \in F^*.\, F \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_rec$\bot$*)*
- $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*

- ev_pair

1. $\Gamma; \eta; FC; SC \vdash \langle A_1; P_2\rangle \hookrightarrow V$ *(By Assumption)*
2. $\langle A_1; P_2\rangle \stackrel{fact}{\rightsquigarrow} \langle F_1; F_2\rangle$ *(By Assumption and* fact_pair*)*
3. $A_1 \stackrel{fact}{\rightsquigarrow} F_1$ *(By Inversion using* fact_pair*)*
4. $P_2 \stackrel{fact}{\rightsquigarrow} F_2$ *(By Inversion using* fact_pair*)*
5. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P2\rangle)) \vdash A_1 \hookrightarrow V$ *(By Definition of* ev_pair*)*
6. $\eta \stackrel{*}{\Rightarrow} \eta_1$ *(By Assumption)*
7. *By the Induction Hypothesis on Line 5, we know that:*
   (a) Either $\Gamma; \eta_1 \vdash F_1 \hookrightarrow\!\!\!\!\!\rightarrow W_1'$ and there exists a W_1 such that $W_1 \stackrel{*}{\Rightarrow} W_1'$ and $FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair1}}\langle z; P2\rangle)) \vdash W_1 \stackrel{SC}{\hookrightarrow} V$
      i. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle W_1; z\rangle)) \vdash P_2 \hookrightarrow V$ (By Inversion using ev_SC *and* ev_pair_SC1*)*
      ii. By the Induction Hypothesis on Line 7(a)i, we know that:
         A. Either $\Gamma; \eta_1 \vdash F_2 \hookrightarrow\!\!\!\!\!\rightarrow W_2'$ and there exists a W_2 such that $W_2 \stackrel{*}{\Rightarrow} W_2'$ and $FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{pair2}}\langle W_1; z\rangle)) \vdash W_2 \stackrel{SC}{\hookrightarrow} V$
            - $\Gamma; \eta_1 \vdash \langle F_1; F_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \langle W_1'; W_2'\rangle$ *(By* ev_delphin_pair*)*
            - $\langle W_1; W_2\rangle \stackrel{*}{\Rightarrow} \langle W_1'; W_2'\rangle$ *(By Definition 1)*
            - $FC; SC \vdash \langle W_1; W_2\rangle \stackrel{SC}{\hookrightarrow} V$ *(By Inversion using* ev_SC *and* ev_pair_SC2*)*
         B. or both $\Gamma; \eta_1 \vdash F_2 \hookrightarrow\!\!\!\!\!\rightarrow \bot$ and $FC \stackrel{FC}{\hookrightarrow} V$
            - $\Gamma; \eta_1 \vdash \langle F_1; F_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_pair$\bot_2$*)*
            - $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*
   (b) or both $\Gamma; \eta_1 \vdash F_1 \hookrightarrow\!\!\!\!\!\rightarrow \bot$ and $FC \stackrel{FC}{\hookrightarrow} V$
      - $\Gamma; \eta_1 \vdash \langle F_1; F_2\rangle \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_pair$\bot_1$*)*
      - $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*

- ev_app

1. $\Gamma; \eta; FC; SC \vdash P_1\ A_2 \hookrightarrow V$ *(By Assumption)*
2. $P_1\ A_2 \stackrel{fact}{\rightsquigarrow} F_1\ F_2$ *(By Assumption and* fact_app*)*

3. $P_1 \overset{fact}{\rightsquigarrow} F_1$ *(By Inversion using* fact_app*)*
4. $A_2 \overset{fact}{\rightsquigarrow} F_2$ *(By Inversion using* fact_app*)*
5. $\Gamma; \eta; FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash P_1 \hookrightarrow V$ *(By Definition of* ev_app*)*
6. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*
7. *By the Induction Hypothesis on Line 5, we know that:*
   (a) *Either* $\Gamma; \eta_1 \vdash F_1 \hookrightarrow W_1'$ *and there exists a* W_1 *such that* $W_1 \overset{*}{\Rightarrow} W_1'$ *and* $FC; (SC, (\Gamma; \eta; \lambda z.\textbf{\textit{app1}}\ z\ A_2)) \vdash W_1 \overset{SC}{\hookrightarrow} V$
      i. $W_1 = \{\eta'; \Lambda \mathbf{x} \in A.\ P_1'\}$ *and* $\Gamma; \eta; FC; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash A_2 \hookrightarrow V$ *(By Inversion using* ev_SC *and* ev_app_SC1*)*
      ii. $W_1' = \{\eta''; \Lambda \mathbf{x} \in A.\ F_1'\}$ *where* $\eta' \overset{*}{\Rightarrow} \eta''$ *and* $P_1' \overset{fact}{\rightsquigarrow} F_1'$ *(By Definition 1)*
      iii. *By the Induction Hypothesis on Line 7(a)i, we know that:*
         – *Either* $\Gamma; \eta_1 \vdash F_2 \hookrightarrow W_2'$ *and there exists a* W_2 *such that* $W_2 \overset{*}{\Rightarrow} W_2'$ *and* $FC; (SC, (\Gamma; \eta'; \lambda z.\textbf{\textit{app2}}\ P_1'\ z)) \vdash W_2 \overset{SC}{\hookrightarrow} V$
            * $\Gamma; (\eta', W_2/\mathbf{x}); FC; SC \vdash P_1' \hookrightarrow V$ *(By Inversion using* ev_SC *and* ev_app_SC2*)*
            * $(\eta', W_2/\mathbf{x}) \overset{*}{\Rightarrow} (\eta'', W_2'/\mathbf{x})$ *(By Definition 1)*
            * *By the Induction Hypothesis (with last two lines), we know that*

              ———————————————————————————

              (1) Either $\Gamma; (\eta'', W_2'/\mathbf{x}) \vdash F_1' \hookrightarrow W_3'$ *and there exists a* W_3 *such that* $W_3 \overset{*}{\Rightarrow} W_3'$ *and* $FC; SC \vdash W_3 \overset{SC}{\hookrightarrow} V$

              ———————————————————————————

              – $\Gamma; \eta_1 \vdash F_1\ F_2 \hookrightarrow W_3'$ *(By* ev_delphin_app*)*
              – $W_3 \overset{*}{\Rightarrow} W_3'$ *(Given)*
              – $FC; SC \vdash W_3 \overset{SC}{\hookrightarrow} V$ *(Given)*

              ———————————————————————————

              (2) or both $\Gamma; (\eta'', W_2'/\mathbf{x}) \vdash F_1' \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$

              ———————————————————————————

              – $\Gamma; \eta_1 \vdash F_1\ F_2 \hookrightarrow \bot$ *(By* ev_delphin_app$\bot_3$*)*
              – $FC \overset{FC}{\hookrightarrow} V$ *(Given)*

              ———————————————————————————
         – *or both* $\Gamma; \eta_1 \vdash F_2 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
            * $\Gamma; \eta_1 \vdash F_1\ F_2 \hookrightarrow \bot$ *(By* ev_delphin_app$\bot_2$*)*
            * $FC \overset{FC}{\hookrightarrow} V$ *(Given)*
   (b) *or both* $\Gamma; \eta_1 \vdash F_1 \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
      – $\Gamma; \eta_1 \vdash F_1\ F_2 \hookrightarrow \bot$ *(By* ev_delphin_pair$\bot_1$*)*
      – $FC \overset{FC}{\hookrightarrow} V$ *(Given)*

- ev_case

1. $\Gamma; \eta; FC; SC \vdash \text{case } \Omega \hookrightarrow V$ *(By Assumption)*
2. $\text{case } \Omega \overset{fact}{\rightsquigarrow} \text{case } \Omega'$ *(By Assumption and Inversion)*
3. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Inversion on* fact_case*)*
4. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_case*)*
5. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*
6. *By Part 2 of this Lemma, we know:*
   - *Either* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow W'$ *and there exists a* W *such that* $W \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
     - $\Gamma; \eta_1 \vdash \text{case } \Omega' \hookrightarrow W'$ *if and only if* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow W'$ *(By* ev_delphin_case*)*
     - *Therefore,* $\Gamma; \eta_1 \vdash \text{case } \Omega' \hookrightarrow W'$ *and there exists a* W *such that* $W \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
   - *or both* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
     - $\Gamma; \eta_1 \vdash \text{case } \Omega' \hookrightarrow \bot$ *if and only if* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow \bot$ *(By* ev_delphin_case$\bot$*)*
     - $\Gamma; \eta_1 \vdash \text{case } \Omega' \hookrightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$

Proof of Part 2

- ev_nil

  1. $\Gamma; FC; SC \vdash \eta \sim \cdot \hookrightarrow V$ *(By Assumption)*
  2. $\cdot \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \cdot$ *(By* case_empty*)*
  3. $\Gamma; \eta_1 \sim \cdot \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_nil*)*
  4. *Therefore we just need to show that* $FC \overset{FC}{\hookrightarrow} V$
  5. $FC \overset{FC}{\hookrightarrow} V$ *(By Definition of* ev_nil*)*

- ev_no

  1. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
  2. $\psi \circ \eta' \neq \eta$ *for all* η' *(By Definition of* ev_no*)*
  3. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Definition of* ev_no*)*
  4. *By Assumption, we know that* $(\Omega, (\Psi \triangleright \psi \mapsto P))$ *can be factored. So, by Inversion we have two possibilities:*
     - case_nonempty
       - $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega''$ *(By Assumption)*
       - $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Inversion using* case_nonempty*)*
       - $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Lemma 3.2)*
       - $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) = \Omega''$ *(By Inversion using* case_nonempty*)*
       - $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*
       - $\psi \circ \eta' \neq \eta_1$ *for all* η' *(By Lemma 2)*
       - *By the Induction Hypothesis on Line 3, we know that:*

       (a) Either $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow W'$ *and* $W \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
         i. $\Gamma \vdash \eta_1 \sim \Omega'' \hookrightarrow\!\!\!\!\!\rightarrow W'$ *(By Lemma 13.6)*
         ii. $W \overset{*}{\Rightarrow} W'$ *(Given)*
         iii. $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *(Given)*

       (b) or both $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
         i. $\Gamma \vdash \eta_1 \sim \Omega'' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By Lemma 13.7)*
         ii. $FC \overset{FC}{\hookrightarrow} V$ *(Given)*
     - case_new
       - $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega', (\Psi \triangleright \psi \mapsto F)$ *(By Assumption)*
       - $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Inversion using* case_new*)*
       - $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Lemma 3.2)*

* $\eta \stackrel{*}{\Rightarrow} \eta_1$ *(By Assumption)*
* $\psi \circ \eta' \neq \eta_1$ *for all* η' *(By Lemma 2)*
* *By the Induction Hypothesis on Line 3, we know that:*

(a) Either $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow W'$ *and* $W \stackrel{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \stackrel{SC}{\hookrightarrow} V$

i. $\Gamma \vdash \eta_1 \sim \Omega', (\Psi \triangleright \psi \mapsto F) \hookrightarrow\!\!\!\!\!\rightarrow W'$ *(By* ev_delphin_no*)*

ii. $W \stackrel{*}{\Rightarrow} W'$ *(Given)*

iii. $FC; SC \vdash W \stackrel{SC}{\hookrightarrow} V$ *(Given)*

(b) or both $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *and* $FC \stackrel{FC}{\hookrightarrow} V$

i. $\Gamma \vdash \eta_1 \sim \Omega', (\Psi \triangleright \psi \mapsto F) \hookrightarrow\!\!\!\!\!\rightarrow \bot$ *(By* ev_delphin_no$\bot$*)*

ii. $FC \stackrel{FC}{\hookrightarrow} V$ *(Given)*

- ev_yes

1. *Here we prove a property that will be used in this case*
   If
   (a) $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$
   (b) *and* $\eta \overset{*}{\Rightarrow} \eta_1$
   (c) *and there exists a* W *such that* $W \overset{*}{\Rightarrow} W'$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$
   (d) *and either* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow W'$ *or* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$
   Then

   $$\textit{there exists an } \hat{W} \textit{ such that } \hat{W} \overset{*}{\Rightarrow} W' \textit{ and } FC; SC \vdash \hat{W} \overset{SC}{\hookrightarrow} V$$

   Proof
   We are given that $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$. *By Lemma 9.1, we know that either*
   - $\cdot; SC \vdash W \overset{SC}{\hookrightarrow} V$
     * $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *(By Lemma 10.1)*
   - *or* $(\cdot; SC \vdash W \overset{SC}{\hookrightarrow} \bot$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V)$
     * $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Inversion using* ev_FC_nonempty*)*
     * *We are given that either:*
       · $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow W'$. *In this case there exists a* $\hat{W}$ *such that* $\hat{W} \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash \hat{W} \overset{SC}{\hookrightarrow} V$ *(By Induction Hypothesis)*
       · *or* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$. *In this case, we know that both*
       $FC \overset{FC}{\hookrightarrow} V$ *(By Induction Hypothesis) and*
       $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *(By Lemma 11.1)*
2. $\Gamma; FC; SC \vdash \eta \sim (\Omega, (\Psi \triangleright \psi \mapsto P)) \hookrightarrow V$ *(By Assumption)*
3. $\psi \circ \eta' = \eta$ *(By Definition of* ev_yes*)*
4. $\eta \overset{*}{\Rightarrow} \eta_1$ *(By Assumption)*
5. $\psi \circ \eta'' = \eta_1$ *where* $\eta' \overset{*}{\Rightarrow} \eta''$ *(By Lemma 1)*
6. *By Assumption, we know that* $(\Omega, (\Psi \triangleright \psi \mapsto P))$ *can be factored. So, there are two possibilities. Either we use* case_nonempty *or we use* case_new

   ————————————————————————————

7. *We first focus on* case_nonempty
   ————————————————————————————
8. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega''$ *(By Assumption)*

9. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Inversion using* case_nonempty*)*

10. $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Lemma 3.2)*

11. $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) = \Omega''$ *(By Inversion using* case_nonempty*)*

12. $P \overset{fact}{\rightsquigarrow} F$ *(By Lemma 4)*

13. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Definition of* ev_yes*)*

14. *By Part 1 of this Lemma on Line 13, we know that:*

    (a) *Either* $\Gamma; \eta'' \vdash F \hookrightarrow\!\!\!\!\rightarrow W'$ *and* $W \overset{*}{\Rightarrow} W'$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$

        i. $\Gamma \vdash \eta_1 \sim \Omega'' \hookrightarrow\!\!\!\!\rightarrow W'$ *(By Lemma 13.2 with Line 11)*

        ii. $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$ *(Given)*

        iii. *Either* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow W'$ *or* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Lemma 13.5 with Line 11)*

        iv. *There exists a* W_1 *such that* $W_1 \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W_1 \overset{SC}{\hookrightarrow} V$ *(By Line 1)*

    (b) *or both* $\Gamma; \eta'' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$

        i. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Inversion using* ev_FC_nonempty*)*

        ii. *By the Induction Hypothesis on the previous line, we know that:*

        - *Either* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow W'$ *and* $W \overset{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$
            * $\Gamma \vdash \eta_1 \sim \Omega'' \hookrightarrow\!\!\!\!\rightarrow W'$ *(By Lemma 13.3 with Line 11)*
            * $W \overset{*}{\Rightarrow} W'$ *(Given)*
            * $FC; SC \vdash W \overset{SC}{\hookrightarrow} V$ *(Given)*
        - *or both* $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $FC \overset{FC}{\hookrightarrow} V$
            * $\Gamma \vdash \eta_1 \sim \Omega'' \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Lemma 13.4 with Line 11)*
            * $FC \overset{FC}{\hookrightarrow} V$ *(Given)*

---

15. *We now focus on* case_new

---

16. $\Omega, (\Psi \triangleright \psi \mapsto P) \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega', (\Psi \triangleright \psi \mapsto F)$ *(By Assumption)*

17. $\Omega \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Inversion using* case_new*)*

18. $\Omega' \overset{fact-case}{\rightsquigarrow\!\!\!\!\rightarrow} \Omega'$ *(By Lemma 3.2)*

19. $P \overset{fact}{\rightsquigarrow} F$ *(By Definition of* case_new*)*

20. $\Omega' \oplus (\Psi \triangleright \psi \mapsto P) \uparrow$ *(By Definition of* case_new*)*

21. $\Gamma; \eta'; (FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash P \hookrightarrow V$ *(By Definition of* ev_yes*)*

22. *By Part 1 of this Lemma on Line 13, we know that:*

(a) *Either* $\Gamma; \eta'' \vdash F \hookrightarrow\!\!\!\!\rightarrow W'$ *and* $W \stackrel{*}{\Rightarrow} W'$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$

i. $\Gamma \vdash \eta_1 \sim \Omega', (\Psi \triangleright \psi \mapsto F) \hookrightarrow\!\!\!\!\rightarrow W'$ *(By* ev_delphin_yes*)*

ii. $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)); SC \vdash W \overset{SC}{\hookrightarrow} V$ *(Given)*

iii. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega'$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Lemma 3.9 with Line 20)*

iv. $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no *and* ev_delphin_nil *and Definition 2)*

v. *There exists a* W_1 *such that* $W_1 \stackrel{*}{\Rightarrow} W'$ *and* $FC; SC \vdash W_1 \overset{SC}{\hookrightarrow} V$ *(By Line 1)*

(b) *or both* $\Gamma; \eta'' \vdash F \hookrightarrow\!\!\!\!\rightarrow \bot$ *and* $(FC, (\Gamma; \eta; SC; \textbf{\textit{case}}\ \Omega)) \overset{FC}{\hookrightarrow} V$

i. *For all* $(\Psi_A \triangleright \psi_A \mapsto P_A) \in \Omega'$*, there does not exist a renaming substitution* t *such that* $\psi = \psi_A \circ t$ *(By Lemma 3.9 with Line 20)*

ii. $\Gamma \vdash \eta_1 \sim \Omega' \hookrightarrow\!\!\!\!\rightarrow \bot$ *(By Inversion using* ev_delphin_no *and* ev_delphin_nil *and Definition 2)*

iii. $\Gamma; FC; SC \vdash \eta \sim \Omega \hookrightarrow V$ *(By Inversion using* ev_FC_nonempty*)*

iv. *By the Induction Hypothesis with the previous two lines, we know that* $FC \overset{FC}{\hookrightarrow} V$

□

10.7 Bringing it Together

We want to prove the following theorem:

Theorem 1 (Main) *If* $P \overset{fact}{\rightsquigarrow} F$ *and* $\Gamma; \cdot; \cdot; \cdot \vdash P \hookrightarrow V$ *and* V *does not contain any* Λ *terms, then* $\Gamma; \cdot \vdash F \hookrightarrow V$

PROOF. *Apply Lemma 14 with* $FC = \cdot$ *and* $SC = \cdot$ *and* $\eta_1 = \eta = \cdot$

- *If* $\Gamma; \eta \vdash F \hookrightarrow \bot$, *then* $\cdot \overset{FC}{\hookrightarrow} V$. *However, we know that* $\cdot \overset{FC}{\hookrightarrow} \bot$ *(By* ev_FC_empty*), so this must not be the case.*
- *Therefore, we know that* $\Gamma; \eta \vdash F \hookrightarrow W'$ *and there exists a* W *such that* $\cdot; \cdot \vdash W \overset{SC}{\hookrightarrow} V$ *and* $W \overset{*}{\Rightarrow} W'$.
  - *Since* $\cdot; \cdot \vdash W \overset{SC}{\hookrightarrow} W$ *(By* ev_SC_empty*), we know that* $V = W$
  - *Since* $V \overset{*}{\Rightarrow} W'$ *and* V *does not contain any* Λ *terms, we know that* $V = W'$ *(By Definition 1)*
  - *Therefore,* $\Gamma; \cdot \vdash F \hookrightarrow V$

□