代数的エフェクトハンドラを持つ言語のプログラムに対するエフェクト列の静的な見積り

山﨑 陽介　川端 英之　弘中 哲夫

代数的エフェクトは制御の流れに対する様々な操作を抽象化する仕組みの一つで，CPS を用いずにモジュラーに記述できるようにした言語機構である．しかし，その機能を十分に活用するためには，複雑になりがちな制御フローを適切に把握したプログラミングが必要で，簡単ではない．本論文では，代数的エフェクトを用いたプログラミングの支援を目的として，プログラミング中で発生するエフェクト列を静的に見積もりプログラマに提示する手法を提案する．本アルゴリズムは，型検査とは独立しており，型にエフェクト列の情報を組み込むものとは異なる．エフェクトのハンドリング部分を詳細に検査することで，精度の高い見積りを得ることができる．本手法を組み込んだプログラミング支援ツールは，エフェクト発生の順序や，未ハンドルエフェクトの有無などの情報をユーザに提供でき，代数的エフェクトを用いるプログラマの負担を大幅に軽減できることが期待される．

1　はじめに

代数的エフェクトとそのハンドラ [5] は制御の流れに対する様々な操作を抽象化する仕組みの一つで，CPS を用いずにモジュラーに記述できるようにした言語機構である．プログラム中のエフェクトの発生を抽象化してハンドラで定義するものである．例外ハンドラの一般化と見ることもでき，例外ハンドラと異なる点は，処理がハンドラに移る際に，その地点での継続を切り取り，それを再開することができる点である．

代数的エフェクトとそのハンドラを十分に活用するためには，複雑になりがちな制御フローを適切に把握したプログラミングが必要で，簡単ではない．

本論文では，Pretnar の代数的エフェクトとそのハンドラの言語体系 [6] を元に，代数的エフェクトを用いたプログラミングの支援を目的として，プログラミング中で発生するエフェクトの発生順序を静的に見積もりプログラマに提示する手法を提案する．Pretnar の代数的エフェクトの型システム [6] では，エフェクトの情報を集合によって表しているため，エフェクトの発生順序までは把握することができない．他に，エフェクトの情報を集合で扱わない型システムとして，川俣ら [8] のものがある．これは，エフェクトの発生順序を型システムに組み込んでいる．しかし，型の性質を満たすために，エフェクトの見積りが保守的になりすぎてしまい，false positive が多くなってしまっている．それらに対して，本論文で提案するアルゴリズムでは，型とは独立に扱い，エフェクトのハンドリング部分を詳細に検査することで，より精度の高い，false positive が少ない見積りを得ることができる．また，川俣ら [8] のシステムでは，シャローハンドラ [3] を採用していたが，我々のシステムでは，ディープハンドラを採用している．

本論文の貢献は，代数的エフェクトを持つ言語に対してのエフェクト列の静的な見積もりを実現するアルゴリズムの提案と実装である．本アルゴリズムの健全性の証明は完成していないが，妥当と予想している．実装は GitHub[†1] にて公開している．

本論文の構成は次のとおりである．2 節では代数的

Static Estimation of Effect Row for programs in an Language with Algebraic Effect Handlers.

Yosuke Yamasaki, Hideyuki Kawabata, Hironaka Tetsuo, 広島市立大学, Hiroshima City University.

†1 https://github.com/zaki5m/effect_analyzer

$$
\begin{array}{lcll}
v & ::= & x, y, k & \text{変数} \\
 & | & \texttt{true} \mid \texttt{false} & \text{ブール定数} \\
 & | & \textbf{fun}\ x \mapsto c & \text{関数} \\
 & | & h & \text{ハンドラ} \\
h & ::= & \textbf{handler}\ \{\ \textbf{return}\ x \mapsto c_r, [\texttt{op}_i(x; k) \mapsto c_{\texttt{op}_i}]_{i \in 1 \ldots n}\ \} & \text{ハンドラ} \\
c & ::= & \textbf{return}\ v & \text{return} \\
 & | & \texttt{op}\ (v; y.c) & \texttt{op}\ \text{呼び出し} \\
 & | & \textbf{do}\ x \leftarrow c_1\ \text{in}\ c_2 & \text{逐次実行} \\
 & | & \textbf{if}\ v\ \textbf{then}\ c_1\ \textbf{else}\ c_2 & \text{条件分岐} \\
 & | & v_1\ v_2 & \text{関数適用} \\
 & | & \textbf{with}\ v\ \textbf{handle}\ c & \text{ハンドリング}
\end{array}
$$

図 1　対象言語の構文

エフェクトハンドラの概要について述べる．3 節では，本論文で対象とする言語を定義する．4 節では，エフェクトの時間的な見積りであるエフェクト列について述べる．5 節では，プログラムからエフェクト列を見積もるアルゴリズムについて述べる．6 節では関連研究を述べる．最後に 7 節では，まとめと今後の課題を述べる．

2　代数的エフェクトとそのハンドラ

代数的エフェクトとそのハンドラ [5] を持つ言語では，エフェクトを発生させるとその地点での継続 (その地点での残りの計算) を切り取って，ハンドラへ処理が渡される．ハンドラは，渡された継続を用いて処理を行うことができる．Pretnar の代数的エフェクトとそのハンドラのチュートリアル [6] を元に例を示す．以下の例の言語は，3 節で定義する．また，以下で出現する記号 ^ は文字列の結合を表している．

以下は，`read` が呼ばれると常に “Bob” を継続に渡すようなハンドラである．

$$
\begin{aligned}
h \stackrel{\text{def}}{=} \textbf{handler}\{&\textbf{return}\ x \mapsto \textbf{return}\ x, \\
&\texttt{read}(\_; k) \mapsto k\ \texttt{"Bob"}\}
\end{aligned}
$$

このハンドラの元で実行するプログラムを以下に定義する．

$$
\begin{aligned}
c \stackrel{\text{def}}{=} &\textbf{do}\ name1 \leftarrow \texttt{read}()\ \textbf{in} \\
&\textbf{do}\ name2 \leftarrow \texttt{read}()\ \textbf{in} \\
&\textbf{return}\ name1\hat{}name2 \\
\textbf{with}\ h\ \textbf{handle}\ c&
\end{aligned}
$$

そこで，**with** h **handle** c を実行することを考える．このプログラムは，`read`() というエフェクトが発生すると，ハンドラに捕捉され k `"Bob"` が実行される．継続 k は **fun** $y \mapsto$ **with** h **handle do** $name1 \leftarrow y$ **in do** $name2 \leftarrow$ `read`() **in return** $name1\hat{}name2$ である．

k `"Bob"` の評価を進めると，**with** h **handle return** `"Bob"^"Bob"` となり，h の **return** 節に`"BobBob"`を引数として渡して，最終的には，**return** `"BobBob"` となる．

次にハンドラを以下のように変更した例を考える．

$$
\begin{aligned}
h' \stackrel{\text{def}}{=} \textbf{handler}\{&\textbf{return}\ x \mapsto \textbf{return}\ x, \\
&\texttt{read}(\_; k) \mapsto \\
&\quad \textbf{do}\ x \leftarrow k\ \texttt{"Bob"}\ \textbf{in} \\
&\quad k\ x\}
\end{aligned}
$$

ハンドラ h と h' が異なる点は `read` というエフェクトが発生し，ハンドラが捕捉した時の動作が異なる．ハンドラ h は，継続 k に “Bob” を渡す処理のみだったが，ハンドラ h' は継続 k に “Bob” を渡した結果 x を，再び継続 k に x を渡している．つまりハンドラ h' では継続を 2 回再開している．

そこで，**with** h' **handle** c を実行することを考え

る．エフェクト read が 3 回呼び出され，最終的には，**return** "BobBobBobBobBobBob" となる．

このように，ハンドラは継続を受け取ることで，制御フローを変化させることが可能である．また，継続を複数回再開するなどの処理も可能なためより複雑なシステムの際には，制御フローを正しく認識することが難しくなる．

3 対象言語

本節では，アルゴリズムの対象とする言語の構文，操作的意味論，型システムについて述べる．

3.1 対象言語の構文

本論文で用いる言語の構文を図 1 に示す．この構文は Pretnar の体系 [6] と同一のものである．この言語は，エフェクトが発生しない**値** v とエフェクトが発生する可能性のある**コンピュテーション** c で構成される．

プログラムで発生するエフェクトはオペレーションと呼ばれる項で抽象化し，ハンドラにオペレーションに対応する実装を定義する．オペレーション呼び出し (op 呼び出し)，すなわち $\mathsf{op}(v; y.c)$ は，v がオペレーション op への引数としてハンドラに渡され，y が op から継続に渡される値を受け取る変数，c が継続である．op 呼び出しが起こると，引数 v がハンドラに渡され，ハンドラ内で継続が再開されると，変数 y に値が束縛され，c が実行される．**ハンドラ**は **return** 節と op 節を持っており，op 節の $[\mathsf{op}_i(x; k) \mapsto c_{\mathsf{op}_i}]_{i \in 1 \ldots n}$ は，$\mathsf{op}_1(x; k) \mapsto c_{\mathsf{op}_1}, \ldots, \mathsf{op}_n(x; k) \mapsto c_{\mathsf{op}_n}$ の略記である．**with** v **handle** c はハンドラ v のもとで，項 c を実行する．

また，オペレーション呼び出しに明示的な継続とらない generic effect [4] を Pretnar の体系 [6] に合わせて以下のように導入する．

$$\mathsf{op} \stackrel{\text{def}}{=} \mathbf{fun}\ x \mapsto \mathsf{op}(x; y.\mathbf{return}\ y)$$

generic effect はシンプルに使えるが，表現力は同じである．

また，本論文内での例の際に，unit や文字列などのブール定数以外の定数の導入はできるものとする．

3.2 操作的意味論

評価関係の定義を図 2 に示す．Pretnar のもの [6] にイベントトレースが加わっており，川俣らのもの [8] のハンドラをシャローハンドラからディープハンドラへ変更したものである (E-HandlingInOp)．また，川俣らのもの [8] は再帰を許す体系だが，本言語では再帰を許さない．

ディープハンドラとシャローハンドラは表現力が等価なことが知られており，ディープハンドラはシャローハンドラを模倣することが可能であり，シャローハンドラは再帰を用いてディープハンドラを模倣することが可能である [3]．

評価関係は，コンピュテーション c とイベントトレース η のペア c, η の二項関係 $c, \eta \rightsquigarrow c', \eta'$ となっており，これは川俣らのもの [8] と同一である．イベントトレースとは，評価の途中でハンドルしたイベント (エフェクト) を記録している，ハンドル済みオペレーションの列である．イベントトレース η の構文は $\eta ::= \epsilon \mid \mathsf{op}^{\checkmark} \mid \eta; \eta$ と定義される．ϵ は空のイベントトレースを表している．$\mathsf{op}^{\checkmark}$ は，ハンドル済みオペレーションを表している．また，; はイベントトレースの連結を表している．(E-HandlingInOp) でオペレーションがハンドラに捕捉された際にハンドル済みオペレーション $\mathsf{op}^{\checkmark}$ が末尾に追加される．例えば，評価の結果イベントトレースが $\mathsf{op}_1^{\checkmark}; \mathsf{op}_2^{\checkmark}$ であった場合は，評価の途中で，op_1 がハンドルされた後に op_2 がハンドルされたことを表している．

3.3 型システム

型システムは Pretnar [6] のものと同一である．Pretnar [6] の型システムの構文を図 3 に示す．値型 A とコンピュテーション型 $\underline{C}$ に分かれており，コンピュテーション型 $A!\Delta$ の Δ はエフェクトの集合である．このエフェクトの集合 Δ は，そのコンピュテーションが発生させる可能性のあるエフェクトの集合を表している．型付け規則は付録 (図 10) に掲載する．

また，Pretnar の型システム [6] で示されている性質は全て満たす．つまり，以下の 2 つの性質を満たす．

- 適切に型付けされたプログラムは行き詰まり状

$$\frac{c_1, \eta_1 \rightsquigarrow c_1', \eta_1'}{\mathbf{do}\ x \leftarrow c_1\ \mathbf{in}\ c_2, \eta \rightsquigarrow \mathbf{do}\ x \leftarrow c_1'\ \mathbf{in}\ c_2, \eta'}\ (\text{E-SEQ}) \qquad \frac{}{\mathbf{do}\ x \leftarrow \mathbf{return} v\ \mathbf{in}\ c, \eta \rightsquigarrow c[v/x], \eta}\ (\text{E-SEQRET})$$

$$\frac{}{\mathbf{do}\ x \leftarrow \mathtt{op}(v; y.c_1)\ \mathbf{in}\ c_2, \eta \rightsquigarrow \mathtt{op}(v; y.\mathbf{do}\ x \leftarrow c_1 \mathbf{in}\ c_2), \eta}\ (\text{E-SEQOP}) \qquad \frac{}{(\mathbf{fun}\ x \to c)\ v, \eta \rightsquigarrow c[v/x], \eta}\ (\text{E-APP})$$

$$\frac{}{\mathbf{if\ true\ then}\ c_1\ \mathbf{else}\ c_2, \eta \rightsquigarrow c_1, \eta}\ (\text{E-IFTRUE}) \qquad \frac{}{\mathbf{if\ false\ then}\ c_1\ \mathbf{else}\ c_2, \eta \rightsquigarrow c_2, \eta}\ (\text{E-IFFALSE})$$

以下では $h = \mathbf{handler}\{\mathbf{return}\ x \mapsto c_r, [\mathtt{op}_i(x; k) \mapsto c_{\mathtt{op}_i}]_{i \in 1 \ldots n}\}$ とする

$$\frac{c, \eta \rightsquigarrow c', \eta'}{\mathbf{with}\ h\ \mathbf{handle}\ c, \eta \rightsquigarrow \mathbf{with}\ h\ \mathbf{handle}\ c', \eta'}\ (\text{E-HANDLING})$$

$$\frac{}{\mathbf{with}\ h\ \mathbf{handle}\ (\mathbf{return}\ v), \eta \rightsquigarrow c_r[v/x], \eta}\ (\text{E-HANDLINGRET})$$

$$\frac{\mathtt{op} \in \{\mathtt{op}_1, \ldots, \mathtt{op}_n\}}{\mathbf{with}\ h\ \mathbf{handle}\ \mathtt{op}_i(v; y.c), \eta \rightsquigarrow c_i[v/x, (\mathbf{fun}\ y \mapsto \mathbf{with}\ h\ \mathbf{handle}\ c)/k], \eta; \mathtt{op}_i^{\checkmark}}\ (\text{E-HANDLINGINOP})$$

$$\frac{\mathtt{op} \notin \{\mathtt{op}_1, \ldots, \mathtt{op}_n\}}{\mathbf{with}\ h\ \mathbf{handle}\ \mathtt{op}(v; y.c), \eta \rightsquigarrow \mathtt{op}(v; y.\mathbf{with}\ h\ \mathbf{handle}\ c), \eta}\ (\text{E-HANDLINGOUTOP})$$

図 2　評価規則

A, B	::=	$\mathtt{bool}$	ブール型
	|	$A \to \underline{C}$	関数型
	|	$\underline{C} \Rightarrow \underline{D}$	ハンドラ型
$\underline{C}, \underline{D}$	::=	$A!\{\mathtt{op}_1, \ldots, \mathtt{op}_n\}$	

図 3　型の構文

態にならないことが保証される [1]

- 以下に示す型安全性 (定理 3.1)

定理 3.1 $\vdash c : A!\Delta$ ならば，以下のいずれかが成り立つ．

- $c = \mathbf{return}\ v$ であり，$\vdash v : A$ である．
- $c = \mathtt{op}(v; y.c')$ であり，$\mathtt{op} \in \Delta$ である
- $c, \eta \rightsquigarrow c', \eta'$ であり，$\vdash c' : A!\Delta$ である．

なお，本論文で扱う言語と Pretnar [6] の言語との違いとして，評価規則にイベントトレースが含まれている部分が異なるが，型情報にはイベントトレースは含まれないため，対象言語の型システムとして，Pretnar [6] の型システムを用いる．

4　エフェクト列

エフェクトの見積りアルゴリズムで用いるエフェクト列について述べる．エフェクト列は，評価規則で用いたイベントトレースの見積りである．

エフェクト列は，構文的にはイベントトレースを拡張したものであり，イベントトレースで表すことができるものはエフェクト列でも表すことができる．

4.1　エフェクト列の構文

エフェクト列の構文を図 4 に示す．$\bot$ は見積りアルゴリズムの中で使用されるものであり，見積もったエフェクト列には現れないものである．ϵ は純粋であること，つまり空のエフェクト列であることを示している．$\mathtt{op}_i^j$ は未ハンドルのオペレーションを表しており，$\mathtt{op}_i^{\checkmark}$ はハンドル済みのオペレーションを表している．肩の部分についている添え字は，プログラム中の $\mathtt{op}$ を一意に識別するものであり，エフェクト列の見積りアルゴリズムの際に使用する．ここでの一意とは，プログラム中で同名のオペレーション呼び出しがあった場合でも，プログラム中の位置が異なる場合

δ	::=	a, b, c, d	エフェクト変数	
	$\mid$	$\bot$	ボトム	$s ::= (\delta_1,\delta_2,\Sigma)$　エフェクト状態
	$\mid$	ϵ	純粋	$\Sigma ::= \{\bigsqcup j : \{x \triangleright a, k \triangleright b \to s\}\}$
	$\mid$	$\mathtt{op}_i^j$	未ハンドルオペレーション	$h^{ops} ::= \{\mathtt{return} : (a, s),$
	$\mid$	$\mathtt{op}_i^{\checkmark}$	ハンドル済みオペレーション	$\bigsqcup_{\mathtt{op}\in\mathtt{ops}} \mathtt{op} : ((a, b, c, cd), s)\}$
	$\mid$	$\delta_1;\delta_2$	連結	
	$\mid$	$\delta_1\mid\delta_2$	非決定的選択	
	$\mid$	$a \to s$	エフェクト関数	
	$\mid$	h^{ops}	ハンドラ	

図 4　エフェクト列の構文

は，それぞれ明確に区別するという意味である．また，δ;δ，$\delta|\delta$ はそれぞれ，連結と非決定的選択を表しており，非決定的選択よりも連結の方が強く結合するものとする．エフェクト関数 $a \to s$ は対象言語の関数に対応しており，エフェクト変数 a の中身が具体的なエフェクト列で置き換えられた時に，エフェクト状態 s を返すものである．

エフェクト状態 s は (δ, δ, Σ) の 3 つ組からなる．1 つ目の δ はそのコンピュテーションで発生するエフェクト列を表しており，2 つ目の δ は，そのコンピュテーションでリターンされる値が持っているエフェクト列を表している．Σ はオペレーションの引数とその地点での継続をラベル付けしているレコードである．ここのラベルは，$\mathtt{op}^j$ の添え時 j に対応する．

ハンドラ h^{ops} は，レコードになっており，キーを `return`，および，ハンドラに含まれる各オペレーション `op` としている．キーに対応する中身は，キーが `return` の時は，1 個のエフェクト変数とエフェクト状態のペアになっており，キーが `op` の時は，4 個のエフェクト変数とエフェクト状態のペアになっている．また，ops はそのハンドラで補足することのできるオペレーションの集合である．

また，見積りアルゴリズムで対象言語からエフェクト列を見積もると，対象言語での値はエフェクト列では δ に対応し，ハンドラは h^{ops}，コンピュテーションは s にそれぞれ対応している．

$$\frac{}{\delta \equiv^{\spadesuit} \delta} \qquad \frac{}{\epsilon;\delta \equiv^{\spadesuit} \delta} \qquad \frac{}{\delta;\epsilon \equiv^{\spadesuit} \delta}$$

$$\frac{\delta_1 \equiv^{\spadesuit} \delta_1' \qquad \delta_2 \equiv^{\spadesuit} \delta_2'}{\delta_1;\delta_2 \equiv^{\spadesuit} \delta_1';\delta_2'} \qquad \frac{\delta_1 \equiv^{\spadesuit} \delta_1'}{\delta_1 \equiv \delta_1'}$$

$$\frac{\delta_1 \equiv \delta_1'}{\delta_1;\mathtt{op}_i^j;\delta_2 \equiv \delta_1'} \qquad \frac{\delta_1 \equiv \delta_1'}{\delta_1;a;\delta_2 \equiv \delta_1'}$$

図 5　エフェクト列の同値関係

4.2　エフェクト列の同値関係

エフェクト列の同値関係を図 5 に示す．$\equiv$ は弱い同値関係であり，$\equiv^{\spadesuit}$ は強い同値関係である．弱い同値関係は強い同値関係を含む．エフェクト列中に出てくる ϵ は，読み飛ばしても強い同値と定義する．また，未ハンドルオペレーション `op` とエフェクト変数 a が表れた際には，その直前までのエフェクト列が弱い同値であれば，弱い同値と定義する．

例えば，$\delta_1 = \mathtt{op}_1^{\checkmark};\epsilon;\mathtt{op}_2^{\checkmark}$，$\delta_2 = \mathtt{op}_1{\checkmark};\mathtt{op}_2^{\checkmark}$ の時，$\delta_1 \equiv \delta_2$ である．他に，$\delta_1' = \mathtt{op}_1^{\checkmark};\mathtt{op}_3^1;\mathtt{op}_2^{\checkmark}$ とする．これは，ハンドル済みエフェクトの中に，$\mathtt{op}_3^1$ という未ハンドルエフェクトが含まれている．$\delta_1' \not\equiv \delta_2$ であり，$\delta_1' \equiv \mathtt{op}_1^{\checkmark}$ であるが，$\delta_1' \not\equiv^{\spadesuit} \mathtt{op}_1^{\checkmark}$ である．

4.3　エフェクト列の順序関係

エフェクト列同士の順序関係 $\sqsubseteq$ を定義する．この順序関係は，左辺のエフェクト列が，右辺に含まれるかを確かめるものである．この順序関係は $\mathtt{op}^{\checkmark}$ に焦点を当てている．先頭からの $\mathtt{op}^{\checkmark}$ のエフェクト列が左辺と右辺で一致している場合にこの順序関係を満

$$\frac{\delta \equiv^{\spadesuit} \delta'}{\delta \sqsubseteq^{\spadesuit} \delta'} \qquad \frac{\delta \equiv^{\spadesuit} \delta'}{\delta \sqsubseteq \delta'}$$

$$\frac{\delta \equiv \delta'}{\delta \sqsubseteq \delta'}$$

$$\frac{\delta_1 \sqsubseteq \delta_1'}{(\delta_1|\delta_2) \sqsubseteq \delta_1'}\ (\text{LHS})$$

$$\frac{\delta_1 \sqsubseteq \delta_1'}{\delta_1 \sqsubseteq (\delta_1'|\delta_2')}\ (\text{RHS})$$

$$\frac{\delta_1 \sqsubseteq \delta_1' \qquad \delta_2 \sqsubseteq \delta_2'}{\delta_1;\delta_2 \sqsubseteq \delta_1';\delta_2'}\ (\text{CONCAT})$$

図 6　エフェクト列の順序関係

$$\frac{\delta \sqsubseteq^{\spadesuit} \delta'}{\mathit{diff}(\delta,\delta') = \epsilon}$$

$$\frac{\delta_2 \sqsubseteq^{\spadesuit} \delta_2'}{\mathit{diff}(\delta_1;\delta_2,\delta_2') = \delta_1}$$

$$\frac{\delta_2 \sqsubseteq^{\spadesuit} \delta_2'}{\mathit{diff}(\delta_2,\delta_1';\delta_2') = \delta_1'}$$

$$\frac{\delta_2 \sqsubseteq^{\spadesuit} \delta_2'}{\mathit{diff}(\delta_1;\delta_2,\delta_1';\delta_2') = \mathit{diff}(\delta_1,\delta_1')}$$

図 7　エフェクト列の差分抽出

たす．図 6 はエフェクト列の順序関係の定義である．

順序関係も同値関係と同様，弱い順序関係と強い順序関係が存在する．2 つの違いは，前提の同値関係が弱いか強いかである．

例えば，$\delta_1 = \mathtt{op}_1^{\checkmark};\mathtt{op}_2^{\checkmark}$，$\delta_2 = \mathtt{op}_1^{\checkmark};(\mathtt{op}_2^{\checkmark}|\epsilon)$ の時，$\delta_1 \sqsubseteq \delta_2$ は成り立つ．また，$\delta_3 = \mathtt{op}_1^{\checkmark};(\mathtt{op}_2^{\checkmark}|\mathtt{op}_3)$ とした時，$\mathtt{op}_1^{\checkmark} \sqsubseteq \delta_3$ も成り立つ．なぜなら，$\mathtt{op}_1^{\checkmark} \not\equiv \mathtt{op}_1^{\checkmark};\mathtt{op}_2^{\checkmark}$ であるが，$\mathtt{op}_1^{\checkmark} \equiv \mathtt{op}_1^{\checkmark};\mathtt{op}_3$ が成り立つためである．

イベントトレースはエフェクト列の部分集合であるため，エフェクト列上の二項関係 $\sqsubseteq$ はイベントトレースとエフェクト列を比較することにも使用することができる．この順序関係は 5.2 節でのアルゴリズムの健全性を示す際に用いる．

4.4　エフェクト列の差分抽出

エフェクト列の差分抽出関数を図 7 に示す．この関数は，後方から二つのエフェクト列がマッチする部分を取り除くものである．また，$\mathit{diff}(\delta_1,\delta_1')$ がどのルールにも当てはまらなかった場合は，δ_1 を返す．

例えば，$\delta_1 = \mathtt{op}_1^{\checkmark};\mathtt{op}_2;\epsilon$ と $\delta_2 = \mathtt{op}_2;\epsilon$ について，$\mathit{diff}(\delta_1,\delta_2)$ を適用した結果は，$\mathtt{op}_1^{\checkmark}$ になる．

4.5　イベントトレースの見積りにおけるエフェクト列の意味

イベントトレースの見積りを行って得られたエフェクト列が，それぞれどのような意味を持つのかについて述べる．本節では，直感的な説明を行い，詳細な見積りアルゴリズムについては 5 節で述べる．

例 4.1

$$\begin{aligned} h \stackrel{\text{def}}{=} \mathbf{handler}\{&\mathbf{return}\ x \mapsto \mathbf{return}\ x \\ &\mathtt{read}(\_;k) \mapsto k\ \text{`Bob`} \\ &\mathtt{write}(x;k) \mapsto k\ ()\} \\ c \stackrel{\text{def}}{=} &\mathbf{do}\ \mathit{name} \leftarrow \mathtt{read}()\ \mathbf{in} \\ &\mathbf{do}\ \_ \leftarrow \mathtt{write}(\mathit{name})\ \mathbf{in} \\ &\mathbf{return}\ \mathit{name} \end{aligned}$$

この定義のもとで，**with** h **handle** c のイベントトレースを見積もったエフェクト列は，$\mathtt{read}^{\checkmark};\mathtt{write}^{\checkmark}$ となる．✓ がついているエフェクトはハンドル済みのエフェクトを表している．c で発生したエフェクトがハンドラ h で捕捉されるためである．このエフェクト列を見ると，このプログラムが，どのようなエフェクトの発生順序になる可能性があるかを確かめることができる．

一方，ハンドラで包まない c のイベントトレースを見積もったエフェクト列は $\mathtt{read};\mathtt{write}$ となる．上述のプログラムとの違いは，c の外側のハンドラの有無である．ハンドラがない c のエフェクト列は，未ハンドルオペレーションの列となる．c を実際に評価した場合はエフェクト $\mathtt{read}$ が発生した地点で評価が止まってしまうため，実際のイベントトレースは ϵ となる．しかし，エフェクト列を構築する際には，川俣らの手法 [8] と同様に，ハンドルされないオペレーションは仮想的にハンドルされると見做す．この時，仮想的にハンドルされるとは，ただ継続を再開するハンドラによってハンドルされることを意味する．つまり，ハンドラがない場合にオペレーション呼び出し以降のイベントトレースを見積もらないのではなく，仮想

的にハンドルされたと見做して見積る．このように，未ハンドルオペレーションを含むエフェクト列を見ると，このプログラム中でハンドルされていないエフェクトを確認することができる．

5 見積りアルゴリズム

正しく型付けされたプログラムから，イベントトレースの見積りであるエフェクト列を生成するアルゴリズムについて述べる．また，本アルゴリズムの健全性についても議論する．

5.1 見積りアルゴリズム

エフェクト列の見積りアルゴリズムを図 8 に示す．また，図 9 はハンドラ部分の見積りアルゴリズムを示している．

エフェクト列の見積りアルゴリズムは，エフェクト環境 K，項，エフェクト列の 3 つ組で表される．エフェクト環境 K は，変数とそれに対応するエフェクト列のペアの集合である．

また，ハンドラ部分の見積りアルゴリズムはエフェクト集合 Σ，ハンドラ H，エフェクト列とリターンエフェクトのペア (δ_1, δ_2)，エフェクト状態 s の 4 つ組からなる．

値のエフェクト列は純粋 ϵ または，エフェクト関数 $a \to s$ である．コンピュテーションはエフェクト状態 s，ハンドラはハンドラ h^{ops} である．また，本アルゴリズム中で出てくる $\cup$ に関しては，レコードの拡張を意味する．

op 呼び出しを見積もる，(R-Op) について説明する．(E-HandlingInOp) からわかるように，$\mathtt{op}(v; y.c)$ の v はハンドラに渡す引数であり，y はハンドラが継続を再開するときにハンドラから渡される引数であり，c は継続が再開された後に行われる処理である．このコンピュテーションで発生するエフェクト列は，op と c で発生するエフェクト列 C_t を連結させた，$\mathtt{op}^i; C_t$ となる．これは，ハンドラがない状況では，op 呼び出しは継続を単に再開するだけの仮想的なハンドラに捕捉されると見做すためである．また，外側にハンドラが存在した時，詳細に見積もるための情報を Σ に記録しておく．記録する情報は，レコードでキーを $\mathtt{op}^i$ の肩の部分 i，キーに対応する中身にハンドラに渡す引数と，継続の情報を記録する．$\mathtt{op}^i$ の肩の部分 i は常にフレッシュなものになっている．

次に，**handler** を見積もる (R-Handler) について説明する．**handler** は return 節とハンドラが持つ各 op 節からなる．return 節は，**with** h **handle** c の c で return された値を受け取り c_r の評価を行う．各 op 節は，c で発生した op 呼び出しの時の引数と継続を受け取り c_{op} の評価を行う．h^{ops} はレコードであり，キーを return と各 op とし，キーに対応する中身をハンドラに渡す引数や継続などを一時的にエフェクト変数で置いた時のエフェクト変数と，評価を行うコンピュテーションのエフェクト状態をペアにしたものとする．

ハンドラ部分の見積りは，エフェクト列のパターンマッチによって処理を変える．エフェクト列が単体 (エフェクト列の末尾) の場合は，return 節の部分のエフェクト列を末尾に加える．エフェクト列が連結の場合に，先頭がハンドルできる op の場合は，ハンドラ h^{ops} の op 節の部分を取り出しエフェクト列を op の直後につける．ハンドルされた際の継続は Σ から取り出して利用する．

ハンドラ部分の見積りで出てくる関数 ρ, σ, ι はそれぞれ，return 節の見積り関数，op 節の見積り関数，レコードのフィールド参照関数である．

例えば，例 4.1 の条件のもとで，**with** h **handle** c に対して，本アルゴリズムを適用する．アルゴリズムを適用した結果のエフェクト列は $\mathtt{op}^{\checkmark}_{read}; \mathtt{op}^{\checkmark}_{write}$ となる．アルゴリズムは，(R-Handling) を適用し，h に対して (R-Handler)，c に対して (R-SeqOp) を適用する．h に対して (R-Handler) を適用した結果は，$h^{ops} = \{return \triangleright (\epsilon, A, \emptyset), \mathtt{read} \triangleright K_{read}[\epsilon/B_{read}], \mathtt{write} \triangleright K_{write}[\epsilon/B_{wdrite}]\}(ops = \{\ \mathtt{read}, \mathtt{write}\})$ となる．c に対して (R-SeqOp) を適用した結果は $(\mathtt{op}^0_{read}; \mathtt{op}^1_{write}, \epsilon, \Sigma)$ となる．尚，$\Sigma = \{0 : \{x \triangleright \epsilon, k \triangleright \epsilon \to (\mathtt{op}^1_{write}, \epsilon, \Sigma')\}\} \cup \Sigma'$ $(\Sigma' = \{1 : \{x \triangleright \epsilon, k \triangleright \epsilon \to (\epsilon, \epsilon, \emptyset)\}\})$．ここで得られたエフェクト列をハンドラ部分の詳細なエフェクト列の見積り $\blacktriangleright$ によって詳細なエフェクト列を算出する．$\Sigma, h^{ops}, ((\mathtt{op}^0_{read}; \mathtt{op}^1_{write}, \epsilon)) \blacktriangleright$

$$\frac{x \triangleright A \in K}{K \vdash x \triangleright A}\ (\text{R-Var}) \qquad \frac{}{K \vdash \mathbf{true} \triangleright \epsilon}\ (\text{R-True}) \qquad \frac{}{K \vdash \mathbf{false} \triangleright \epsilon}\ (\text{R-False})$$

$$\frac{K, x \triangleright a \vdash c \triangleright (A_t, A_r, \Sigma)}{K \vdash \mathbf{fun}\ x \mapsto c \triangleright a \to (A_t, A_r, \Sigma)}\ (\text{R-Fun}) \quad \frac{K \vdash v \triangleright A}{K \vdash \mathbf{return}\ v \triangleright (\epsilon, A, \phi)}\ (\text{R-Return})$$

$$\frac{K \vdash c_1 \triangleright (A_t, A_r, \Sigma_a) \qquad K, x \triangleright A_r \vdash c_2 \triangleright (B_t, B_r, \Sigma_b)}{K \vdash \mathbf{do}\ x \leftarrow c_1 \mathbf{in}\ c_2 \triangleright (A_t; B_t, B_r, \Sigma_a \cup \Sigma_b)}\ (\text{R-Seq})$$

$$\frac{K \vdash c_1 \triangleright (A_t, A_r, \Sigma_a) \qquad K \vdash c_2 \triangleright (B_t, B_r, \Sigma_b)}{K \vdash \mathbf{if}\ v\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \triangleright (A_t | B_t, A_r | B_r, \Sigma_a \cup \Sigma_b)}\ (\text{R-If})$$

$$\frac{K \vdash \mathtt{op}(v; y.\mathbf{do}\ x \leftarrow c_1\ \mathbf{in}\ c_2) \triangleright (A_t, A_r, \Sigma)}{K \vdash \mathbf{do}\ x \leftarrow \mathtt{op}(v; y.c_1)\ \mathbf{in}\ c_2 \triangleright (A_t, A_r, \Sigma)}\ (\text{R-SeqOp})$$

$$\frac{K \vdash v_1 \triangleright a \to (A_t, A_r, \Sigma) \qquad K \vdash v_2 \triangleright B}{K \vdash v_1\ v_2 \triangleright (A_t, A_r, \Sigma)[B/a]}\ (\text{R-App})$$

$$\frac{K \vdash v \triangleright A \qquad K, y \triangleright b \vdash c \triangleright (C_t, C_r, \Sigma)}{K \vdash \mathtt{op}(v; y.c) \triangleright (\mathtt{op}^i; C_t, C_r, \{i : \{x \triangleright A, k \triangleright b \to (C_t, C_r, \Sigma)\}\} \cup \Sigma))}\ (\text{R-Op})$$

$$\frac{K \vdash v \triangleright h^{ops} \qquad K \vdash c \triangleright (A_t, A_r, \Sigma_a) \qquad \Sigma_a, h^{ops}, (A_t, A_r) \blacktriangleright (B_t, B_r, \Sigma_b)}{K \vdash \mathbf{with}\ v\ \mathbf{handle}\ c \triangleright (B_t, B_r, \Sigma_b)}\ (\text{R-Handling})$$

$$\frac{\begin{array}{c}\forall \mathtt{op} \in ops[\ K, x \triangleright a_{op}, k \triangleright b_{op} \to (c_{opt}, c_{opr}, \emptyset) \vdash c_{op} \triangleright (C_{opt}, C_{opr}, \Sigma_{op})] \\ K, x \triangleright a_{ret} \vdash c_r \triangleright (C_{rett}, C_{retr}, \Sigma_{ret}) \\ h^{ops} = \{return : (a_{ret}, (C_t, C_r, \Sigma_{ret})), \bigsqcup_{\mathtt{op} \in ops} \mathtt{op} : ((a_{op}, b_{op}, c_{opt}, c_{opr}), (C_{opt}, C_{opr}, \Sigma_{op}))\}\end{array}}{K \vdash \mathbf{handler}\ \{\mathbf{return}\ x \mapsto c_r, \mathtt{op}_1(x; k) \mapsto c_{\mathtt{op}_1}, \ldots, \mathtt{op}_n(x; k) \mapsto c_{\mathtt{op}_n}\} \triangleright h^{ops}}\ (\text{R-Handler})$$

図 8　エフェクト列の見積り

$(\mathtt{op}^{\checkmark}_{read}; \mathtt{op}^{\checkmark}_{write}, \epsilon, \Sigma)$ となり，**with** h **handle** c に対して本アルゴリズムを適用した結果は $\mathtt{op}^{\checkmark}_{read}; \mathtt{op}^{\checkmark}_{write}$ となる．

5.2　アルゴリズムの健全性

本アルゴリズムで見積もられるエフェクト列は，実際に評価した時のイベントトレースを包含していることを示すことによって，本アルゴリズムの健全性を示すことができる．また，本アルゴリズムで導出された未ハンドルオペレーションの集合は，型システム (エフェクトシステム) の未ハンドルエフェクトの集合の部分集合になっていることを示すことで，エフェクトシステムで保証されている性質を本アルゴリズムが出力するエフェクト列が満たすことを示す．

尚，これらの定理の証明については，(E-SeqOp) と (E-HandlingOutOp) のルールを適用できない条件付きで，付録の補題を組み合わせることで証明ができると予測している．これらの完全な証明に関しては今後の課題とする．

定理 5.1 $c, \eta \rightsquigarrow c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C'_t, C'_r, \Sigma')$ ならば $\eta' \sqsubseteq \eta; \mathit{diff}(C_t, C'_t)$

補題 5.1 $c, \eta \rightsquigarrow^* c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C'_t, C'_r, \Sigma')$ ならば $C'_t \neq \mathtt{op}^{\checkmark}; (*)$

定理 5.2 $c, \eta \rightsquigarrow^* c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ ならば $\eta' \sqsubseteq \eta; C_t)$

定理 5.1 によって，ワンステップの評価でのエフェクト列が正しいことが保証される．また，定理 3.1(型安全性) から補題 5.1 が導かれる．補題 5.1 は評価が停止した時のエフェクト列にハンドル済みエフェク

$$\frac{(\delta, A', \Sigma_{ret}) = \rho(A, h^{ops}, \Sigma)}{\Sigma, h^{ops}, (\epsilon, A) \ \blacktriangleright \ (\epsilon; \delta, A', \Sigma \cup \Sigma_{ret})} \text{ (H-Epsilon)}$$

$$\frac{(\delta, A', \Sigma_{ret}) = \rho(A, h^{ops}, \Sigma)}{\Sigma, h^{ops}, (\mathtt{op}^{\checkmark}, A) \ \blacktriangleright \ (\mathtt{op}^{\checkmark}; \delta, A', \Sigma \cup \Sigma_{ret}))} \text{ (H-OpCheck)}$$

$$\frac{(\delta, A', \Sigma_{ret}) = \rho(A, h^{ops}, \Sigma) \qquad (\mathtt{op} \notin ops)}{\Sigma, h^{ops}, (\mathtt{op}, A) \ \blacktriangleright \ (\mathtt{op}; \delta, A', \Sigma \cup \Sigma_{ret})} \text{ (H-OpO)}$$

$$\frac{((c_{opt}, c_{opr}), (\delta, A', \Sigma_{op}), (C_t, C_r)) = \sigma(\mathtt{op}, h^{ops}, \Sigma) \qquad \Sigma, h^{ops}, (C_t, C_r) \ \blacktriangleright \ (\delta', A'', \Sigma') \qquad (\mathtt{op} \in ops)}{\Sigma, h^{ops}, (\mathtt{op}, A) \ \blacktriangleright \ (\mathtt{op}^{\checkmark}; \delta, A', \Sigma \cup \Sigma_{op})[(\delta''/c_{opt}, A''/c_{opr}]} \text{ (H-OpI)}$$

$$\frac{\Sigma, h^{ops}, (\delta, A) \ \blacktriangleright \ (\delta', A', \Sigma') \qquad (\mathtt{op} \notin ops)}{\Sigma, h^{ops}, (\mathtt{op}; \delta, A) \ \blacktriangleright \ (\mathtt{op}; \delta', A', \Sigma')} \text{ (H-SeqOpO)}$$

$$\frac{((c_{opt}, c_{opr}), (\delta', A', \Sigma_{op}), (C_t, C_r)) = \sigma(op, h^{ops}, \Sigma) \qquad \Sigma, h^{ops}, (C_t, C_r) \ \blacktriangleright \ (\delta'', A'', \Sigma') \qquad (\mathtt{op} \in ops)}{\Sigma, h^{ops}, (\mathtt{op}; \delta, A) \ \blacktriangleright \ (\mathtt{op}^{\checkmark}; \delta', A', \Sigma \cup \Sigma_{op})[\delta''/c_{opt}, A''/c_{opr}]} \text{ (H-SeqOpI)}$$

$$\frac{\Sigma, h^{ops}, (\delta_1, A_1) \ \blacktriangleright \ (\delta_1', A_1', \Sigma_1') \qquad \Sigma, h^{ops}, (\delta_2, A_2) \ \blacktriangleright \ (\delta_2', A_2', \Sigma_2')}{\Sigma, h^{ops}, (\delta_1 | \delta_2, A_1 | A_2) \ \blacktriangleright \ (\delta_1' | \delta_2', A_1' | A_2', \Sigma_1' \cup \Sigma_2')} \text{ (H-Nondterministic)}$$

$$\frac{\Sigma, h^{ops}, (\delta_1, \bot) \ \blacktriangleright \ (\delta_1', \bot, \Sigma_1') \qquad \Sigma, h^{ops}, (\delta_2, A) \ \blacktriangleright \ (\delta_2', A', \Sigma_2') \qquad (\delta_1 \neq \mathtt{op})}{\Sigma, h^{ops}, (\delta_1; \delta_2, A) \ \blacktriangleright \ (\delta_1'; \delta_2', A', \Sigma_1' \cup \Sigma_2')} \text{ (H-Seq)}$$

$$\begin{aligned}
\rho(A, h^{ops}, \Sigma) = & \text{if } A = \bot \text{ then } \bot \\
& \text{else let } (a_{ret}, (C_{rett}, C_{retr}, \Sigma_{ret})) = \iota(\text{return}, h^{ops}) \text{ in} \\
& (C_{rett}, C_{retr}, \Sigma_{ret})[A/a_{ret}]
\end{aligned}$$

$$\begin{aligned}
\sigma(\mathtt{op}^i, h^{ops}, \Sigma) = & \text{let } ((a_{op}, b_{op}, c_{opt}, c_{opr}), (C_{opt}, C_{opr}, \Sigma_{op})) = \iota(\mathtt{op}, h^{ops}) \text{ in} \\
& \text{let } \{x \triangleright A, k \triangleright b \to (C_t, C_r, \Sigma_{op}')\} = \iota(i, \Sigma) \text{ in} \\
& ((c_{opt}, c_{opr}), (C_{opt}, C_{opr}, \Sigma_{op})[A/a_{op}], (C_t, C_r)[b_{op}/b])
\end{aligned}$$

$$\begin{aligned}
\iota(\Gamma\Gamma, S) = & \text{let } \{key : body\} \cup S' = S' \text{ in} \\
& body
\end{aligned}$$

図 9　ハンドラ部分の見積り

トから始まるエフェクト列が現れないことが保証される．そのため，定理 5.1 と補題 5.1 から定理 5.2 が示され，健全性が証明される．

定理 5.3 $\vdash c : A!\Delta$ かつ $K \vdash c \triangleright (A_t, A_r, \Sigma)$ ならば，$unHandleOp(A_t) \subseteq \Delta$

また，型情報に含まれるエフェクトの集合 K に対して，エフェクト列の中の未ハンドルエフェクトの集合が K の部分集合になっていることを示す．なお，関数 *unHandleOp* はエフェクト列から，そこに含まれる未ハンドルエフェクトの集合を取り出す関数である．これは，発生し得ないエフェクトが記録されて

いることはないことを示している．

5.3 アルゴリズムの実装と評価

本アルゴリズムの実装は OCaml にて行なっている．また，対象言語のインタープリタに関しても実装を行なっており，いずれも GitHub 上で公開している．本アルゴリズムを実際に実行した例を 2 節の例を用いて示す．

$$
\begin{aligned}
h' \stackrel{\text{def}}{=} \mathbf{handler}\{&\mathbf{return}\ x \mapsto \mathbf{return}\ x,\\
&\texttt{read}(\_; k) \mapsto\\
&\quad \mathbf{do}\ x \leftarrow k\ \texttt{"Bob"}\ \mathbf{in}\\
&\quad k\ x\}
\end{aligned}
$$

$$
\begin{aligned}
c \stackrel{\text{def}}{=} &\mathbf{do}\ name1 \leftarrow \texttt{read}()\ \mathbf{in}\\
&\mathbf{do}\ name2 \leftarrow \texttt{read}()\ \mathbf{in}\\
&\mathbf{return}\ name1\hat{}name2
\end{aligned}
$$

本アルゴリズムで **with** h' **handle** c のエフェクト列を見積ると $\texttt{read}^{\checkmark};\texttt{read}^{\checkmark};\texttt{read}^{\checkmark}$ となる．**with** h' **handle** c, ϵ から始め，評価が停止した時の実際のイベントトレースも $\texttt{read}^{\checkmark};\texttt{read}^{\checkmark};\texttt{read}^{\checkmark}$ となり，見積もったエフェクト列と一致する．

他にも以下のような例を考える．

$$
\begin{aligned}
c' \stackrel{\text{def}}{=} &\mathbf{if}\ v\ \mathbf{then}\\
&\quad \mathbf{do}\ name1 \leftarrow \texttt{read}()\ \mathbf{in}\\
&\quad \mathbf{do}\ name2 \leftarrow \texttt{read}()\ \mathbf{in}\\
&\quad \mathbf{return}\ name1\hat{}name2\\
&\mathbf{else}\\
&\quad \mathbf{do}\ \_ \leftarrow \texttt{write}(\texttt{"Bob"})\ \mathbf{in}\\
&\quad \mathbf{do}\ name \leftarrow \texttt{read}()\ \mathbf{in}\\
&\quad \mathbf{return}\ name
\end{aligned}
$$

本アルゴリズムで **with** h' **handle** c' のエフェクト列を見積ると $\texttt{read}^{\checkmark};\texttt{read}^{\checkmark};\texttt{read}^{\checkmark}|\texttt{write};\texttt{read}^{\checkmark}$ となる．**with** h' **handle** c', ϵ から始め，評価が停止した時の実際のイベントトレースは，v が **true** の時，$\texttt{read}^{\checkmark};\texttt{read}^{\checkmark};\texttt{read}^{\checkmark}$ となり，v が **false** の時，write の部分で評価が停止するが，write がただ継続を再開すると考えると，$\texttt{write};\texttt{read}^{\checkmark}$ となり，見積もったエフェクト列と一致する．

川俣らの手法 [8] で，同等なプログラムに対してイベントトレースを見積る [†2] と，川俣らの手法 [8] 特有の $\texttt{op}_{dum}$ を除いて，12 パターン出現することとなり，本アルゴリズムよりも false positive が多くなる．

6 関連研究

本論文では，Pretnar の代数的エフェクトの体系 [6] をもとにしている．これは，実行時に発生しうるエフェクトの情報を型情報に集合として定義したものである．型情報に含まれた集合は未ハンドルオペレーションで評価が停止した際に，その集合にその未ハンドルオペレーションが必ず含まれていることを型安全性として保証している．

川俣ら [8] の手法も同じく大いに参考にしたものである．この手法は本論文と同じく Pretnar の代数的エフェクトの体系 [6] をもとにしているが，ディープハンドラではなくシャローハンドラを採用しており，再帰関数が扱える点が異なる．また，イベントトレースを用いた評価規則も同じである．この手法では，型情報にエフェクトの時間的情報をトレースエフェクトとして埋め込んでいる．エフェクトの時間的な情報をトレースエフェクトの集合で表している．本論文の手法では，エフェクトをグラフとして表しているため，その部分が異なる．また，ハンドラにエフェクトがハンドルされた際の false positive が多くなるのに対して，本論文の手法ではハンドラ部分を詳細に見積もることで false positive を少なくしている．

他に継続などを用いてコントロールフローを変えるものに対して時間的な情報を扱う手法として Gordon の sequential effects [2] や，Sekiyama らの AEW(Answer Effect Modification) に対応した時間的情報を扱えるエフェクトシステム [7] がある．前者は，タグ付き限定継続演算子を持つ体系におけるエフェクトの順序の見積りを行っている．後者は，shift0/reset0 の体系において依存型を用いて flow sensitive なエフェクトの順序の見積りを行っている．また，エフェクトの順序を無限と有限を分けて管理している．

†2 川俣らの手法 [8] はディープハンドラを扱えないので，ハンドラ h' を 3 重にネストさせるようにエンコードして見積もっている．

7 まとめと今後の課題

本論文では，代数的エフェクトハンドラを持つ言語のプログラムに対して，エフェクトの発生の時間的な見積りを静的に行うアルゴリズムを提案した．ハンドラ部分の見積りをより正確に行うことによって，既存手法よりも，false positive の少ない見積りを行うことを可能にした．また，提案したアルゴリズムの実装は GitHub[†3] にて公開している．

今後の課題としては，本論文で示した定理の証明がある．また，対象言語をディープハンドラだけでなくシャローハンドラも含む体系に対しても見積りを行えるアルゴリズムを構築すること，再帰を含む体系に対しても見積りを行えるアルゴリズムを構築することがある．特に，再帰を含む体型に関しては，無限のエフェクト列をどのように表すかという問題がある．無限の扱いに関しては，有限と無限を分けて管理する方法や，エフェクトの列に対して再帰的な構文を追加する方法などがある．

また，このアルゴリズムを用いた代数的エフェクトハンドラを持つ体系に対してのプログラミング支援の応用に関しても今後の課題である．

参 考 文 献

[1] Bauer, A. and Pretnar, M.: An effect system for algebraic effects and handlers, *Logical methods in computer science*, Vol. 10(2014).

[2] Gordon, C. S.: Lifting sequential effects to control operators, *34th European Conference on Object-Oriented Programming (ECOOP 2020)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[3] Hillerström, D. and Lindley, S.: Shallow effect handlers, *Programming Languages and Systems: 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2–6, 2018, Proceedings 16*, Springer, 2018, pp. 415–435.

[4] Plotkin, G. and Power, J.: Algebraic operations and generic effects, *Applied categorical structures*, Vol. 11(2003), pp. 69–94.

[5] Plotkin, G. and Pretnar, M.: Handlers of algebraic effects, *European Symposium on Programming*, Springer, 2009, pp. 80–94.

[6] Pretnar, M.: An introduction to algebraic effects and handlers. invited tutorial paper, *Electronic notes in theoretical computer science*, Vol. 319(2015), pp. 19–35.

[7] Sekiyama, T. and Unno, H.: Temporal Verification with Answer-Effect Modification: Dependent Temporal Type-and-Effect System with Delimited Continuations, *Proceedings of the ACM on Programming Languages*, Vol. 7, No. POPL(2023), pp. 2079–2110.

[8] 川俣楓河, 寺内多智弘: 代数的エフェクトハンドラを持つ言語のためのトレースエフェクト, コンピュータ ソフトウェア, Vol. 40, No. 2(2023), pp. 19–48.

†3 `https://github.com/zaki5m/effect_analyzer`

付録

定理 5.1 $c, \eta \rightsquigarrow c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C'_t, C'_r, \Sigma')$ ならば $\eta' \sqsubseteq \eta; \mathit{diff}(C_t, C'_t)$

証明．評価規則の構造に関する帰納法．

(E-SeqRet の時)

R-Seq より，以下が成り立つ．

1. $c_1 = \mathbf{return}\ v$
2. $K \vdash v \triangleright A_r$
3. $K \vdash c_1 \triangleright (\epsilon, A_r, \Sigma_a)$
4. $K, x \triangleright A_r \vdash c_2 \triangleright (B_t, B_r, \Sigma_b)$
5. $K \vdash \mathbf{do}\ x \leftarrow c_1 \mathbf{in} c_2 \triangleright (\epsilon; B_t, B_r, \Sigma_a \cup \Sigma_b)$

また，補題 5.2 より $K \vdash c_2[v/x] \triangleright B_t$ である．

さらに，$\mathit{diff}(\epsilon; B_t, B_t) = \epsilon$ である．よって，

$\eta \sqsubseteq \eta; \epsilon$ となり成り立つ．

(E-App)(E-IfTrue)(E-IfFalse) も同様に成り立つ．

(E-Seq の時)

R-Seq より，以下が成り立つ．

1. $K \vdash c_1 \triangleright (A_t, A_r, \Sigma_a)$
2. $K, x \triangleright A_r \vdash c_2 \triangleright (B_t, B_r, \Sigma_b)$
3. $K \vdash \mathbf{do}\ x \leftarrow c_1 \mathbf{in} c_2 \triangleright (A_t; B_t, B_r, \Sigma_a \cup \Sigma_b)$
1. $K \vdash c'_1 \triangleright (A'_t, A'_r, \Sigma'_a)$
2. $K, x \triangleright A'_r \vdash c_2 \triangleright (B'_t, B'_r, \Sigma_b)$
3. $K \vdash \mathbf{do}\ x \leftarrow c_1 \mathbf{in} c_2 \triangleright (A'_t; B_t, B_r, \Sigma'_a \cup \Sigma_b)$

補題 5.5，補題 5.6 より $\mathit{diff}(B_t, B'_t) = \epsilon$ である．また帰納法の仮定より，$\eta' \sqsubseteq \eta; \mathit{diff}(A_t, A'_t)$ が成り立つ．よって，$\eta' \sqsubseteq \eta; \mathit{diff}(A_t; B_t, A'_t; B'_t)$ が成り立つ．

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{}{\Gamma \vdash \mathbf{true} : \mathsf{bool}} \qquad \frac{}{\Gamma \vdash \mathbf{false} : \mathsf{bool}} \qquad \frac{\Gamma, x : A \vdash c : \underline{C}}{\Gamma \vdash \mathbf{fun}\ x \mapsto c : A \to \underline{C}}$$

$$\frac{\Big[(\mathsf{op}_i : A_i \to B_i) \in \Sigma \qquad \Gamma, x : A_i, k : B_i \to B\,!\,\Delta' \vdash c_i : B\,!\,\Delta'\Big]_{1 \le i \le n} \qquad \Delta \setminus \{\mathsf{op}_i\}_{1 \le i \le n} \subseteq \Delta'}{\Gamma \vdash \mathbf{handler}\ \{\mathbf{return}\ x \mapsto c_r, \mathsf{op}_1(x; k) \mapsto c_1, \ldots, \mathsf{op}_n(x; k) \mapsto c_n\} : A\,!\,\Delta \Rightarrow B\,!\,\Delta'}$$

$$\frac{\Gamma \vdash v : A}{\Gamma \vdash \mathbf{return}\ v : A\,!\,\Delta} \qquad \frac{(\mathsf{op} : A_{\mathsf{op}} \to B_{\mathsf{op}}) \in \Sigma \qquad \Gamma \vdash v : A_{\mathsf{op}} \qquad \Gamma, y : B_{\mathsf{op}} \vdash c : A\,!\,\Delta \qquad \mathsf{op} \in \Delta}{\Gamma \vdash \mathsf{op}(v; y.\, c) : A\,!\,\Delta}$$

$$\frac{\Gamma \vdash c_1 : A\,!\,\Delta \qquad \Gamma, x : A \vdash c_2 : B\,!\,\Delta}{\Gamma \vdash \mathbf{do}\ x \leftarrow c_1\ \mathbf{in}\ c_2 : B\,!\,\Delta} \qquad \frac{\Gamma \vdash v_1 : A \to \underline{C} \qquad \Gamma \vdash v_2 : A}{\Gamma \vdash v_1\, v_2 : \underline{C}}$$

$$\frac{\Gamma \vdash v : \mathsf{bool} \qquad \Gamma \vdash c_1 : \underline{C} \qquad \Gamma \vdash c_2 : \underline{C}}{\Gamma \vdash \mathbf{if}\ v\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 : \underline{C}} \qquad \frac{\Gamma \vdash v : \underline{C} \Rightarrow \underline{D} \qquad \Gamma \vdash c : \underline{C}}{\Gamma \vdash \mathbf{with}\ v\ \mathbf{handle}\ c : \underline{D}}$$

図 10　型システム ([6] の Fig.6)

(E-Handling) も同様に成り立つ.

(E-HandlingRet の時)

R-Handling より，以下が成り立つ.

1. $c = \mathbf{return}\ v$
2. $K \vdash v \triangleright A_r$
3. $K \vdash c \triangleright (\epsilon, A_r, \Sigma_a)$
4. $\Sigma_a, H, (\epsilon, A_r) \blacktriangleright (\epsilon; C_{rett}, C_{retr}, \Sigma_a \cup \Sigma_r et)[A_r/A]$
5. $K \vdash \mathbf{with}\ h\ \mathbf{handle}\ c \triangleright (C_{rett}, C_{retr}, \Sigma_a \cup \Sigma_r et)[A_r/A]$

また, $K \vdash c_r[v/x] \triangleright (C_{rett}, C_{retr}, \Sigma_r et)[A_r/A]$ である．さらに, $diff\,(\epsilon; C_{rett}[A_r/A], C_{rett}[A_r/A]) = \epsilon$ である．よって，$\eta \sqsubseteq \eta; \epsilon$ となり成り立つ.

(E-HandlingInOp の時)

R-Handling より，以下が成り立つ.

1. $c = op(v; y.c')$
2. $K \vdash v \triangleright A$
3. $K \vdash c' \triangleright (C'_t, C'_r, \Sigma_{c'})$
4. $K, y \triangleright B \vdash c \triangleright (C'_t; , C'_r, \Sigma_{c'})$
5. $\Sigma_c = \{i : \{x \triangleright A, k \triangleright B \to (C'_t; , C'_r, \Sigma_{c'})\}\} \cup \Sigma_{c'}$
6. $K \vdash c \triangleright (op^i; C'_t; , C'_r, \Sigma_c)$
7. $\Sigma_{c'}, H, (C'_t, C'_r) \blacktriangleright (\delta'', A'', \Sigma'')$
8. $\Sigma_c, H, (op^i; C'_t; , C'_r) \blacktriangleright (op^{\checkmark}; C_{opt}, C_{opr}, \Sigma_c \cup \Sigma_{op})[A/A_{op}, (\delta'', A'', \Sigma'')/K_{op}]$
9. $K \vdash \mathbf{with}\ h\ \mathbf{handle}\ c \triangleright (op^{\checkmark}; C_{opt}, C_{opr}, \Sigma_a \cup \Sigma_{op})[A_r/A, B_{op}/B, (\delta'', A'', \Sigma'')/K_{op}]$

また, $K \vdash c_j[v/x, \mathbf{fun}\ y \mapsto \mathbf{handle}\ h\ \mathbf{with}\ c] \triangleright (C_{opt}, C_{opr}, \Sigma_{ret})[A_r/A, B_{op}/B, (\delta'', A'', \Sigma'')/K_{op}]$ である．さらに，$diff\,(op^{\checkmark}; C_{opt}, C_{opt}) = op^{\checkmark}$ である．よって，$\eta; op^{\checkmark} \sqsubseteq \eta; op^{\checkmark}$ となり成り立つ.

□

定理 5.2 $\vdash c : A!\Delta$ かつ $K \vdash c \triangleright (A_t, A_r, \Sigma)$ ならば, $unHandleOp(A_t) \subseteq \Delta$

証明. 型の構造に関する帰納法.

($c = \mathbf{return}\ v$ の時)

R-Return より，$K \vdash \mathbf{return}\ v \triangleright (\epsilon, A, \Sigma)$. $unHandleOp(\epsilon) = \emptyset$ である．$\emptyset \subseteq \Delta$ よって満たす.

($c = op(v; y.c')$ の時)

R-Op より，$K \vdash op(v; y.c') \triangleright (op; C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C_t, C_r, \Sigma)$ である．仮定より，$\Gamma \vdash c' : A!\Delta$ かつ $op \in \Delta$ かつ

$unHandleOp(C_t) \subseteq \Delta$ である．なので，$unHandleOp(op; C_t) \subseteq \Delta$ よって満たす．

($c = \mathbf{handle}\ h\ \mathbf{with}\ c'$) の時)

R-Handling より，$K \vdash c' \triangleright (C_t', C_r', \Sigma)$ かつ $\Gamma \vdash c' \triangleright A!\Delta$ かつ $unHandleOp(C_t') \subseteq \Delta$ である．なので，$unHandleOp(op; C_t) \subseteq \Delta$ よって満たす．

□

補題 5.1 $c, \eta \rightsquigarrow^* c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C_t', C_r', \Sigma')$ ならば $C_t' \neq op^{\checkmark}; (*)$

証明. 型安全性より，c の評価が停止するとき $c = \mathbf{return}\ v$ または，$c = op(v; y.c')$ のいずれかである．($c = \mathbf{return}\ v$ のとき)

R-Return より，$K \vdash \mathbf{return}\ v \triangleright (\epsilon, A, \Sigma)$

$\epsilon \neq op^{\checkmark}; (*)$ である．よって成り立つ．

($c = op(v; y.c')$ のとき)

$K \vdash op(v; y.c') \triangleright (op; A_t, A_r, \Sigma)$

$op; A_t \neq op^{\checkmark}; (*)$ である．よって成り立つ． □

補題 5.2

- $K, x \triangleright A \vdash u \triangleright B$ かつ $K \vdash v \triangleright A$ ならば $K \vdash u[v/x] \triangleright B$
- $K, x \triangleright A \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash v \triangleright A$ ならば $K \vdash c[v/x] \triangleright (C_t, C_r, \Sigma)$

証明. アルゴリズムの構造に関する帰納法. □

補題 5.3 $K, x \triangleright A \vdash c : (C_t, C_r, \Sigma)$ かつ $K, x \triangleright A' \vdash c : (C_t, C_r, \Sigma)$ かつ $A' \sqsubseteq^{\spadesuit} A$ ならば $C_r' \sqsubseteq^{\spadesuit} C_r$

未解決

補題 5.4 $c, \eta \rightsquigarrow c', \eta'$ かつ $K \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K \vdash c' \triangleright (C_t', C_r', \Sigma')$ ならば $C_r' \sqsubseteq^{\spadesuit} C_r$ である．

証明. 補題 5.3 を用いて，評価規則の構造に関する帰納法. □

補題 5.5 $K, x \triangleright A \vdash c \triangleright (C_t, C_r, \Sigma)$ かつ $K, x \triangleright A' \vdash c \triangleright (C_t', C_r', \Sigma')$ かつ $A' \sqsubseteq^{\spadesuit} A$ ならば $C_t' \sqsubseteq^{\spadesuit} C_t$ である．

証明. アルゴリズムの構造に関する帰納法. □

補題 5.6 $\Sigma, h^{ops}, (\delta, A) \blacktriangleright (C_t, C_r, \Sigma_a)$ かつ $\Sigma', h^{ops}, (\delta', A') \blacktriangleright (C_t', C_r', \Sigma_a')$ かつ $\delta' \sqsubseteq^{\spadesuit} \delta$ かつ $A' \sqsubseteq^{\spadesuit} A$ ならば，$C_t' \sqsubseteq^{\spadesuit} C_t$ かつ $C_r' \sqsubseteq^{\spadesuit} C_r$ である．

証明. $\delta' \sqsubseteq^{\spadesuit} \delta$，$A' \sqsubseteq^{\spadesuit} A$ に関する構造帰納法 □