

関数型言語の組み込みソフトウェア開発への適用に向けて

今井 敬吾 (有限会社 IT プランニング)

1 はじめに

弊社は関数型プログラミング言語によるソフトウェア開発を指向している。関数型言語^{*1}のプログラミング言語としての特徴は (1) 型安全性 (2) 型推論 (3) パラメトリック多相 (4) 高階関数を用いた関数プログラミングである。本稿では関数型言語の信頼性とそれ以外のいくつかの観点から関数型言語の概略を述べる。さらに、組み込みソフトウェアの開発に関数型言語をする際の課題について述べる。

2 関数型言語の信頼性

関数型言語の特徴の一つは高い信頼性である。関数型言語で作成したソフトウェアでは、コアダンプや `NullPointerException` に類する実行時エラーが発生しない。この理由は、関数型言語ではヌル値や未初期化の値が存在し得ないためである。

このような関数型言語の信頼性は型を中心とした言語設計により成立している。関数型言語では、C 言語の `enum` 型に相当する概念を拡張した代数的データ型という種類の型を使うことができる。代数的データ型は C 言語の `switch` 文に相当するパターンマッチの機能で場合分けを記述できる。このとき、場合分けが網羅的でないとコンパイラが警告メッセージを表示して、分岐記述の抜け漏れを指摘してくれる。

関数型にはヌル値がない、と聞いて、読者諸氏には、型にヌル値を許したい時はどうするのか、という疑問を持つ方もおられよう。そのような局面では、データが内容をもたない場合を表現できる `option` 型 (OCaml) や `Maybe` 型 (Haskell) で既存の型から新しい型を合成する。これらの型は代数的データ型であり、ヌルチェックの抜け漏れはコンパイラにより検出できる。

3 産業界と関数型言語

関数型言語は主にヨーロッパの大学で開発された `Haskell` と `OCaml` がよく知られている。`Haskell` と `OCaml` は外資系の投資銀行において特に利用されている。プログラムの誤りが損失に直結するため、関数型言語の高信頼性が好まれている。ここ数年では、Java の実行環境で動作する `Scala` が Twitter で利用されており注目を浴びている。さらに、Microsoft が Visual Studio 2010 に `F#` を導入した。

4 組み込み開発への導入における課題

4.1 技術的側面

大きな問題点はリソースの制約である。関数型言語の利点は自由に関数を扱えることであるが、このような言

語においてはガベージコレクションが必須である。ガベージコレクションのオーバーヘッドを許容できる程度の環境でないと導入はきわめて困難である。さらに、`Haskell` に代表される遅延評価の言語はメモリ消費量と実行時間の予測が難しい場合がある。これらの条件から、`OCaml` と `F#` が組み込みソフトウェア開発のために有力な候補ではないかと筆者は考える。`OCaml` は現代的で効率的な世代別かつインクリメンタルなガベージコレクションを備えており、かつその歴史は古く枯れたランタイムを持つ。`F#` は .NET Compact Framework により組み込みソフトウェア開発に直接適用できる。

4.2 管理的側面

関数型言語の教育は一部の大学でしか行われていないため、はじめから関数型言語を扱えるプログラマーを集めることは困難である。しかしながら関数型言語に関する書籍は多く出版されており、今後は関数型セミナーに関するセミナーも増えると予想される。一例として、名古屋ソフトウェアセンターは弊社人員を講師として 2010 年 9 月に `F#` のセミナーを行う。また、`OCaml` や `F#` と手続き型言語はそれなりに似通っており、再代入可能な変数やクラス・オブジェクト等を標準で備えているため、移行のコストは低減できる。

5 OCaml と CIL による C 言語ソースコードの構文解析

組み込み分野における関数型言語のもう一つの有望な利用法として、C 言語のソースコードの構文解析がある。`OCaml` では `CIL`[2] というライブラリを使用して C 言語のソースコードを構文解析できる。`CIL` は C 言語のソースコードのモデル検査器 `BLAST`[1] など多数のプロジェクトで用いられている。さらに、`CIL` を用いたカバレッジテスト自動実行ツールも存在する。

著者について

2009 年 3 月名古屋大学情報科学研究科博士後期課程単位取得退学。同年 4 月名古屋大学組み込みシステム研究センター研究員。2010 年 3 月任期満了に伴い退職。2010 年 4 月より有限会社 IT プランニング (<http://www.itpl.co.jp/>) 勤務。2007 年 5 月、書籍 入門 `OCaml`[3] を `OCaml-Nagoya` 名義で執筆。

参考文献

- [1] `BLAST`: Berkeley Lazy Abstraction Software Verification Tool. <http://mtc.epfl.ch/software-tools/blast/index-epfl.php>.
- [2] `CIL` (C Intermediate Language). <http://cil.sourceforge.net/>.
- [3] `OCaml-Nagoya`. 入門 *OCaml* ~プログラミング基礎と実践理解~. 毎日コミュニケーションズ, 2007.

*1 Lisp 等の静的型検査のない言語はここでは議論しない。