システムソフトウェア特論 課題#4

学籍番号:1831013 泉 祐太

2020 年 2 月 17 日

1　取り組んだ課題

このレポートで扱う課題は、第 4 回講義の演習 4-1(c) である。

課題の内容は、『”a+(b+—c)*d+”の正規表現の認識器を作成せよ』というものであった。

この正規表現をオートマトンで表現する場合には状態が 5 つ (初期状態、a...d の 4 文字に対して 1 つずつ) 必要なので、これらの状態に対して遷移条件を記述する方針でプログラムを作成した。資料にある Sam41 クラスからの変更点を以下に示す。

```
while(true) {
  switch(stat) {
    case 0: if(tok.chkfwd("a")) { stat = 1; continue; }
            break;
    case 1: if(tok.chkfwd("a")) { stat = 1; continue; }
            if(tok.chkfwd("b")) { stat = 2; continue; }
            if(tok.chkfwd("c")) { stat = 3; continue; }
            if(tok.chkfwd("d")) { stat = 4; continue; }
            break;
    case 2: if(tok.chkfwd("b")) { stat = 2; continue; }
            if(tok.chkfwd("d")) { stat = 4; continue; }
            break;
    case 3: if(tok.chkfwd("d")) { stat = 4; continue; }
            break;
    case 4: if(tok.chkfwd("d")) { stat = 4; continue;}
            if(tok.chkfwd("$")) { System.out.println("accept."); return; }
            break;
    }
    System.out.println("error."); return;
}
```

各状態での遷移条件を以下に示す。

- 初期状態: a →状態 1
- 状態 1: a →状態 1, b →状態 2, c →状態 3, d →状態 4
- 状態 2: b →状態 2, d →状態 4
- 状態 3: d →状態 4
- 状態 4: d →状態 4 (eof なら受理)

この状態 1〜4 が case1〜4 に対応し、初期状態が case0 に対応する。

1.1 実行例

```
E:\programs\java>java Sam41c abddd
accept.
E:\programs\java>java Sam41c aaddd
accept.
E:\programs\java>java Sam41c aaabbddd
accept.
E:\programs\java>java Sam41c aaacddd
accept.
E:\programs\java>java Sam41c aaaddd
accept.
E:\programs\java>java Sam41c aaaccddd
error.
E:\programs\java>java Sam41c aaabcddd
error.
```

b と c が同じ文字列にあったり、c が複数個ある入力を排除できていることが確認できる。

2　アンケート回答

A1.

形式言語は学部生の頃に何かの講義で学習したような気がするが、あまり定かではないので、形式言語について知っていることはあまりない、といった状況である。正規表現についてはもともと興味はあった。

A2.

オートマトンにより認識器は比較的親しみやすいタイプだったと感じる。もっともこれは学部生の頃に受講したオートマトン理論 (だったと思う) の講義に際して購入した参考書によって、オートマトンの構成が苦でないことが理由としてあると考えている。CYK による認識器はかなり難しいと感じたし、いまだにきちんと理解できていない。

A3.

最低限、前半のオートマトンによる認識器について学習できたことはよかったと思うし、形式言語、特に正規表現について学習できたことは今後も応用がききそうだなと感じている。実行速度が劣るかもしれないが、Python でも実装してみようと思った。