Online Type-Directed Partial Evaluation for Dynamically-Typed Languages

Eijiro Sumii and Naoki Kobayashi

University of Tokyo

Partial Evaluation

p: プログラム, **s**: 入力の一部

$$\textit{peval}(\mathbf{p}, \mathbf{s}) = \mathbf{p}_\mathbf{s}$$

d: 残りの入力

$$\textit{eval}(\mathbf{p}, (\mathbf{s}, \mathbf{d})) = \textit{eval}(\mathbf{p}_\mathbf{s}, \mathbf{d})$$

Type-Directed PE

- 古典的な PE (Syntax-Directed PE)

p と s の式 (expression)

↓ ***peval***

p_s の式

- Type-Directed PE [Danvy 96]

p と s の式

↓ ***eval*** + type inference

λd. p(s, d) の値 (value) **と型**

↓ ***reify***

p_s の式

Type-Directed PE

- 古典的な PE (Syntax-Directed PE)

素朴な方法だと
interpretive overhead
が大きく遅い

p と s の式 (expression)
↓ ***peval***
p_s の式

- Type-Directed PE [Danvy 96]

どちらも直接の
実行なので速い

p と s の式
↓ ***eval*** + type inference
λd. p(s, d) の値 (value) **と型**
↓ ***reify***
p_s の式

従来の TDPE の問題点

- 適用できる言語 ($\lambda^{\rightarrow,\times,+}$) が限定
  - ✗ 再帰型や動的型を持つ言語
- 与えられた型に盲従
  - ⇨ PE の処理と結果の両方が冗長
- SDPE との関係が不明確
  - ⇨ 従来の技術の応用が困難

我々の貢献

- 様々な関数型言語に適用でき、
  冗長性のない TDPE を提案
- Higher-Order Abstract Syntax を用いた
  ある単純な SDPE から、deforestation で
  我々の TDPE を導出

  ⇒ より複雑な SDPE からは
  より高度な TDPE が得られる

発表の概要

- Introduction
- Danvy の TDPE
- 我々の TDPE
- 比較実験
- SDPE との関係
- 関連研究
- 結論

Danvy の TDPE: Reification

reify = 「eval の逆」

値 **v** とその型 τ $\mapsto$ ***eval*(e) = v** なる式 **e** の強正規形

$\Uparrow$

関数の場合が問題（他の場合は自明）

reify $(\rho \to \sigma)$ **v** :=

(1) 引数の「代表」となる ρ 型の値を生成

(2) 生成した値に **v** を適用

(3) 適用した結果である σ 型の値を reify

Danvy の TDPE: Reification の例 (1)

例 : ***reify*** $(\alpha \to \beta \to \alpha)$ **v**

関数適用

where $\mathbf{v} = \lambda \mathbf{a}.\ \lambda \mathbf{b}.\ ((\lambda \mathbf{c}.\ \mathbf{c})\ @\ \mathbf{a})$

$\mathbf{v}\ @\ \underline{\mathbf{x}}\ =\ \lambda \mathbf{b}.\ ((\lambda \mathbf{c}.\ \mathbf{c})\ @\ \underline{\mathbf{x}})$

シンボル

$\mathbf{v}\ @\ \underline{\mathbf{x}}\ @\ \underline{\mathbf{y}}\ =\ (\lambda \mathbf{c}.\ \mathbf{c})\ @\ \underline{\mathbf{x}}$

$= \underline{\mathbf{x}}$

λ 抽象の
syntax constructor

よって

$$\textit{reify}\ (\alpha \to \beta \to \alpha)\ \mathbf{v}\ =\ \underline{\lambda \mathbf{x}}.\ \underline{\lambda \mathbf{y}}.\ \underline{\mathbf{x}}$$

Danvy の TDPE: Reification の例 (2)

例 : ***reify*** **((int → α) → α) v**

where **v = λa. (a @ (1 + 2))**

v @ x = x @ (1 + 2) ⇒ Error!

そこで **x** を **λb. x @ b** と η 展開して

関数適用の syntax constructor

v @ (λb. x @ b) = (λb. x @ b) @ (1 + 2)
= x @ 3

よって

reify **((int → α) → α) v = λx. x @ 3**

Danvy の TDPE: Reification の一般的な定義

reify : *type* → *value* → *exp*

reify α v = v

reify (α → β) v = λx. (*reify* β (v @ (*reflect* α x)))

(where x *is fresh)*

reflect : *type* → *exp* → *value*

reflect α e = e

reflect (α → β) e = λa. (*reflect* β (e @ (*reify* α a)))

Danvy の TDPE: 問題点

- 適用できる言語が限定されている

  整数型や再帰型には「代表」となる値がない

  – ***reflect int*** **x = ???**

  – ***reflect*** **(α *list*)** **y = ???**

- 特化時も実行時も効率が悪い

  型が冗長だと、無駄な reflection が発生して
  結果も冗長になる

  reify **((α → α) → α → α) (λa. a) = λx. λy. (x @ y)**

我々の TDPE: Basic Idea (1)

Reflection を行わない！

$$\mathit{reify}(\lambda x.\ t) = \underline{\lambda y}.\ \mathit{reify}((\lambda x.\ t)\ @\ \underline{y})$$

(where $\underline{y}$ is fresh)

$$\mathit{reify}(t_1\ \underline{@}\ t_2) = \mathit{reify}(t_1)\ \underline{@}\ \mathit{reify}(t_2)$$

$$\mathit{reify}(\underline{y}) = \underline{y}$$

我々の TDPE: Basic Idea (2)

primitive な value destructor たちを、

「式に対しては式を返す」

ように拡張する。

例: 関数適用

$$\mathbf{e_1 \mathbin{@'} e_2 = e_1 \mathbin{@} e_2}$$

(e_1 が関数の場合)

$$\mathbf{e_1 \mathbin{@'} e_2 = e_1 \mathbin{@} (reify\ e_2)}$$

(e_1 が式の場合)

我々の TDPE: 全体像

p: プログラム, **s**: 入力の一部, **d**: 残りの入力

p と s の式

↓ ***eval′***

λd. p(s, d) の値

↓ ***reify***

p_s の式

我々の TDPE: 全体像

p: プログラム, **s**: 入力の一部, **d**: 残りの入力

p と **s** の式

↓ ***preprocess***

λd. p(s, d) の式′

↓ ***eval***

λd. p(s, d) の値′

↓ ***reify***

$\mathbf{p_s}$ の式

- @ や + などを @′ や +′ などでおきかえる
- 静的に型つけされた言語では tag をつける

我々の TDPE: 例

元のプログラム: **f @ (n + (s + 2))**

静的な入力: **s = 1**

動的な入力: **f と n**

⇓

λf. λn. f @ (n + (1 + 2)) を強正規化したい

⇓

λf. λn. f @' (n +' (1 +' 2)) を ***eval*** して ***reify*** する

我々の TDPE: 例

***reify*(λf. λn. f @′ (n +′ (1 +′ 2)))**

我々の TDPE: 例

$$\mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))$$

$$= \lambda x.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ x)$$

（関数に対する *reify* の定義）

我々の TDPE: 例

$$\mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n +' (1 +' 2)))$$

$$= \underline{\lambda x}.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n +' (1 +' 2)))\ @\ \underline{x})$$

$$= \underline{\lambda x}.\ \mathit{reify}(\lambda n.\ \underline{x}\ @'\ (n +' (1 +' 2)))$$

（β 簡約）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \lambda x.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ x) \\
= {} & \lambda x.\ \mathit{reify}(\lambda n.\ x\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \lambda x.\ \lambda y.\ \mathit{reify}((\lambda n.\ x\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ y)
\end{aligned}
$$

（関数に対する *reify* の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \lambda \underline{x}.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{x}) \\
= {} & \lambda \underline{x}.\ \mathit{reify}(\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \lambda \underline{x}.\ \lambda \underline{y}.\ \mathit{reify}((\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{y}) \\
= {} & \lambda \underline{x}.\ \lambda \underline{y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ (1\ +'\ 2)))
\end{aligned}
$$

（β 簡約）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{x}) \\
= {} & \underline{\lambda x}.\ \mathit{reify}(\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}((\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{y}) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ 3))
\end{aligned}
$$

（整数に対する $+'$ の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{x}) \\
= {} & \underline{\lambda x}.\ \mathit{reify}(\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}((\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{y}) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ 3)) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +\ 3))
\end{aligned}
$$

（式に対する +′ の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \mathit{reify}((\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{x}) \\
= {} & \underline{\lambda x}.\ \mathit{reify}(\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}((\lambda n.\ \underline{x}\ @'\ (n\ +'\ (1\ +'\ 2)))\ @\ \underline{y}) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ (1\ +'\ 2))) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +'\ 3)) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ @'\ (\underline{y}\ +\ 3)) \\
= {} & \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ \underline{@}\ (\underline{y}\ +\ 3))
\end{aligned}
$$

（式に対する @′ の定義）

我々の TDPE: 例

$$\mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))$$

$$= \ldots$$

$$= \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(x\ \underline{@}\ (y\ \underline{+}\ 3))$$

$$= \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(x)\ \underline{@}\ \mathit{reify}(y\ \underline{+}\ 3)$$

（$\underline{@}$ に対する *reify* の定義）

我々の TDPE: 例

$$\mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2)))$$

$$= \ldots$$

$$= \lambda x.\ \lambda y.\ \mathit{reify}(x\ @\ (y + 3))$$

$$= \lambda x.\ \lambda y.\ \mathit{reify}(x)\ @\ \mathit{reify}(y + 3)$$

$$= \lambda x.\ \lambda y.\ x\ @\ \mathit{reify}(y + 3)$$

（シンボルに対する *reify* の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
&= \ldots \\
&= \lambda \underline{x}.\ \lambda \underline{y}.\ \mathit{reify}(\underline{x}\ \underline{@}\ (\underline{y}\ \underline{+}\ 3)) \\
&= \lambda \underline{x}.\ \lambda \underline{y}.\ \mathit{reify}(\underline{x})\ \underline{@}\ \mathit{reify}(\underline{y}\ \underline{+}\ 3) \\
&= \lambda \underline{x}.\ \lambda \underline{y}.\ \underline{x}\ \underline{@}\ \mathit{reify}(\underline{y}\ \underline{+}\ 3) \\
&= \lambda \underline{x}.\ \lambda \underline{y}.\ \underline{x}\ \underline{@}\ (\mathit{reify}(\underline{y})\ \underline{+}\ \mathit{reify}(3))
\end{aligned}
$$

（$\underline{+}$ に対する *reify* の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
&= \ldots \\
&= \lambda x.\ \lambda y.\ \mathit{reify}(x\ @\ (y + 3)) \\
&= \lambda x.\ \lambda y.\ \mathit{reify}(x)\ @\ \mathit{reify}(y + 3) \\
&= \lambda x.\ \lambda y.\ x\ @\ \mathit{reify}(y + 3) \\
&= \lambda x.\ \lambda y.\ x\ @\ (\mathit{reify}(y) + \mathit{reify}(3)) \\
&= \lambda x.\ \lambda y.\ x\ @\ (y + \mathit{reify}(3))
\end{aligned}
$$

（シンボルに対する *reify* の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
&= \ldots \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x}\ \underline{@}\ (\underline{y}\ \underline{+}\ 3)) \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \mathit{reify}(\underline{x})\ \underline{@}\ \mathit{reify}(\underline{y}\ \underline{+}\ 3) \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \underline{x}\ \underline{@}\ \mathit{reify}(\underline{y}\ \underline{+}\ 3) \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \underline{x}\ \underline{@}\ (\mathit{reify}(\underline{y})\ \underline{+}\ \mathit{reify}(3)) \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \underline{x}\ \underline{@}\ (\underline{y}\ \underline{+}\ \mathit{reify}(3)) \\
&= \underline{\lambda x}.\ \underline{\lambda y}.\ \underline{x}\ \underline{@}\ (\underline{y}\ \underline{+}\ 3)
\end{aligned}
$$

（整数に対する *reify* の定義）

我々の TDPE: 例

$$
\begin{aligned}
& \mathit{reify}(\lambda f.\ \lambda n.\ f\ @'\ (n\ +'\ (1\ +'\ 2))) \\
&= \ldots \\
&= \lambda x.\ \lambda y.\ \mathit{reify}(x\ @\ (y + 3)) \\
&= \lambda x.\ \lambda y.\ \mathit{reify}(x)\ @\ \mathit{reify}(y + 3) \\
&= \lambda x.\ \lambda y.\ x\ @\ \mathit{reify}(y + 3) \\
&= \lambda x.\ \lambda y.\ x\ @\ (\mathit{reify}(y) + \mathit{reify}(3)) \\
&= \lambda x.\ \lambda y.\ x\ @\ (y + \mathit{reify}(3)) \\
&= \lambda x.\ \lambda y.\ x\ @\ (y + 3)
\end{aligned}
$$

我々の TDPE: 効率の比較

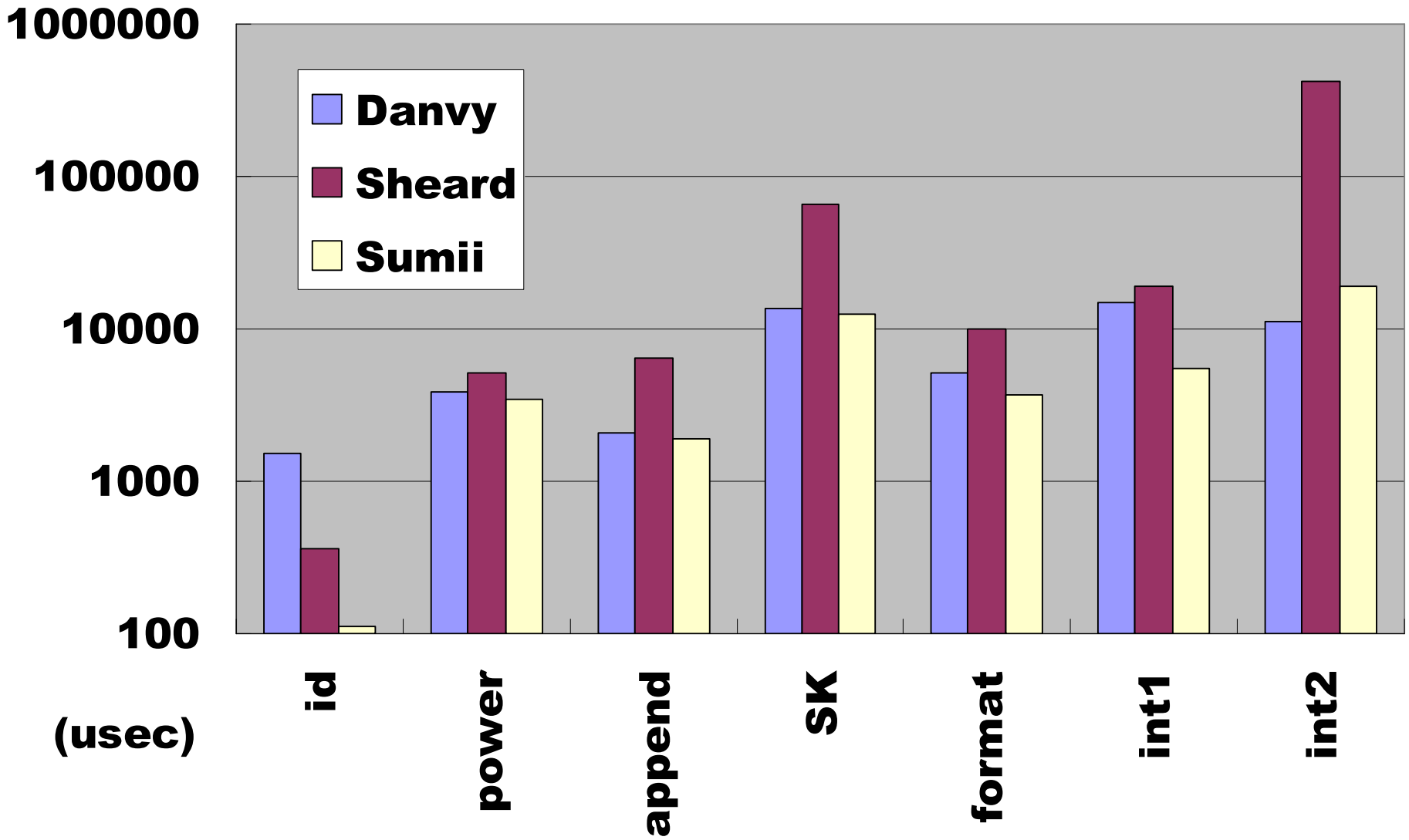
	Danvy	我々
Reflection	必要	不要
Destructor の拡張	不要	必要

どちらのほうが効率が良いのか？

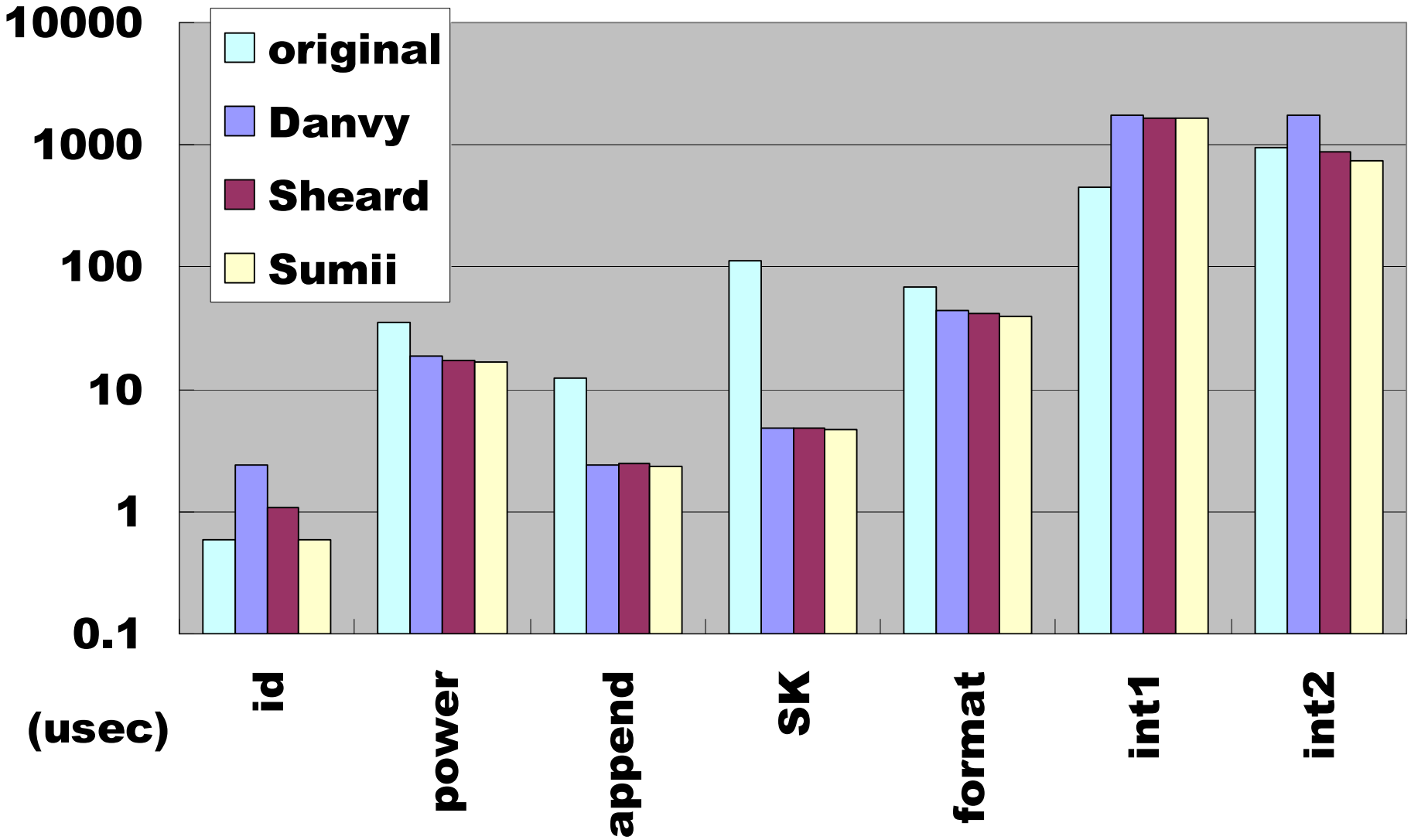
- Reflection は、Destructor の拡張と違って PE の処理だけでなく結果も悪化させる。
- 実験によれば、Destructor の拡張は Reflection よりオーバーヘッドが小さい。

⇒ 我々の TDPE のほうが効率が良い

比較実験: 特化にかかる時間



比較実験: 実行にかかる時間



発表の概要

- Introduction
- Danvy の TDPE
- 我々の TDPE
- 比較実験
- SDPE との関係
- 関連研究
- 結論

SDPE との関係: 全体像

単純な SDPE with HOAS & deforestation

+

HOAS から FOAS への変換

⇓

我々の TDPE

SDPE との関係: HOAS

Higher-Order Abstract Syntax
[Pfenning & Elliott 88]:

object level の binding を
meta level の binding であらわす

例:

λx. λy. (x @ y) ⇒ λ(λa. λ(λb. (a @ b)))

SDPE との関係:
HOAS を用いた SDPE

$$R(\underline{\lambda}(\lambda x.\, t)) = \underline{\lambda}(\lambda y.\, R((\lambda x.\, t)\ @\ y))$$

$$R(t_1\ \underline{@}\ t_2) = (\lambda x.\, t)\ @\ R(t_2) \qquad (\text{if } R(t_1) = \underline{\lambda}(\lambda x.\, t))$$

$$R(t_1\ \underline{@}\ t_2) = R(t_1)\ \underline{@}\ R(t_2) \qquad (\text{otherwise})$$

SDPE との関係: HOAS を用いた SDPE

$$
\begin{array}{lcll}
R(\underline{\lambda}(\lambda x.\, t)) & = & \underline{\lambda}(\lambda y.\, R((\lambda x.\, t)\ @\ y)) & \\
R(t_1\ \underline{@}\ t_2) & = & (\lambda x.\, t)\ @\ R(t_2) & (\text{if } R(t_1) = \underline{\lambda}(\lambda x.\, t)) \\
R(t_1\ \underline{@}\ t_2) & = & R(t_1)\ \underline{@}\ R(t_2) & (\text{otherwise}) \\
 & \Downarrow & & \\
\underline{\lambda'}(\lambda x.\, t) & = & \underline{\lambda}(\lambda y.\, (\lambda x.\, t)\ @\ y) & \\
t_1\ \underline{@'}\ t_2 & = & (\lambda x.\, t)\ @\ t_2 & (\text{if } t_1 = \underline{\lambda}(\lambda x.\, t)) \\
t_1\ \underline{@'}\ t_2 & = & t_1\ \underline{@}\ t_2 & (\text{otherwise})
\end{array}
$$

SDPE との関係: HOAS を用いた SDPE

$$
\begin{array}{lcll}
R(\underline{\lambda}(\lambda x.\, t)) & = & \underline{\lambda}(\lambda y.\, R((\lambda x.\, t)\ @\ y)) & \\
R(t_1\ \underline{@}\ t_2) & = & (\lambda x.\, t)\ @\ R(t_2) & (\text{if } R(t_1) = \underline{\lambda}(\lambda x.\, t)) \\
R(t_1\ \underline{@}\ t_2) & = & R(t_1)\ \underline{@}\ R(t_2) & (\text{otherwise}) \\
 & \Downarrow & & \\
\underline{\lambda}'(\lambda x.\, t) & = & \underline{\lambda}(\lambda y.\, (\lambda x.\, t)\ @\ y) & \\
t_1\ \underline{@}'\ t_2 & = & (\lambda x.\, t)\ @\ t_2 & (\text{if } t_1 = \underline{\lambda}(\lambda x.\, t)) \\
t_1\ \underline{@}'\ t_2 & = & t_1\ \underline{@}\ t_2 & (\text{otherwise})
\end{array}
$$

SDPE との関係: HOAS を用いた SDPE

$$
\begin{array}{lcll}
R(\underline{\lambda}(\lambda x.\, t)) & = & \underline{\lambda}(\lambda y.\, R((\lambda x.\, t)\ @\ y)) & \\
R(t_1\ \underline{@}\ t_2) & = & (\lambda x.\, t)\ @\ R(t_2) & (\text{if } R(t_1) = \underline{\lambda}(\lambda x.\, t)) \\
R(t_1\ \underline{@}\ t_2) & = & R(t_1)\ \underline{@}\ R(t_2) & (\text{otherwise}) \\
 & \Downarrow & & \\
\underline{\lambda}'(\lambda x.\, t) & = & \underline{\lambda}(\lambda x.\, t) & \\
t_1\ \underline{@}'\ t_2 & = & (\lambda x.\, t)\ @\ t_2 & (\text{if } t_1 = \underline{\lambda}(\lambda x.\, t)) \\
t_1\ \underline{@}'\ t_2 & = & t_1\ \underline{@}\ t_2 & (\text{otherwise})
\end{array}
$$

SDPE との関係: HOAS を用いた SDPE

$$
\begin{array}{lcll}
R(\underline{\lambda}(\lambda x.\, t)) & = & \underline{\lambda}(\lambda y.\, R((\lambda x.\, t)\ @\ y)) & \\
R(t_1\ \underline{@}\ t_2) & = & (\lambda x.\, t)\ @\ R(t_2) & (\text{if } R(t_1) = \underline{\lambda}(\lambda x.\, t)) \\
R(t_1\ \underline{@}\ t_2) & = & R(t_1)\ \underline{@}\ R(t_2) & (\text{otherwise}) \\
 & \Downarrow & & \\
\underline{\lambda}'(\lambda x.\, t) & = & \underline{\lambda}(\lambda x.\, t) & \\
t_1\ \underline{@}'\ t_2 & = & (\lambda x.\, t)\ @\ t_2 & (\text{if } t_1 = \underline{\lambda}(\lambda x.\, t)) \\
t_1\ \underline{@}'\ t_2 & = & t_1\ \underline{@}\ t_2 & (\text{otherwise})
\end{array}
$$

SDPE との関係: HOAS を用いた SDPE

$$
\begin{array}{lcll}
R(\lambda x.\, t) & = & \lambda y.\, R((\lambda x.\, t)\ @\ y) & \\
R(t_1\ \underline{@}\ t_2) & = & (\lambda x.\, t)\ @\ R(t_2) & (\text{if } R(t_1) = \lambda x.\, t) \\
R(t_1\ \underline{@}\ t_2) & = & R(t_1)\ \underline{@}\ R(t_2) & (\text{otherwise}) \\
 & \Downarrow & & \\
\underline{\lambda}'(\lambda x.\, t) & = & \lambda x.\, t & \\
t_1\ \underline{@}'\ t_2 & = & (\lambda x.\, t)\ @\ t_2 & (\text{if } t_1 = \lambda x.\, t) \\
t_1\ \underline{@}'\ t_2 & = & t_1\ \underline{@}\ t_2 & (\text{otherwise})
\end{array}
$$

$\Rightarrow$ @′ の定義と一致する！

SDPE との関係: HOAS から FOAS への変換

$$F(\lambda x.\, t) = \underline{\lambda y}.\, F((\lambda x.\, t)\ @\ \underline{y}) \quad \text{(where } \underline{y} \text{ is fresh)}$$

$$F(t_1\ \underline{@}\ t_2) = F(t_1)\ \underline{@}\ F(t_2)$$

$$F(\underline{y}) = \underline{y}$$

⇒ *reify* の定義と一致する！

SDPE との関係: まとめ

より複雑な SDPE を用いれば
より高度な TDPE が得られる

単純な SDPE with HOAS & deforestation

+

HOAS から FOAS への変換

⇓

我々の TDPE

発表の概要

- Introduction
- Danvy の TDPE
- 我々の TDPE
- 比較実験
- SDPE との関係
- 関連研究
- 結論

関連研究

- [Danvy 96 & 97] [Sheard 97]:
  reflection のできない型には ad hoc に対応
  ⇒ 複雑で効率が悪い
- [Thiemann 96 & 99]:
  HOAS を用いた offline SDPE から
  我々と同様の変換で offline PGG を導出
- [Helsen & Thiemann 98]:
  上の offline PGG と Danvy の TDPE の
  類似性を指摘

結論

- TDPE の改良を提案
  - 広く適用できる
  - 単純で効率が良い
- 改良された TDPE と SDPE の関係を明確化

  ⇒ TDPE と SDPE の技術を融合できる

例: コードや計算の重複や消滅がなく、
任意の副作用について健全な SDPE
[Lawall & Thiemann 97]

◆今後の課題

- PE の停止性の保証
- 副作用を複雑に用いたプログラムの特化