I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

http://www.jaist.ac.jp/~hirokawa/lectures/fp/

Schedule

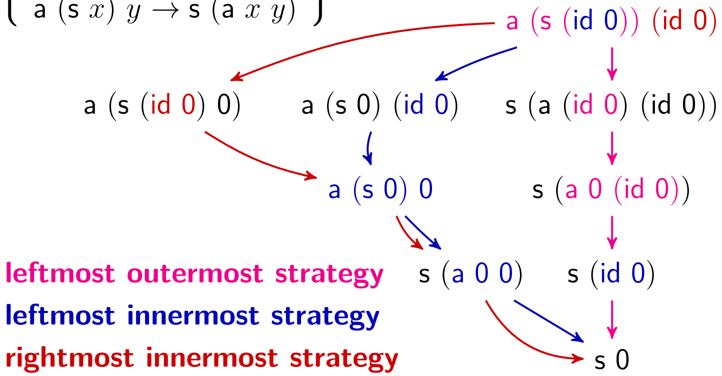
10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	applications	11/18	confluence
10/26	program reasoning	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

Evaluation

exam (60) + reports (40)

Rewrite Strategies

$$\text{TRS } \mathcal{R} = \left\{ \begin{array}{r} \mathsf{id}\ x \to x \\ \mathsf{a}\ 0\ y \to y \\ \mathsf{a}\ (\mathsf{s}\ x)\ y \to \mathsf{s}\ (\mathsf{a}\ x\ y) \end{array} \right\}$$



Innermost Strategy (Innermost Rewriting)

Definition

$s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ if there exist rule $\ell \to r \in \mathcal{R}$, context C, and σ such that

- $s = C[\ell\sigma]$, $t = C[r\sigma]$ and
- $u \in \mathsf{NF}(\mathcal{R})$ for all subterms u of $\ell\sigma$ with $u \neq \ell\sigma$

Example

$$\text{consider TRS } \mathcal{R} = \left\{ \begin{array}{r} 0 + y \to y \\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ \mathsf{d}(x) \to x + x \end{array} \right\}$$

1. compute all rewrite sequences from $\mathsf{d}(\mathsf{d}(\mathsf{s}(0)))$
2. which of them are sequences of innermost strategy?

Naive Innermost Rewriting

consider TRS

$$\mathcal{R} = \left\{ \begin{array}{r} 0 + y \to y \\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ \mathsf{main} \to \mathsf{s}(0) + \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \end{array} \right\}$$

we have:

$$\underbrace{\mathsf{main}}_{\text{matched}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \underbrace{\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}}_{\text{matched}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\text{matched}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}$$

Efficient Rewriting by Keeping Normal Form Positions

consider TRS

$$\mathcal{R} = \left\{ \begin{array}{r} 0 + y \to y \\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ \mathsf{main} \to \mathsf{s}(0) + \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \end{array} \right\}$$

we have:

$$\underbrace{\mathsf{main}}_{\text{matched}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \underbrace{\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}}_{\text{matched}} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{s}(\underbrace{\underbrace{0}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))}_{\mathsf{NF}}}_{\text{matched}}) \xrightarrow{\mathsf{i}}_{\mathcal{R}} \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))}_{\mathsf{NF}})}_{\mathsf{NF}}$$

Bottom-Up Computation of Normal Forms

assume that for every term s there uniquely exists term t with $s \to_{\mathcal{R}}^{*} t \in \mathsf{NF}(\mathcal{R})$

Definition

$s{\downarrow_{\mathcal{R}}}$ denotes normal form of s with respect to $\mathcal{R}$

Fact

- if s is variable then $s{\downarrow_{\mathcal{R}}} = s$
- if $s = f(s_1, \ldots, s_n)$ then for $t = f(s_1{\downarrow_{\mathcal{R}}}, \ldots, s_n{\downarrow_{\mathcal{R}}})$

$$s{\downarrow_{\mathcal{R}}} = \begin{cases} (r\sigma){\downarrow_{\mathcal{R}}} & \text{if } \ell \to r \in \mathcal{R} \text{ and } t = \ell\sigma \\ t & \text{otherwise} \end{cases}$$

Report Assignment II: Mini-Haskell

program → **MiniHS** → normal form of `main`

sample input:

```
                    :
                    :
qsort []            = [] .
qsort (cons X XS) =
  append   (qsort (filter (geq X) XS))
   (cons X (qsort (filter (lt X) XS))) .

main = qsort [5,1,4,2,3] .
```

output:

```
[1,2,3,4,5]
```

Submission

- submit by email

  ```
  Subject: report2
  To: hirokawa@jaist.ac.jp
  ```

  attaching two files: `M`*YourStudentID*`.hs` and `M`*YourStudentID*`.trs`

- code must start from information

  ```
  -- name: your full name
  -- id: your student ID
  -- acknowledgements: name if anybody has assisted you
  ```

- deadline: **Nov 28 (Mon) 10:00 AM JST**

Remark on Report Submission

- deadline is strict
- submission style is strict, and do not submit any additional file
- use GHC 8.*.* or 9.*.*;
- do not import in `Mxxxxxxx.hs` any module except `Data.List`, `TRS`, `Parser`, and `System.Environment`
- do not change program specification; especially the syntax of input programs.
- do not ask me to test your code
- **no plagiarism:** investigate by yourself what is regarded as plagiarism

Instructions on This Task

- Download `minihs.zip` from the course page.
- Rename `Main.hs` to `M`*yourStudentID*`.hs`.
- Rename `Mxxxxxxx.trs` to `M`*yourStudentID*`.trs`.
- The interpreter must take a text file of a Haskell-like program and output a normal form of constant symbol `main`.
- Its computation should be done by rewriting with **innermost strategy**.
  You get no point unless innermost strategy is adopted.

- On my environment (Intel Core i7-5500U CPU 2.40GHz), your program will be compiled and executed as follows:

  ```
  ghc -o minihs Mxxxxxxx.hs
  ./minihs sum10.trs
  ./minihs qsort100.trs
  ./minihs fact10.trs
  ./minihs Mxxxxxxx.trs
  ```

Point Allocation

- $+5$ points if `sum10.trs` outputs correct number within 60 seconds
- $+5$ points if `qsort100.trs` outputs correct list within 120 seconds
- $+5$ points if `fact10.trs` outputs number within 120 seconds
- $+5$ points if you complete magic square solver `Mxxxxxxx.trs` and if it outputs correct list within 180 seconds (in your or my minihs)

Hints

Applicative Terms

```haskell
data Term = Var String
          | Con String
          | App Term Term  deriving Eq
```

- add 0 Y

  ```haskell
  App (App (Con "add") (Con "0")) (Var "Y")
  ```

- add (s X) Y

  ```haskell
  App (App (Con "add") (App (Con "s") (Var "X")))
      (Var "Y")
  ```

Exercise

use Term to represent s (add X Y)

Parser.hs

input text	ouput term
1	s 0
2	s (s 0)
$[0, 1, 2]$	cons 0 (cons (s 0) (cons (s (s 0)) nil))

Substitutions

```haskell
type Substitution = [(String, Term)]
```

substitute :: Term → Substitution → Term

substitute $t\ \sigma = t\sigma$

match :: Term → Term → Maybe Substitution

$$\texttt{match}\ \ell\ t = \begin{cases} \text{Just } \sigma & \text{if } \ell\sigma = t \\ \text{Nothing} & \text{otherwise} \end{cases}$$

Naive Rewriting

$\texttt{rewriteAtRoot}$ :: TRS $\to$ Term $\to$ Maybe Term

$$\texttt{rewriteAtRoot}\ \mathcal{R}\ t = \begin{cases} \text{Just } (r\sigma) & \text{if } t = \ell\sigma \text{ and } \ell \to r \in \mathcal{R} \\ \text{Nothing} & \text{otherwise} \end{cases}$$

$\texttt{rewrite}$ :: TRS $\to$ Term $\to$ Maybe Term

$$\texttt{rewrite}\ \mathcal{R}\ t = \begin{cases} \text{Just } u & \text{if } t \xrightarrow{\text{i}}_{\mathcal{R}} u \\ \text{Nothing} & \text{otherwise} \end{cases}$$

$\texttt{nf1}$ :: TRS $\to$ Term $\to$ Term

$\texttt{nf1}\ \mathcal{R}\ t$ returns term u with $t \xrightarrow{\text{i}}{}^{*}_{\mathcal{R}} u \in \text{NF}(\mathcal{R})$

$\texttt{nf1}\ \mathcal{R}\ t$ can be computed by repeatedly applying $\texttt{rewrite}$

Rewriting with Marked Terms

```
data MarkedTerm = MApp MarkedTerm MarkedTerm
                | MCon String
                | NF Term
```

E.g. MApp (MCon "s") (NF (App (Con "s") (Con "s"))) stands for $\mathsf{s} \cdot \underset{\text{NF}}{\underline{(\mathsf{s}\ 0)}}$

$\texttt{substitute2}$ :: Term $\to$ Substitution $\to$ MarkedTerm

$\texttt{substitute2}\ (\mathsf{s}\ x)\ [(x, \mathsf{s}\ 0)] = \mathsf{s} \cdot \underset{\text{NF}}{\underline{(\mathsf{s}\ 0)}}$

$\texttt{rewriteAtRoot2}$ :: TRS $\to$ Term $\to$ MarkedTerm

$\texttt{rewriteAtRoot2}\ \mathcal{R}\ (\mathsf{f}\ (\mathsf{s}\ 0)) = \mathsf{id} \cdot \underset{\text{NF}}{\underline{(\mathsf{s}\ 0)}}$

use $\texttt{rewriteAtRoot2}$ to implement counterparts of $\texttt{rewrite}$ and $\texttt{nf1}$:

$\texttt{rewrite2}$ :: TRS $\to$ MarkedTerm $\to$ MarkedTerm

$\texttt{nf2}$ :: TRS $\to$ MarkedTerm $\to$ Term

Bottom-Up Computation of Normal Forms

implement approach described on slide 7

$\texttt{nf3}$ :: TRS $\to$ MarkedTerm $\to$ Term

Homework 1/3

1 Recall the Leibniz formula

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Define $\texttt{leibniz}\ n$ that sums up the first n fractional numbers in the formula. For instance,

$$\texttt{leibniz}\ 4 = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7}$$

Hint: In the case that you have a type error, please check $\texttt{fromIntegral}\ n$, which converts integer n to the corresponding number of other type.

Homework 2/3

2 Consider the following types for arithmetic expressions:

```
data Exp = Val Int | Var String
         | Add Exp Exp | Mul Exp Exp
type Env = [(String, Int)]
```

Implement the evaluation function

$$\texttt{eval1} :: \text{Env} \to \text{Exp} \to \text{Int}$$

For instance, we have

$$\texttt{eval1}\ [(\texttt{"x"}, 10), (\texttt{"y"}, 30)]\ e = 50$$

for $e = \text{Add (Mul (Var } \texttt{"x"}\text{) (Val } 2\text{)) (Var } \texttt{"y"}\text{)}$

Homework 3/3

3 Implement the evaluation function

$$\texttt{eval2} :: \text{Env} \to \text{Exp} \to \text{Either String Int}$$

For instance, we have

$$\begin{aligned} \texttt{eval2}\ [(\texttt{"x"}, 10), (\texttt{"y"}, 30)]\ e &= \text{Right } 50 \\ \texttt{eval2}\ [(\texttt{"x"}, 10)]\ e &= \text{Left } \texttt{"x is undefined"} \end{aligned}$$

for $e = \text{Add (Mul (Var } \texttt{"x"}\text{) (Val } 2\text{)) (Var } \texttt{"y"}\text{)}$