令和２年度 学士論文

代数的エフェクトを持つ計算体系 System F^{ϵ} へのエフェクト強制の導入と健全性の証明

東京工業大学 情報理工学院数理・計算科学系
学籍番号 17B01160
池守 和槻

指導教員
増原 英彦 教授

令和３年２月５日

概要

代数的エフェクトとは，例外や状態などの計算エフェクトを表すための機構である．代数的エフェクトを特徴に持つプログラミング言語として Koka 言語がある．Koka 言語の中間言語は高階の多相型と代数的エフェクト，エフェクト強制を備える言語である．この中でエフェクト強制を除いた計算体系は形式化されているが，エフェクト強制を含めた計算体系は形式化されていない．本研究では，Koka 言語の中間言語に即してエフェクト強制を導入した計算体系 System F^{ϵ}+open を定義し，健全性を証明する．この計算体系に対して健全性を示すことにより，Koka 言語を用いてエフェクト安全なプログラムを記述できることが保証される．

謝辞

まず本研究を進めるにあたり多くのアドバイスやご指導をしていただいた増原英彦教授，叢悠悠助教，ならびに研究室の皆様に心より感謝いたします．そして，本研究に関して議論する機会を多く設けていただき，様々なアドバイスをしていただいた Microsoft Research Redmond の Daan Leijen 氏に心より感謝いたします．

目次

図目次

第 1 章

はじめに

代数的エフェクト [1] とは，例外や状態などの計算エフェクトを表すための機構である．代数的エフェクトは例外処理の一般化と捉えることができ，計算エフェクトを発生させるオペレーションと計算エフェクトを処理するハンドラからなる．代数的エフェクトでは，計算エフェクトを統一的に扱うことができ，プログラマは独自の計算エフェクトを定義することができる．また，ハンドラを変更することでオペレーションの意味を変更でき，プログラムの再利用性を高めることができる．

Koka 言語 [2, 3] は代数的エフェクトを組み込みの機能として提供している静的型付け言語である．Koka 言語 はプログラムの実行中に発生する計算エフェクトがハンドラによって適切に処理されているかを型システムによって静的に検査することができ，エフェクト安全なプログラムを記述することができる．さらに，優れた型推論を併せ持つため多くの型注釈を省略することができる．

Koka 言語 のコンパイラではプログラマが記述したプログラムは明示的に型付けされた中間言語への変換を経て，代数的エフェクトを組み込みの言語機能として持たないプログラミング言語へコンパイルされる．代数的エフェクトを組み込みの機能として提供する Koka 言語 から代数的エフェクトの機能を持たない言語へ変換されることは大きな特徴である [3]．また，Koka 言語では発生する計算エフェクトに対応したハンドラの配列を関数に引数として渡すことで，最も内側の適切なハンドラの探索を効率的にしている点も大きな特徴である [3]．

Koka 言語の中間言語の健全性はエフェクト強制を含めた計算体系では示されていない．この中間言語は高階の多相型と代数的エフェクト，エフェクト強制を備える言語である．この中からエフェクト強制を除いた計算体系が System F^{ϵ} [3] として形式化されている．System F^{ϵ} の関数適用では，関数の本体が発生させるエフェクトと引数が発生させるエフェクトが等しくなければならない．したがって，例外のみを発生させる関数を例外と入出力を発生させる項に適用することができない．Koka 言語の中間言語では，この制約をエフェクト強制を導入することで解決している．エフェクト強制によって，関数を関数の本体が実際に発生させるエフェクトよりも多くのエフェクトを発生させる関数として型付けすることができる．例えば，関数の本体が発生させるエフェクトを例外のみから例外と入出力に強制することができる．よって，例外のみを発生させる関数に対して，例外と入出力を発生させる項を適用することができる．

本研究の貢献は次の 2 つである．

- System F^{ϵ} にエフェクト強制を導入した計算体系 System F^{ϵ}+open を定義する．ただし，エフェクト強制の導入は Koka 言語の中間言語に即した形で定義する．

- System F^{ϵ}+open の健全性を証明する．証明は [3] の証明に従って示す．

本論文の構成は以下の通りである．まず，第 2 章では代数的エフェクトの概略を説明し，先行研究の計算体系とエフェクト強制について説明する．次に，第 3 章では System F^{ϵ} にエフェクト強制を導入した System F^{ϵ}+open の構文と型システム，操作的意味論を定義し，System F^{ϵ}+open の健全性について説明する．第 4 章で関連研究を紹介し，第 5 章でまとめと今後の課題について述べる．

第 2 章

背景

2.1 代数的エフェクト

代数的エフェクト [1] とは，例外や状態などの計算エフェクトを表すための機構である．また，代数的エフェクトは計算エフェクト発生させるオペレーションとエフェクトを処理するハンドラからなる．代数的エフェクトではエフェクトを処理するハンドラを変更することによってオペレーションの意味を変えるこができる．オペレーション呼び出しによりエフェクトが発生すると，プログラムの実行はオペレーションの処理が定義されている最も内側のハンドラに移る．また，ハンドラはオペレーション呼び出しを捉えるとオペレーション呼び出しをした場所からハンドラまでの継続を得る．継続とは，ある地点からのその後の計算のことである．この場合の継続はオペレーション呼び出しからオペレーションの処理が定義されている最も内側のハンドラまでの計算のことである．

Koka 言語 [2] を用いた代数的エフェクトによる例外の実装を示す．ここでは例外を表すエフェクトを `exc` という名前で宣言し，その例外を表すオペレーションを `raise` として定義する．

```
1 effect exc {
2     control raise(s : string) : int
3 }
```

例外を表すエフェクト `exc` を用いた計算として，`safe-div` 関数を考える．`safe-div` 関数は引数として 2 つの `int` 型の値 `x`, `y` を受け取り，`y` が `0` の時は `raise` オペレーションを呼び出して例外を発生させ，`y` が `0` でない時は `x` の `y` による商を返り値として出力する．ここで発生させる例外は 0 除算による実行時エラーを起こすのではなく，プログラマーが独自に定義したエフェクトが発生することに注意されたい．

```
1 fun safe-div(x, y) {
2     if (y == 0) then raise("division by zero")
3     else x / y
4 }
```

先述の safe-div 関数を用いて例外が発生するプログラムを以下に示す．main 関数では，まず safe-div(10, 0) が実行され，y が 0 なので raise オペレーションが呼び出される．次に，プログラムの実行が 2 行目のハンドラで定義される raise オペレーションに移る．したがって，main 関数は "division by zero" を標準出力に出力して終了する．

```
1 fun main() {
2     with handler { control raise(s) -> println(s) }
3     val x = safe-div(10, 0)
4     println(x)
5 }
```

次に，ハンドラによって取得される継続を用いて例外が発生したあとも処理を続行するプログラムを以下に示す．先述の例と同様に，main 関数ではまず safe-div(10, 0) が実行され，y が 0 なので raise オペレーションが呼び出される．次に，プログラムの実行が 2 行目のハンドラで定義される raise オペレーションに移る．2 行目の resume は raise オペレーション呼び出しから 4 行目の println(x) までのその後の計算を表す継続である．main 関数の 2 行目で resume に引数として 42 が渡されているので，safe-div(10, 0) の返り値は 42 となる．そして，main 関数では x が 42 に束縛され，println(x) で 42 を標準出力に出力して終了する．

```
1 fun main() {
2     with handler { control raise(s) -> resume(42) }
3     val x = safe-div(10, 0)
4     println(x)
5 }
```

2.2　System F^{ϵ}

System F^{ϵ} [3] は明示的に型付けされた計算体系である．この計算体系は代数的エフェクトを持つ計算体系であり，言語要素として System Fω [4] のような高階の多相型とエフェクトを表すためのエフェクト列を持つ．

例外 exc を発生させる System F^{ϵ} の関数を以下に示す．この関数は int 型の引数 x を受け取り，perform *raise* "raise exception" で raise オペレーションを呼び出し，例外 exc を発生させる．perform はオペレーション呼び出しをするための構文であり，"raise exception" は raise オペレーションの引数である．そして，関数は 1 を返り値として出力する．この関数の型は int → ⟨exc⟩ int となる．⟨exc⟩ は関数に int 型の引数が渡され，関数の本体を評価した際に発生するエフェクトをエフェクト列を用いて表している．

$$\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1 \tag{2.1}$$

次に，上記の関数に 2 を渡した System F^{ϵ} の項を以下に示す．この項の型は `int` 型であり，項が発生させるエフェクトは ⟨exc⟩ となる．

$$(\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1)(2) \tag{2.2}$$

2.3　System F^{ev}

System F^{ϵ} の項は，エビデンス変換 [3] という変換によって System F^{ev} [3] の項に変換される．System F^{ev} は明示的に型付けされた計算体系であり，代数的エフェクトを持つ計算体系を System Fω のような高階の多相型と計算エフェクトを表すエフェクト列で拡張した点が System F^{ϵ} と類似している．一方で，代数的エフェクトのオペレーションの実装が定義されているハンドラを関数の引数として渡す点が System F^{ϵ} と異なる．

(2.1) の System F^{ϵ} の関数をエビデンス変換によって変換した System F^{ev} の関数を以下に示す．関数の引数の $z : \mathtt{evv}\ \langle \mathtt{exc} \rangle$ はエフェクト ⟨exc⟩ の `raise` オペレーションの実装が定義されているハンドラの配列であり，エビデンス配列という．関数に引数として渡されるエビデンス配列は関数の本体が発生させるエフェクトと対応した配列である．

$$\lambda^{\langle \mathtt{exc} \rangle} z : \mathtt{evv}\ \langle \mathtt{exc} \rangle, x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ z\ \mathtt{"raise\ exception"}; 1$$

同様に (2.2) の System F^{ϵ} の項を変換した System F^{ev} の項を以下に示す．System F^{ev} の関数適用では関数の本体が発生させるエフェクトに対応したエビデンス配列も引数として関数に適用する．以下の例では，`ev` がエフェクト ⟨exc⟩ に対応したエビデンス配列であり，`raise` オペレーションの実装が定義されているハンドラが含まれている．

$$(\lambda^{\langle \mathtt{exc} \rangle} z : \mathtt{evv}\ \langle \mathtt{exc} \rangle, x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ z\ \mathtt{"raise\ exception"}; 1)(\mathtt{ev})(2)$$

2.4　エフェクト強制

エフェクト強制とは，あるエフェクトから他のエフェクトへの変換である．Koka 言語のプログラムは System F^{ϵ} を明示的にエフェクト強制を行えるように拡張した計算体系に基づく中間言語にコンパイルされる．`open` はエフェクト強制を行うための構文であり，関数型につくエフェクトを拡張することができる．

System F^{ϵ} における関数 `inc` と `open` によってエフェクトを拡張された関数 `inc-exc` を以下に示す．

$$\mathtt{inc} = \lambda^{\langle\rangle} x : \mathtt{int}.\ x + 1$$
$$\mathtt{inc\text{-}exc} = \mathtt{open}[\langle\rangle, \langle \mathtt{exc} \rangle](\mathtt{inc})$$

関数 `inc` の型は $\mathtt{int} \to \langle\rangle\ \mathtt{int}$ である．⟨⟩ は関数 `inc` に `int` 型の引数が渡されて，関数の本体を評価した際に発生するエフェクトはないことを表している．次に，関数 `inc-exc` の型は $\mathtt{int} \to \langle \mathtt{exc} \rangle\ \mathtt{int}$ であり，エフェクトが ⟨⟩ から ⟨exc⟩ に拡張されている．

System F^{ϵ} の関数適用の型規則を以下に示す．

$$\frac{\Gamma \vdash e : \sigma' \to \epsilon\ \sigma \mid \epsilon \qquad \Gamma \vdash e' : \sigma' \mid \epsilon}{\Gamma \vdash e\ e' : \sigma \mid \epsilon}\ [\mathrm{App}]$$

$\Gamma \vdash e : \sigma' \to \epsilon\ \sigma \mid \epsilon$ は項 e の型が $\sigma' \to \epsilon\ \sigma$ であり，縦棒の右側の ϵ は e が発生するエフェクトが ϵ であることを表している．同様に，$\Gamma \vdash e' : \sigma' \mid \epsilon$ は e' の型が σ' であり，e’ が発生するエフェクトが ϵ であることを表している．特に，System F^{ϵ} での関数適用の型規則は関数が発生するエフェクトと関数の本体が発生するエフェクト，引数が発生するエフェクトの三つが等しくなければならない．

次に，System F^{ϵ} で型付け不可能な項の例を示す．

$$(\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1)(\mathtt{inc(1)})$$

関数適用に着目すると，$\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1$ の型は $\mathtt{int} \to \langle \mathtt{exc} \rangle\ \mathtt{int}$ であり, 発生するエフェクトは exc である．また，引数の inc(1) の型は int 型であり，関数の本体が発生させるエフェクトは $\langle\rangle$ である．したがって，関数のエフェクト $\langle \mathtt{exc} \rangle$ と引数のエフェクト $\langle\rangle$ が異なるので型付け不可能である．

上記の項を open によって型付け可能にした項を次に示す．

$$(\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1)(\mathtt{open}[\langle\rangle, \langle \mathtt{exc} \rangle](\mathtt{inc})(1)$$

関数適用に着目すると，$\lambda^{\langle \mathtt{exc} \rangle} x : \mathtt{int}.\ \mathtt{perform}\ \mathit{raise}\ \mathtt{"raise\ exception"}; 1$ の型は $\mathtt{int} \to \langle \mathtt{exc} \rangle\ \mathtt{int}$ であり，関数の本体が発生させるエフェクトは exc である．また，引数の $\mathtt{open}[\langle\rangle, \langle \mathtt{exc} \rangle](\mathtt{inc})$ の型が $\mathtt{int} \to \langle \mathtt{exc} \rangle\ \mathtt{int}$ であるので，$\mathtt{open}[\langle\rangle, \langle \mathtt{exc} \rangle](\mathtt{inc})(1)$ の型は int 型であり，発生するエフェクトが $\langle \mathtt{exc} \rangle$ となる．したがって，関数のエフェクト $\langle \mathtt{exc} \rangle$ と引数のエフェクト $\langle \mathtt{exc} \rangle$ のエフェクトが一致するので型付け可能な項となる．

第 3 章

System F^{ϵ}+open

本章では，System F^{ϵ} をエフェクト強制で拡張した System F^{ϵ}+open を提案する．まず，3.1 節では，エフェクトの集まりを表すエフェクト列の説明をする．次に，3.2 節では，エフェクトを発生させるためのオペレーションの説明をする．そして，3.3 節と 3.4 節で構文と型システムについて説明し，3.5 節で操作的意味論について説明する．最後に，3.6 節で健全性について説明する．

3.1 エフェクト列

エフェクトラベルとエフェクト列 [3] はエフェクトの集まりを表すために用いられ，プログラマが自由に定義することができる．例えば，`exc` は例外，`state` は状態を表すエフェクトラベルとして定義できる．また，$\langle \texttt{exc} \rangle$ は例外を表すエフェクト列であり，$\langle \texttt{exc}, \texttt{state} \rangle$ は例外と状態を表すエフェクト列である．

エフェクト列は [3] に従って，図 3.1 のように型コンストラクタと構文が定義される．$\langle\rangle$ は空のエフェクト列を表す型コンストラクタである．また，$\langle \_ \mid \_ \rangle$ はエフェクトラベル l とエフェクト列 ϵ を受け取って拡張されたエフェクト列 $\langle l \mid \epsilon \rangle$ を作る型コンストラクタである．エフェクト列の特徴はエフェクトラベルの重複が許される点である．したがって，$\langle \texttt{exc} \rangle$ と $\langle \texttt{exc}, \texttt{exc} \rangle$ は異なるエフェクト列である．

(空エフェクト)	$\langle\rangle$	:	`eff`
(エフェクト列の拡張)	$\langle \_ \mid \_ \rangle$	:	$\texttt{lab} \to \texttt{eff} \to \texttt{eff}$

(構文)

$\epsilon ::= \sigma^{\texttt{eff}}$	(エフェクト列)
$\mu ::= \alpha^{\texttt{eff}}$	(エフェクト列の型変数)
$l ::= c^{\texttt{lab}}$	(エフェクトラベル)
$\langle l_1, \ldots, l_n \rangle \doteq \langle l_1 \mid \ldots \mid \langle l_n \mid \langle\rangle \rangle \ldots \rangle$	(閉じたエフェクト列)
$\langle l_1, \ldots, l_n \mid \epsilon \rangle \doteq \langle l_1 \mid \ldots \mid \langle l_n \mid \epsilon \rangle \ldots \rangle$	(拡張されたエフェクト列)

図 3.1　エフェクト列の定義

エフェクト列に対する同値関係についても [3] にしたがって，図 3.2 のように定義される．[refl] 規則，[trans] 規則は通常の同値関係における反射律と推移律である．[eq-head] 規則は等しいエフェクト列 ϵ_1 と ϵ_2 をそれぞれ同じエフェクトラベル l で拡張したエフェクト列 $\langle l \mid \epsilon_1 \rangle$ と $\langle l \mid \epsilon_2 \rangle$ も等しいとする規則である．また，[eq-trans] 規則はエフェクトラベルの交換に関する規則である．エフェクトラベルの交換は異なるエフェクトラベル同士に限定されている．

$$\frac{}{\epsilon \equiv \epsilon}\ [\text{refl}] \qquad \frac{\epsilon_1 \equiv \epsilon_2 \qquad \epsilon_2 \equiv \epsilon_3}{\epsilon_1 \equiv \epsilon_3}\ [\text{eq-trans}]$$

$$\frac{l_1 \neq l_2 \qquad \epsilon_1 \equiv \epsilon_2}{\langle l_1, l_2 \mid \epsilon_1 \rangle \equiv \langle l_2, l_1 \mid \epsilon_2 \rangle}\ [\text{eq-swap}] \qquad \frac{\epsilon_1 \equiv \epsilon_2}{\langle l \mid \epsilon_1 \rangle \equiv \langle l \mid \epsilon_2 \rangle}\ [\text{eq-head}]$$

図 3.2　エフェクト列に対する同値関係の定義

3.2　オペレーション

オペレーションはエフェクトを発生させるために用いられ，エフェクトラベルと合わせてプログラマが自由に定義することができる．例えば，エフェクトラベル `exc` のオペレーションとして `raise` を定義することができ，`raise` を用いると `exc` エフェクトを発生させることができる．

オペレーションとエフェクトシグネチャ，エフェクトシグネチャの集合は [3] に従って図 3.3 のように定義される．オペレーションは名前と型によって定義される．このような名前と型の組の集合をエフェクトシグネチャと呼び，sig と表記する．また，エフェクトラベル l_i と l_i に対応するエフェクトシグネチャ sig_i の組の集合をエフェクトシグネチャの集合と呼び，Σ と表記する．ただし，各オペレーションと各エフェクトラベルはそれぞれ相異なる名前であり，Σ はトップレベルで定義されている．例えば，$\Sigma_1 = \{\texttt{exc} : \{\texttt{raise} : \texttt{string} \to \texttt{int}\}, \texttt{state} : \{\texttt{put} : \texttt{int} \to (), \texttt{get} : () \to \texttt{int}\}\}$ をエフェクトシグネチャの集合のひとつとして定義することができる．`exc` はエフェクトラベルであり，オペレーションとして `raise` が定義されている．同様に，`state` はエフェクトラベルであり，オペレーションとして `put` と `get` が定義されている．

最後にオペレーションとエフェクトシグネチャ，エフェクトシグネチャの集合に対する記法について説明する．まず，エフェクトラベル l がエフェクトシグネチャ Σ に属している場合，$l \in \Sigma$ と表記する．次に，エフェクトラベル l_i に対応するエフェクトシグネチャ sig_i を $\Sigma(l_i)$ と表記する．さらに，オペレーション op がエフェクトラベル l に対応するエフェクトシグネチャに属している場合，$op \in \Sigma(l)$ と表記する．例えば，上記の Σ_1 を用いると $\texttt{exc} \in \Sigma_1$ や $\Sigma_1(\texttt{exc}) = \{\texttt{raise} : \texttt{string} \to \texttt{int}\}$，$\texttt{raise} \in \Sigma_1(\texttt{exc})$ と表記できる．

(エフェクトシグネチャ)	$sig ::= \{op_1 : \forall\overline{\alpha}.\sigma_1 \to \sigma_1', \cdots, op_1 : \forall\overline{\alpha}.\sigma_n \to \sigma_n'\}$
(エフェクトシグネチャの集合)	$\Sigma \quad ::= \{l_1 : sig_1, \ldots, l_n : sig_n\}$

図 3.3　エフェクトシグネチャとエフェクトシグネチャの集合の定義

3.3　構文

本節では，まず 型とカインドについて説明する．次に，System F$^{\epsilon}$ の項について説明する．

3.3.1　型とカインドの構文

System F$^{\epsilon}$+open の型とカインドは 図 3.4 で定義される．

(型)	σ	::=	$\alpha^k \mid \sigma \to \epsilon\ \sigma \mid \forall \alpha^k.\sigma \mid c^k \sigma, \ldots, \sigma$
(カインド)	k	::=	$* \mid k \to k \mid \texttt{eff} \mid \texttt{lab}$

図 3.4　System F$^{\epsilon}$+open の型

まず，型について説明する．α^k は型変数，$\sigma \to \epsilon\ \sigma$ は関数型, $\forall \alpha^k.\sigma$ は全称型である．この三つは Sytem Fω と概ね同様であるが，関数型 $\sigma \to \epsilon\ \sigma$ に関数の本体が発生させるエフェクトを表すエフェクト列 ϵ が付く点が異なる．$c^k \sigma, \ldots, \sigma$ は組み込みの型コンストラクタ c^k に対する型適用である．k は型コンストラクタのカインドを表している．

次に，カインドについて説明する．$*$ は真の型のカインド，$k \to k$ は高階の型のカインドである．この二つは Sytem Fω と同様である．また，`eff` はエフェクト列のカインド，`lab` はエフェクトラベルのカインドである．

3.3.2　項の構文

System F$^{\epsilon}$+open の構文は 図 3.5 で定義される．灰色で強調されている部分が本研究で新たに加えた構文である．

(値)	v	::=	$x \mid \lambda^{\epsilon} x : \sigma.e \mid \Lambda \alpha^k.v \mid \texttt{handler}^{\epsilon}\ h \mid \texttt{perform}^{\epsilon}\ op\ \overline{\sigma}$
		|	$\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v)$
(項)	e	::=	$v \mid e\ e \mid e[\sigma] \mid \texttt{handle}^{\epsilon}\ h\ e \mid \texttt{val}\ x = e;\ e$
		|	$\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e)$
		|	$\texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e)$
(ハンドラ)	h	::=	$\{op_1 \to f_1, \ldots, op_n \to f_n\}$

図 3.5　System F$^{\epsilon}$+open の構文

まず，値について説明する．x は変数, $\lambda^{\epsilon} x : \sigma.e$ はラムダ抽象, $\Lambda \alpha^k.v$ は型抽象の構文である．この三つは Sytem Fω と概ね同様であるが，型抽象が値のみに制限されている点が異なる．$\texttt{handler}^{\epsilon}\ h$ はハンドラ h を値として扱うための構文である．$\texttt{perform}^{\epsilon}\ op\ \overline{\sigma}$ はオペレーション呼び出しをす

るための構文である．この構文によって，オペレーション呼び出しを行いエフェクトを発生させる．$\texttt{open}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](v)$ は関数型の値 v の関数型につくエフェクト列に対してエフェクト強制するための構文である．この構文は関数型の値 v の本体が発生させるエフェクトが $\langle l_1,\ldots,l_n\rangle$ から $\langle l_1,\ldots,l_n \mid \epsilon_0\rangle$ に変換されることを表す．

次に，項について説明する．v は 値，$e\ e$ は関数適用, $e[\sigma]$ は型適用の構文である．この三つは通常の Sytem Fω と同様である．$\texttt{handle}^{\epsilon}\ h\ e$ は項 e をハンドラ h でハンドルするための構文である．$\texttt{val}\ x = e;\ e$ は変数束縛のための構文である．$\texttt{open}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](e)$ は値の場合と同様である．$\texttt{under}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](e)$ は項 e のエフェクトに対するエフェクト強制のための構文である．この構文は項 e が発生させるエフェクトが $\langle l_1,\ldots,l_n\rangle$ から $\langle l_1,\ldots,l_n \mid \epsilon_0\rangle$ に変換されることを表す．

最後に，ハンドラについて説明する．ハンドラ h はオペレーション op と関数 f の組 $op \to f$ の集合である．ただし，$op_1,\ldots,op_n$ は相異なる名前である．

3.4　型システム

本節では，まず 3.4.1 節で型環境について説明する．次に，3.4.2 節でカインド導出について説明する．最後に，3.4.3 節で型導出について説明する．

3.4.1　型環境

本論では，型環境 Γ を以下のように定義する．

$$(\text{型環境}) \qquad \Gamma ::= \varnothing \mid \Gamma, x:\sigma \mid \Gamma, \alpha^k$$

型環境は項変数と型の組 $x:\sigma$ と型変数とカインドの組 α^k の列である．ここで，$x:\sigma$ は項変数 x の型が σ であることを表しており，α^k は型変数 α のカインドが k であることを表している．

3.4.2　カインド導出

System F$^{\epsilon}$+open のカインド導出は以下の形式で与えられる．

$$(\text{カインド導出}) \qquad \begin{array}{c} \Gamma \vdash_{\texttt{wf}} \sigma : k \\ \uparrow \quad \uparrow\ \downarrow \end{array}$$

上向きの矢印 (↑) はカインド導出をするために事前に与えられる要素である．一方で，下向きの矢印 (↓) は導出される要素である．つまり，型 σ は型環境 Γ の下で，カインド k が導出される．

図 3.6 は Sytem F$^{\epsilon}$+open のカインド導出規則であり，[3] と同様に定義される．[KIND-VAR] 規則と [KIND-QUANT] 規則, [KIND-APP] 規則，[KIND-ARROW] 規則は System Fω におけるカインド導出規則と同様である．[KIND-CON] 規則は，組み込みの型コンストラクタに対するカインド導出である．[KIND-LABEL] 規則はエフェクトラベルに対するカインド導出規則であり，エフェクトラベル l がエフェクトシグネチャの集合 Σ に属するかを検査する．[KIND-TOTAL] 規則は空のエフェクト列に対

$$\frac{\alpha^k \in \Gamma}{\Gamma \vdash_{\mathtt{wf}} \alpha^k : k}\ [\text{KIND-VAR}] \qquad \frac{\Gamma, \alpha^k \vdash_{\mathtt{wf}} \sigma : * \qquad k \neq lab}{\Gamma \vdash_{\mathtt{wf}} \forall \alpha^k.\sigma : *}\ [\text{KIND-QUANT}]$$

$$\frac{\Gamma \vdash_{\mathtt{wf}} \sigma_1 : k_1 \to k \qquad \Gamma \vdash_{\mathtt{wf}} \sigma_2 : k_1}{\Gamma \vdash_{\mathtt{wf}} \sigma_1\ \sigma_2 : k}\ [\text{KIND-APP}]$$

$$\frac{\Gamma \vdash_{\mathtt{wf}} \sigma_1 : * \qquad \Gamma \vdash_{\mathtt{wf}} \sigma_2 : * \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{wf}} \sigma_1 \to \epsilon\ \sigma_2 : *}\ [\text{KIND-ARROW}]$$

$$\frac{}{\Gamma \vdash_{\mathtt{wf}} c^k : k}\ [\text{KIND-CON}] \qquad \frac{l \in \Sigma}{\Gamma \vdash_{\mathtt{wf}} l : \mathtt{lab}}\ [\text{KIND-LABEL}]$$

$$\frac{}{\Gamma \vdash_{\mathtt{wf}} \langle\rangle : \mathtt{eff}}\ [\text{KIND-TOTAL}] \qquad \frac{\Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff} \qquad \Gamma \vdash_{\mathtt{wf}} l : \mathtt{lab}}{\Gamma \vdash_{\mathtt{wf}} \langle l \mid \epsilon \rangle : \mathtt{eff}}\ [\text{KIND-ROW}]$$

図 3.6　System F$^{\epsilon}$+open のカインド導出規則

するカインド導出規則であり，カインド `eff` が導出される．最後に [KIND-ROW] 規則は拡張したエフェクト列 $\langle l \mid \epsilon \rangle$ に対するカインド導出規則であり，l のカインドが `lab`, ϵ のカインドが `eff` であることを検査する．

[KIND-LABEL] 規則と [KIND-TOTAL] 規則，[KIND-ROW] 規則は [KIND-APP] 規則と [KIND-ARROW] 規則，[KIND-CON] 規則を用いて導出することができるので必ずしも必要ではない．しかし，本論では [3] と同様にカインド導出規則として上記の三つの規則を定める．

3.4.3　型導出

System F$^{\epsilon}$+open の型導出は以下の形式で与えられる．上向きの矢印 (↑) と下向きの矢印 (↓) の役割はカインド導出と同様である．

$$\text{(型導出)} \qquad \underset{\uparrow \quad \uparrow\ \downarrow}{\Gamma \vdash_{\mathtt{val}} v : \sigma} \qquad \underset{\uparrow\ \uparrow\ \downarrow\ \uparrow}{\Gamma \vdash e : \sigma \mid \epsilon} \qquad \underset{\uparrow \quad \uparrow\ \downarrow\ \downarrow\ \uparrow}{\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon}$$

まず，$\Gamma \vdash_{\mathtt{val}} v : \sigma$ は値の型導出の形式である．値 v は型環境 Γ の下で，型 σ が導出される．次に，$\Gamma \vdash e : \sigma \mid \epsilon$ はエフェクトを発生させる項の型導出の形式である．項 e は型環境 Γ とエフェクト列 ϵ の下で，型 σ が導出される．最後に，$\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$ はハンドラの型導出の形式である．ハンドラ h は型環境 Γ とエフェクト列 ϵ の下で，型 σ とハンドラ h に対応するエフェクトラベル l が導出される．

図 3.4.3 は Sytem F$^{\epsilon}$+open の型導出規則である．灰色で強調された規則が本研究で新たに追加した型規則である．各型規則について結論の形式に分けて順に説明する．

まず，規則の結論が $\Gamma \vdash_{\mathtt{val}} v : \sigma$ となる型規則について説明する．[Var] 規則と [Abs] 規則, [TAbs] 規則は，それぞれ項変数とラムダ抽象，型適用に対する型規則である．この三つの規則は System Fω における規則と概ね同様である．ただし，[Abs] 規則で構文に表れる ϵ は関数の本体が発生させるエフェクトを表すエフェクト注釈である．[Perform] 規則はオペレーション呼び出しに対する型規則である．[Perform] 規則では，オペレーション op がエフェクトシグネチャ $\Sigma(l)$ に属していることを検査する．また，導出された関数型の入力と出力の型はオペレーション op の入力と出力の型に $\overline{\sigma}$ を型代入した型である．さらに，関数型に付くエフェクトは構文に表れるエフェクト注釈 ϵ とオペレーション op が属するエフェクトラベル l で拡張されたエフェクト列 $\langle l \mid \epsilon \rangle$ である．[Handler] 規則はハンドラに対する型

$$\frac{x : \sigma \in \Gamma \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{val}} x : \sigma}\ [\mathrm{Var}] \qquad \frac{\Gamma \vdash_{\mathtt{val}} v : \sigma \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash v : \sigma \mid \epsilon}\ [\mathrm{Val}]$$

$$\frac{\Gamma \vdash e_1 : \sigma_1 \to \epsilon\ \sigma_2 \mid \epsilon \qquad \Gamma \vdash e_2 : \sigma_1 \mid \epsilon}{\Gamma \vdash e_1\ e_2 : \sigma \mid \epsilon}\ [\mathrm{App}] \qquad \frac{\Gamma, x : \sigma_1 \vdash e : \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \sigma_1 : *}{\Gamma \vdash_{\mathtt{val}} \lambda^{\epsilon} x : \sigma_1 . e : \sigma_1 \to \epsilon\ \sigma_2}\ [\mathrm{Abs}]$$

$$\frac{\Gamma \vdash e : \forall \alpha^k . \sigma_1 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \sigma : k}{\Gamma \vdash e[\sigma] : \sigma_1[\alpha := \sigma] \mid \epsilon}\ [\mathrm{TApp}] \qquad \frac{\Gamma, \alpha^k \vdash_{\mathtt{val}} v : \sigma \qquad k \neq lab}{\Gamma \vdash_{\mathtt{val}} \Lambda \alpha^k . v : \forall \alpha^k . \sigma}\ [\mathrm{TAbs}]$$

$$\frac{op : \forall \overline{\alpha} . \sigma_1 \to \sigma_2\ \in\ \Sigma(l) \qquad \overline{\alpha} \notin ftv(\Gamma) \qquad \Gamma \vdash_{\mathtt{wf}} \overline{\sigma} : \overline{k} \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{val}} \mathrm{perform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_1[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_2[\overline{\alpha} := \overline{\sigma}]}\ [\mathrm{Perform}]$$

$$\frac{op_i : \forall \overline{\alpha} . \sigma_1 \to \sigma_2 \in \Sigma(l) \qquad \overline{\alpha} \notin ftv(\epsilon, \sigma) \qquad \Gamma \vdash_{\mathtt{val}} f_i : \forall \overline{\alpha} . \sigma_1 \to \epsilon\ ((\sigma_2 \to \epsilon\ \sigma) \to \epsilon\ \sigma)}{\Gamma \vdash_{\mathtt{ops}} \{op_1 \to f_1, \ldots, op_n \to f_n\} : \sigma \mid l \mid \epsilon}\ [\mathrm{Ops}]$$

$$\frac{\Gamma \vdash h : \sigma \mid l \mid \sigma}{\Gamma \vdash_{\mathtt{val}} \mathrm{handler}^{\epsilon}\ h : (() \to \langle l \mid \epsilon \rangle\ \sigma) \to \epsilon\ \sigma}\ [\mathrm{Handler}]$$

$$\frac{\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon \qquad \Gamma \vdash e : \sigma \mid \langle l \mid \epsilon \rangle}{\Gamma \vdash \mathrm{handle}^{\epsilon}\ h\ e : \sigma \mid \epsilon}\ [\mathrm{Handle}] \qquad \frac{\Gamma \vdash e_1 : \sigma_1 \mid \epsilon \qquad \Gamma, x : \sigma_1 \vdash e_2 : \sigma_2 \mid \epsilon}{\Gamma \vdash \mathtt{val}\ x =\ e_1;\ e_2 : \sigma_2 \mid \epsilon}\ [\ \mathrm{Let}\ ]$$

$$\frac{\Gamma \vdash e : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash \mathtt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon}\ [\ \mathrm{Open}\ ]$$

$$\frac{\Gamma \vdash e : \sigma \mid \langle l_1, ..., l_n \rangle \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash \mathtt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) : \sigma \mid \langle l_1, ..., l_n \mid \epsilon_0 \rangle}\ [\ \mathrm{Under}\ ]$$

図 3.7　System F$^{\epsilon}$+open の型規則

規則である．規則の前提における $\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$ はハンドラ h に対する型導出である．

次に，規則の結論が $\Gamma \vdash e : \sigma \mid \epsilon$ となる規則について説明する．[App] 規則と [TApp] 規則はそれぞれ関数適用と型適用に対する型規則であり，System Fω における規則と概ね同様である．[Val] 規則は値をエフェクトを発生させる項として型付けするための型規則である．この規則によって，値は任意のエフェクトを発生させる項とすることができる．[Handle] 規則は，項 e をハンドラ h でハンドルする項 $\mathtt{handle}^{\epsilon}\ h\ e$ に対する型規則である．構文に表れる注釈 ϵ は項 e をハンドラ h でハンドルした後のエフェクト列を表す．[Let] 規則は変数束縛に対する型規則である．[Open] 規則は，関数型の項 e に対して，関数型に付くエフェクト列を拡張する型規則である．同様に，[Under] 規則は項 e に付くエフェクト列を拡張する型規則である．ただし，どちらの規則においても拡張前のエフェクト列は閉じたエフェクト列でなければならない．

最後に, 規則の結論が $\Gamma \vdash_{\mathtt{ops}} h \mid l \mid \epsilon$ となる規則について説明する．当てはまる規則は [Ops] 規則だけであり，この型規則はハンドラに対する型規則である．各オペレーション op_i と対になっている f_i が値であることを検査する．

3.5　操作的意味論

System F$^{\epsilon}$+open の意味論は値呼び評価戦略である．System F$^{\epsilon}$+open の評価文脈は [3] にしたがって，図 3.8 のように定義する．また，入れ子になった評価文脈に対する記法も同様に定義する．例えば，

$\texttt{handle}^{\epsilon}\ h\ E[x]$ は $\texttt{handle}^{\epsilon}\ h \cdot E \cdot x$ と表記される．灰色で強調された部分が本研究で新たに加えた部分である．

$bop(E)$ は評価文脈 E 中の全てのオペレーションの集合を表す．$bop(E)$ を [3] にしたがって，図 3.9 のように帰納的に定義する．例えば，$E = \texttt{handle}^{\epsilon}\ \{\texttt{get} \to f_1, \texttt{put} \to f_2\}\ \square$ とすると，$bop(E) = \{\texttt{get}, \texttt{put}\}$ である．

(純評価文脈)

$$
\begin{array}{lll}
F & ::= & \square \mid F\ e \mid v\ F \mid F[\sigma] \mid \texttt{val}\ x = F;\ e \\
 & \mid & \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](F) \\
 & \mid & \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](F)
\end{array}
$$

(評価文脈)

$$
\begin{array}{lll}
E & ::= & \square \mid E\ e \mid v\ E \mid E[\sigma] \mid \texttt{handle}^{\epsilon}\ h\ E \mid \texttt{val}\ x = E;\ e \\
 & \mid & \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E) \\
 & \mid & \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E)
\end{array}
$$

(入れ子になった評価文脈に対する記法)

$$
\begin{array}{ll}
E \cdot e \doteq E[e] & v\ \square \cdot E \doteq v \cdot E \doteq v\ E \\
\square\ e \cdot E \doteq E\ e & \square\,[\sigma] \cdot E \doteq E[\sigma] \\
\multicolumn{2}{l}{\texttt{handle}^{\epsilon}\ h\ \square \cdot E \doteq \texttt{handle}^{\epsilon}\ h \cdot E \doteq \texttt{handle}^{\epsilon}\ h\ E} \\
\multicolumn{2}{l}{\texttt{val}\ x = \square; e \cdot E \doteq \texttt{val}\ x = E; e} \\
\multicolumn{2}{l}{\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle] \cdot E \doteq \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E)} \\
\multicolumn{2}{l}{\texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle] \cdot E \doteq \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E)}
\end{array}
$$

図 3.8　評価文脈と入れ子になった評価文脈に対する記法

$$
\begin{aligned}
&bop(\square) && = \varnothing \\
&bop(E\ e) && = bop(E) \\
&bop(v\ E) && = bop(E) \\
&bop(E[\sigma]) && = bop(E) \\
&bop(\texttt{handle}^{\epsilon}\ h\ E) && = bop(E) \cup \{op \mid (op \to f) \in h\} \\
&bop(\texttt{val}\ x = E;\ e) && = bop(E) \\
&bop(\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E)) && = bop(E) \\
&bop(\texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E)) && = bop(E)
\end{aligned}
$$

図 3.9　*bop* の定義

(簡約規則)

$$
\begin{array}{lll}
(app) & (\lambda^{\epsilon} x : \sigma.e)\ v & \to\ e[x := v] \\
(tapp) & (\Lambda\alpha^{k}.v)[\sigma] & \to\ v[\alpha^{k} := \sigma] \\
(handler) & (\texttt{handler}^{\epsilon}\ h)\ v & \to\ \texttt{handle}^{\epsilon} h \cdot v\ () \\
(return) & \texttt{handle}^{\epsilon}\ h \cdot v & \to\ v \\
(perform) & \texttt{handle}^{\epsilon}\ h \cdot E \cdot \texttt{perform}\ op\ \overline{\sigma}\ v & \to\ f[\overline{\sigma}]\ v\ k \\
 & \text{iff } op \notin bop(E) \wedge (op \to f) \in h & \\
 & \text{where } op : \forall\overline{\alpha}.\sigma_1 \to \sigma_2 \in \Sigma(l) & \\
 & k = \lambda^{\epsilon} x : \sigma_2[\overline{\alpha} := \overline{\sigma}].\texttt{handle}^{\epsilon}\ h \cdot E \cdot x & \\
(let) & \texttt{val}\ x = v;\ e & \to\ e[x := v] \\
(open) & o\texttt{pen}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v)\ v' & \to\ \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v\ v') \\
(under) & \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v) & \to\ v
\end{array}
$$

(評価規則)

$$
\frac{e \to e'}{E[e] \mapsto E[e']}\ [\text{STEP}]
$$

図 3.10　評価文脈と評価規則

簡約規則と評価規則についても [3] にしたがって，図 3.10 のように定義する．灰色で強調されている簡約規則が本研究で新たに加えた規則である．簡約規則と評価規則を順に説明する．

まず，簡約規則について説明する．(app) と $(tapp)$ は System Fω と同様である．$(handler)$ はハンドラ $\texttt{handler}^{\epsilon}\ h$ で関数 v にユニット型の値 $()$ を適用した項 $v\ ()$ をハンドルする簡約規則である．$(return)$ はハンドラでハンドルされる項が値になった場合に，値に簡約する規則である．$(perform)$ はオペレーション呼び出し $\texttt{perform}^{\epsilon}\ op\ \overline{\sigma}$ を $(op \to f) \in h$ となる最も内側のハンドラ h でハンドルする簡約規則である．条件 $op \notin bop(E) \wedge (op \to f) \in h$ によって，h が op を含む最も内側のハンドラであることを保証している．また，関数 k はオペレーション呼び出しからハンドラ h までの継続を表している．(let) は一般的な変数束縛の簡約規則である．$(open)$ は関数型のエフェクトに対してエフェクト強制

された値 $\texttt{open}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](v)$ に値 v' を適用する簡約規則である．$(under)$ はエフェクト強制をされた項 $\texttt{under}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](v)$ を値 v に簡約する規則である．

次に，評価規則 [STEP] について説明する．通常の評価文脈を用いた評価規則と同様である．つまり，項 e が e' に簡約されるならば $E[e]$ の 1 ステップ評価は $E[e']$ である．

3.6　System F$^{\epsilon}$+open の健全性

System F$^{\epsilon}$+open の健全性について説明する．System F$^{\epsilon}$+open における健全性とは，進行定理と保存定理の二つに加えて，空のエフェクト列の下で正しく型付けされた項に含まれるオペレーション呼び出しが適切にハンドルされるという性質である．健全性は [3] と同様に以下の 補題 4 と 補題 5, 定理 2, 定理 3 を用いて示した．補題と定理の番号は [3] と対応させている．証明の詳細は付録で示す．

補題 4 は空のエフェクト列の下で型付けされた項の全てのオペレーション呼び出しはハンドルされることを主張している．

補題 4 (エフェクト列が空ならばオペレーションはハンドルされる)**.** $\varnothing \vdash E[\texttt{perform}^{\epsilon_0}\ \texttt{op}\ \overline{\tau}\ v] : \sigma \mid \langle\rangle$ ならば，$E = E_1 \cdot \texttt{handle}^{\epsilon}\ h\ \cdot E_2$ かつ，$op \notin bop(E_2)$ かつ，$op \to f \in h$.

さらに，補題 5 はハンドルされずに発生するエフェクトは全てエフェクト列に含まれるを主張している．

補題 5 (ハンドルされないエフェクトはエフェクト列に含まれる)**.** $\varnothing \vdash E[\texttt{perform}^{\epsilon}\ \texttt{op}\ \overline{\tau}\ v] : \sigma \mid \epsilon$ かつ，$op \notin bop(E)$ ならば，$op \in \Sigma(l)$ かつ，$l \in \epsilon$.

最後に，定理 2 と 定理 3 は進行定理と保存定理 [5] である．ただし，保存定理は型だけでなくエフェクト列も簡約の前後で保存されることを主張している．

定理 2 (進行)**.**
$\varnothing \vdash e_1 : \sigma \mid \langle\rangle$ ならば，e_1 は値，または $e_1 \mapsto e_2$.

定理 3 (保存)**.**
$\varnothing \vdash e_1 : \sigma \mid \langle\rangle$ かつ，$e_1 \mapsto e_2$ ならば，$\varnothing \vdash e_2 : \sigma \mid \langle\rangle$.

これらの補題と定理によって，正しく型付けされた System F$^{\epsilon}$+open の項は適用可能な簡約規則がなく値でもないような，行き詰まり状態にならないことが示された．

第 4 章

関連研究

4.1 代数的エフェクトを持つプログラミング言語

■Eff 言語　Eff 言語 [6, 7] は ML に類似した構文のプログラミング言語であり，代数的エフェクトを組み込みの機能として提供している言語である．Eff 言語ではエフェクトはオペレーションとして定義され，エフェクトの集まりはオペレーションの名前の集合によって表される．また，Eff 言語は Koka 言語と同様にオペレーションの集合上のパラメータ多相によって，エフェクトの多相性を実現している．Eff 言語では実行時に最も内側の適切なハンドラを探索する．

■Koka 言語　Koka 言語は代数的エフェクトを組み込みの機能として提供するプログラミング言語である．Koka 言語では，エフェクトはエフェクトラベルとエフェクト列によって定義され，エフェクトの集まりはエフェクト列を用いて表される．また，Koka 言語ではエフェクト列上のパラメータ多相によって，エフェクトの多相性を実現している．Koka 言語はエビデンス変換によって，関数の本体が発生するエフェクトに対応したハンドラの配列を関数の引数として渡すように変換される．この変換によって，実行時にハンドラを探索するコストが削減されプログラムの実行が高速になる．

■Effekt 言語　Effekt 言語 [8] は Scala 言語に似た構文のプログラミング言語であり，代数的エフェクトを組み込みの機能として提供している．Effekt 言語は Eff 言語と同様にエフェクトはオペレーションとして定義され，エフェクトの集まりはオペレーションの名前の集合によって表される．Effekt 言語では Koka 言語や Eff 言語とは異なり，パラメータ多相を用いずに文脈的エフェクト多相性 [8] によってエフェクト多相性を実現している．Effekt 言語では，ハンドラを関数の引数として渡す点が Koka 言語と類似している．一方で，関数に要求されたハンドラのみを引数として渡す点が Koka 言語とは異なる．

4.2 型強制・エフェクト強制

Eff 言語の中間言語は ExEff [7] という計算体系に基づいている．この計算体系は代数的エフェクトを持つ計算体系を System F [4] のような多相型と部分型で拡張した計算体系である．さらに，明示的に型強制とエフェクト強制を行う．ExEff において，エフェクト強制された項の例を以下に示す．ここで表れる `Raise` は例外を起こすオペレーションの名前とする．

$$(\texttt{fun}\ (\texttt{x}\ :\ \texttt{Unit}) \mapsto (\texttt{return}\ \texttt{x}))) \rhd \langle \texttt{Unit} \rangle \to \langle \texttt{Unit} \rangle ! \varnothing_{\{\texttt{Raise}\}}$$

`fun (x : Unit) ↦ (return x)` の型は $\texttt{Unit} \to \texttt{Unit}!\varnothing$ であり，$\triangleright \langle \texttt{Unit} \rangle \to \langle \texttt{Unit} \rangle !\varnothing_{\{\texttt{Raise}\}}$ によって関数の型が $\texttt{Unit} \to \texttt{Unit}!\{\texttt{Raise}\}$ に明示的にエフェクト強制されている．このエフェクト強制は本論の $\texttt{open}[\langle\rangle, \langle \texttt{exc} \rangle]$ によるエフェクト強制に相当する．

第 5 章

まとめと今後の課題

本研究では，System F^{ϵ} を拡張した計算体系 System F^{ϵ}+open を定義し，健全性を証明した．これにより，Koka 言語の中間言語で記述された型付け可能なプログラムは行き詰まり状態にならないことが示された．

今後の課題としては，まず代数的エフェクトを持つ暗黙的に型付けされた計算体系を Koka 言語に即した形式で定義し，この計算体系の型推論と本研究で定義した System F^{ϵ}+open への変換を定義する．本研究で新たに定義したエフェクト強制によって，Koka コンパイラでのエフェクト強制に関する変換を形式的に扱うことができる．よって，Koka コンパラでの型推論とエフェクト強制を含めた中間言語への変換の正しさを形式的に保証できる．

次に，エフェクト強制 $\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle]$ を削減するためのアルゴリズムを開発する．中間言語に表れる $\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle]$ を削減することによって，エビデンス配列への操作を減らすことができ，操作に必要な時間とメモリを削減できることが期待できる．このアルゴリズムは古殿ら [9] と共に開発中である．

参考文献

[1] Matija Pretnar. An introduction to algebraic effects and handlers. invited tutorial paper. *Electronic Notes in Theoretical Computer Science*, Vol. 319, pp. 19–35, 2015. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI).

[2] Daan Leijen. Type directed compilation of row-typed algebraic effects. In *Proceedings of Principles of Programming Languages (POPL'17), Paris, France*, January 2017.

[3] Ningning Xie, Jonathan Brachthauser, Daniel Hillerstrom, Philipp Schuster, and Daan Leijen. Effect handlers, evidently. In *The 25th ACM SIGPLAN International Conference on Functional Programming (ICFP)*. ACM SIGPLAN, ACM, August 2020.

[4] Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002.

[5] A. K. Wright and M. Felleisen. A Syntactic Approach to Type Soundness. *Information and Computation*, Vol. 115, No. 1, pp. 38–94, November 1994. ZSCC: 0001328.

[6] Andrej Bauer and Matija Pretnar. An effect system for algebraic effects and handlers. *Log. Methods Comput. Sci.*, Vol. 10, No. 4, 2014.

[7] Georgios Karachalias, Matija Pretnar, Amr Hany Saleh, Stien Vanderhallen, and Tom Schrijvers. Explicit effect subtyping. 2020.

[8] Jonathan Immanuel Brachthäuser, Philipp Schuster, and Klaus Ostermann. Effects as capabilities: Effect handlers and lightweight effect polymorphism. *Proc. ACM Program. Lang.*, Vol. 4, No. OOPSLA, November 2020.

[9] 古殿直也. 代数的エフェクトを備えた関数型言語 koka に対するエフェクト割り当ての最適化. 東京工業大学学士論文, 2021.

付録 A

System F^{ϵ}+open

A.1 Effect row

Type constructors	$\langle\rangle : \mathtt{eff}$
	$\langle \_ \mid \_ \rangle : \mathtt{lab} \to \mathtt{eff} \to \mathtt{eff}$
Closed effect row	$\langle l_1, \ldots, l_n \rangle \doteq \langle l_1 \mid \cdots \mid \langle l_n, \mid \langle\rangle \cdots \rangle$
Extended effect row	$\langle l_1, \ldots, l_n \mid \mu \rangle \doteq \langle l_1 \mid \cdots \mid \langle l_n, \mid \mu \rangle \cdots \rangle$
Meta variables	$\epsilon ::= \sigma^{\mathtt{eff}} \quad \mu ::= \alpha^{\mathtt{eff}} \quad l ::= c^{\mathtt{lab}}$

図 A.1 Type constructors

$$\frac{}{\epsilon \equiv \epsilon}\ [\text{refl}] \qquad \frac{\epsilon_1 \equiv \epsilon_2 \qquad \epsilon_2 \equiv \epsilon_3}{\epsilon_1 \equiv \epsilon_3}\ [\text{eq-trans}]$$

$$\frac{l_1 \neq l_2 \qquad \epsilon_1 \equiv \epsilon_2}{\langle l_1, l_2 \mid \epsilon_1 \rangle \equiv \langle l_2, l_1 \mid \epsilon_2 \rangle}\ [\text{eq-swap}] \qquad \frac{\epsilon_1 \equiv \epsilon_2}{\langle l \mid \epsilon_1 \rangle \equiv \langle l \mid \epsilon_2 \rangle}\ [\text{eq-head}]$$

図 A.2 Equivalance of row types

A.2 Effect signatures

Effects signature	$sig ::= \{op_1 : \forall \overline{\alpha}.\sigma_1 \to \sigma_1', \cdots, op_1 : \forall \overline{\alpha}.\sigma_n \to \sigma_n'\}$
Effect signatures	$\Sigma ::= \{l_1 : sig_1, \ldots, l_n : sig_n\}$

図 A.3 Effects signatures

A.3 Syntax

$$\begin{array}{llll} \text{Types} & \sigma & ::= & \alpha^k \mid \forall \alpha^k.\sigma \mid c^k \sigma, \ldots, \sigma \mid \sigma \to \epsilon\ \sigma \\ \text{Kinds} & k & ::= & * \mid k \to k \mid \texttt{eff} \mid \texttt{lab} \end{array}$$

図 A.4　Types and Kinds

$$\begin{array}{llll} \text{Value} & v & ::= & x \mid \lambda^{\epsilon} x : \sigma.e \mid \texttt{handler}^{\epsilon}\ h \mid \Lambda \alpha^k.v \mid \texttt{perform}^{\epsilon}\ op\ \overline{\sigma} \\ & & \mid & \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v) \\ \text{Expression} & e & ::= & v \mid e\ e \mid e[\sigma] \mid \texttt{handle}^{\epsilon}\ h\ e \mid \texttt{val}\ x = e;\ e \\ & & \mid & \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) \\ & & \mid & \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) \\ \text{Handler} & h & ::= & \{op_1 \to f_1, \ldots, op_n \to f_n\} \end{array}$$

図 A.5　Value, Expression and Handler

A.4 Type rules

$$\Gamma\ ::=\ \varnothing \mid \Gamma, x : \sigma \mid \Gamma, \alpha^k$$

図 A.6　Context

$$\frac{\alpha^k \in \Gamma}{\Gamma \vdash_{\texttt{wf}} \alpha^k : k}\ [\text{KIND-VAR}] \qquad \frac{}{\Gamma \vdash_{\texttt{wf}} c^k : k}\ [\text{KIND-CON}]$$

$$\frac{}{\Gamma \vdash_{\texttt{wf}} \langle\rangle : \texttt{eff}}\ [\text{KIND-TOTAL}] \qquad \frac{\Gamma, \alpha^k \vdash_{\texttt{wf}} \sigma : * \qquad k \neq \texttt{lab}}{\Gamma \vdash_{\texttt{wf}} \forall \alpha^k.\sigma : *}\ [\text{KIND-QUANT}]$$

$$\frac{\Gamma \vdash_{\texttt{wf}} \sigma_1 : k_1 \to k \qquad \Gamma \vdash_{\texttt{wf}} \sigma_2 : k_1}{\Gamma \vdash_{\texttt{wf}} \sigma_1\ \sigma_2 : k}\ [\text{KIND-APP}]$$

$$\frac{\Gamma \vdash_{\texttt{wf}} \epsilon : \texttt{eff} \qquad \Gamma \vdash_{\texttt{wf}} l : \texttt{lab}}{\Gamma \vdash_{\texttt{wf}} \langle l \mid \epsilon \rangle : \texttt{eff}}\ [\text{KIND-ROW}] \qquad \frac{l \in \Sigma}{\Gamma \vdash_{\texttt{wf}} l : \texttt{lab}}\ [\text{KIND-LABEL}]$$

$$\frac{\Gamma \vdash_{\texttt{wf}} \sigma_1 : * \qquad \Gamma \vdash_{\texttt{wf}} \sigma_2 : * \qquad \Gamma \vdash_{\texttt{wf}} \epsilon : \texttt{eff}}{\Gamma \vdash_{\texttt{wf}} \sigma_1 \to \epsilon\ \sigma_2 : *}\ [\text{KIND-ARROW}]$$

図 A.7　Kinding rules

$$\frac{x : \sigma \in \Gamma \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{val}} x : \sigma}\ [\text{Var}] \qquad \frac{\Gamma \vdash_{\mathtt{val}} v : \sigma \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash v : \sigma \mid \epsilon}\ [\text{Val}]$$

$$\frac{\Gamma \vdash e_1 : \sigma_1 \to \epsilon\ \sigma_2 \mid \epsilon \qquad \Gamma \vdash e_2 : \sigma_1 \mid \epsilon}{\Gamma \vdash e_1\ e_2 : \sigma \mid \epsilon}\ [\text{App}] \qquad \frac{\Gamma, x : \sigma_1 \vdash e : \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \sigma_1 : *}{\Gamma \vdash_{\mathtt{val}} \lambda^{\epsilon} x : \sigma_1 . e : \sigma_1 \to \epsilon\ \sigma_2}\ [\text{Abs}]$$

$$\frac{\Gamma \vdash e : \forall \alpha^k . \sigma_1 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \sigma : k}{\Gamma \vdash e[\sigma] : \sigma_1[\alpha := \sigma] \mid \epsilon}\ [\text{TApp}] \qquad \frac{\Gamma, \alpha^k \vdash_{\mathtt{val}} v : \sigma \qquad k \neq \mathtt{lab}}{\Gamma \vdash_{\mathtt{val}} \Lambda \alpha^k . v : \forall \alpha^k . \sigma}\ [\text{TAbs}]$$

$$\frac{op : \forall \overline{\alpha} . \sigma_1 \to \sigma_2\ \in\ \Sigma(l) \qquad \overline{\alpha} \notin ftv(\Gamma) \qquad \Gamma \vdash_{\mathtt{wf}} \overline{\sigma} : \overline{k} \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{val}} \mathtt{perform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_1[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_2[\overline{\alpha} := \overline{\sigma}]}\ [\text{Perform}]$$

$$\frac{op_i : \forall \overline{\alpha} . \sigma_1 \to \sigma_2 \in \Sigma(l) \qquad \overline{\alpha} \# ftv(\epsilon, \sigma) \qquad \Gamma \vdash_{\mathtt{val}} f_i : \forall \overline{\alpha} . \sigma_1 \to \epsilon\ ((\sigma_2 \to \epsilon\ \sigma) \to \epsilon\ \sigma)}{\Gamma \vdash_{\mathtt{ops}} \{op_1 \to f_1, \ldots, op_n \to f_n\} : \sigma \mid l \mid \epsilon}\ [\text{Ops}]$$

$$\frac{\Gamma \vdash h : \sigma \mid l \mid \sigma}{\Gamma \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon}\ h : (() \to \langle l \mid \epsilon \rangle\ \sigma) \to \epsilon\ \sigma}\ [\text{Handler}]$$

$$\frac{\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon \qquad \Gamma \vdash e : \sigma \mid \langle l \mid \epsilon \rangle}{\Gamma \vdash \mathtt{handle}^{\epsilon}\ h\ e : \sigma \mid \epsilon}\ [\text{Handle}] \qquad \frac{\Gamma \vdash e_1 : \sigma_1 \mid \epsilon \qquad \Gamma, x : \sigma_1 \vdash e_2 : \sigma_2 \mid \epsilon}{\mathtt{val}\ x = e_1;\ e_2 : \sigma_2 \mid \epsilon}\ [\text{Let}]$$

$$\frac{\Gamma \vdash e : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash \mathtt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) : \sigma_1 \to \langle l_1, \ldots, l_n | \epsilon_0 \rangle\ \sigma_2 \mid \epsilon}\ [\text{Open}]$$

$$\frac{\Gamma \vdash e : \sigma \mid \langle l_1, \ldots, l_n \rangle \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash \mathtt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle}\ [\text{Under}]$$

図 A.8　Typing rules

$$\frac{\Gamma \vdash_{\mathtt{wf}} \sigma : * \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}}{\Gamma \vdash_{\mathtt{ec}} [\,] : \sigma \to \sigma \mid \epsilon}\ [\text{CEMPTY}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} e : \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to (\sigma_2 \to \epsilon\ \sigma_3) \mid \epsilon}{\Gamma \vdash_{\mathtt{ec}} E\ e : \sigma_1 \to \sigma_3 \mid \epsilon}\ [\text{CAPP1}]$$

$$\frac{\Gamma \vdash_{\mathtt{val}} v : \sigma_2 \to \epsilon\ \sigma_3 \qquad \Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to \sigma_2 \mid \epsilon}{\Gamma \vdash_{\mathtt{ec}} v\ E : \sigma_1 \to \sigma_3 \mid \epsilon}\ [\text{CAPP2}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to \forall \alpha^k . \sigma_2 \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \sigma : k}{\Gamma \vdash_{\mathtt{ec}} E[\sigma] : \sigma_1 \to \sigma_2\ [\alpha := \sigma] \mid \epsilon}\ [\text{CTAPP}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} h : \sigma \mid l \mid \epsilon \qquad \Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to \sigma \mid \langle l \mid \epsilon \rangle}{\Gamma \vdash_{\mathtt{ec}} \mathtt{handle}^{\epsilon}\ h\ E : \sigma_1 \to \sigma \mid \epsilon}\ [\text{CHANDLE}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to \sigma_2 \mid \epsilon \qquad \Gamma \vdash e : \sigma_3 \mid \epsilon}{\Gamma \vdash_{\mathtt{ec}} \mathtt{val}\ x = E;\ e : \sigma_1 \to \sigma_3 \mid \epsilon}\ [\text{CLET}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to (\sigma_2 \to \langle l_1, \ldots, l_n \rangle\ \sigma_3) \mid \epsilon \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash_{\mathtt{ec}} \mathtt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E) : \sigma_1 \to (\sigma_2 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_3) \mid \epsilon}\ [\text{COPEN}]$$

$$\frac{\Gamma \vdash_{\mathtt{ec}} E : \sigma_1 \to \sigma_2 \mid \langle l_1, \ldots, l_n \rangle \qquad \Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}}{\Gamma \vdash_{\mathtt{ec}} \mathtt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E) : \sigma_1 \to \sigma_2 \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle}\ [\text{CUNDER}]$$

図 A.9　Evaluation context typing

A.5 Evalutation

Reduction rules

$$
\begin{array}{lll}
(app) & (\lambda^{\epsilon} x : \sigma.e)\ v & \to\ e[x := v] \\
(tapp) & (\Lambda\alpha^{k}.v)[\sigma] & \to\ v[\alpha^{k} := \sigma] \\
(handler) & (\texttt{handler}^{\epsilon}\ h)\ v & \to\ \texttt{handle}^{\epsilon}\ h \cdot v\ () \\
(return) & \texttt{handle}^{\epsilon}\ h \cdot v & \to\ v \\
(perform) & \texttt{handle}^{\epsilon}\ h \cdot E \cdot \texttt{perform}^{\epsilon}\ op\ \overline{\sigma}\ v & \to\ f[\overline{\sigma}]\ v\ k \\
 & \texttt{iff}\ op \notin bop(E) \wedge (op \to f) \in h & \\
 & \texttt{where}\ op : \forall\overline{\alpha}.\sigma_1 \to \sigma_2 \in \Sigma(l) & \\
 & k = \lambda x : \sigma_2[\overline{\alpha} := \overline{\sigma}].\texttt{handle}^{\epsilon}\ h \cdot E \cdot x & \\
(let) & \texttt{val}\ x = v;\ e & \to\ e[x := v] \\
(open) & \texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v)\ v' & \to\ \texttt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v\ v') \\
(under) & \texttt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v) & \to\ v
\end{array}
$$

Evalutation rule

$$\frac{e \to e'}{E[e] \mapsto E[e']}\ [\text{STEP}]$$

図 A.10　Reduction rules and Evaluation rule

$$
\begin{array}{lll}
F & ::= & \square \mid F\ e \mid v\ F \mid F[\sigma] \mid \texttt{val}\ x = F;\ e \\
 & \mid & \texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](F) \\
 & \mid & \texttt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](F) \\
E & ::= & \square \mid E\ e \mid v\ E \mid E[\sigma] \mid \texttt{handle}^{\epsilon}\ h\ E \mid \texttt{val}\ x = E;\ e \\
 & \mid & \texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E) \\
 & \mid & \texttt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E)
\end{array}
$$

図 A.11　Evalutation context

A.6　Soundness Proof of System F$^{\epsilon}$+open

Prop 1. Type variable substitution preserves the kinding.
If $\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma : k$ and $\Gamma_1 \vdash_{\mathtt{wf}} \sigma_1 : k_1$ then $\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma[\alpha^{k_1} := \sigma_1] : k$

Proof. By induction on kinding derivation.

case　[KIND-VAR] : $\sigma = \alpha^{k_1}$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \alpha^{k_1} : k_1$	(given)
$\alpha^{k_1}[\alpha^{k_1} := \sigma_1] = \sigma_1$	(by substitution)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \alpha^{k_1}[\alpha^{k_1} := \sigma_1] : k_1$	(follows)

case　[KIND-VAR] : $\sigma = \beta^{k}$ and $\alpha^{k_1} \neq \beta^{k}$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \beta^{k} : k$	(given)
$\beta^{k}[\alpha^{k_1} := \sigma_1] = \beta^{k}$	(by substitution)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \beta^{k}[\alpha^{k_1} := \sigma_1] : k$	(follows)

case　[KIND-CON] : $\sigma = c^{k}$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} c^{k} : k$	(given)
$c^{k}[\alpha^{k_1} := \sigma_1] = c^{k}$	(by substitution)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} c^{k}[\alpha^{k_1} := \sigma_1] : k$	(KIND-CON)

case　[KIND-TOTAL] : $\sigma = \langle\rangle$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \langle\rangle : \mathtt{eff}$	(given)
$\langle\rangle[\alpha^{k_1} := \sigma_1] = \langle\rangle$	(by substitution)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \langle\rangle[\alpha^{k_1} := \sigma_1] : \mathtt{eff}$	(KIND-TOTAL)

case　[KIND-QUANT] : $\sigma = \forall\beta^{k_2}.\sigma_2$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \forall\beta^{k_2}.\sigma_2 : *$	(given)
$\Gamma_1, \alpha^{k_1}, \Gamma_2, \beta^{k_2} \vdash_{\mathtt{wf}} \sigma_2 : *3$	(KIND-QUANT)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1], \beta^{k_2} \vdash_{\mathtt{wf}} \sigma_2[\alpha^{k_1} := \sigma_1] : *$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \forall\beta^{k_2}.\sigma_2[\alpha^{k_1} := \sigma_1] : *$	(KIND-QUANT)

case　[KIND-APP] : $\sigma = \sigma_2\ \sigma_3$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_2\ \sigma_3 : k_2$	(given)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_2 : k_3 \to k_2$	(KIND-APP)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_3 : k_3$	(KIND-APP)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_2[\alpha^{k_1} := \sigma_1] : k_3 \to k_2$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_3[\alpha^{k_1} := \sigma_1] : k_3$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_2[\alpha^{k_1} := \sigma_1]\ \sigma_3[\alpha^{k_1} := \sigma_1] : k_3$	(KIND-APP)

case　[KIND-ROW] : $\sigma = \langle l \mid \epsilon \rangle$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \langle l \mid \epsilon \rangle : \mathtt{eff}$	(given)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} l : \mathtt{lab}$	(KIND-ROW)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}$	(KIND-ROW)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} l : \mathtt{lab}$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \epsilon[\alpha^{k_1} := \sigma_1] : \mathtt{eff}$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \langle l \mid \epsilon \rangle[\alpha^{k_1} := \sigma_1] : \mathtt{eff}$	(KIND-ROW)

case　[KIND-ARROW] : $\sigma = \sigma_2 \to \epsilon\ \sigma_3$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_2 \to \epsilon\ \sigma_3 : *$	(given)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_2 : *$	(KIND-ARROW)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma_3 : *$	(KIND-ARROW)
$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}$	(KIND-ARROW)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_2[\alpha^{k_1} := \sigma_1] : *$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_3[\alpha^{k_1} := \sigma_1] : *$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \epsilon[\alpha^{k_1} := \sigma_1] : \mathtt{eff}$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash_{\mathtt{wf}} \sigma_2[\alpha^{k_1} := \sigma_1] \to \epsilon[\alpha^{k_1} := \sigma_1]\ \sigma_3[\alpha^{k_1} := \sigma_1] : *$	(KIND-ARROW)

case　[KIND-LABEL] : $\sigma = l$

$\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} l : \mathtt{lab}$	(given)
$l \in \Sigma$	(KIND-LABEL)
$\Gamma_1, \Gamma_2[\alpha^{k_1} := \sigma_1] \vdash l : \mathtt{lab}$	(KIND-LABEL)

□

Prop 2. Weakening.

1. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{wf}} \sigma : k$ and $x \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, x : \sigma_0, \Gamma_2 \vdash_{\mathtt{wf}} \sigma : k$.
2. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} v : \sigma$ and $x \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, x : \sigma_0, \Gamma_2 \vdash_{\mathtt{val}} v : \sigma$.
3. If $\Gamma_1, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $x \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, x : \sigma_0, \Gamma_2 \vdash e : \sigma \mid \epsilon$.
4. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$ and $x \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, x : \sigma_0, \Gamma_2 \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$.
5. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{wf}} \sigma : k$ and $\alpha \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{wf}} \sigma : k$.
6. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} v : \sigma$ and $\alpha \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{val}} v : \sigma$.
7. If $\Gamma_1, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $\alpha \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash e : \sigma \mid \epsilon$.
8. If $\Gamma_1, \Gamma_2 \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$ and $\alpha \notin dom(\Gamma_1, \Gamma_2)$ then $\Gamma_1, \alpha^{k_1}, \Gamma_2 \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$.

Proof. straightforward. □

Prop 3. Derived type and effect row are well-formed.
If $\Gamma \vdash e : \sigma \mid \epsilon$ and Γ is well-formed then $\Gamma \vdash_{\mathtt{wf}} \sigma : *$ and $\Gamma \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}$.

Proof. straightforward. □

Prop 4. Replacement.
If $\Gamma \vdash E[e] : \sigma \mid \epsilon$ then $\exists\ \sigma', \epsilon'$ s.t.

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$
2. $\Gamma \vdash_{\mathsf{ec}} E : \sigma' \to \sigma \mid \epsilon$
3. $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E[e'] : \sigma \mid \epsilon$

Proof. By induction on E.

case $E = []$

We have a derivation: $\Gamma \vdash e : \sigma \mid \epsilon$. Let $\sigma' = \sigma$ and $\epsilon' = \epsilon$ then

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$
2. $\Gamma \vdash_{\mathsf{ec}} [] : \sigma' \to \sigma \mid \epsilon$ by Prop3
3. $\forall e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E[e'] : \sigma \mid \epsilon$

case $E = E_0\ e_0$

$\Gamma \vdash E_0[e]\ e_0 : \sigma \mid \epsilon$	(given)
$\Gamma \vdash E_0[e] : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\Gamma \vdash e_0 : \sigma_1 \mid \epsilon$	(App)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.

(a) $\Gamma \vdash e : \sigma' \mid \epsilon'$

(b) $\Gamma \vdash_{\mathsf{ec}} E_0 : \sigma' \to (\sigma_1 \to \epsilon\ \sigma) \mid \epsilon$

(c) $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$

Therefore,

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a)
2. $\Gamma \vdash_{\mathsf{ec}} E_0\ e_0 : \sigma' \to \sigma \mid \epsilon$ by (b) and (CAPP1)
3. $\forall\ e',\ \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e']\ e_0 : \sigma \mid \epsilon$ by (c) and (App)

case $E = v\ E_0$

$\Gamma \vdash v\ E_0[e] : \sigma \mid \epsilon$	(given)
$\Gamma \vdash v : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\Gamma \vdash E_0[e] : \sigma_1 \mid \epsilon$	(App)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.

(a) $\Gamma \vdash e : \sigma' \mid \epsilon'$

(b) $\Gamma \vdash_{\mathsf{ec}} E_0 : \sigma' \to \sigma_1 \mid \epsilon$

(c) $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma \mid \epsilon$

Therefore,

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a)
2. $\Gamma \vdash_{\mathsf{ec}} v\ E_0 : \sigma' \to \sigma_1 \mid \epsilon$ by (b) and (CAPP2)
3. $\forall\ e',\ \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash v\ E_0[e'] : \sigma \mid \epsilon$ by (c) and (App)

case　$E = E_0[\sigma_0]$

$\Gamma \vdash E_0[e][\sigma_0] : \sigma_1[\alpha^k := \sigma_0] \mid \epsilon$　(given)
$\Gamma \vdash E_0[e] : \forall \alpha^k.\sigma_1 \mid \epsilon$　(TApp)
$\Gamma \vdash_{\mathtt{wf}} \sigma_0 : k$　(TApp)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.
(a)　$\Gamma \vdash e : \sigma' \mid \epsilon'$
(b)　$\Gamma \vdash_{\mathtt{ec}} E_0 : \sigma' \to (\forall \alpha^k.\sigma_1) \mid \epsilon$
(c)　$\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \forall \alpha^k.\sigma_1 \mid \epsilon$
Therefore,
1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a)
2. $\Gamma \vdash_{\mathtt{ec}} E_0[\sigma_0] : \sigma' \to (\forall \alpha^k.\sigma_1) \mid \epsilon$ by (b) and (CTAPP)
3. $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'][\sigma_0] : \sigma_1[\alpha^k := \sigma_0] \mid \epsilon$ by (c) and (TApp)

case　$E = \mathtt{handle}^{\epsilon}\ h\ E_0$

$\Gamma \vdash \mathtt{handle}^{\epsilon}\ h\ E_0[e] : \sigma \mid \epsilon$　(given)
$\Gamma \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$　(Handle)
$\Gamma \vdash E_0[e] : \sigma \mid \langle l \mid \epsilon \rangle$　(Handle)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.
(a)　$\Gamma \vdash e : \sigma' \mid \epsilon'$
(b)　$\Gamma \vdash_{\mathtt{ec}} E_0 : \sigma' \to \sigma \mid \langle l \mid \epsilon \rangle$
(c)　$\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma \mid \langle l \mid \epsilon \rangle$
Therefore,
1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a).
2. $\Gamma \vdash_{\mathtt{ec}} \mathtt{handle}^{\epsilon}\ h\ E_0 : \sigma' \to \sigma \mid \langle l \mid \epsilon \rangle$ by (c) and (CHANDLE)
3. $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash \mathtt{handle}^{\epsilon}\ h\ E_0[e'] : \sigma \mid \epsilon$ by (c) and (Handle)

case　$E = \mathtt{val}\ x = E_0;\ e_0$

$\Gamma \vdash \mathtt{val}\ x = E_0[e];\ e_0 : \sigma \mid \epsilon$　(given)
$\Gamma \vdash E_0[e] : \sigma_1 \mid \epsilon$　(Let)
$\Gamma, x : \sigma_1 \vdash e_0 : \sigma \mid \epsilon$　(Let)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.
(a)　$\Gamma \vdash e : \sigma' \mid \epsilon'$
(b)　$\Gamma \vdash_{\mathtt{ec}} E_0 : \sigma' \to \sigma_1 \mid \epsilon$
(c)　$\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma_1 \mid \epsilon$
Therefore,
1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a).
2. $\Gamma \vdash_{\mathtt{ec}} \mathtt{val}\ x = E_0; e_0 : \sigma' \to \sigma \mid \epsilon$ by (b) and (CLET)
3. $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash \mathtt{val}\ x = E_0[e'];\ e_0 : \sigma \mid \epsilon$ by (c) and (Let)

case　$E = \mathtt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0)$

$\Gamma \vdash \mathtt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0[e]) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0\rangle\ \sigma_2 \mid \epsilon$　(given)
$\Gamma \vdash E_0[e] : \sigma_1 \to \langle l_1, \ldots, l_n\rangle\ \sigma_2 \mid \epsilon$　(Open)
$\Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}$　(Open)

By I.H. $\exists\ \sigma', \epsilon'$ s.t.

(a)　$\Gamma \vdash e : \sigma' \mid \epsilon'$

(b)　$\Gamma \vdash_{\mathtt{ec}} E_0 : \sigma' \to (\sigma_1 \to \langle l_1, \ldots, l_n\rangle\ \sigma_2) \mid \epsilon$

(c)　$\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma_1 \to \langle l_1, \ldots, l_n\rangle\ \sigma_2 \mid \epsilon$

Therefore,

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a).
2. $\Gamma \vdash_{\mathtt{ec}} \mathtt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0) : \sigma' \to (\sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0\rangle\ \sigma_2) \mid \epsilon$
   by (b) and (COPEN)
3. $\forall\ e',\ \Gamma \vdash e' : \sigma' \mid \epsilon'$
   $\Rightarrow \Gamma \vdash \mathtt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0[e']) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0\rangle\ \sigma_2 \mid \epsilon$
   by (c) and (Open)

case　$E = \mathtt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0)$

$\Gamma \vdash \mathtt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0[e]) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0\rangle$　(given)
$\Gamma \vdash E_0[e] : \sigma \mid \langle l_1, \ldots, l_n\rangle$　(Under)
$\Gamma \vdash_{\mathtt{wf}} \epsilon_0 : \mathtt{eff}$

By I.H. $\exists\ \sigma', \epsilon'$ s.t.

(a)　$\Gamma \vdash e : \sigma' \mid \epsilon'$

(b)　$\Gamma \vdash_{\mathtt{ec}} E_0 : \sigma' \to \sigma \mid \langle l_1, \ldots, l_n\rangle$

(c)　$\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash E_0[e'] : \sigma \mid \langle l_1, \ldots, l_n\rangle$

Therefore,

1. $\Gamma \vdash e : \sigma' \mid \epsilon'$ by (a).
2. $\Gamma \vdash_{\mathtt{ec}} \mathtt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0) : \sigma' \to \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0\rangle$
   by (b) and (CUNDER)
3. $\forall\ e', \Gamma \vdash e' : \sigma' \mid \epsilon' \Rightarrow \Gamma \vdash \mathtt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](E_0[e']) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0\rangle$
   by (c) and (Under)

□

Prop 5. Labels that are not in the effects row are caught in the context.
If $\varnothing \vdash E[\mathtt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \epsilon$ and $op \in \Sigma(l)$ and $l \notin \epsilon$ then $l \in \lceil E \rceil^l$

Proof. By induction on E.

case　$E = [\,]$. trivial

case　$E = E_0\ e_0$

$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]\ e_0 : \sigma \mid \epsilon$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\varnothing \vdash e_0 : \sigma_1 \mid \epsilon$	(App)
$l \in \lceil E_0 \rceil^l$	(I.H.)
$l \in \lceil E \rceil = \lceil E_0 \rceil$	(definition)

case　$E = v_0\ E_0$

$\varnothing \vdash v_0\ E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \epsilon$	(given)
$\varnothing \vdash v_0 : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \mid \epsilon$	(App)
$l \in \lceil E_0 \rceil$	(I.H.)
$l \in \lceil E \rceil = \lceil E_0 \rceil$	(definition)

case　$E = E_0[\sigma_0]$

$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v][\sigma_0] : \sigma_1[\alpha^k := \sigma_0] \mid \epsilon$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \forall \alpha^k.\sigma_1 \mid \epsilon$	(TApp)
$\varnothing \vdash_{\texttt{wf}} \sigma_0 : k$	(TApp)
$l \in \lceil E_0 \rceil$	(I.H.)
$l \in \lceil E \rceil^l = \lceil E_0 \rceil^l$	(definition)

case　$E = \texttt{handle}^{\epsilon}\ h\ E_0$

$\varnothing \vdash \texttt{handle}^{\epsilon}\ h\ E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \epsilon$	(given)
$\varnothing \vdash_{\texttt{ops}} h : \sigma \mid l_0 \mid \epsilon$	(Handle)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle l_0 \mid \epsilon \rangle$	(Handle)

- $l_0 = l$.
  $l \in \lceil E \rceil^l$ ($\because \Sigma(l) = h$)
- $l_0 \neq l$.
  $l \in \lceil E_0 \rceil^l$ ($\because$ *I.H.*)
  $\therefore l \in \lceil E \rceil^l$

case　$E = \texttt{val}\ x = E_0;\ e_0$

$\varnothing \vdash \texttt{val}\ x = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v];\ e_0 : \sigma \mid \epsilon$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \mid \epsilon$	(Let)
$x : \sigma_1 \vdash e_0 : \sigma \mid \epsilon$	(Let)
$l \in \lceil E_0 \rceil^l$	(I.H.)
$l \in \lceil E \rceil^l = \lceil E_0 \rceil^l$	(definition)

case　$E = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$

$\varnothing \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon$	(Open)
$\varnothing \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$	(Open)
$l \in \lceil E_0 \rceil^l$	(I.H.)
$l \in \lceil E \rceil^l = \lceil E_0 \rceil^l$	(definition)

case　$E = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$

$\varnothing \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle l_1, \ldots, l_n \rangle$	(Open)
$\varnothing \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$	(Open)
$l \in \lceil E_0 \rceil^l$	(I.H.)
$l \in \lceil E \rceil^l = \lceil E_0 \rceil^l$	(definition)

□

Lem 4. Well typed operations are handled
If $\varnothing \vdash E[\texttt{peform}^{\epsilon}\ \texttt{op}\ \overline{\sigma}\ \texttt{v}] : \sigma \mid \langle\rangle$ and $op \in \Sigma(l)$
then E has the form $E_1 \cdot \texttt{handle}^{\epsilon}\ h \cdot E_2$ with $op \notin bop(E_2)$ and $op \to f \in h$.

Proof.

$\varnothing \vdash E[\texttt{peform}^{\epsilon}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle\rangle$	(given)
$op \notin \Sigma(l)$	(given)
$l \notin \langle\rangle$	(trivial)
$l \in \lceil E \rceil^l$	($Prop5$)
$E = E_1 \cdot \texttt{handle}^{\epsilon}\ h \cdot E_2$	(by definition of $\lceil E \rceil^l$)
$op \to f \in h$	(above)
$op \notin bop(E_2)$	(Let $\texttt{handle}^{\epsilon}\ h$ be the innermost one)

□

Lem 5. Effect types are meaningful
If $\varnothing \vdash E[\texttt{peform}^{\epsilon}\ \texttt{op}\ \overline{\sigma}\ \texttt{v}] : \sigma \mid \epsilon$ with $op \notin bop(E)$, then $op \in \Sigma(l)$ and $l \in \epsilon$,
i.e. effect types cannot be discarded without a handler.

Proof. By induction on E.

case　$E = []$

$\varnothing \vdash \texttt{peform}^{\epsilon}\ op\ \overline{\sigma}\ v : \sigma \mid \epsilon$	(given and definition of evaluation context)
$\varnothing \vdash \texttt{peform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\varnothing \vdash_{\texttt{val}} \texttt{peform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_1 \to \epsilon\ \sigma$	(Val)
$op \in \Sigma(l)\ and\ l \in \epsilon$	(Perform)

case $E = E_0\ e_0$

$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]\ e_0 : \sigma \mid \epsilon$ (given)
$l \notin bop(E) = bop(E_0)$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$ (App)
$op \in \Sigma(l)\ and\ l \in \epsilon$ (I.H.)

case $E = v_0\ E_0$

$\varnothing \vdash v_0\ E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \epsilon$ (given)
$op \notin bop(E) = bop(E_0)$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \mid \epsilon$ (App)
$op \in \Sigma(l)\ and\ l \in \epsilon$ (I.H.)

case $E = E_0[\sigma_0]$

$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v][\sigma_0] : \sigma_1 \mid \epsilon$ (given)
$op \notin bop(E) = bop(E_0)$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \forall \alpha^k.\sigma_1 \mid \epsilon$ (TApp)
$op \in \Sigma(l)\ and\ l \in \epsilon$ (I.H.)

case $E = \texttt{handle}^{\epsilon}\ h\ E_0$

$\varnothing \vdash \texttt{handle}^{\epsilon}\ h\ E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \epsilon$ (given)
$op \notin bop(E) = bop(E_0)\ \cup\ \{op' \mid (op' \to f) \in h\}$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle l_0 \mid \epsilon \rangle$ (Handle)
$op \notin bop(E_0)$ $(bop(E_0) \subseteq bop(E_0)\ \cup\ \{op' \mid (op' \to f) \in h\})$
$op \in \Sigma(l)\ and\ l \in \langle l_0 \mid \epsilon \rangle$ (I.H.)
$l \in \epsilon$ $(op \notin \{op' \mid (op' \to f) \in h\} = \{op' \mid op' \in \Sigma(l_0)\}$

case $E = \texttt{val}\ x = E_0;\ e_0$

$\varnothing \vdash \texttt{val}\ x = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v];\ e_0 : \sigma \mid \epsilon$ (given)
$op \notin bop(E) = bop(E_0)$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \mid \epsilon$ (Let)
$op \in \Sigma(l)\ and\ l \in \epsilon$ (I.H.)

case $E = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$

$\varnothing \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon \rangle](E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon$ (given)
$op \notin bop(E) = bop(E_0)$ (given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon$ (Open)
$op \in \Sigma(l)\ and\ l \in \epsilon$ (I.H.)

case　$E = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$

$\varnothing \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(given)
$op \notin bop(E) = bop(E_0)$	(given)
$\varnothing \vdash E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle l_1, \ldots, l_n \rangle$	(Under)
$op \in \Sigma(l)\ and\ l \in \langle l_1, \ldots, l_n \rangle$	(I.H.)
$l \in \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(therefore)

□

Lem 11. Variable substitution
If $\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v : \sigma$, then $\Gamma_1, \Gamma_2 \vdash e[x := v] : \sigma \mid \epsilon$

Proof. By induction on typing derivation.

Part1 If $\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{val}} v' : \sigma$ and $\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v : \sigma$, then $\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v'[x := v] : \sigma$.
Part2 If $\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v : \sigma$, then $\Gamma_1, \Gamma_2 \vdash e[x := v] : \sigma \mid \epsilon$.
Part3 If $\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{ops}} h : \sigma \mid l \mid \epsilon$ and $\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v : \sigma$, then $\Gamma_1, \Gamma_2 \vdash_{\texttt{ops}} h[x := v] : \sigma \mid l \mid \epsilon$.

Part1　Typing derivation for expressions.

case　[Val]: $e = v_0$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash x : \sigma \mid \epsilon$	(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{val}} v_0 : \sigma$	(Val)
$\Gamma_1, \Gamma_2 \vdash_{\texttt{val}} v_0[x := v] : \sigma$	($Part2$)
$\Gamma_1, \Gamma_2 \vdash v_0[x := v] : \sigma \mid \epsilon$	(Val)

case　[App]: $e = e_1\ e_2$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1\ e_2 : \sigma \mid \epsilon$	(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1 : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_2 : \sigma_1 \mid \epsilon$	(App)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v] : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$	(I.H.)
$\Gamma_1, \Gamma_2 \vdash e_2[x := v] : \sigma_1 \mid \epsilon$	(I.H.)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v]\ e_2[x := v] : \sigma \mid \epsilon$	(App)

case　[TApp]: $e = e_1[\sigma]$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1[\sigma] : \sigma_1[\alpha^k := \sigma] \mid \epsilon$	(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1 : \forall \alpha^k . \sigma_1 \mid \epsilon$	(TApp)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v] : \forall \alpha^k . \sigma_1 \mid \epsilon$	(I.H.)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v][\sigma] : \sigma_1[\alpha^k := \sigma] \mid \epsilon$	(TApp)

case　[Handle]: $e = \texttt{handle}^{\epsilon}\ h\ e$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash \texttt{handle}^{\epsilon}\ h\ e : \sigma \mid \epsilon$　(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{ops}} h : \sigma \mid l \mid \epsilon$　(Handle)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e : \sigma \mid \langle l \mid \epsilon \rangle$　(Handle)
$\Gamma_1, \Gamma_2 \vdash_{\texttt{ops}} h[x := v] : \sigma \mid l \mid \epsilon$　($Part3$)
$\Gamma_1, \Gamma_2 \vdash e[x := v] : \sigma \mid \langle l \mid \epsilon \rangle$　(I.H.)
$\Gamma_1, \Gamma_2 \vdash \texttt{handle}^{\epsilon}\ h[x := v]\ e[x := v] : \sigma \mid \epsilon$　(Handle)

case　[Let]: $e = \texttt{val}\ y = e_1;\ e_2$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash \texttt{val}\ y = e_1;\ e_2 : \sigma \mid \epsilon$　(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1 : \sigma_2 \mid \epsilon$　(Let)
$\Gamma_1, x : \sigma_1, \Gamma_2, y : \sigma_2 \vdash e_2 : \sigma \mid \epsilon$　(Let)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v] : \sigma_2 \mid \epsilon$　(I.H.)
$\Gamma_1, \Gamma_2.y : \sigma_2 \vdash e_2[x := v] : \sigma \mid \epsilon$　(I.H.)
$\Gamma_1, \Gamma_2 \vdash \texttt{val}\ y = e_1[x := v];\ e_2[x := v] : \sigma \mid \epsilon$　(Let)

case　[Open]: $e = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1)$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon$　(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1 : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon$　(Open)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　(Open)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v] : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon$　(I.H.)
$\Gamma_1, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　($Prop2$)
$\Gamma_1, \Gamma_2 \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1[x := v]) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon$　(Open)

case　[Under]: $e = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1)$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$　(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash e_1 : \sigma \mid \langle l_1, \ldots, l_n \rangle$　(Under)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　(Under)
$\Gamma_1, \Gamma_2 \vdash e_1[x := v] : \sigma \mid \langle l_1, \ldots, l_n \rangle$　(I.H.)
$\Gamma_1, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　($Prop2$)
$\Gamma_1, \Gamma_2 \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1[x := v]) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$　(Under)

Part2　Typing derivation for values.

case　[Var]: $v_0 = x$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{val}} x : \sigma_1$　(given)
$x[x := v] = v$　(by substitution)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{val}} v : \sigma_1$　(given)
$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\texttt{val}} x[x := v] : \sigma_1$　(follows)

case　[Var]: $v_0 = y$ and $y \neq x$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\mathtt{val}} y : \sigma$	(given)
$y : \sigma \in \Gamma_1, x : \sigma_1, \Gamma_2$	(Var)
$y \neq x$	(given)
$y : \sigma \in \Gamma_1, \Gamma_2$	(follows)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} y : \sigma$	(Var)
$y[x := v] = y$	(by substitution)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} y[x := v] : \sigma$	(Var)

case　[Abs]: $v_0 = \lambda^{\epsilon} y : \sigma_2.e$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\mathtt{val}} \lambda^{\epsilon} y : \sigma_2.e : \sigma_2 \to \epsilon\ \sigma_3$	(given)
$\Gamma_1, x : \sigma_1, \Gamma_2, y : \sigma_2 \vdash_{\mathtt{val}} e : \sigma_3 \mid \epsilon$	(Abs)
$\Gamma_1, \Gamma_2, y : \sigma_2 \vdash e[x := v] : \sigma_3 \mid \epsilon$	$(Part1)$
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} \lambda^{\epsilon} y : \sigma_2.e[x := v] : \sigma_2 \to \epsilon\ \sigma_3$	(Abs)

case　$[TAbs]$: $v_0 = \Lambda\alpha^k.v_1$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\mathtt{val}} \Lambda\alpha^k.v_1 : \forall\alpha^k.\sigma_2$	(given)
$\Gamma_1, x : \sigma_1, \Gamma_2, \alpha^k \vdash_{\mathtt{val}} v_1 : \sigma_2$	(TAbs)
$\Gamma_1, \Gamma_2, \alpha^k \vdash_{\mathtt{val}} v_1[x := v] : \sigma_2$	(I.H.)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} \Lambda\alpha^k.v_1[x := v] : \forall\alpha^k.\sigma_2$	(TAbs)

case　[Perform]: $v_0 = \mathtt{perform}^{\epsilon}\ op\ \overline{\sigma}$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\mathtt{val}} \mathtt{perform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_3[\overline{\alpha} := \overline{\sigma}]$	(given)
$op : \forall\overline{\alpha}.\sigma_2 \to \sigma_3 \in \Sigma(l)$	(Perform)
$(\mathtt{perform}^{\epsilon}\ op\ \overline{\sigma})[x := v] = \mathtt{perform}^{\epsilon}\ op\ \sigma$	(by substitution)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} (\mathtt{perform}^{\epsilon}\ op\ \overline{\sigma})[x := v] : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_3[\overline{\alpha} := \overline{\sigma}]$	(Perform)

case　[Handler]: $v_0 = \mathtt{handler}^{\epsilon}\ h$

$\Gamma_1, x : \sigma_1, \Gamma_2 \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon}\ h : (() \to \langle l \mid \epsilon \rangle\ \sigma) \to \epsilon\ \sigma$	(given)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$	(Handler)
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{ops}} h[x := v] : \sigma \mid l \mid \epsilon$	$(Part3)$
$\Gamma_1, \Gamma_2 \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon}\ h[x := v] : (() \to \langle l \mid \epsilon \rangle\ \sigma) \to \epsilon\ \sigma$	(Handler)

Part3　Typing derivation for handlers. Follows directly from Part2.

□

Lem 12. Type variable substitution
If $\Gamma_1, \alpha^k, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $\Gamma_1 \vdash_{\mathtt{wf}} \sigma_1 : k$, then $\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$

Proof. By induction on typing derivation.

Part1 if $\Gamma_1, \alpha^k, \Gamma_2 \vdash e : \sigma \mid \epsilon$ and $\Gamma_1 \vdash_{\texttt{wf}} \sigma_1 : k$,
then $\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$

Part2 if $\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{val}} v : \sigma \mid \epsilon$ and $\Gamma_1 \vdash_{\texttt{wf}} \sigma_1 : k$,
then $\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{val}} v[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$

Part3 if $\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{ops}} h : \sigma \mid l \mid \epsilon$ and $\Gamma_1 \vdash_{\texttt{wf}} \sigma_1 : k$,
then $\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{ops}} h[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid l[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$

Part1　Typing derivation for expressions.

case　[Val]: $e = v_0$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash v_0 : \sigma \mid \epsilon$	(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{val}} v_0 : \sigma$	(Val)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{val}} v_0[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1]$	($Part2$)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash v_0[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(Val)

case　[App]: $e = e_1\ e_2$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1\ e_2 : \sigma \mid \epsilon$	($givem$)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \sigma_2 \to \epsilon\ \sigma \mid \epsilon$	(App)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_2 : \sigma_2 \mid \epsilon$	(App)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1] : \sigma_2[\alpha^k := \sigma_1] \to \epsilon[\alpha^k := \sigma_1]\ \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_2[\alpha^k := \sigma_1] : \sigma_2[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1]\ e_2[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(App)

case　[TApp]: $e = e_1[\sigma]$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1[\sigma] : \sigma_2 \mid \epsilon$	(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \forall \beta^{k'}.\sigma_2 \mid \epsilon$	(TApp)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{wf}} \sigma : k'$	(TApp)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1 : \forall \beta^{k'}.\sigma_2[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{wf}} \sigma[\alpha^k := \sigma_1] : k'$	($Prop1$)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1][\sigma[\alpha^k := \sigma_1]] : \sigma_2[\alpha^k := \sigma_1][\beta^{k'} := \sigma[\alpha^k := \sigma_1]] \mid \epsilon[\alpha^k := \sigma_1]$	(TApp)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\sigma][\alpha^k := \sigma_1] : \sigma_2[\beta^{k'} := \sigma][\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(by substitution)

case　[Handle]: $e = \texttt{handle}^{\epsilon}\ h\ e_1$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash \texttt{handle}^{\epsilon}\ h\ e_1 : \sigma \mid \epsilon$	(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{ops} h : \sigma \mid l \mid \epsilon$	(Handle)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \sigma \mid \langle l \mid \epsilon \rangle$	(Handle)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{ops} h[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid l \mid \epsilon[\alpha^k := \sigma_1]$	(I.H.)
$\Gamma_1, \alpha^k, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1 : \sigma[\alpha^k := \sigma_1] \mid \langle l \mid \epsilon \rangle[\alpha^k := \sigma_1]$	(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash (\texttt{handle}^{\epsilon}\ h\ e_1)[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$	(Handle)

case　[Let]: $e = \texttt{val}\ x = e_1;\ e_2$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash \texttt{val}\ x = e_1;\ e_2 : \sigma \mid \epsilon$　(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \sigma_2 \mid \epsilon$　(Let)
$\Gamma_1, \alpha^k, \Gamma_2, x : \sigma_2 \vdash e_2 : \sigma$　(Let)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1] : \sigma_2[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1], x : \sigma_2[\alpha^k := \sigma_1] \vdash e_2[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash \texttt{val}\ x = e_1[\alpha^k := \sigma_1];\ e_2[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　(Let)

case　[Open]: $e = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1)$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1) : \sigma_2 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_3 \mid \epsilon$　(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \sigma_2 \to \langle l_1, \ldots, l_n \rangle\ \sigma_3 \mid \epsilon$　(Open)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　(Open)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1] : (\sigma_2 \to \langle l_1, \ldots, l_n \rangle\ \sigma_3)[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{wf}} \epsilon_0[\alpha^k := \sigma_1] : \texttt{eff}$　(*Prop*1)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle[\alpha^k := \sigma_1]](e_1[\alpha^k := \sigma_1]) :$
$(\sigma_2 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_3)[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　(Open)

case　[Under]: $e = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1)$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_1) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$　(given)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash e_1 : \sigma \mid \langle l_1, \ldots, l_n \rangle$　(Under)
$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$　(Under)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash e_1[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid \langle l_1, \ldots, l_n \rangle[\alpha^k := \sigma_1]$　(I.H.)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{wf}} \epsilon_0[\alpha^k := \sigma_1] : \texttt{eff}$　(*Prop*1)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle[\alpha^k := \sigma_1]](e_1[\alpha^k := \sigma_1]) :$
$\sigma[\alpha^k := \sigma_1] \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle[\alpha^k := \sigma_1]$　(Under)

Part2　Typing derivation for values.

case　[Var]: $v = x$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\texttt{val}} x : \sigma$　(given)
$x : \sigma \in \Gamma_1, \Gamma_2[\alpha^k := \sigma_1]$　(Var)
$x : \sigma[\alpha^k := \sigma_1] \in \Gamma_1, \Gamma_2[\alpha^k := \sigma_1]$　(therefore)
$x[\alpha^k := \sigma_1] = x$　(by substitution)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{val}} x : \sigma[\alpha^k := \sigma_1]$　(Var)
$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\texttt{val}} x[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1]$　(follows)

case　[Abs]: $v = \lambda^{\epsilon} y : \sigma_2 . e$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\mathtt{val}} \lambda^{\epsilon} y : \sigma_2 . e : \sigma_2 \to \epsilon\ \sigma_3$　(given)

$\Gamma_1, \alpha^k, \Gamma_2, y : \sigma_2 \vdash e : \sigma_3 \mid \epsilon$　(Abs)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1], y : \sigma_2[\alpha^k := \sigma_1] \vdash e[\alpha^k := \sigma_1] : \sigma_2[\alpha^k := \sigma_1] \mid \epsilon[\alpha^k := \sigma_1]$　($Part1$)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} (\lambda^{\epsilon} y : \sigma_2 . e)[\alpha^k := \sigma_1] : (\sigma_2 \to \epsilon\ \sigma_3)[\alpha^k := \sigma_1]$　(Abs)

case　[TAbs]: $v = \Lambda \beta^{k'} . v_0$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\mathtt{val}} \Lambda \beta^{k'} . v : \forall \beta^{k'} . \sigma_2$　(given)

$\Gamma_1, \alpha^k, \Gamma_2, \beta^{k'} \vdash_{\mathtt{val}} v_0 : \sigma_2$　(TAbs)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1], \beta^{k'}[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} v_0[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1]$　(I.H.)

$\beta^{k'}[\alpha^k := \sigma_1] = \beta^{k'}$　($\beta^k\ is\ fresh$)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1], \beta^{k'} \vdash_{\mathtt{val}} v_0[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1]$　(therefore)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} \Lambda \beta^{k'} . v_0[\alpha^k := \sigma_1] : \forall \beta^{k'} . \sigma[\alpha^k := \sigma_1]$　(TAbs)

case　[Perform]: $v = \mathtt{perform}^{\epsilon}\ op\ \overline{\sigma}$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\mathtt{val}} \mathtt{perform}^{\epsilon}\ op\ \overline{\sigma} : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_3[\overline{\alpha} := \overline{\sigma}]$　(given)

$op : \forall \overline{\alpha} . \sigma_2 \to \sigma_3 \in \Sigma(l)$　(Perform)

$\overline{\alpha} \notin ftv(\Gamma_1, \alpha^k, \Gamma_2)$　(Perform)

$(\mathtt{peform}^{\epsilon]}\ op\ \overline{\sigma})[\alpha^k := \sigma_1] = \mathtt{perform}^{\epsilon[\alpha^k := \sigma_1]}\ op\ \overline{\sigma}[\alpha^k := \sigma_1]$　(by substitution)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} (\mathtt{perform}^{\epsilon}\ op\ \overline{\sigma})[\alpha^k := \sigma_1] : (\sigma_2[\overline{\alpha} := \overline{\sigma}] \to \langle l \mid \epsilon \rangle\ \sigma_3[\overline{\alpha} := \overline{\sigma}])[\alpha^k := \sigma_1]$　(Perform)

case　[Handler]: $v = \mathtt{handler}^{\epsilon}\ h$

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon}\ h : \sigma$　(given)

$\Gamma_1, \alpha^k, \Gamma_2 \vdash_{\mathtt{val}} h : \sigma \mid l \mid \epsilon$　(Handler)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} h[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1] \mid l \mid \epsilon[\alpha^k := \sigma_1]$　($Part3$)

$\Gamma_1, \Gamma_2[\alpha^k := \sigma_1] \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon[\alpha^k := \sigma_1]}\ h[\alpha^k := \sigma_1] : \sigma[\alpha^k := \sigma_1]$　(Handler)

Part3　Typing derivation for handlers. Follows directly from Part2

□

Lem 13. Progress with effects
If $\varnothing \vdash e_1 : \sigma \mid \epsilon$ then either e_1 is a value, or $e_1 \mapsto e_2$, or $e_1 = E[\mathtt{peform}^{\epsilon_0}\ \mathtt{op}\ \overline{\sigma}\ \mathtt{v}]$,
where $op : \forall \overline{\alpha} . \sigma_1 \to \sigma_2 \in \Sigma(l)$, and $op \notin bop(E)$

Proof. By induction on typing derivation.

case　[App]: $e_1 = e_3\ e_4$.

$\varnothing \vdash e_3\ e_4 : \sigma \mid \epsilon$　(given)

$\varnothing \vdash e_3 : \sigma_1 \to \epsilon\ \sigma \mid \epsilon$　(App)

$\varnothing \vdash e_4 : \sigma_1 \mid \epsilon$　(above)

By I.H., we know that either e_3 is a value, or $e_3 \mapsto e_5$, or $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e_3 \mapsto e_5$. Then we know $e_3\ e_4 \mapsto e_5\ e_4$ by [STEP] and the goal holds.
- $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = E_0\ e_4$, then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
- e_3 is a value. By I.H., we know either e_4 is a value, or $e_4 \mapsto e_6$, or $e_4 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
  - $e_4 \mapsto e_6$. Then we know $e_3\ e_4 \mapsto e_3\ e_6$ by [STEP] and the goal holds.
  - $e_4 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = e_3\ E_0$, then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
  - e_4 is a value, then we do case analysis on the form of e_3.

subcase　$e_3 = x$. This is impossible because x is not well-typed under an empty context.

subcase　$e_3 = \lambda x : \sigma.e$. Then by (*app*) and [STEP] we have $(\lambda x : \sigma.e)\ e_4 \mapsto e[x := e_4]$.

subcase　$e_3 = \Lambda\alpha^k.e$. This is impossible because it does not have a function type.

subcase　$e_3 = \texttt{peform}^{\epsilon}\ op\ \overline{\sigma}$. Let $E = [\,]$, then we have $e_1 = E[\texttt{peform}^{\epsilon}\ op\ \overline{\sigma}\ e_4]$.

subcase　$e_3 = \texttt{handler}^{\epsilon}\ h$.

Then by (*handler*) and [STEP] we have $\texttt{handler}^{\epsilon}\ h\ e_4 \mapsto \texttt{handle}^{\epsilon}\ h\ (e_4())$.

subcase　$e_3 = \texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v)$. Then by (*open*) and [STEP] we have $\texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v)\ e_4 \mapsto \texttt{under}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v\ e_4)$.

case　[TApp] $e_1 = e_3[\sigma_1]$.

$\varnothing \vdash e_3[\sigma_1] : \sigma_2[\alpha^k := \sigma_1] \mid \epsilon$	(given)
$\varnothing \vdash e_3 : \forall\alpha^k.\sigma_2 \mid \epsilon$	(TApp)
$\varnothing \vdash_{\texttt{wf}} \sigma_1 : k$	(TApp)

By I.H., we know that either e_3 is a value, or $e_3 \mapsto e_4$, or $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e_3 \mapsto e_4$. Then we know $e_3 \mapsto e_4$ by [STEP] and the goal holds.
- $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = E_0[\sigma_1]$, then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
- e_3 is a value. Then we do case analysis on the form of e_3.

subcase　$e_3 = x$. This is impossible because x is not well-typed under an empty context.

subcase　$e_3 = \lambda x : \sigma.e$. This is impossible because it does not have a polymorhic type.

subcase　$e_3 = \Lambda\alpha^k.e$. Then by (*tapp*) and [STEP] we have $(\Lambda\alpha^k.e)[\sigma] \mapsto e[\alpha^k := \sigma_1]$.

subcase　$e_3 = \texttt{peform}^{\epsilon}\ op\ \overline{\sigma}$. This is impossible because it does not have a polymorhic type.

subcase　$e_3 = \texttt{handler}^{\epsilon}\ h$. This is impossible because it does not have a polymorhic type.

subcase　$e_3 = \texttt{open}[\langle l_1, \ldots, l_n\rangle, \langle l_1, \ldots, l_n \mid \epsilon_0\rangle](v)$. This is impossible because it does not have a polymorhic type.

case　[Handle]: $e_1 = \texttt{handle}^{\epsilon}\ h\ e$.

$\varnothing \vdash \texttt{handle}^{\epsilon}\ h\ e$	(given)
$\varnothing \vdash e : \sigma \mid \langle l \mid \epsilon\rangle$	(Handle)

By I.H., we know that either e is a value, or $e \mapsto e_3$, or $e = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e \mapsto e_3$. Then we know $\texttt{handle}^{\epsilon}\ h\ e \mapsto \texttt{handle}^{\epsilon}\ h\ e_3$ by [STEP] and the goal holds.
- $e = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$, and $op \notin bop(E_0)$. We discuss whether op is bound h.
  - $op \to f \in h$.

    Then by (*perform*) and [STEP] we have $handle^{\epsilon} \cdot h \cdot E_0 \cdot \texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v \mapsto f\ \overline{\sigma}\ v\ k$.

- $op \notin h$. Let $E = \texttt{handle}^{\epsilon}\ h\ E_0$, then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- e is a value. Then by $(return)$ and [STEP] we have $\texttt{handle}^{\epsilon}\ h\ e \mapsto e$.

case [Let]: $e_1 = \texttt{val}\ x = e_3;\ e_4$

$\varnothing \vdash \texttt{val}\ x = e_3;\ e_4 : \sigma \mid \epsilon$	(given)
$\varnothing \vdash e_3 : \sigma_1 \mid \epsilon$	(Let)
$x : \sigma_1 \vdash e_4 : \sigma \mid \epsilon$	(Let)

By I.H., we know that either e_3 is a value, or $e_3 \mapsto e_5$, or $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e_3 \mapsto e_5$. Then we know $\texttt{val}\ x = e_3;\ e_4 \mapsto \texttt{val}\ x = e_5;\ e_4$ by $[STEP]$ and the goal holds.
- $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = val\ x = E_0;\ e_4$, then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
- e_3 is a value. Then by (let) and [STEP] we have $\texttt{val}\ x = e_3;\ e_4 \mapsto e_4[x \coloneqq e_3]$

case [Open]: $e_1 = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3)$

$\varnothing \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma_2 \mid \epsilon$	(given)
$\varnothing \vdash e_3 : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma_2 \mid \epsilon$	(Open)
$\varnothing \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$	(Open)

By I.H., we know that either e_3 is a value, or $e_3 \mapsto e_4$, or $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e_3 \mapsto e_4$. Then we know
  $\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3) \mapsto \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_4)$
  by step and the goals holds.
- $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$ then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
- e_3 is a value. Then we know $\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3)$ is value and the goal holds.

case [Under]: $e_1 = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3)$

$\varnothing \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3) : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(given)
$\varnothing \vdash e_3 : \sigma \mid \langle l_1, \ldots, l_n \rangle$	(Under)
$\varnothing \vdash_{\texttt{wf}} \epsilon_0 : \texttt{eff}$	(Under)

By I.H. we know that either e_3 is a value, or $e_3 \mapsto e_4$, or $e_3 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.

- $e_3 \mapsto e_4$. Then we know
  $\texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3) \mapsto \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_4)$
  by step and the goals holds.
- $e_3 = E_0[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$. Let $E = \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](E_0)$
  then we have $e_1 = E[\texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v]$.
- e_3 is a value.
  Then by $(under)$ and step, we have $\texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](e_3) \mapsto e_3$.

□

Lem 14. Small step preservation

If $\varnothing \vdash e_1 : \sigma \mid \epsilon$ and $e_1 \to e_2$, then $\varnothing \vdash e_2 : \sigma \mid \epsilon$

Proof. By induction on reduction.

case $(\lambda^{\epsilon} x : \sigma_1.e)\ v \to e[x := v]$.

$\varnothing \vdash (\lambda^{\epsilon} x : \sigma_1.e)\ v : \sigma_2 \mid \epsilon$	(given)
$\varnothing \vdash \lambda^{\epsilon} x : \sigma_1.e : \sigma_1 \to \epsilon\ \sigma_2 \mid \epsilon$	(App)
$\varnothing \vdash v : \sigma_1 \mid \epsilon$	(App)
$x : \sigma_1 \vdash e : \sigma_2 \mid \epsilon$	(Abs)
$\varnothing \vdash e[x := v] : \sigma_2 \mid \epsilon$	($Lem11$)

case $(\Lambda\alpha^k.v)[\sigma] \to v[\alpha^k := \sigma]$.

$\varnothing \vdash (\Lambda\alpha^k.v)[\sigma] : \sigma_1[\alpha^k := \sigma] \mid \epsilon$	(given)
$\varnothing \vdash \Lambda\alpha^k.v : \forall\alpha^k.\sigma_1 \mid \epsilon$	(TApp)
$\varnothing \vdash_{\mathtt{wf}} \sigma : k$	(TApp)
$\varnothing \vdash_{\mathtt{val}} \Lambda\alpha^k.v : \forall\alpha^k.\sigma_1$	(Val)
$\varnothing \vdash_{\mathtt{wf}} \epsilon : \mathtt{eff}$	(Val)
$\alpha^k \vdash_{\mathtt{val}} v : \sigma_1$	(TAbs)
$\alpha^k \vdash v : \sigma_1 \mid \epsilon$	(Val)
$\varnothing \vdash v[\alpha^k := \sigma] : \sigma_1[\alpha^k := \sigma] \mid \epsilon$	($Lem12$)

case $(\mathtt{handler}^{\epsilon}\ h)\ v \to \mathtt{handle}^{\epsilon}\ h\ (v\ ())$.

$\varnothing \vdash (\mathtt{handler}^{\epsilon}\ h)\ v : \sigma \mid \epsilon$	(given)
$\varnothing \vdash \mathtt{handler}^{\epsilon}\ h : (() \to \langle l \mid \epsilon\rangle\ \sigma) \to \epsilon\ \sigma \mid \epsilon$	(App)
$\varnothing \vdash v : (() \to \langle l \mid \epsilon\rangle)\ \sigma \mid \epsilon$	(App)
$\varnothing \vdash_{\mathtt{val}} \mathtt{handler}^{\epsilon}\ h : (() \to \langle l \mid \epsilon\rangle\ \sigma) \to \epsilon\ \sigma$	(Val)
$\varnothing \vdash_{\mathtt{ops}} h : \sigma \mid l \mid \epsilon$	(Handler)
$\varnothing \vdash v : () \to \langle l \mid \epsilon\rangle \mid \langle l \mid \epsilon\rangle$	(Val)
$\varnothing \vdash v\ () : \sigma \mid \langle l \mid \epsilon\rangle$	(App)
$\varnothing \vdash \mathtt{handle}^{\epsilon}\ h\ (v\ ()) : \sigma \mid \epsilon$	(Handle)

case $\mathtt{handle}^{\epsilon}\ h \cdot v \to v$.

$\varnothing \vdash \mathtt{handle}^{\epsilon}\ h \cdot v : \sigma \mid \epsilon$	(given)
$\varnothing \vdash v : \sigma \mid \langle l \mid \epsilon\rangle$	(Handle)
$\varnothing \vdash v : \sigma \mid \epsilon$	(Val)

case　$\texttt{handle}^{\epsilon}\ h \cdot E \cdot \texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v \to f\ [\overline{\sigma}]\ v\ k.$

$op \notin bop(E)\ and\ op \to f \in h$	(given)
$op : \forall \overline{\alpha}.\sigma_1 \to \sigma_2 \in \Sigma(l)$	(given)
$k = \lambda^{\epsilon} x : \sigma_2[\overline{\alpha} := \overline{\sigma}].\texttt{handle}^{\epsilon}\ h \cdot E \cdot x$	(given)
$\varnothing \vdash \texttt{handle}^{\epsilon}\ h \cdot E \cdot \texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v : \sigma \mid \epsilon$	(given)
$\varnothing \vdash_{\texttt{ops}} h : \sigma \mid l \mid \epsilon$	(Handle)
$\varnothing \vdash_{\texttt{val}} f : \overline{\alpha}.\sigma_1 \to \epsilon\ ((\sigma_2 \to \epsilon\ \sigma) \to \epsilon\ \sigma)$	(Ops)
$\varnothing \vdash_{\texttt{val}} f : \overline{\alpha}.\sigma_1 \to \epsilon\ ((\sigma_2 \to \epsilon\ \sigma) \to \epsilon\ \sigma) \mid \epsilon$	(Val)
$\varnothing \vdash f\ [\overline{\sigma}] : \sigma_1[\overline{\alpha} := \overline{\sigma}] \to \epsilon\ ((\sigma_2[\overline{\alpha} := \overline{\sigma}] \to \epsilon\ \sigma) \to \epsilon\ \sigma)$	(TApp)
$\varnothing \vdash \texttt{peform}^{\epsilon_0}\ op\ \overline{\sigma}\ v : \sigma_2[\overline{\alpha} := \overline{\sigma}] \mid \epsilon'$	($Prop4$ *and* (Perform))
$\varnothing \vdash_{\texttt{ec}} \texttt{handle}^{\epsilon}\ h \cdot E : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \sigma \mid \epsilon$	($Prop4$)
$\varnothing \vdash v : \sigma_1[\overline{\alpha} := \overline{\sigma}] \mid \epsilon'$	(*App and Tapp*)
$\varnothing \vdash v : \sigma_1[\overline{\alpha} := \overline{\sigma}] \mid \epsilon$	(Val)
$\varnothing \vdash f\ [\overline{\sigma}]\ v : (\sigma_2[\overline{\alpha} := \overline{\sigma}] \to \epsilon\ \sigma) \to \epsilon\ \sigma \mid \epsilon$	(App)
$x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \vdash_{\texttt{val}} x : \sigma[\overline{\alpha} := \overline{\sigma}]$	(Var)
$x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \vdash_{\texttt{val}} x : \sigma[\overline{\alpha} := \overline{\sigma}] \mid \epsilon'$	(Val)
$x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \vdash_{\texttt{ec}} \texttt{handle}^{\epsilon}\ h \cdot E : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \sigma \mid \epsilon$	($Weakening$)
$x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \vdash \texttt{handle}^{\epsilon}\ h \cdot E \cdot x : \sigma \mid \epsilon$	($Prop4$)
$\varnothing \vdash_{\texttt{val}} \lambda^{\epsilon} x : \sigma_2[\overline{\alpha} := \overline{\sigma}].\texttt{handle}^{\epsilon}\ h \cdot E \cdot x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \epsilon\ \sigma$	(Abs)
$\varnothing \vdash_{\texttt{val}} \lambda^{\epsilon} x : \sigma_2[\overline{\alpha} := \overline{\sigma}].\texttt{handle}^{\epsilon}\ h \cdot E \cdot x : \sigma_2[\overline{\alpha} := \overline{\sigma}] \to \epsilon\ \sigma \mid \epsilon$	(Val)
$\varnothing \vdash f\ [\overline{\sigma}]\ v\ k : \sigma \mid \epsilon$	(App)

case　$\texttt{val}\ x = v;\ e \to e[x := v]$

$\varnothing \vdash \texttt{val}\ x = v;\ e : \sigma \mid \epsilon$	(given)
$\varnothing \vdash v : \sigma_1 \mid \epsilon$	(Let)
$x : \sigma_1 \vdash e : \sigma \mid \epsilon$	(Let)
$\varnothing \vdash e[x := v] : \sigma \mid \epsilon$	($Lem11$)

case　$\texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v)\ v' \to \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v\ v')$

$\varnothing \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v)\ v' : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(given)
$\varnothing \vdash \texttt{open}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v) : \sigma_1 \to \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle\ \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(App)
$\varnothing \vdash v' : \sigma_1 : \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(App)
$\varnothing \vdash v : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(Open)
$\varnothing \vdash v' : \sigma_1 \mid \langle l_1, \ldots, l_n \rangle$	(Val)
$\varnothing \vdash v : \sigma_1 \to \langle l_1, \ldots, l_n \rangle\ \sigma \mid \langle l_1, \ldots, l_n \rangle$	(Val)
$\varnothing \vdash v\ v' : \sigma \mid \langle l_1, \ldots, l_n \rangle$	(App)
$\varnothing \vdash \texttt{under}[\langle l_1, \ldots, l_n \rangle, \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle](v\ v') : \sigma \mid \langle l_1, \ldots, l_n \mid \epsilon_0 \rangle$	(Under)

case　$\texttt{under}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle](v) \to v$

$\varnothing \vdash \texttt{under}[\langle l_1,\ldots,l_n\rangle,\langle l_1,\ldots,l_n \mid \epsilon_0\rangle(v) : \sigma \mid \langle l_1,\ldots,l_n \mid \epsilon_0\rangle$	(given)
$\varnothing \vdash v : \sigma \mid \langle l_1,\ldots,l_n\rangle$	(Under)
$\varnothing \vdash v : \sigma \mid \langle l_1,\ldots,l_n \mid \epsilon_0\rangle$	(Val)

□

Thm 2. Progress
If $\varnothing \vdash e_1 : \sigma \mid \langle\rangle$ then either e_1 is a value, or $e_1 \mapsto e_2$

Proof. Apply Lem13, then we know that either e_1 is a value, or $e_1 \mapsto e_2$, or $e_1 = E[\texttt{peform}^{\epsilon}\ op\ \overline{\sigma}\ v]$, where $op : \forall\alpha.\sigma_1 \to \sigma_2 \in \Sigma(l)$, and $l \notin bop(E)$. For the first two cases, we have proved the goal. For the last casse, we prove it by contradiction.

$\varnothing \vdash E[\texttt{proform}^{\epsilon}\ op\ \overline{\sigma}\ v] : \sigma \mid \langle\rangle$	(given)
$l \notin bop(E)$	(given)
$l \in \langle\rangle$	($Lem5$)
Contradiction	

□

Thm 3. Preservation
If $\varnothing \vdash e_1 : \sigma \mid \langle\rangle$ and $e_1 \mapsto e_2$, then $\varnothing \vdash e_2 : \sigma \mid \langle\rangle$

Proof.

$e_1 = E[e_1']$	([STEP])
$e_1' \to e_2'$	(above)
$e_2 = E[e_2']$	(above)
$\varnothing \vdash E[e_1'] : \sigma \mid \langle\rangle$	(given)
$\varnothing \vdash e_1' : \sigma' \mid \epsilon'$	($Prop4$)
$\varnothing \vdash E : \sigma_1 \to \sigma \mid \langle\rangle$	(above)
$\varnothing \vdash e_2' : \sigma' \mid \epsilon'$	($Lem14$)
$\varnothing \vdash E[e_2'] : \sigma \mid \langle\rangle$	($Prop4$)

□