

Fragen, Antworten und Kommentare zur aktuellen Vorlesung

Das Video zur Lösung der Aufgabe 8a finden Sie unter: <https://youtu.be/1qx6II810Ng>

Das Video zur Lösung der Aufgabe 8b finden Sie unter: <https://youtu.be/umY2Hsif48I>.

für $n=0$ ist der Start nicht ok, korrigiert (erste Zeile streichen):

```
Start # suche0 # L
suche0 c sucheb # L
suche0 # S # S
weiterhin fehlen:
sucheb B sucheb B L
neuesc B neuesc B R
neuesB A neuesB A R
```

Frage: (zum Lösungsvideo Blatt 2, Aufgabe 4) Dadurch, dass es die Überföhrungsfunktion "einb A zwei # L" gibt, wird doch auch die Sprache ccA akzeptiert? Das ist doch nicht korrekt, oder?

Antwort: Das ist zunäcst eine sehr gute Feststellung. In der Vorlesung nutzen wir eine auch von Kollegen genutzte, an dieser Stelle zu vereinfachte, vereinfachte Definition einer Turing-Maschine, was auch bei der Bearbeitung von Blatt 3, Aufgabe 9 deutlich werden kann.

Genauer wird in der formal korrekten Definition das Bandalphabet aufgeteilt und es gibt ein Eingabealphabet und das uns bekannte Bandalphabet. Das Eingabealphabet ist eine Teilmenge des Bandalphabets. Für die Turing-Maschine sind dann nur Eingaben aus dem Bandalphabet* erlaubt. Mit einem Eingabealphabet {a,b,c} muss Ihr Wort ccA also nicht betrachtet werden.

Für unsere Veranstaltung wird angenommen, dass das Eingabealphabet durch die Aufgabenstellung definiert wird, also keine weiteren Zeichen zu verarbeiten sind, die nicht in den angegebenen Sprachen vorkommen. Mit dieser auch für die Klausur geltenden Vereinbarung ist die Lösung korrekt.

Frage: Beim Quiz3 zur Lernnotiz 2 ist $f1(I^n) = f1(I^{n+1})$ nicht berechenbar, warum?

Antwort: Die Antwort im Quiz ist falsch, da so nur gefordert wird, dass $f1$ eine konstante Funktion ist, z. B., das Ergebnis immer das leere Wort ist. Die Aufgabe wurde heimlich korrigiert durch

$f1(I^n) = f1(I^{n+1})I$ (das letzte I bedeutet, es wird angehängt)

Dies ist keine ordentliche Funktionsdefinition und deshalb nicht berechenbar.

$f1(\varepsilon) = f1(I)I$, d.h. das Ergebnis von $f1(\varepsilon)$ hat mindestens ein I am Ende,

da $f1(I) = f1(II)I$, d. h. $f1(I)$ hat mindestens ein I am Ende, d. h. aber $f1(\varepsilon)$ hat mindestens II am Ende.

Wird dies schrittweise fortgeführt, muss $f1(\varepsilon)$ dann unendlich viele I am Ende haben, was als Ergebnis einer Turing-Maschine nicht zugelassen ist.

Frage: Wie funktioniert die universelle Turing-Maschine?

Antwort: Wichtig ist die Idee dahinter. Sie wollen eine Turing-Maschine schreiben die Turing-Maschinen ausführt, ähnlich zu einem Simulator für Turing-Maschinen, nur dass dieser selbst als Turing-Maschine geschrieben wird. Der Ansatz ist genau wie für einen Simulator, als Eingabe wird die Turing-Maschine, die ausgeführt werden soll und das Eingabewort auf das Band geschrieben. Dann startet die universelle Turing-Maschine, schreibt zunächst die Ausgangskonfiguration der auszuführenden Turing-Maschine auf das Band, liest das Zeichen unter dem Schreiblesekopf und den Zustand und sucht dann auf dem Band in der auszuführenden Turing-Maschine nach dem Folgezustand und dem neuen Bandinhalt. Jede dieser Aktionen benötigt einige Zustände und Hilfszeichen, um sich Details zu merken. Wichtig ist aber, dass nur endlich viele Zustände und endlich viele Zusatzzeichen benötigt werden, da für jede Turingmaschine die Kombination aus Zeichen und Zuständen immer endlich ist. So kann letztendlich garantiert werden, dass es eine universelle Turing-Maschine gibt.

Frage: Wenn ich es richtig verstehe, sind die Begriffe „rekursiv aufzählbar“ und „aufzählbar“ Synonyme?

Antwort: Ja, es werden manchmal in den Folien mehrere Begriffe angegeben, die das Gleiche bedeuten, da sie diese auch in anderen Lehrbüchern in diesen Varianten finden. Der Begriff „semi-entscheidbar“ ist hier ein weiteres Synonym.

Frage: Was ist die Halteproblemsprache genau?

Die Halteproblemsprache besteht aus allen Wörtern, die die folgende Form haben: Turingmaschine#Eingabewort und die Turingmaschine hält angesetzt auf das Eingabewort an. Wir wissen dazu, dass eine Turingmaschine auch nur aus endlich vielen Zeichen besteht und so auf das Band geschrieben werden kann.

Bei der speziellen Halteproblemsprache wird davon nur die Teilmenge betrachtet, bei der das Eingabewort wieder die Turing-Maschine selbst ist, also die Eingabe der Form $TMX\#TMX$ hat.

Eine weitere oft betrachtete Variante beschäftigt sich mit der Sprache, die als Worte alle Turing-Maschinen beinhaltet, die angesetzt auf das leere Wort anhalten.

Für alle diese Sprachen gilt, dass sie nicht entscheidbar, aber aufzählbar sind.

Frage: Wie hängen jetzt charakteristische Funktion und Entscheidbarkeit zusammen?

Antwort: Erstmal sind charakteristische Funktionen unabhängig von Turing-Maschinen und Entscheidbarkeit definiert. Für eine beliebige Menge M gilt $f(x) = 1$, wenn x in M ist, sonst $f(x) = 0$.

Die charakteristische Funktion kann aber auch mit Hilfe einer Turing-Maschine berechnet/programmiert werden. Da gilt dann, dass genau dann, wenn M entscheidbar ist, ist die charakteristische Funktion Turing-berechenbar.

Frage: Was ist der Busy Beaver genau?

Antwort: Die Aufgabe ist: Sie können n Zustände nutzen und müssen eine Turing-Maschine schreiben, die möglichst viele Striche auf dem Band als Ergebnis liefert. Das bedeutet natürlich auch, dass die Turing-Maschine anhalten muss. Die Frage ist dann nach der zu n gehörenden maximalen Strichanzahl, wobei es zwischen den Strichen Leerzeichen geben darf. Diese Funktion ist nicht berechenbar.

Frage: Ich habe mit der Diagonalisierung bei dem Halte-Problem ein Problem.

Antwort: Schauen Sie sich nochmals die Idee der Überabzählbarkeit der reellen Zahlen an, die Idee ist die gleiche. Grob ist der Ansatz immer, wenn es ein Verfahren geben soll, dann gibt es auch ein zweites Verfahren, das recht einfach auf dem ersten basiert und damit auch existiert. Dann wird gezeigt, dass das zweite Verfahren für einen bestimmten Wert zwei verschiedene Ergebnisse liefern müsste, was nicht geht. Damit kann das zweite und damit auch das erste Verfahren nicht existieren.

Frage: Welches Wort ist $a^n b^{2^n} c^{3^n}$ für $n=0$?

Antwort: „hoch 0“ ergibt für jedes Zeichen und Wort immer ϵ , so dass das auch das Gesamtergebnis ist.

Frage: Welches Wort ist $a^n b^{2^n} c^{3^n}$ für $n=-1$?

Antwort: Das ist nicht definiert, da unsere Definition mit $n=0$ startet. [Man kann auch darüber nachdenken und interessante Konstruktionen basteln, aber das ist eher reine Mathematik jenseits der Theoretischen Informatik. Wer sich dafür genauer interessiert, unsere Sprachen mit der Konkatenationsverknüpfung bilden ein freies Monoid. Rest ist Algebra.

Frage: Wie gehören die Begriffe Entscheidbarkeit, Aufzählbarkeit und Akzeptierbarkeit zusammen?

Akzeptierbarkeit einer Sprache L ist zunächst die Forderung, dass es eine Turing-Maschine gibt, die irgendwann anhält, wenn sie auf ein Wort der Sprache L angesetzt wird.

Dies ist eng verwandt mit dem Begriff der Aufzählbarkeit, die nur die zusätzliche Form der Ergebnisausgabe der Turing-Maschine fordert. Wenn es für L eine akzeptierende Turing-Maschine gibt, ist es trivial daraus eine aufzählende Turing-Maschine konstruieren (erste Diagonalisierungsidee, alle Worte schrittweise ausprobieren). Genauso kann aus der aufzählenden Turing-Maschine eine akzeptierende gebaut werden, es werden mit der gleichen Diagonalisierungsidee alle Turing-Maschinen mit I^n als Eingabe ausgeführt und angehalten, wenn das Wort gefunden wurde. Etwas unsauber kann damit bei Aufzählbarkeit und Akzeptierbarkeit von Synonymen gesprochen werden.

Die Entscheidbarkeit ist anders, da für eine Sprache L die Turing-Maschine irgendwann mit einem bestimmten Ergebnis anhalten muss, wenn das Wort in L liegt und mit einem anderen Ergebnis anhalten muss, wenn das Wort nicht in L liegt. Das „und“ macht es anders, da so die Akzeptierbarkeit bzw. Aufzählbarkeit von L (wie vorher auch) und zusätzlich die Akzeptierbarkeit bzw. Aufzählbarkeit des Komplements von L gefordert wird.

Damit ist L entscheidbar, wenn L akzeptierbar und auch das Komplement von L akzeptierbar ist.

Frage: Ich finde das Thema mit der Entscheidbarkeit ziemlich schwer, wie sehen Sie das bei den anderen Themen.

Antwort: Vom Inhalt her haben Sie Recht, das ist das schwerste Kapitel der Veranstaltung. Die nachfolgenden sind meist deutlich leichter. Ich habe das Thema an den Anfang der Veranstaltung gestellt, da die Aussage „es gibt viele Eigenschaften von Programmen, die wir nicht automatisch nachweisen können“ die zentrale Aussage der Veranstaltung ist, die gerade auch in der Praxis im Hinterkopf bleiben muss. (Weiterhin sind Sie ja am Anfang des Semesters noch am Frischesten :))

Frage: Gibt es auch zu den anderen Aufgaben Beispiellösungen?

Antwort: Ist aktuell nicht geplant. Sie können aber Ihre Lösungen mit mir besprechen (VL-Zeit und Übungszeit). Das ist auch möglich, wenn Sie keinen Ansatz gefunden haben.

Frage: Gibt es für die Reduktionen ein Kochrezept?

Antwort: Ganz allgemein leider nicht. Trotzdem besteht immer die Idee mit einer Turing-Maschine ein Problem der einen Art in ein Problem der anderen Art zu verwandeln. Statt der formalen Forderung:

Definition: Gegeben seien Sprache1 und Sprache2 nicht notwendigerweise über den gleichen Alphabeten Alphabet1 und Alphabet2 sowie eine **total berechenbare Funktion** $f: \text{Alphabet1}^* \rightarrow \text{Alphabet2}^*$; gelte dann für alle $w \in \text{Alphabet1}^*$
 $w \in \text{Sprache1}$ genau dann wenn $f(w) \in \text{Sprache2}$,

Kann man auch äquivalent etwas anschaulicher versuchen zu zeigen:

$w \in \text{Sprache1}$ daraus folgt $f(w) \in \text{Sprache2}$ und

$w \notin \text{Sprache1}$ daraus folgt $f(w) \notin \text{Sprache2}$

Frage: Warum wird die Software als Prototyp bezeichnet?

Antwort: Ist genauer ein funktionaler Prototyp, der die (von mir gewünschte) Basisfunktionalität vollständig umsetzt, also komplett nutzbar ist, aber an vielen Stellen Optimierungen benötigt. Dies umfasst zumindest deutlich zu lange Methoden, eine zu geringe Testüberdeckung und die Performance diverser Methoden. Gerade bei der Performance wurde auf die Machbarkeit und eine erste einfache Lösung fokussiert; dies wäre aber im nächsten Schritt zu überarbeiten. Dies ist also kein Wegwerf-Prototyp der für Beispiel-Use-Cases die Machbarkeit zeigt, aber könnte zum wegwerfen animieren, um z. B. den domänengetriebenen Ansatz komplett umzusetzen. Da eine solche Überarbeitung Zeit kostet, ist unklar, wann sie in der Prioritätenliste weiter nach oben wandert.

