↑しらべもの[*1]

Mark P. Jones, Dictionary-free Overloading by Partial Evaluation

目次

1. なにについて書かれた論文か？

Haskell のようなアドホック多相の実装において、Dictionary Passing が非効率であることを改善するために、部分評価によって dictionary を渡さなくてすむような方法について

2. なにが知りたくてこれを読んだ？

Dictionary-free のまえに、Dictionary Passing のことを知らなかったので、それが知りたくて。この論文の section 3 に dictionary passing の説明がある。

3. わかったこと

やっぱり、Boxing では戻り値の型に依存して動作が決まる多相に対処できない。(section 3 3rd paragraph)

この問題を解決するためのエレガントな方法が、型情報を値から切り離して、それ自体をオブジェクトとしてあつかい、引数として渡すこと（Type Passing)。戻り値の型は、それが overloading に関係するときだけ渡せばよい。

(これらは、自分で悩んで到達しようとしていたのと同じ結論だった）

Another approach が dictionary value である。（別の方法だったのか！）

論文には明には書かれていないが、型クラスのインスタンスは、あとから定義されるのでディクショナリー渡しがいるのだと思う。

たとえば、(+) d 2 3 において、(+) 関数は Prelude に含まれるライブラリで、すでにコンパイルされている状況で、Num のインスタンスを定義して用いようとすると、新しいインスタンスに関する方法は、(+) の中には含められない。ので、辞書を渡す必要あり。

Dictionary Passing がわかりやすい上に、分割コンパイラに適している、とは書かれている。

だが、以下のような問題がある：

- Dictionary に含まれる未使用要素は、コードサイズの無用な増大をまねく
- 一般に、メソッドディスパッチに使われるセレクターは高階関数になる。これにより、効率的なコード

[*1] https://uhideyuki.sakura.ne.jp/studs/index.cgi/ja/FrontPage#p6

の生成や静的解析が難しくなる。

- Dictionary の生成、渡しは、実行時の余分なオーバーヘッドとなる。

3.1 節以降でこれらは、それぞれもう少し詳しく述べられる。

4. わからなかったこと

Fully Boxing って、fromInteger 実現できない方法なの？ ”Fully” になにか秘訣がある？

5. 参考文献のなかで、有益そうなもの

以下は 2016-07-22 追記

Dictionary-passing について、Further details about the translation process are given by [1, 17, 19, 20]. とあるので、これらを読んでみたい。

[1] L. Augustsson, Implementing Haskell overloading, Conference on Functional Programming Languages and Computer Architecture, Copenhagen, Denmark, June 1993.

[17] J. Peterson and M. Jones, Implementing Type Classes, ACM SIGPLAN ’93 Conference on Programming Language Design and Implementation, Albuquerque, New Mexico, June 1993.

[19] S. L. Peyton Jones and P. Wadler, A static semantics for Haskell (draft), Manuscript, Department of Computing Science, University of Glasgow, February 1992.

[20] P. Wadler and S. Blott, How to make ad-hoc polymorphism less ad-hoc, In 16th ACM annual symposium on Principles of Programming Languages, Austin, Texas, January 1989.