(Bi-)^3 directional Typing for Answer Type Modification

Takuma Yoshioka (Kyoto University)

Goal: a foundation for shift0/reset0 and ATM easy to extend to advanced typing features

Background

- **shift0/reset0** [Danvy and Filinski, 1989] can represent various computational effects, such as exceptions, mutable states, and non-determinism, and have the same expressive power as **effect handlers** [Plotkin and Pretnar, 2009]
- Some type systems for shift0/reset0 accommodates **answer type modification** (ATM) [Danvy and Filinski, 1989; Materzok and Biernacki, 2011]
- ATM enables us to write a type-safe printf, let-insertions, and A-normalization in a direct style [Asai, 2005; 2009]
- ATM can be applied to **temporal verification** [Sekiyama and Unno, 2023] and **refinement types** [Kawamata et al., 2024]

Our approach: extend bidirectional typing to shift0/reset0 and ATM

- We propose **(bi-)^3 directional typing**, an extension of bidirectional typing [Pierce and Turner, 2000], in a simply typed setting
- Bidirectional typing can be used for refinement types [Xi and Pfenning, 1999] and higher-rank polymorphism [Dunfield and Krishnaswami, 2013]
- **(Bi-)^3 directional typing** could be a basis for introducing research results in bidirectional typing to shift0/reset0

shift0/reset0 and ATM

Syntax

shift0 reset0

```
  e ::= ... | S0 k. e | ⟨e⟩   (expressions)
A, B ::= b | A -> C           (value types)
C, D ::= A / S                (computation types)
  S ::= □ | C ▷ D             (control effects)
```

initial answer type — final answer type

Typing judgment

`Γ ⊢ e : C`

Pure case: `Γ ⊢ e : A / □`
ATM case: `Γ ⊢ e : A / C ▷ D`

Typing example

$$\frac{\Gamma,\ k : \texttt{int} \to \texttt{int}/\square \vdash \texttt{int2str}\ (k\ 1) : \texttt{str}/\square}{\Gamma \vdash \texttt{S0}\ k.\ \texttt{int2str}\ (k\ 1) : \texttt{int} / (\texttt{int}/\square) \triangleright (\texttt{str}/\square)}$$

Example program

The reset0 expression is reduced to the body of shift0, binding k to the eval. ctx. enclosing shift0, called a delimited continuation

```
  ⟨ 1 + (S0 k. int2str (k 2)) ⟩ ^ “is three”
→ int2str (k 2) ^ “is three” ( where k = λx. ⟨1 + x⟩ )
→ int2str ⟨1 + 2⟩ ^ “is three”
→ ... → “3 is three”
```

(Bi-)^3 directional typing

Bidirectional typing judgment [Pierce and Turner, 2000]

`Γ ⊢ e <=> A` with `<=> ::= <= | =>`

- The type A is an **output** if <=> is => (**synthesis** mode)
- The type A is an **input** if <=> is <= (**checking** mode)

(Bi-)^3 directional typing judgment (our work)

`Γ ⊢ e <=> A / <=> C ▷ <=> D`

these types separately have synthesis or checking mode

Key assumptions for making (bi-)^3 directional typing rules:

- Annotations for value types are acceptable, but annotations for initial/final answer types are difficult to write
- Annotating types for delimited continuations (`S0 (k:A->C). e`) is difficult especially when two or more shift0 occur

Typing rule for shift0

$$\frac{\Gamma,\ k : A \to C \vdash e : D}{\Gamma \vdash \texttt{S0}\ k.\ e : A / C \triangleright D}$$

- We must know A and C to extend Γ
- We want to avoid an annotation for k

Our choice:

- Checking initial answer types
- Allowing annotating value types

Typing rule for let-expressions

$$\frac{\Gamma \vdash e1 : A / C' \triangleright D \qquad \Gamma,\ x : A \vdash e2 : B / C \triangleright C'}{\Gamma \vdash \texttt{let}\ x = e1\ \texttt{in}\ e2 : B / C \triangleright D}$$

- Assume that the initial answer type C is given in checking mode
- To use C’ in checking mode for e1, we must synthesize it from e2

Our choice:

- Synthesizing final answer types
- In a let-expression, first typing e2 and then typing e1

Current status

- A core calculus and its implementation
- Well-typed programs **without an annotation for k**
  - `⊢ ⟨1 + (S0 k. int2str (k 2))⟩ => str / <= □`
  - `⊢ ⟨(S0 k. (k 1) ^ “x”) + (S0 k. int2str (k 2)⟩ => str / <= □`

Future work

- Establish a design principle for (bi-)^3 directional typing
- Extend (bi-)^3 directional typing to refinement types and higher-rank polymorphism
- Apply (bi-)^3 directional typing to effect handlers

(Bi-)^3 directional Typing for Answer Type Modification

Takuma Yoshioka (Kyoto University)

Goal: easy implementation of advance typing features for shift0/reset0

- **shift0/reset0** [Danvy and Filinski, 1989] can represent some computational effects, such as exceptions, mutable states, and non-determinism, and have the same expressive power with **effect handlers** [Plotokin and Pretnar, 2009]
- Some type systems for shift0/reset0 [Danvy and Filinski, 1989; Materzok and Biernacki, 2011] accommodates **answer type modification** (ATM)
- ATM enables us to write a type-safe printf, let-insertions, and A-normalization in a direct style [Asai, 2005; 2009]
- ATM can be applied to **temporal verification** [Sekiyama and Unno, 2023] and **refinement types** [Kawamata et al., 2024]

Our approach: extend bidirectional typing to shift0/reset0 and ATM

- We propose **(bi-)^3 directional typing**, an extension of bidirectional typing [Pierce and Turner, 2000] to shift0/reset0 and ATM, in a simple typed setting as a first step toward the goal
- Bidirectional typing is used for implementing refinement types [Jhala and Vazou, 2020]

shift0/reset0 and ATM

Syntax

shift0 reset0

```
  e ::= ... | S0 k. e | ⟨e⟩      (expressions)
A, B ::= b | A -> C              (value types)
C, D ::= A / S                   (computation types)
  S ::= □ | C ▷ D                (control effects)
```

initial answer type **final answer type**

Typing judgment

`Γ ⊢ e : C`

Pure case: `Γ ⊢ e : A / □`
ATM case: `Γ ⊢ e : A / C ▷ D`

Typing example

$$\frac{\Gamma,\ k : \mathtt{int} \to \mathtt{int}/\square \vdash \mathtt{int2str}\ (k\ 1) : \mathtt{str}/\square}{\Gamma \vdash \mathtt{S0}\ k.\ \mathtt{int2str}\ (k\ 1) : \mathtt{int} / (\mathtt{int}/\square) \triangleright (\mathtt{str}/\square)}$$

Example program

The reset0 expression is reduced to the body of shift0, binding k to the eval. ctx. enclosing shift0, called a delimited continuation

```
  ⟨(S0 k. int2str (k 1)) + 2⟩
→ (int2str (k 1)) (where k = λx. ⟨x + 2⟩)
→ (int2str ⟨1 + 2⟩)
→ ... → "3"
```

(Bi-)^3 directional typing: assigning synthesis/checking modes [Pierce and Turner, 2000] to answer types

Inputs a typing context and an expression, and outputs the type `Γ ⊢ e => A`

Inputs a typing context, an expression, and a type, and outputs yes/no `Γ ⊢ e <= A`

Key observations:

- Annotations for value types are acceptable, but annotations for initial/final answer types are difficult to write
- Annotating types for delimited continuations (`S0 k:A->C . e`) is difficult especially when two or more shift0 occurs

Typing rule for shift0

$$\frac{\Gamma,\ k : A \to C \vdash e : D}{\Gamma \vdash \mathtt{S0}\ k.\ e : A / C \triangleright D}$$

- We must know `A` and `C` to extend `Γ`
- We want to avoid an annotation for `k`

→ Our choice:

- Checking initial answer types

Typing rule for let-expressions

$$\frac{\Gamma \vdash e1 : A / C' \triangleright D \qquad \Gamma,\ x : A \vdash e2 : B / C \triangleright C'}{\Gamma \vdash \mathtt{let}\ x = e1\ \mathtt{in}\ e2 : B / C \triangleright D}$$

- Assume that the initial answer type `C` is given in checking mode
- To use `C'` in checking mode for `e1`, we must synthesize it from `e2`

→ Our choice:

- Synthesizing final answer types
- In a let-expression, first typing `e2` and then typing `e1`

Current status

- A core calculus and its implementation
- Well-typed programs:
  - `⟨(S0 k. int2str (k 1)) + 2⟩`
  - `⟨(S0 k. (k 1) ^ "x") + (S0 k. int2str (k 2))⟩`

No annotation required

Future work

- Establish a design principle for (bi-)^3 directional typing
- Extend (bi-)^3 directional typing to polymorphism and refinement types
- Apply (bi-)^3 directional typing to effect handlers

(Bi-)^3 directional Typing for Answer Type Modification

Takuma Yoshioka (Kyoto University)

Goal: Bring a bidirectional typing to answer type modification (ATM)

Can represent some computational effects such as exceptions, mutable states, and non-determinism

ATM enables us to write a type-safe printf, let-insertion, and A-normalization in a direct style [Asai 2005; 2009]

Delimited Control Operators

- …
- shift0/reset0 [Danvy and Filinski, 1989]

Same expressive power [Forster et al, 2017; Pirog et al., 2019]

Effect Handlers [Plotkin and Pretnar, 2009]

Frank [Lindley et al., 2017], Alef [Locascio, 2020]

Bidirectional Typing [Pierce and Turner, 2000]

Lightweight and scalable way for type inference, while requiring some type annotations

Type Systems with Answer Type Modification

+subtyping

$\lambda_{\leq}^{S_0}$ [Materzok and Biernacki, 2011]

$\lambda_{\mathbf{ev}}^{S_0}$ [Sekiyama and Unno, 2023]: Temporal verification with shift0/reset0 using answer **effect** modification

RCaml [Kawamata et al., 2024]: Refinement type system for effect handlers, which is realized by adopting answer **refinement** modification

(Bi-)^3 directional Typing (our work)

Expected benefits: decidable type inference for refinement types, basics for type error localization [Zhao et al., 2024] with ATM

shift0/reset0 and ATM

Example program: printf

shift0: `S0 k. e`

reset0: `⟨e⟩`

The reset0 expression is reduced to the body of shift0, binding k to the eval. ctx. enclosing shift0

```
Γ, k : str->str/□ ⊢ λx. k (int2str x): (int->str/□)/□
-------------------------------------------------------------------
Γ ⊢ S0 k. λx. k (int2str x) : str / (str/□) ▷ ((int->str/□)/□)

⟨(S0 k. λx. k (int2str x)) ^ "one"⟩ 1
→ (λx. k (int2str x)) 1
    ( where k = ⟨[] ^ "one"⟩ )
→ (λx. ⟨(int2str x) ^ "one"⟩) 1
→ ⟨(int2str 1) ^ "one"⟩
→ ... → ⟨"1one"⟩ → "1one"
```

The value type of shift0 is `str`

The return type of k is `str`, **initial answer type**

ATM

The type of the reset0 is `int->str`, **final answer type**

(Bi-)^3 directional typing: assigning synthesis/checking modes [Pierce and Turner, 2000] to answer types

Inputs a typing context and an expression, and outputs the type `Γ ⊢ e => A`

Inputs a typing context, an expression, and a type, and outputs yes/no `Γ ⊢ e <= A`

Declarative rule for shift0

```
Γ, k : A -> C ⊢ e : D
---------------------
Γ ⊢ S0 k. e : A / C ▷ D
```

Value type A and initial answer type C must be known to extend a typing context with k

(Bi-)^3 directional rule for shift0

```
Γ, k : A -> C ⊢ e => D
----------------------------------
Γ ⊢ S0 k. e (<= A) / (<= C) ▷ (=> D)
```

We assign the checking mode to A and C

After inferring the type of e, we outputs final answer type D

Design policy: assigning checking mode to initial answer type, the synthesis mode to final answer type

Propagating initial answer types

```
⟨let x1 = e1 in
 let x2 = e2 in
 v⟩
```

```
e1: A1 / C1 ▷ D1
e2: A2 / C2 ▷ D2
v : A3 / C3 ▷ D3
```

let requires C1 = D2

C2 = D3

reset0 requires A3/□ = C3

value requires C3 = D3

(Bi-)^3 directional typing proceeds **reversely** to evaluation

```
e1 (<= A1) / (<= D2) ▷ (=> D1)
e2 (<= A2) / (<= A3/□) ▷ (=> D2)
v  (=> A3) / (=> A3/□) ▷ (=> A3/□)
```

Current status: ✓ `∅ ⊢ ⟨(S0 k. λx:int. k (int2str x)) ^ "one"⟩ 1 (=> str) / (<= □)`