Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Theoretische Informatik

Sommersemester 2023
2. Aufgabenblatt

Aufgabe 3 (Ausführung einer Turing-Maschine [klausurtypisch], 5+2 Punkte)

a) Gegeben sei die Turing-Maschine auf der rechten Seite. Geben Sie für die Startworte abc und ε die zugehörigen Berechnungen als Folge von Konfigurationen an, bis keine Folgekonfiguration mehr möglich ist.
b) Was vermuten Sie, was die Turing-Maschine generell macht?

```
% Zustaende
Z: Start z1 z2 z3
% Alphabet, Leerzeichen # automatisch dabei
A: a b c
% Start
S: Start
% Ueberfuehrungsfunktion
% alt   lesen   neu  schreiben Richtung
Start     #     z1      #         L
z1        a     z2      b         L
z1        b     z2      b         L
z1        c     z2      c         L
z1        #     z3      #         R
z2        a     z1      b         L
z2        b     z1      b         L
z2        c     z1      c         L
z2        #     z3      #         R
z3        a     z3      b         R
z3        b     z3      b         R
z3        c     z3      b         R
z3        #     z3      #         S
```

Aufgabe 4 (Erstellung einer Turing-Maschine [klausurtypisch], 10+4 Punkte)

a) Zeigen Sie durch die Angabe einer Turing-Maschine, dass die Sprache $\{cca^nb^n \mid n \geq 0\}$ von einer Turing-Maschine akzeptiert wird. Beschreiben Sie vorher Ihren Ansatz mit mindestens 2 Sätzen.
b) Geben Sie für die Startworte ccab, ε, und cb die zugehörigen Berechnungen als Folge von Konfigurationen an, bis keine Folgekonfiguration mehr möglich ist.

Aufgabe 5 (Ausführung einer Turing-Maschine [klausurtypisch], 5+2 Punkte)

a) Gegeben sei die Turing-Maschine auf der rechten Seite. Geben Sie für die Startworte aaaa und aaa die zugehörigen Berechnungen als Folge von Konfigurationen an, bis keine Folgekonfiguration mehr möglich ist.
b) Was vermuten Sie, was die Turing-Maschine generell macht?

```
% Zustaende
Z: Start z1 z2 z3 z4 S
% Alphabet, Leerzeichen # automatisch dabei
A: a b
% Start
S: Start
% Ueberfuehrungsfunktion
% alt  lesen neu schreiben Richtung
 Start   #   z1       #        L
 z1      a   z2       b        L
 z1      b   z2       b        L
 z1      #   z3       #        R
 z2      a   z1       a        L
 z2      b   z1       a        L
 z2      #   z3       #        R
 z3      a   z4       a        R
 z3      b   z4       #        R
 z4      a   z4       a        R
 z4      b   z4       b        R
 z4      #   S        #        S
```

Aufgabe 6 (Erstellung einer Turing-Maschine [klausurtypisch], 10+4 Punkte)

a) Zeigen Sie durch die Angabe einer Turing-Maschine, dass die Sprache $\{a^nccb^n \mid n > 0\}$ von einer Turing-Maschine akzeptiert wird. Beschreiben Sie vorher Ihren Ansatz mit mindestens 2 Sätzen.
b) Geben Sie für die Startworte accb, ε, und ab die zugehörigen Berechnungen als Folge von Konfigurationen an, bis keine Folgekonfiguration mehr möglich ist.

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Theoretische Informatik

Sommersemester 2023
2. Aufgabenblatt

Aufgabe 7 (Weitere Experimente mit Turing-Maschinen)
Von der Veranstaltungsseite ist ein Eclipse-Projekt erhältlich, dass u. a. die Simulation von Turing-Maschinen erlaubt. Eine kurze Einführung befindet sich am Ende dieses Aufgabenblatts. Lesen Sie diese zuerst.

a) Übertragen Sie Ihre Turing-Maschine aus 4a) in das Format des Simulators in die vorhandene Datei beispiele/turingmaschinen/TMcca_nb_n.tm. Prüfen Sie mit den JUnit-Tests aus test.turingMaschine.TMcca_nb_nTest.java ob Ihre Maschine die enthaltenen Tests besteht. Korrigieren Sie gegebenenfalls Ihre Turing-Maschine.

b) Übertragen Sie Ihre Turing-Maschine aus 6a) in das Format des Simulators in die vorhandene Datei beispiele/turingmaschinen/TMa_nccb_n.tm. Prüfen Sie mit den JUnit-Tests aus test.turingMaschine. TMa_nccb_nTest.java ob Ihre Maschine die enthaltenen Tests besteht. Korrigieren Sie gegebenenfalls Ihre Turing-Maschine.

c) Zeigen Sie mit Hilfe einer Turing-Maschine, dass die Funktion $f(I^n) = I^n$ div 2 (div für die ganzzahlige Division) turing-berechenbar ist. Beschreiben Sie Ihren Ansatz mit mindestens 2 Sätzen. Schreiben Sie Ihre Maschine in die Datei beispiele/turingmaschinen/TMHalbieren.tm. Prüfen Sie Ihr Ergebnis mit test.turingMaschine.TMHalbierenTest.java.

d) In der Vorlesung wurde versucht zu zeigen, dass die Funktion $f(I^n) = I^{2n}$ turing-berechenbar ist. Dies ist bis auf unnötige Leerzeichen am Anfang auch gelungen. Korrigieren Sie die Turing-Maschine TM2malN.tm (oder schreiben Sie sie neu) in der Datei TM2malNVersion2.tm, so dass das geforderte Ergebnisformat mit nur einem Leerzeichen am Anfang eingehalten wird. Sie dürfen das Alphabet natürlich erweitern. Beschreiben Sie Ihren Ansatz mit mindestens 2 Sätzen. Prüfen Sie Ihr Ergebnis mit test.turingMaschine.TM2malNVersion2Test.java.

e) Die JUnit-Tests aus den vorherigen Teilaufgaben lassen vermuten, dass Turing-Maschinen problemlos mit Tests geprüft werden können. Dies ist nicht der Fall. Machen Sie das deutlich, indem Sie eine Turing-Maschine zu c) in der Datei Tm2malNFalsch.tm entwickeln, die die Tests aus c) alle erfüllen, aber trotzdem fehlerhaft ist. Zur Vereinfachung sind die Tests aus c) auch in mit test.turingMaschine.Tm2malNFakeTest.java enthalten. Ergänzen Sie dann einen Test, der zeigt, dass Ihre Maschine fehlerhaft ist.

f) Konvertieren Sie Ihre Turing-Maschine aus c) und bringen Sie diese auf https://turingmachinesimulator.com/ zum Laufen.

g) Konvertieren Sie Ihre Turing-Maschine aus c) und bringen Sie diese mit Tursi zum Laufen.

Turing-Maschinen-Bibliothek

Zum Experimentieren steht Ihnen eine prototype Bibliothek von der Veranstaltungsseite zur Verfügung, mit der Turing-Maschinen simuliert und so auch berechnete Ergebnisse überprüft werden können. Zur Beschreibung von Turing-Maschinen wird folgendes Format genutzt, hier erklärt an der Maschine aus der Veranstaltung, die a in b und b in a verwandelt.

```
% Zustaende
Z: Start wandel S
% Alphabet, Leerzeichen # automatisch dabei
A: a b
% Start
S: Start
% Ueberfuehrungsfunktion
% alt  lesen neu    schreiben Richtung
 Start   #   wandel      #        L
 wandel  a   wandel      b        L
```

```
wandel  b   wandel      a       L
wandel  #   S           #       S // Zustand S auch Endzustand
```

Generell müssen Zustände, das verwendete Alphabet, der Startzustand und die Überführungsfunktion angegeben werden. Beginnt eine Zeile mit „%“ handelt es sich um einen Kommentar, diese Zeilen könnten also weggelassen werden. Beginnt eine Zeile mit „Z:“ folgen dahinter durch Leerzeichen getrennt die Namen der Zustände, die so selbst keine Leerzeichen enthalten dürfen. Beginnt eine Zeile mit „A:“ folgen danach wieder durch Leerzeichen getrennt die Zeichen des verwendeten Alphabets. Diese Zeichen dürfen durchaus selbst aus mehreren Zeichen bestehen, eine Randbedingung ist dabei, dass ein Zeichen nicht der Präfix eines anderen Zeichen sein darf, so darf es nur eines der Zeichen „O“ und „OK“ geben. Das in der Veranstaltung benutzte Leerzeichen (Blank) # ist nicht beim Alphabet anzugeben und wird automatisch intern hinzugefügt. Beginnt eine Zeile mit einem „S:“ folgt danach genau der eine Name des Startzustands, der vorher in einer mit „Z:“ markierten Zeile definiert werden musste. Die Überführungsfunktion wird als 5-Tupel getrennt mit Leerzeichen zeilenweise angegeben. Die generelle Form ist: aktueller Zustand, gerade eingelesenes Zeichen, neuer Zustand, zu schreibendes Zeichen, Bewegung des Schreib-Lese-Kopfs. Die Elemente müssen mit mindestens einem Leerzeichen getrennt sein, es sind auch mehrere Leerzeichen möglich, so dass es einfache Formatierungsmöglichkeiten gibt. Die Richtungen links, rechts und stopp können ausgeschrieben oder mit ihrem ersten Buchstaben dargestellt werden. Am Ende von Zeilen mit einem Schritt der Überführungsfunktion kann hinter „//“ ein Kommentar folgen.

Ein Turing-Maschine kann entweder aus einer Datei oder als String gelesen werden.

```
TuringMaschine tm = new TuringMaschine();
tm.dateiEinlesen("Erste.tm");
```

oder

```
TuringMaschine tm = new TuringMaschine();
tm.stringAlsTuringMaschine("""
    % Zustaende
    Z: Start wandel S
    % Alphabet, Leerzeichen # automatisch dabei
    A: a b
    % Start
    S: Start
    % Ueberfuehrungsfunktion
    % alt  lesen neu    schreiben Richtung
    Start   #   wandel      #       L
    wandel  a   wandel      b       L
    wandel  b   wandel      a       L
    wandel  #   S           #       S // Ende
    """);
```

Die Klasse TuringMaschine stellt einige meist dokumentierte Methoden zur Verfügung. Die beiden wichtigsten sind:

```
Wort ergebnis = tm.berechnen(eingabe) // eingabe ist Wort oder String

boolean erg = tm. akzeptieren(eingabe); // eingabe ist Wort oder String
```

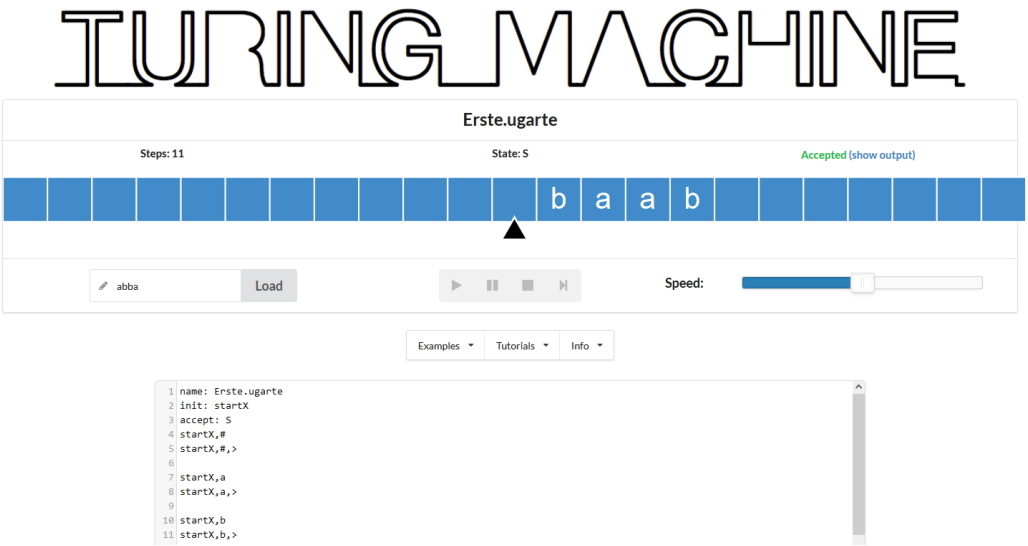
Soll eine Turing-Maschine schrittweise ausgeführt werden, ist das z. B. wie folgt möglich.

```
public static void main(String...strings) {
  TuringMaschine tm = new TuringMaschine();
  tm.stringAlsTuringMaschine("""
      % Zustaende
      Z: Start wandel S
      % Alphabet, Leerzeichen # automatisch dabei
      A: a b
      % Start
      S: Start
      % Ueberfuehrungsfunktion
      % alt      lesen  neu    schreiben Richtung
       Start       #   wandel      #        L
       wandel      a   wandel      b        L
       wandel      b   wandel      a        L
       wandel      #   S           #        S
      """);
  tm.bearbeite("abba");
  tm.ausfuehren(true);
}
```

Es erfolgt eine schrittweise Abarbeitung mit Anzeige der aktuellen Konfiguration,

```
Start: (Start, #abba#)
(wandel, #abba#)
Weiter mit <Return>
(wandel, #abbb#)
Weiter mit <Return>
(wandel, #abab#)
Weiter mit <Return>
```

Es gibt sehr viele Umsetzungen von Turing-Maschinen mit Simulationsprogrammen, da dies ein zentrales Informatik-Grundlagenthema ist. Generell sind viele Varianten leicht zu ergoogeln, wobei es keine einheitliche Form der syntaktischen Darstellung gibt. Allgemein sind aber praktisch alle Modelle logisch äquivalent, so dass die Übersetzung einer Turing-Maschine leicht in eine andere erfolgen kann. Exemplarisch sollen hier zwei Varianten vorgestellt werden.



Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Theoretische Informatik

Sommersemester 2023
2. Aufgabenblatt

Eine graphisch gelungene Simulation mit einer guten Visualisierung des Ablaufs befindet sich auf der Seite https://turingmachinesimulator.com/. Bei einer typischen Nutzung wird zuerst die Turing-Maschine in einem Text-Editor entwickelt und dann in das Eingabefeld kopiert. Die Syntax-Prüfung erfolgt mit dem unteren Knopf „Compile“. Links vom Knopf „Load“ wird das Eingabewort geschrieben, mit dem Knopf „Load“ auf das Band im oberen Bild kopiert und die Ausführung über den „Run-Button >“ oder über den „Schritt-Button >|“ gestartet. Der Ablauf der Schritte kann oben auf dem Band verfolgt werden, etwas tiefer rechts ist die Geschwindigkeit spezifizierbar. Da die Turing-Maschine auf dem ersten (linken) Zeichen der Eingabe beginnt, müssen unsere Turing-Maschinen so ergänzt werden, dass sie am Anfang erst an das Ende des Wortes laufen und dann mit dem eigentlichen Startzustand beginnen. Die oberhalb des Programm-Fensters ausgegebenen Fehlermeldungen sind leider nicht sehr detailliert, Details zur Darstellung können den auf der Web-Seite erhältlichen Beispielen und dem Tutorial entnommen werden. Es ist zu beachten, dass die Aufgabenstellungen in den Beispielen teilweise unpräzise formuliert sind.

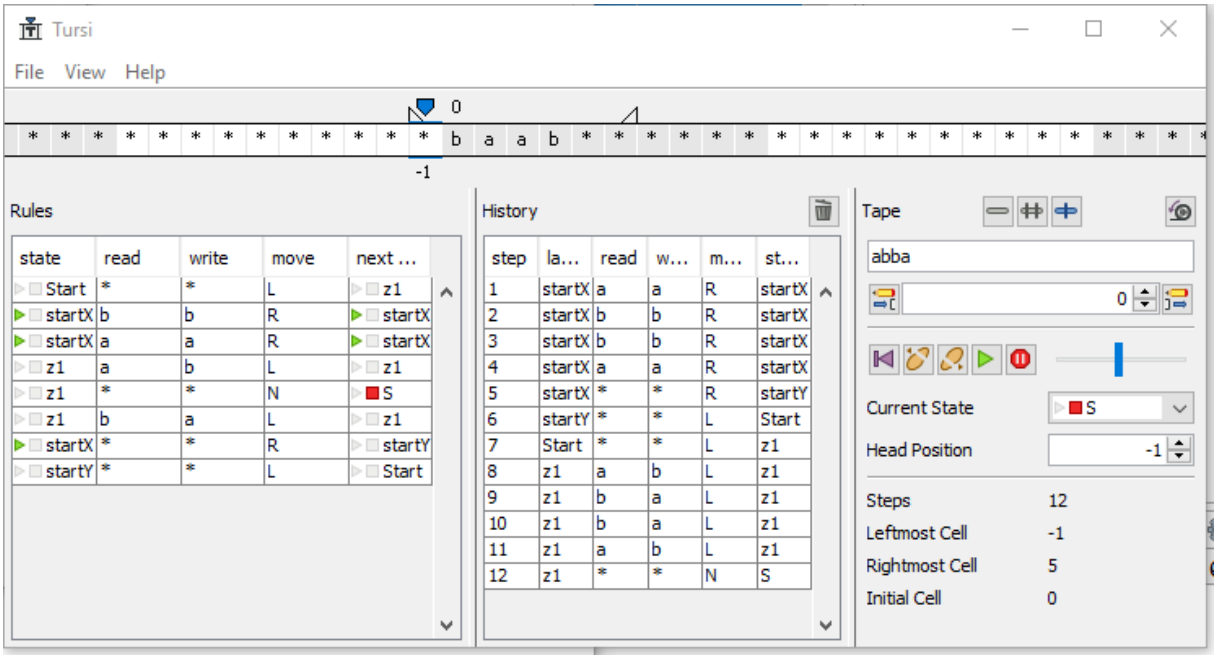
Die gegebene Bibliothek unterstützt die Umwandlung in das hier genutzte Modell, dazu stehen zwei Methoden zur Verfügung. Mit der folgenden Methode werden automatisch Schritte ergänzt, so dass am Anfang hinter die Eingabe gelaufen wird und so die Ausgangssituation der Vorlesung entsteht, dazu wird ein spezieller Zustand „startX“ genutzt.

Weiterhin muss es einen Endzustand mit Namen S geben.

```
    tm.dateiSpeichernUgarte("Erste.ugarte");
```

Soll nicht an das Ende der Eingabe gelaufen werden, ist als zweiter Parameter „false zu übergeben.

```
    tm.dateiSpeichernUgarte("Erste.ugarte", false);
```



Ein zweiter interessanter Turing-Maschinen-Simulator ist Tursi, das als Standalone-Programm von der Webseite https://schaetzc.github.io/tursi/ , genauer https://github.com/schaetzc/tursi/releases

geladen werden kann. Die jar-Datei ist z.B. über die Kommandozeile startbar, die z. B. über StartKonsole.bat aus der kleukerSEU ausgeführt werden kann.

```
F:\Lehre\Material_Theorie\TMSimulatoren> java -jar Tursi.jar
```

Turing-Maschinen werden über „File > Open…“ geladen und dann auf der linken Seite angezeigt. Rechts unter „Tape“ wird die Eingabe für das Band eingegeben und durch Drücken der Return-Taste übergeben. Darunter befinden sich dann die Steuerungsmöglichkeiten, u. a. auch um einen Schritt vorwärts und rückwärts zu machen. Interessant ist u. a. der Mittelteil, der die durchgeführten Schritte protokolliert. Der Schreib-Lese-Kopf kann in der oberen graphischen Darstellung auch genutzt werden, um ihn mit gedrückter linker Maustaste auf dem Bildschirm neu zu platzieren, falls interessante Bandinhalte nicht sichtbar sind.

Die gegebene Bibliothek der Veranstaltung unterstützt die Umwandlung in das hier genutzte Modell, dazu stehen zwei Methoden zur Verfügung. Mit der folgenden Methode werden automatisch Schritte ergänzt, so dass am Anfang hinter die Eingabe gelaufen wird und so die Ausgangssituation der Vorlesung entsteht, dazu werden zwei spezielle Zustände „startX“ und „startY“ genutzt.

```
    tm.dateiSpeichernTursi("Erste.tursi");
```

Soll nicht an das Ende der Eingabe gelaufen werden, ist als zweiter Parameter „false“ zu übergeben.

```
    tm.dateiSpeichernTursi("Erste.tursi", false);
```