B-22　関数型プログラミング学習のためのリスト処理関数の振る舞いの可視化

ビジュアルインタフェース研究室　　瀧屋　考平

1. 背景と目的

近年, 関数型プログラミングの利用が拡大している. 関数型プログラミングとは, 数学的な意味に近い関数の組み合わせでプログラムを記述するプログラミング方法である. 関数型プログラミングには, 高階関数や遅延評価などの通常の命令型プログラミングにはない特徴がある.

関数型プログラミングの学習は, プログラムの処理の流れを把握することが難しいため, Java や Python などの命令型プログラミングを習得している者でも難しいと言われている.

そこで本研究では, 関数型プログラミングの初学者を対象に, 共同研究者の関口が開発したシステム[5]を使用する. システム[5]で作成した Haskell のプログラムを用いて, 関数型プログラミングの特性やリスト処理関数の振る舞いを可視化して見せることで, 関数型プログラミングの学習を支援するシステムの開発を目指す.

2. 関連研究

粟野[1]は, 関数型プログラミングにおいて, 関数や引数をブロックで表し, ブロックを組み合わせて関数の状態を表示することで, 高階関数の理解を支援する手法を提案した.

竹内ら[2]は, 関数型プログラムの実行過程を置き換えモデルによって段階的にアニメーションで表示することで, 学習者の理解を支援する手法を提案した.

志村[3]は, 本研究室の卒業研究で, 高階関数と遅延評価の処理の流れをアニメーションで表示することで, 関数型プログラミングの学習を支援する手法を提案した.

浅川ら[4]は, 関数型言語 Haskell をベースとした初学者向けビジュアルプログラミング環境を開発し, 初学者自身による実装とテストの実行によってプログラムの動作に対する理解を支援する手法を提案した.

3. 本研究の提案

本研究の目的は, 関数型プログラミングの学習を支援するユーザインタフェースの開発である. 昨年度の卒業研究で開発されたアプリケーション[3]は, 学習者が入力したソースコードに対応する可視化を表示するのではなく, 関数型プログラミングの機能に合わせたソースコードの例を表示するものであった. そのため本研究では, 学習者が入力したプログラムに対応したリスト処理関数の振る舞いの可視化を表示する学習支援システムを開発する.

本研究では, 学習者による Haskell のプログラムの作成に共同研究者の関口が開発した入力インタフェース[5]を利用し, 入力されたプログラムの処理の構造を取得して, それに対応した図解とアニメーションによる可視化を行うシステムを開発する.

本システムによって, 学習者は自分が入力したプログラムの可視化を見ることで, 関数がどのタイミングで実行され, どのような処理が行われているかを理解することが期待される.

4. システムの概要

本システムは, 学習者が入力した Haskell のプログラムに対応した図解とアニメーションを表示する. 本研究では, filter 関数, map 関数の 2 つのリスト処理関数に関する振る舞い

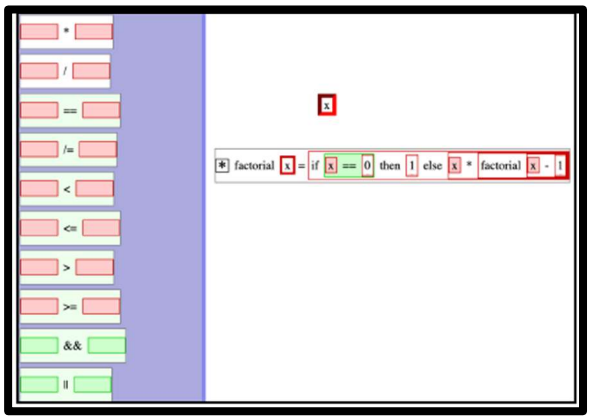

図 1 文献[4]のアプリケーションの実行画面

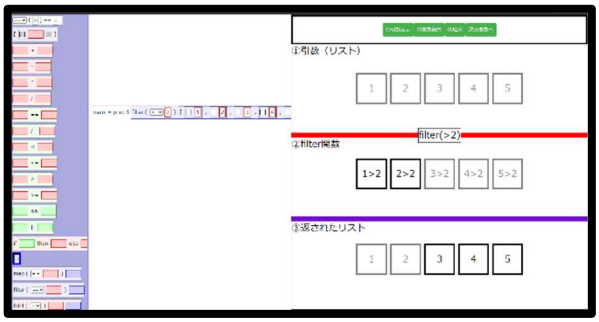

図 2　filter 関数の実行画面

を可視化する機能を実装した.

本システムは, リストを表示する 3 つのエリアとそれぞれのエリアを制御するボタンを設置するエリアの合計 4 つのエリアで構成されている. ①のエリアでは, 関数に適応させるリストを表示する. ②のエリアでは, 関数が適応されているリストを表示する. ③のエリアでは, 関数が適応されたリストを表示する.

4.1 リスト処理関数の振る舞いの可視化

図 2 は map 関数の実行画面の例である. 図 2 の下部の一番左の「①引数表示」ボタンが押されると, ①のエリアで入力インタフェースから受け取ったリストが先頭から順に表示される.

同じく左から 2 番目の「②関数適用」ボタンが押されると, ②のエリアで受け取ったリストに対して行う処理を追加したリストが表示される.

同じく左から 3 番目の「③結果」ボタンが押されると, 長破線で囲われたエリアで関数が適応されて返されたリストが先頭から順に表示される. 図 2 の filter 関数の実行例では, 条件を満たす要素をそのまま表示し, 条件を満たさない要素を半透明にすることで条件を満たしているかどうかを視覚的に区別できるように表現した.

同じく一番右の「次の関数へ」ボタンが押されると, 返されたリストに対して再び適応させる関数がある場合, 次の関数を可視化する画面に移行する. その後, ボタンで操作することで, リスト処理関数の図解とアニメーションを表示する.

4.2 遅延評価の表現方法

本システムの可視化では, 関数が適応された後に返されるリストに格納される値の表記方法が, 要素が評価されるタイミングを表現している. 図 3 の可視化は, 要素内に計算の解が値として格納されているため, 演算がすでに評価されていることを表している. 図 4 の可視化は, 要素内が式の状態で計算が行われていないため, まだ評価されていないことを表している.

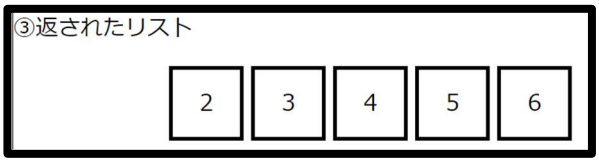

図 3 評価されたリスト

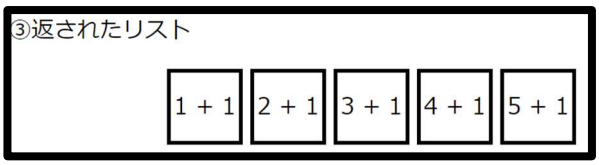

図 4 評価されてないリスト

5. まとめ

本研究では, 学習者が入力インタフェースで作成した Haskell のプログラムに対応した可視化を表示することで, 関数型プログラミングの学習を支援するシステムを開発した. これにより, 高階関数の基本的な処理の流れと遅延評価を可視化することができた.

参考文献

[1] 粟野, 初学者のための高階関数に着目した Haskell ソースコード可視化手法, 平成 27 年度愛知県立大学卒業論文, 2015.

[2] 竹内, 酒井, 置き換えモデルによる Scala の実行過程の可視化, 第 79 回情報処理学会全国大会, pp.687-688, 2017.

[3] 志村, 関数型プログラミング学習のための高階関数の振る舞いの可視化, 令和 4 年度玉川大学卒業論文, 2022.

[4] 浅川, 田中, ユニットテストに注目した関数型ビジュアルプログラミング学習環境, 電子情報通信学会技術研究報, Vol. 123, No.124, pp.71-75, 2023.

[5] 関口, プログラムの可視化に対応した関数型プログラミング学習環境に関する研究, 令和 5 年度玉川大学卒業論文, 2023.