4K-06

関数型言語の諸概念を取り入れたシェル

眞田 龍史 †
電気通信大学 大学院情報理工学研究科 †

小宮 常康 ‡
電気通信大学 大学院情報理工学研究科 ‡

1 はじめに

関数型言語におけるプログラミングでは、高級で直交性が高い機能の組み合わせによってプログラムが記述される。少ない数の機能でも組み合わせによって数多くの処理を実現することができ、高い表現力を持つ特徴がある。関数合成では関数間のデータのやり取りを引数渡しと返り値によって行う。一方、Unix シェルのパイプも一種の関数合成機能であるが、関数間（コマンド間）でやり取りするデータはバイトストリーム（標準入出力）であるため、パイプを流れるデータを様々な一級のデータの形で直接やり取りをすることができない。そこで Unix シェルのパイプにおいても一級のデータをやり取りできるように、引数と返り値を接続するパイプを導入する。ここでの引数と返り値は一級のデータのストリームである。また、ストリーム自体も一級のデータであるため、ストリームのストリームも扱うことが可能となる。本稿ではシェルに関数型言語の持つ表現力を取り入れるための手法について述べる。

2 提案するシェル

提案するシェルは現状では Scheme 言語でマクロを使用することによって実現されている。例えば `ls -l` コマンドの実行結果についてファイルサイズが 1000 以上であるファイル（行）のみ表示するコマンドは図 1 のように記述される。`ls` コマンドは入出力として標準入出力を持つが、本シェルでは `ls` は `(define (ls stream arg1 ...) ...)` のように関数として定義され、引数/返り値によってデータのやり取りが行われる。このため、標準入力とするストリームは第一引数から受け取り、標準出力のストリームは返り値として返される。第二引数以降で受け取った引数はコマンド引数として扱われる。このため、`ls` 関数の中では `ls -l` というコマンドが実行される。コマンドだけでなく `Unixtolist` などの他の関数も第一引数でストリームを受け取るように統一されている。このため、pipe は関数の返り値を次の関数の第一引数に与えるように関数合成を行う。`:in` は仮の引数であり、pipe 式によって前の関数の返り値が与えられることを示す。

```
1 (pipe
2   (ls *stdin* "-l")
3   (Unixtolist :in)
4   (cdr/s :in)
5   (filter/s :in
6     (lambda (x)
7       (>= (caddr (cddr x)) 1000))))
```

図 1 提案するシェルにおけるパイプの利用例

例における各関数の返り値は図 2 で表現したような値になっている。まず `ls` 関数の返り値は本来バイトストリームであるが、一級のデータではないため図では文字のストリームとして表現する。`Unixtolist` 関数ではバイトストリームをリストに変換する。リストの要素はコマンドの 1 行分のデータに対応する。また、1 行分のデータはスペースで区切られ、各文字列や数値を要素とするリストとなる。`cdr/s` はストリーム用の `cdr` であり、ストリームの `cdr` 部に相当する部分を返す。`filter/s` 関数によってストリームの各要素（リスト）に対して第二引数の述語が与えられ、述語の結果が真となった要素のみストリームの要素として残す。述語はリストの 5 番目の要素（ファイルサイズ）が 1000 以上であるか判定を行うものであり、それを満たさない

Introducing concepts of functional languages to UNIX shell

† Ryuji Sanada, Graduate School of Informatics and Engineering, The University of Electro-Communications
‡ Tsuneyasu Komiya, Graduate School of Informatics and Engineering, The University of Electro-Communications

```
('t' 'o' 't' 'a' 'l' ...)
→ (("total" 184)
    ("-rw-rw-rw-" 1 "a" "a" 166 "Sep" 6 "15:38" "a.txt")
    ("-rwxr-xr-x" 1 "a" "a" 8408 "Aug" 29 "16:19" "b")
    ...)
→ (("-rw-rw-rw-" 1 "a" "a" 166 "Sep" 6 "15:38" "a.txt")
    ("-rwxr-xr-x" 1 "a" "a" 8408 "Aug" 29 "16:19" "b")
    ...)
→ (("-rwxr-xr-x" 1 "a" "a" 8408 "Aug" 29 "16:19" "b")
    ...)
```

図 2　ls を使った例におけるパイプを流れる値

要素は filter/s 関数によって除外される。

本シェルではストリームが一級のデータであることから、ストリームのストリームを作ることができる。マージソートをストリームのストリームを使って記述した例を次に示す。

```
1  (pipe
2    (random *stdin* 100)
3    (split :in 4)
4    (map/ss :in
5      (pipe
6        (streamtoUnix :in)
7        (sort :in "-n")
8        (Unixtostream :in "-d" "num")))
9    (map-2/ss :in (pipe (stream-merge
10     :in1 :in2 <)))
11   (map-2/ss :in (pipe (stream-merge
12     :in1 :in2 <)))
13   (car/ss :in))
```

この例においてパイプを流れる値は図 3 のように表現される。random 関数によって 1〜100 までの数値がランダムに並べられた数値ストリームが生成される。これが split 関数によって 4 つのストリームに分割され、ストリームのストリームが作られる。map/ss 関数はストリームのストリームに対して map 処理を行う関数である。与えられたストリームの要素はストリームであるから、関数ではなく pipe 式が適用する。要素の各ストリームは pipe 式の先頭のコマンドの:in に与えられ、それぞれストリームの処理が行われる。その中で各数値ストリームは sort コマンドによって昇順に整列される。これにより、map/ss の返り値として整列された数値ストリームのストリームが得られる。次の map-2/ss 関数はストリームのストリームに対して、2 要素ごとに pipe 式を適用する関数である。このため stream-merge 関数では:in1 と:in2 に数値ストリームがそれぞれ与えられ、それらをマージした一つの数値ストリームが返り値として返される。このため、10 行目の map-2/ss の返り値は要素として二つのストリームを持つストリームとなる。11 行目でも同様のことが行われて一つのストリームを要素に持つストリームが返される。car/ss で要素のストリームを返すことにより、整列された数値ストリームを得ることができる。

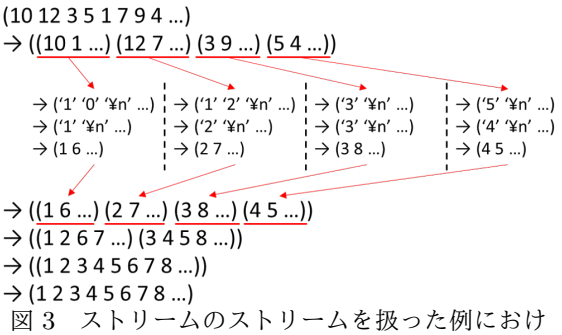

図 3　ストリームのストリームを扱った例におけるパイプを流れる値

3　関連研究

Scheme Shell[1] はプロセスフォームの導入によって複数のコマンドの標準入出力をを合成でき、その中に Scheme プログラムを記述することができるが、パイプはバイトストリームしか扱うことができず、高級なデータを直接扱えない。F shell[2] は直交性が高く高級な機能の組み合わせでコマンドを記述することができる。また、構造化ストリームによってストリームのストリームを扱うことができるが、外部コマンドとの統合を考慮して作られていない。PowerShell など高級なデータ構造をストリームで扱うことができるシェルも存在する。

4　今後の課題

本シェルは Scheme 言語の構文をベースとした構文になっているため、Unix シェルの構文に近い表層構文を設計することが課題としてあげられる。また、pipe 式にプロセス生成機能をつけることによって、ストリームのストリームにおいて各ストリーム処理を並列に動作させるように実装を行う。

参考文献

[1] Olin Shivers, A Scheme shell, Technical report, Massachusetts Institute of Technology, 1994.
[2] Shultis, Jon, A Functional Shell, ACM SIGPLAN, 1983.