OCaml におけるデバッグ出力機構の実装

1211070　櫻井 健二　　中野研究室

1　背景

プログラミング言語 OCaml [1] は、型安全性により高信頼のプログラム記述が可能で、型推論機構によりプログラムを簡潔に記述することができる言語である。OCaml で書かれた実用的なプログラムの例として、Windows や UNIX で利用できるディレクトリ同期ツールの Unison [2] がある。また航空機メーカーのエアバスでも使われている C プログラムの静的解析プログラムである ASTRÉE [3] は約 44000 行と非常に大きいプログラムである。

大きなプログラムを記述する上でデバッグは必要不可欠であり、デバッグを行うときにはデバッグ出力を行うことが有効である。デバッグ出力とはプログラム実行時のデータの値を出力することで、デバッグ出力において型によらず出力できる共通の機構があると便利である。そのような機構の例として、Ruby [4] の p メソッドや Haskell [5] の show 関数などがあるが、標準の OCaml では提供されていない。よって OCaml でデバッグ出力を行う場合、プログラマが出力したいデータの型ごとに出力関数を定義して、出力したい場所に出力関数を挿入する必要があり手間がかかる。

2　目的・方針

本研究では、OCaml におけるプログラミング支援のためのデバッグ出力機構の実現を目指す。

方針としては次の 2 つの機構を実装する。

- 型ごとに出力関数を自動生成する機構
- 型に合った出力関数を自動挿入する機構

実装の方法としては、OCaml の ppx 機能を用いて構文木の書き換えを行う。

3　構文拡張機能 ppx

ppx とは OCaml の構文木の書き換えを行う機能である。通常のコンパイルでは、コードを字句解析と構文解析してできた構文木をそのまま意味解析しコード生成して実行ファイルを生成する。ppx 機能を用いると、図 1 に示すように構文木を書き換えるフェーズを意味解析の前に挿入することができる。この機能を用いることで、コンパイラに直接手を加えずに処理を拡張することができる。

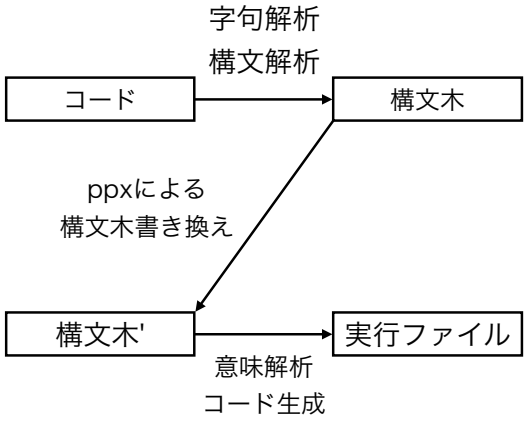

図 1　ppx を用いたコンパイルの流れ

4　設計・実装

設計としては、このシステムを使うユーザーは、変数に対してマーカー `[@p]` をつけることで、その変数を出力できるようにする。

具体的には次の 2 つの機能を実装する。

- 標準のデータ型とユーザー定義のデータ型に対して出力関数を自動生成する機能
- マーカー `[@p]` に対応する変数を出力するための出力関数を適切に自動挿入する機能

図 2 に例を示す。

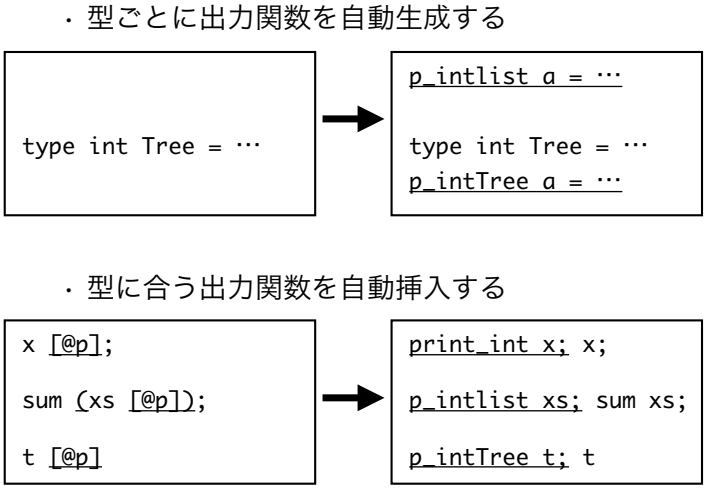

図 2　実装例

図 2 で type はユーザー定義型の宣言である。上では標準のデータ型である int 型のリストの出力関数と、ユーザー定義型である int 型の木の出力関数を自動生成している。また下では、マーカーのつけられた変数に対応した出力関数を挿入している。マーカーは式に対してつけられるので、関数適用式中の引数などを出力したい場合は括弧が必要である。

5 関連ツール

pa_print [6] は本研究と同様にデバッグ出力関数を自動で挿入するツールである。このツールが作られた時点ではまだ ppx 機能が実装されていなかったので、型情報を得るために 2 回コンパイルを行っている。

ppx_implicits [7] は ppx を用いて、型から関数を決定する機構や型クラスの実装方法を提供している。

本研究ではこれらのツールを参考にして、型に合った出力関数の自動挿入を、1 回のコンパイルで行う機構を実装する。

6 現状と今後

現状としては、ppx における構文木の書き換え方法の調査と関連ツールの調査および試用を行った。今後は型ごとに出力関数を自動生成する機能と型推論を利用した出力関数の自動挿入機能の実装、コードの減少による有効性の評価を行う。

参考文献

[1] OCaml : https://ocaml.org
[2] Unison : http://www.cis.upenn.edu/~bcpierce/unison/
[3] ASTRÉE : http://www.astree.ens.fr
[4] Ruby : https://www.ruby-lang.org/ja/
[5] Haskell : https://www.haskell.org
[6] pa_print [Nakano '13]
[7] ppx_implicits [Furuse '15] :
https://bitbucket.org/camlspotter/ppx_implicits