Aufgabe 6: Testrahmen für Aufgabe 5

Erstellung eines Testrahmenns im Stile von "count" wobei die Testüberprüfung automatisiert durch ein sogenanntes Testorakel erfolgt. Wie auch bei Aufgabe 5 sind viele Teile schon vorgegeben.

Hintergrund: Standard Unit Test

Vorgehen ist im allgemeinen wie folgt.

1. Setzen der Eingabe.
2. Ausführung.
3. Protokollierung des Ergebnisses.
4. Überprüfe ob erwartetes Ergebis gleich protokolliertem Ergebnis.

Die Testausführung und Protokollierung lässt sich einfach automatisieren. Die Generierung von Testfällen (Eingaben) und deren Überprüfung muss aber oft manuel vorgenommen werden. Dies ist zeitaufwendig und fehleranfällig.

Wir betrachten die automatisierte Überprüfung des Ergebnisses mit Hilfe eines Testorakels.

Testorakel

Im Fall von Aufgabe 5 möchten wir garantieren, dass die Simplifizierung semantisch korrekt sind. D.h. der simplifizierte reguläre Ausdruck ist gleich dem originalen Ausdruck. Wir sind auf der Suche nach einem Testorakel welches die Überprüfung durchführt. Die Idee ist wie folgt.

Unser Testorakel bekommt drei Eingabewerte.

1. Den originalen Ausdruck R.

2. Den simplifizierten Ausdruck RSimp.

3. Einen beliebigen String W.

Das Testorakel überprüft, ob der String W in R enthalten ist. Falls ja, muss W auch in RSimp enthalten sein und umgekehrt. Ähnliches gilt falsl W nicht in R enthalten ist.

Wie können wir testen, ob ein String W in R enthalten ist? Wir könnenten einen

Automaten aus R bauen und überprüfen, ob der Automata W akzeptiert.

Wir verwenden einen anderen Ansatz der auf Brzozowski's Derivaten beruht. Die theoretischen Grundlagen dazu werden in der Vorlesung besprochen. Hier ist die notwendige Implementierung.

```
RE* deriv(RE* r, char l) {

  switch(r->ofType()) {
  case PhiType:
    return r;
  case EpsType:
    return new Phi();
  case ChType:
    if (((Ch*)r)->getChar() == l) {
      return new Eps();
    }
    else {
      return new Phi();
    }
  case StarType: {
    RE* r1 = ((Star*) r)->getRE();
    return new Conc(deriv(r1,l),r);
  }
  case AltType: {
    RE* r1 = ((Alt*) r)->getLeft();
    RE* r2 = ((Alt*) r)->getRight();
    return new Alt(deriv(r1,l), deriv(r2,l));
  }
  case ConcType: {
    RE* r1 = ((Conc*)r)->getLeft();
    RE* r2 = ((Conc*)r)->getRight();
    if(r1->containsEps()) {
      return new Alt(new Conc(deriv(r1,l),r2), deriv(r2,l));
    }
    else {
      return new Conc(deriv(r1,l),r2);
    }
  }

  }// switch

}

bool match(RE* r, string s) {
  for(int i=0; i < s.length(); i++) {
    r = deriv(r, s[i]);
  }
  return r->containsEps();
}

bool orakel(RE* r, RE* rSimp, string s) {
    return (match(r,s) == match(rSimp,s));
}
```

Eigentliche Aufgabe

Ihre eigentliche Aufgabe besteht darin den Testrahmen auszuarbeiten und ihre

Implementierung in Aufgabe 5 mit Hilfe einer hinreichenden Anzahl von Eingaben zu testen.