まえがき

Haskell でプログラミングするのは楽しい。ところで「Haskell をプログラミングする」のも楽しいのでは？ じゃあやってみよう。というわけで、Haskell からヒラヒラを全て削り落して削りすぎたような関数型言語を作る。

前提知識

以下の知識があると本書をスムーズに読み進められるだろう。

- 入門を終わったレベルの Haskell プログラミング
- 入門を終わったレベルの TypeScript プログラミング
- 入門レベルの GoF デザインパターン
- 文脈自由文法
- 木構造やグラフ構造の用語

本書の概要

関数型言語の構文を設計し、意味を定義し、これらの定義に基づくインタプリタを Haskell と TypeScript で実装する。

Haskell の特徴は遅延評価であるが、処理系を考える場合に直観的で理解の容易な正格評価な処理系にまずは取り組む。1 章で整数を組込型として持つラムダ計算の言語を設計し、Haskell で正格評価のパーサとインタプリタを実装する。2 章で同じインタプリタを TypeScript で実装する。4 章でこのインタプリタのためのパーサを実装する。その準備として、3 章で TypeScript でパーサを作るためのライブラリを構築する。

次に、言語に代数的データ型を追加する。これは代数的データ型の値を分解するためのパターンマッチを必然的に伴う。5 章で、言語の拡張と両言語での正格評価インタプリタの実装を行う。

以上を準備として、遅延評価をする処理系に取り組む。ただし代数的データ型についてはいったん脇に置いて、1 章相当の言語を対象に遅延評価のインタプリタを 6 章にて両言語で実装する。正格評価で計算される整数型の組込演算との協調を実現する。

最後に、遅延評価でのパターンマッチを 7 章にて実現する。正格評価と異なり、遅延評価ではパターンマッチの評価の方法は 2 つの異なるものが必要になる。このマッチの機構

と遅延評価が協調して計算を進行させる。

本書に入らなかった内容

以下の内容については、続編での執筆を予定している。

- その強力な型システムは Haskell から削り落としてはいけない本質であるが、本書では間に合わなかった。表題は「看板に偽りあり」といえる。
- *IO* モナドにより、純粋でない計算を全てその中に封じ込める機構は興味深い Haskell の構成要素である。
- なくてはならないものではないが、Haskell らしさを特徴づけるものとして、インデント構文がある。
- 本書ではソースコードを直接解釈するインタプリタのみを扱っているが、言語処理系のもう一つの重要な形態は、ソースコードに対して同じ計算を行う別の言語のプログラムを生成するコンパイラである。その小規模な例として、JavaScript をターゲットとするトランスパイラを実現する。

フォントとか

和文書体は源ノ明朝・源ノ角ゴシックを使っている。Haskell のソースは `Courier`、TypeScript のソースは `Bitstream Vera Mono`、本書で作成する言語のソースは `Computer Modern Typewriter`、対話コンソール画面は `k14 (JFﾄﾞｿﾙK14)` で示している。

オレは誰？[*1]

関数型帝国名古屋県の東の外れにある愛・地球博記念大学でプログラミング入門と題して右も左もわからない新入生に Haskell を教えている悪人。

[*1] 「オレは誰？」でググってみてください。全国区かと思ったら愛知ローカルなんですねこれ。