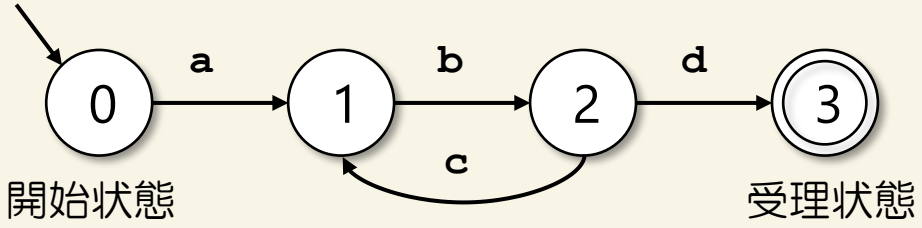
2022年度

C++プログラミングII

第1回レポート課題

課題

決定性有限オートマトンを表すクラス**Automaton**を作成したい．オートマトンの「文字」として文字型以外にも論理型や整数型が利用できるように，クラステンプレートとして実装する．決定性有限オートマトンの例を以下に示す．



状態は整数型，文字はAutomatonの型パラメータ**T**（**T** = 文字型，論理型，整数型のいずれか）によって表現する．決定性有限オートマトンの詳細についてはwebサイト https://w.wiki/5976 などを参照せよ．

1. 以降に示すファイル**automaton.hpp**内のメンバ関数を実装し，**Automaton**の定義を完成させよ．また，テスト用ファイル**automaton-test.cpp**をコンパイル・実行し，実装の正しさを確認せよ．
2. オートマトンの構成を出力する**<<**演算子の多重定義を**Automaton**のクラス定義に追加せよ．出力形式はスライドp.9に示す「期待する実行結果」が得られるようにすること．

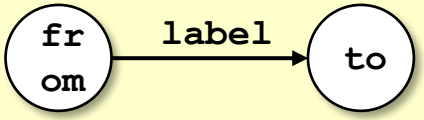
定義ファイル`automaton.hpp`（1）

```
#include <vector>
#include <set>
#include <map>
#include <iostream>

template<typename T>
class Automaton {
  int start_state;                                    // 開始状態
  std::set<int> accept_states;                        // 受理状態集合
  std::map<std::pair<int, T>, int> transitions;       // 遷移関数

public:
  Automaton<T>() {}

  // 開始状態を取得する（本課題では使用しない）
  int get_start_state() {
    return start_state;
  }

  // 開始状態を設定する
  void set_start_state(int q) {
    start_state = q;
  }
```

定義ファイル**automaton.hpp**（2）

```
// 受理状態集合を取得する（本課題では使用しない）
std::set<int> get_accept_states() {
  return accept_states;
}

// 受理状態を追加する
void add_accept_state(int q) {

    関数定義を完成させる

}

// 遷移関数を取得する（本課題では使用しない）
std::map<std::pair<int, T>, int> get_transitions() {
  return transitions;
}

// 遷移関数に遷移を追加する
void add_transition(int from, T label, int to) {

    関数定義を完成させる

}
```

遷移関数**transitions**に指定された遷移



を追加する

定義ファイル**automaton.hpp**（3）

```
  // 状態qが受理状態であればtrue，そうでなければfalseを返す
  bool is_accept_state(int q) {

      関数定義を完成させる

  }

  // 入力文字の列csが受理可能であればtrue，そうでなければfalseを返す
  bool accept(std::vector<T> cs) {

      関数定義を完成させる

  }

  課題2の<<演算子の多重定義は
  ここに加える

};
```

入力文字に対応する遷移先が存在しない場合は受理不可能と判定する．
例: p2.のオートマトンに対する入力文字の列**a c**は，状態1で**c**に対応する遷移先が存在しないため受理不可能と判定する．

状態および文字は，引数として受け取る**std::ostream**型の出力ストリームオブジェクトにそのまま**<<**演算子で出力すればよい（**true**は**1**，**false**は**0**と表示されるが問題ない）．

テスト用ファイルautomaton-test.cpp（1）

```
#include "automaton.hpp"
#include <iostream>

template<typename T>
void test(Automaton<T>& A, const std::vector<T>& cs, bool expected) {
  for(auto c : cs) {
    std::cout << c << " ";
  }
  bool result = A.accept(cs);
  std::cout << "-> "
    << (result? "accepted" : "rejected") << " "
    << (result == expected? "OK" : "NG") << std::endl;
}
```

テスト用ファイルautomaton-test.cpp（2）

```
int main() {
  Automaton<char> A1;
  A1.add_transition(0, 'a', 1); A1.add_transition(1, 'b', 2);
  A1.add_transition(2, 'c', 1); A1.add_transition(2, 'd', 3);
  A1.set_start_state(0); A1.add_accept_state(3);

  Automaton<bool> A2;
  A2.add_transition(0, true, 1); A2.add_transition(0, false, 2);
  A2.add_transition(1, true, 3); A2.add_transition(1, false, 3);
  A2.add_transition(2, true, 3);
  A2.set_start_state(0); A2.add_accept_state(1); A2.add_accept_state(3);

  std::vector<std::vector<char>> inputs1 {
    { 'a', 'b', 'c', 'b', 'd' },
    { 'a', 'b', 'c', 'b', 'c', 'd' },
    { 'a', 'b', 'c', 'b', 'c', 'b', 'd' },
    { 'a', 'b', 'c' }
  };
  std::vector<bool> expected1 { true, false, true, false };
  for(int i = 0; i < inputs1.size(); ++i) {
    test(A1, inputs1[i], expected1[i]);
  }
```

テスト用ファイル automaton-test.cpp（3）

```
  std::vector<std::vector<bool>> inputs2 {
    {},
    { true },
    { false, false },
    { false, true }
  };
  std::vector<bool> expected2 { false, true, false, true };
  for(int i = 0; i < inputs2.size(); ++i) {
    test(A2, inputs2[i], expected2[i]);
  }

  std::cout << A1;
  std::cout << A2;
  return 0;
}
```

課題2の確認用．課題1を解く際はいったんコメントアウトしておくとよい

期待する実行結果

```
a b c b d -> accepted OK
a b c b c d -> rejected OK
a b c b c b d -> accepted OK
a b c -> rejected OK
-> rejected OK
1 -> accepted OK
0 0 -> rejected OK
0 1 -> accepted OK
start state:0 accept states:3
transitions:
0 -[a]-> 1
1 -[b]-> 2
2 -[c]-> 1
2 -[d]-> 3

start state:0 accept states:1 3
transitions:
0 -[0]-> 2
0 -[1]-> 1
1 -[0]-> 3
1 -[1]-> 3
2 -[1]-> 3
```

課題1の出力

課題2の出力
（trueは1，falseは0と出力）

レポートの構成

表紙

タイトル（C++プログラミングII 第1回レポート課題）
学籍番号，氏名，提出年月日

本文

1. レポート課題の説明

2. 課題1の解答
   メンバ関数**`add_accept_state`**, **`add_transition`**, **`is_accept_state`**, **`accept`**の実装コード，および各関数の処理内容の説明

3. 課題2の解答
   <<演算子の実装コード，および処理内容の説明

4. 感想など

提出方法

- Course Powerの「第1回レポート課題」内にあるレポート「第1回レポート課題」から以下を提出する
  - レポートファイル（pdf形式）
  - 作成したソースコード

- レポートファイル名はreport1-学籍番号-氏名.pdfとする
  - 学籍番号と氏名のところは，自分のものに置き換える

- 締め切り: 6/5（日）23:59