

函数式语言程序设计 2019 课程作业

版本和更新记录

版本：2019-05-17

更新记录

1. 2019-05-17 的更新（相对于征求意见稿）：

1. `AST.hs` 里，`Expr` 类型增加了 data constructor `EMod`，表示取余（modulo）操作符。
2. `EDiv` 和 `EMod`，除以 0、对 0 取模（modulo）规定为 undefined behavior。你的实现里可以任意处理。
3. 修改 `AST.hs` 文档措辞（用变量名而不是参数类型来明确指明参数），见 `doc/haddock/AST.html`。
4. 给出了 `Spec.hs`，其中有 80 个 HTF 格式的测试用例。最终评测时我们用 100 个测试用例来测“必做部分”，会包括这 80 个测试用例。`Spec.hs` 也由浅入深地解释了如何用 `AST.hs` 里的各个数据类型来构造一个（你的语言的）程序（的 AST）。
5. 提高了选做部分的分值：现在 k 个人组队时，做 k 个选做项目，就可能得到 110 分。“选做部分分值除以组队人数”的策略不变。
6. 展示环节，允许不方便到场的同学远程展示。

简介

本次作业选题为“编程语言的解释器的实现”。

完成方式

本作业允许组队完成，规模为1~3 人。作业提交内容包括：

1. 项目全部源代码；
2. 实验报告（内容包括但不限于：(1) 提交代码的执行方法描述，(2) 作业的实现目标(实现了哪些语言特性、语言定义是怎样的、实现了哪些得分项)，(3) 作业的实现思路与亮点，(4) 挑战与解决方案、参考资料或参考代码等）；
3. 每位成员在作业中的贡献（请用一个另外的文档简要说明）。

本作业的截止时间为 **第 18 周周日**，请按时提交作业，否则会影响成绩的录入。本学期毕业的同学需要更早提交（第 16 周）。实现选做内容的同学需要给助教展示验收，详见后文“评测方式”。

早提交作业的同学，会被助教优先评判；比起晚交的同学，有更多修改并提高分数的机会。

作业内容

本作业的内容是一个语言的解释器（interpreter）。该语言是一个基于简单类型 lambda 演算（simply typed lambda calculus）的微型的语言。本作业和编译原理课通常做的语言实现作业不太一样。编译原理课通常会用 BNF 给出要实现的语言的具体文法，例如：

```
expression
 ::= True
  | False
  | (not expression)
  | (and expression expression)
  | (or expression expression)
```

然后由你来 parse 成解析树（parsing tree、CST），然后转换成某种形式的抽象语法树（AST），然后从 AST 去解释执行（如果是做解释器），或编译到某种目标语言（如果是做编译器）。

但本作业不一样。我们会提供一个使用 Haskell 的数据类型来定义的抽象语法树（AST），见 `AST.hs` 以及其生成的文档 `doc/haddock/AST.html`，其中给出了 AST 的每种形式的求值规则和类型规则（typing rule）。

你需要做的是（更详细的内容包括 **评分标准** 请参考“语言特性得分项目”部分）：

1. 必做部分：核心语言（80 分）

1. 写一个 `evalType` 函数，求该 AST 的实例的类型；
2. 写一个 `evalValue` 函数，对该 AST 的实例求值；

2. 选做部分

1. 实现代数数据类型（ADT）的支持：包括代数数据类型的声明、代数数据类型的构造函数、模式匹配语句的支持（35 分）；
2. 为这个语言设计一个具体文法，并写一个 parser，它能把符合你的具体文法的字符串 parse 到该 AST（35 分）；
3. 基于具体文法，实现一个 `REPL`（35 分）；
4. 实现编译器（90 分）。

每位同学必做部分 + 选做部分的总分不超过 110 分。对于组队的同学，必做部分的得分每位同学相同，选做部分的得分由组员均摊。例如：

- 3 位同学组队，必做部分得到 75 分，正确实现了共 90 分的选做项目，则每人的课程作业得分为 $\min(110, 75 + (90/3)) = \min(110, 75 + 30) = \min(110, 105) = 105$ ；
- 1 位同学单独成队，必做部分得到 75 分，选做项目得到 40 分，则课程作业得分为 $\min(110, 75 + 40) = \min(110, 115) = 110$ 。

如果你希望实现不在以上列表的选做内容，请联系助教获得许可。

语言特性得分项目

核心语言：必做

包括 `AST.hs` 中：

1. 类型：`Type` 类型中的 `TBool`、`TInt`、`TChar`、`TArrow` 的支持，分别对应布尔类型、整数类型、字符类型、函数类型(包括高阶函数)；

2. 表达式：Expr 类型中的 EBoolLit、EIntLit、ECharLit、ENot、EAnd、EOr、EAdd、ESub、EMul、EDiv、EMod、EEq、ENeq、ELt、EGt、ELe、EGe、EIf、ELambda、ELet、ELetRec、EVar、EApply。即除去 ECase 的所有 Expr。

不需要支持 ADT，即 Program 的形式为 Program [] <some expression>。

注意：

1. ELambda 无法进行递归，ELetRec 允许递归，所以你的实现需要支持递归；
2. 函数是 currying 的，所以你的实现需要支持高阶函数。
3. evalType 的类型是 Program -> Maybe Type。当参数代表的表达式有类型错误时，返回 Nothing；否则返回 Just 该表达式的类型。
evalValue 的类型是 Program -> Result。返回参数代表的表达式的求值结果。
4. src/ 下的 EvalType.hs 和 EvalValue.hs 提供了一种实现方式，供参考。你可以用它的方式来实现（其中用到后面课程要讲的 MonadTransformer），或用你自己的方式实现。

评分方法：会提供 80 个样例实例测试 evalType 和 evalValue 函数（见 Spec.hs），并有 20 个隐藏测试点，最终得分按照这 100 个测试点的得分确定。这些测试点不会纠结在一些细节上，例如不会考虑除数为 0 时的行为、不会考虑命名冲突时的行为等等。

代数数据类型：选做

需要在 evalType 和 evalValue 中完整实现对 AST.hs 中的 Type、Expr 和 Pattern 的支持，评分时有对于这两个函数的黑盒测试。如果思考得比较清楚，该选做项工作量相对同分值的其他选项会小一些。

35 分里有 30 分是基本分数，5 分是亮点得分。

文法设计和 parser：选做

需要提供可以 parse 出一些给定的 test case 的字符串用于黑盒测试。可以参考 Lisp 系方言的语法并增加自己设计的类型标注（Lisp 系容易 parse）。核心语言中去掉了 many 常见语言中的常见语法糖，因此文法设计和 parser 层面可以实现一些语法糖，例如支持在单个 let 语句中进行多个绑定、支持在 let 语句中使用模式匹配等等。

最终评分时我们会提供若干个合法的 AST，你需要构造一些符合你设计的文法的程序（字符串），使得它们经过你的 parser 后可以得到这些 AST。

35 分里有 30 分是基本分数，5 分是亮点得分。

REPL：选做

类似 ghci，支持表达式值打印、表达式类型查看等。

35 分里有 30 分是基本分数，5 分是亮点得分。

实现编译器：选做

把合法的 AST 编译到某种目标代码。可以编译至 `Java Bytecode`、`LLVM IR`、`GIMPLE` 或 `JavaScript`。

如果想编译到其它目标语言，请联系助教获得许可。限制目标语言的用意是：不应编译到太高级的语言，使得转换过程太 trivial。

评测方式

本作业的评测方法为黑盒 + 白盒测试。即通过以下方面进行详细评分：

1. 提交的实现首先会经过基本的测试用例测试；
2. 所提交实现的性能优化和代码风格等均会考察和予以评分。

执行环境为 GHC 8.6.4，允许使用所有 GHC 扩展和 ghc boot libraries (GHC 自带库)。鼓励使用 `stack` 进行项目管理，resolver 版本仍为 `lts-13.13`。使用第三方库 (package) 需要提前向助教确认并获得批准。

展示环节

做了选做部分的同学，需要在第 16 周的某个时间（待定）或第 18 周的某个时间（待定），给助教集中展示验收，以充分体现你所实现的功能。

由于行程冲突，到场展示有困难的同学，允许联系助教远程展示。