Fragen, Antworten und Kommentare zur aktuellen Vorlesung

Das Video zur Lösung der Aufgaben 3 und 4 finden Sie unter: https://youtu.be/hcrGzXdIznk . Bei Aufgabe 4 fehlt ein Übergang `einb c zweic1 # L`

Das Video zur Lösung der Aufgaben 5 und 6 finden Sie unter: https://youtu.be/AbT9PHY7UIE.

Das Thema „Erstellung einer Turing-Maschine“ wird auch in der Übung in der nächsten Woche behandelt.

Frage: In Folie 36 wird gezeigt das der Schreib-Lese-Kopf auf der Position r steht, dies kann aber nicht der Startzustand sein oder? Der Startzustand muss zwingend nach dem Wort stehen und nicht mitten drinnen oder habe ich etwas falsch verstanden? Ist der Startzustand dann immer unmittelbar hinter dem Wort oder kann auch ein # noch dazwischen liegen?

Antwort: Genau auf Folie 36 wird das Konzept eingeführt und das Bild zeigt eine laufende Maschine in Aktion. Am Start würde der der Kopf wie von Ihnen beschrieben rechts neben dem „t“ stehen. Nach Folie 41 darf im Eingabewort kein Leerzeichen stehen, außer die Maschine soll mit n Wörtern (n > 1) starten, dann ist ein Leerzeichen zwischen den einzelnen Wörtern. Das wird allerdings als Verallgemeinerung erst in Folie 59 erklärt.

Die Eingabe ohne Leerzeichen ist dabei keine Einschränkung, man könnte statt der Leerzeichen ein neues Zeichen bei der Eingabe wählen und die Turing-Maschine würde in den ersten Schritten dieses neue Zeichen durch Leerzeichen ersetzen und wieder hinter die Eingabe laufen.

Frage: Soll ich zur Fehlerbehandlung einen speziellen error-Zustand nutzen?

Antwort: Nein, wenn in einem Zustand ein Zeichen gelesen wird, dass nicht erwartet wird, was also einen Fehler anzeigt, wird die Turing-Maschine an dieser Stelle schlicht nicht definiert. Formaler: Es gibt dann keine Folgekonfiguration, da aber die letzte Aktion für den Schreib-Lesekopf kein S war, wird die Eingabe so nicht akzeptiert.

Frage: Muss eine Turing-Maschine vollständig definiert werden?

Antwort: Nein, die Überführungsfunktion darf Lücken haben, also partiell sein. Ist ein Übergang nicht definiert, ist das Verhalten der Turing-Maschine nicht definiert, was bedeutet, dass sie nicht terminiert und es so keine erfolgreiche Berechnung ist.

Frage: Können wir Turing-Maschinen auch in der graphischen Form in der Klausur angeben?

Antwort: Ist auch ok. (Persönlich finde ich die graphische Notation nicht besser lesbar als die textuelle, zumindest wenn Korrekturen angebracht werden müssen.)

Frage: Soll in der Klausur die Turing-Maschine auch immer rechts neben dem Wort starten?

Antwort: Ja (auch wenn in der Literatur meist links vom Wort gestartet wird). Als Anmerkung ergänzt, wenn Sie Abläufe für Turing-Maschinen angeben, ist es wichtig, dass Sie immer vollständige Konfigurationen aufschreiben.

Frage: wenn da a^n steht und n=0, dann ist das doch das leere Wort?

Antwort: Stimmt.

Frage: wenn da cca^nb^n steht , bezieht sich das erste n auf cca (also akzeptierte Worte ε, ccab, ccaccabb, ccaccaccabbb, …) oder auf a (akzeptierte Worte cc, ccab, ccaabb, ccaaabbb, …)

Antwort: Die Potenz bezieht sich nur immer auf das vorgestellte Zeichen, bindet mathematisch als am höchsten. Die zweite angegebene Menge von Worten ist korrekt. Für die erste Sprache würde angegeben: $(cca)^nb^n$

Frage: Ist es wichtig wo der Schreib-Lesekopf am Ende steht?

Antwort: Nein und ja. Nein, wenn Sie überprüfen sollen, ob ein Wort zu einer Sprache gehört. Da ist es nur wichtig, dass die Turing-Maschine genau dann terminiert (S), wenn das Wort zur Sprache gehört. Ja, die Position ist wichtig, wenn es darum geht eine Turing-Maschine für die Berechnung einer Funktion anzugeben. Hier ist gefordert, dass der Schreib-Lesekopf rechts neben dem Ergebnis (also nicht auf dem letzten Zeichen) steht.

- f(w1,…,wm) = (u1,…,un) genau dann wenn es eine terminierende Berechnung Start, #w1#w2#...#wm# ->* z, #u1#u2#...#un# gibt

Frage: Ich habe Schwierigkeiten bei der Eingabe der Turing-Maschine.

Antwort: Wenn Sie die Tests laufen lassen, dann bei der Ausgabe zuerst ganz nach oben scrollen, da könnte es Warnungen bezüglich fehlender Zeichen oder Zustände geben. Bei längeren Zustandsnamen immer genau nach Tippfehlern suchen.

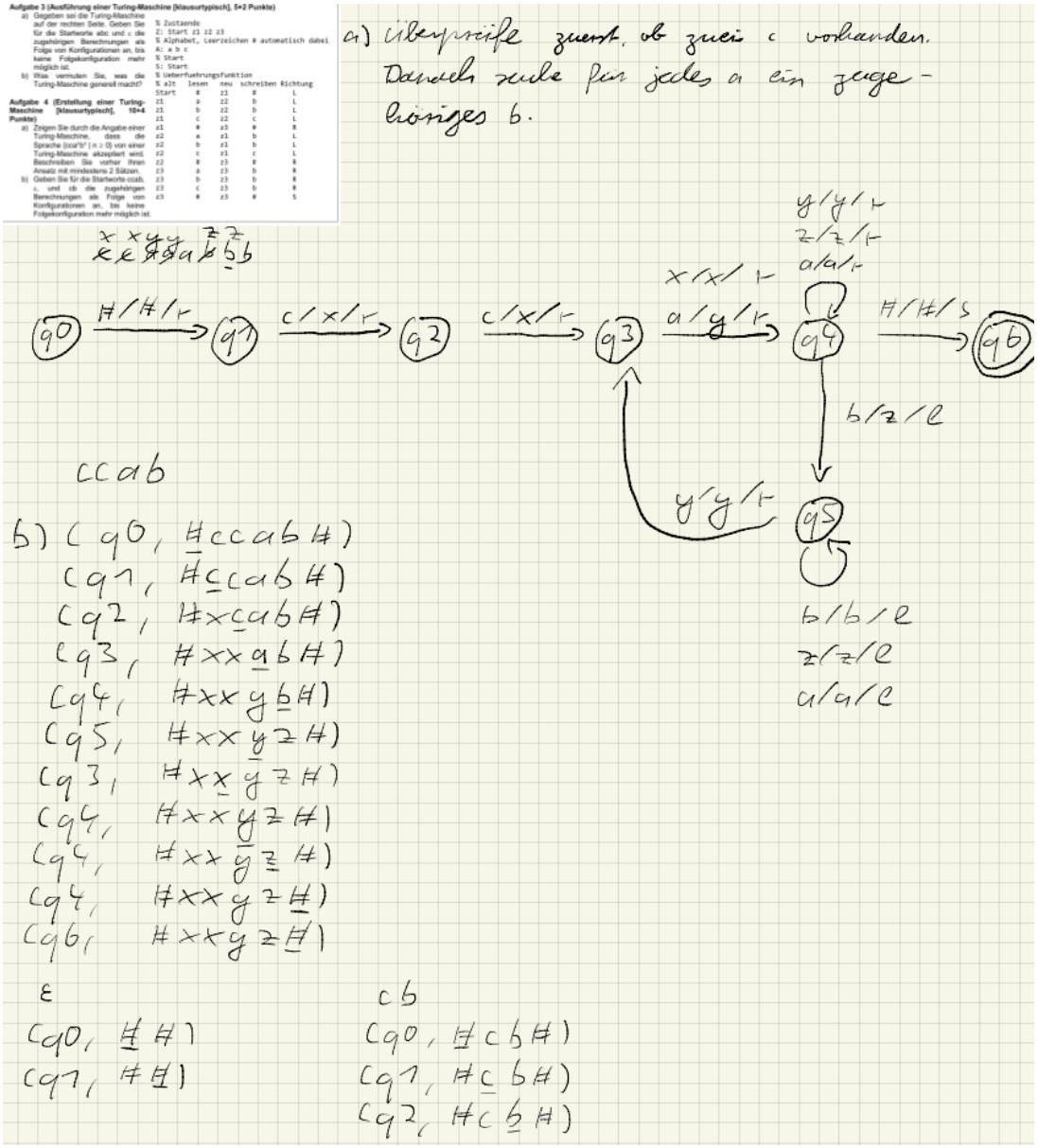
Frage: (zum Lösungsvideo Blatt 2, Aufgabe 4) Dadurch, dass es die Überführungsfunktion "einb A zweic # L" gibt, wird doch auch die Sprache ccA akzeptiert? Das ist doch nicht korrekt, oder?

Antwort: Das ist zunächst 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: Können Sie meine Lösung zu Aufgabe 4 überprüfen?



Antwort: Generell gerne, sollte aber wenn möglich in der Vorlesungs- oder der anschließenden Praktikumszeit geschehen, da man dann gemeinsam über die Ideen reden kann. In der VL-freien Zeit geht einfach eine E-Mail oder der Wunsch nach einem Zoom-Termin mit der Nennung möglicher Zeitintervalle. Meine Empfehlung ist immer, wenn möglich zuerst mit einem Simulator, entweder in meiner Umgebung oder einem im Netz auszuführen und vorhandene JUnit-Tests zu nutzen. Da der Computer auch dann noch einfach „nein“ sagen kann, schicken Sie dann Ihren Lösungsansatz, eventuell mit Ergebnissen der genannten Tests an mich.

zu a) Die grobe Beschreibung wäre in Ordnung, wobei der zweite Satz recht unpräzise ist (was auch zu Problemen führt)

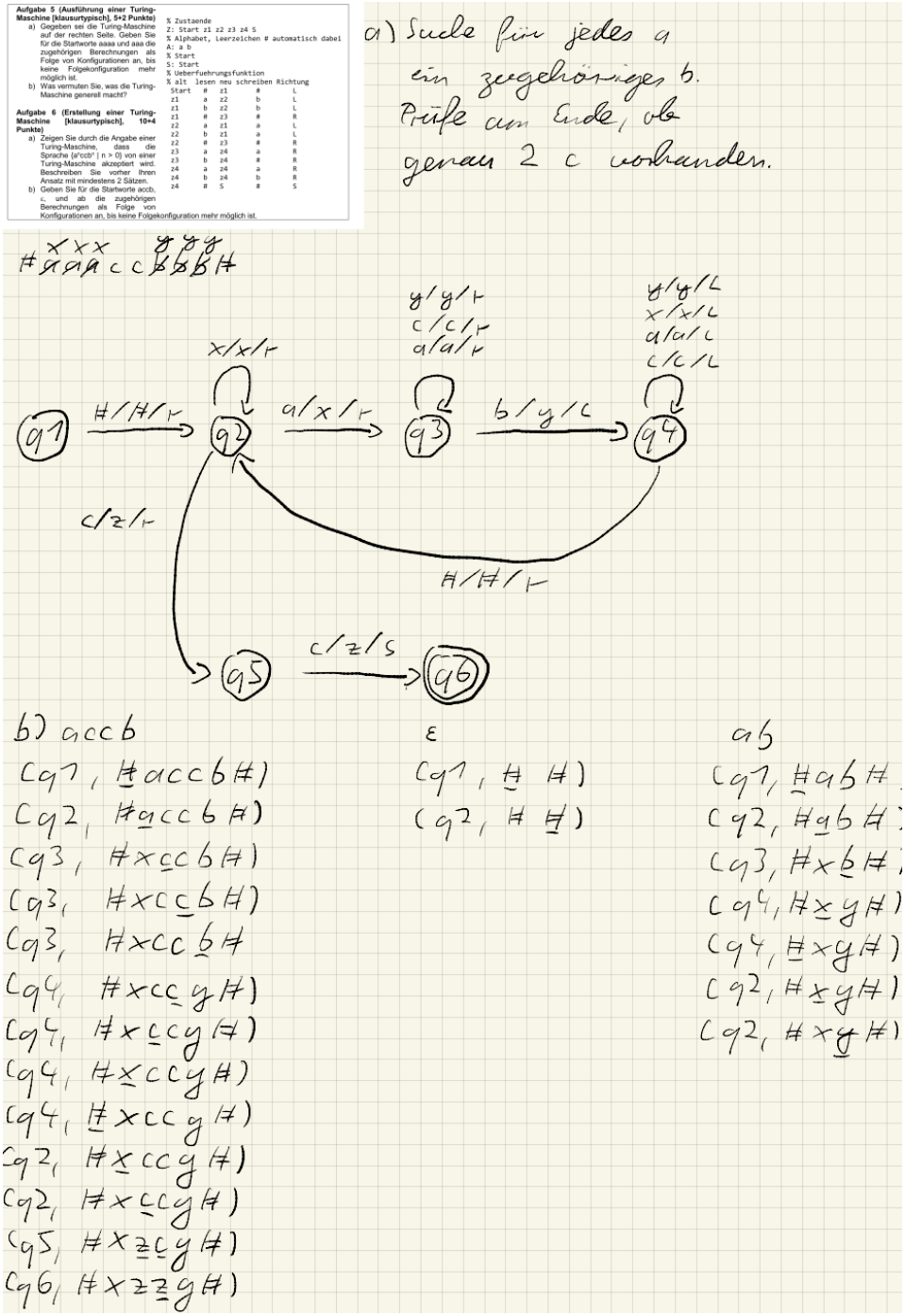
Zunächst fällt auf, dass sie davon ausgehen, dass die Turing-Maschine links vor dem Eingabewort startet. Das findet man häufig in der Literatur, ist aber in der Vorlesung (auch K. Morisse) nicht der Fall. Lässt sich leicht retten, indem man auf die linke der Seite der Eingabe läuft, was allerdings hier fehlt und Punkte kostet. Die Maschine ist fast ok, das Problem ist, dass sie nach einem a und der Suche nach einem b, wenn Sie sofort ein Leerzeichen finden (cca), terminieren. Das passiert auch bei cca^n, n>0 allgemein. Sie müssten in q5 prüfen, ob danach ein Leerzeichen kommt, um dann zu terminieren. Weiterhin muss das Wort cc akzeptiert werden, das ist bei Ihnen nicht der Fall. Der Rest ist ok.

Ihr Problem tritt auch in (q5,#xxyz#) auf, der von Ihnen genutzte Übergang existiert nicht im Diagramm. Die Idee, dass ein x auf das letzte gefundene a hindeutet, könnte zielführend werden.

Bei cb muss das c in der letzten Zeile ein x sein.

Generell würde Ihre Lösung in etwa die Hälfte der Punkte liefern.

Frage: Können Sie meine Lösung zu Aufgabe 6 überprüfen?



Die grobe Beschreibung wäre in Ordnung, wobei der zweite Satz recht unpräzise ist. (Startproblem s. vorherige Frage)

Die Maschine ist fast richtig, das Problem ist, dass zu viele b auch akzeptiert werden, da der Fehler nicht entdeckt wird. Sie akzeptieren $a^n ccb^m$ mit $m \geq n$. Das Problem lässt sich beheben indem Sie prüfen, dass in q6 rechts davon kein b steht. Die Konfigurationsfolgen sind ok.

Generell würde Ihre Lösung in etwa 70% der Punkte liefern.

Generell ist es bei Strukturen der Form $a^n b^n$ einfacher die äußeren a und b zu markieren und dann schrittweise nach innen zu gehen. So ist meist feststellbar, ob die Anzahl gleich ist. Das wurde bei beiden Aufgaben nicht gemacht und hat zu Problemen geführt.

Frage: Können die Lösungsüberprüfungen auch ohne Eclipse genutzt werden?

Antwort: Generell ja, da JUnit unabhängig von einer Entwicklungsumgebung auch von der Konsole aus aufrufbar ist. Allerdings ist der Weg recht aufwendig. Dabei gibt es eine einfachere funktionierende Lösung mit Schwächen in der Ausgabe und eine komplexere und schönere Lösung. Beide werden im Folgenden vorgestellt.

Bei der einfachen Lösung wird die Command-Umgebung (cmd) von Windows genutzt. Die Konsole wird entweder über Windows direkt oder über die KleukerSEU mit StartKonsole.bat aufgerufen. Wird die KleukerSEU nicht genutzt, muss Java installiert sein. Als Beispiel wird folgende Aufgabe betrachtet.

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.

Zur Vorbereitung wird die Datei theoriesammlung.zip von der Veranstaltungsseite geladen und in einem beliebigen Verzeichnis, hier C:\Internet ausgepackt. Weiterhin wird eine Hilfsbibliothek zur einfachen JUnit-Ausführung benötigt, die unter https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/ und der höchsten Nummer heruntergeladen werden kann. Im Beispiel wird https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/6.0.3/junit-platform-console-standalone-6.0.3.jar genutzt und in das Unterverzeichnis der Theoriesammlung kopiert. Das Verzeichnis sieht dann wie folgt aus, in diesem Verzeichnis erfolgen auch die folgenden Schritte:

```
C:\tmp\theoriesammlung>dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: 12B4-1D2C

 Verzeichnis von C:\tmp\theoriesammlung

09.03.2026  12:50    <DIR>          .
09.03.2026  12:50    <DIR>          ..
13.01.2026  11:15               502 .classpath
03.12.2022  10:43               391 .project
26.04.2025  19:12    <DIR>          .settings
09.03.2026  12:49    <DIR>          beispiele
09.03.2026  12:49    <DIR>          bin
09.03.2026  11:14         2.874.645 junit-platform-console-standalone-6.0.3.jar
09.03.2026  12:49    <DIR>          src
24.06.2024  11:32               621 TM2malN.ugarte
24.06.2024  11:32                88 TMa_nb_n_c_n.ugarte
24.06.2024  16:26               451 TMa_nccb_n.ugarte
24.06.2024  11:32               109 TMHalbieren.ugarte
24.06.2024  11:32               126 TMPlus1.ugarte
               8 Datei(en),      2.876.933 Bytes
               6 Verzeichnis(se), 18.174.550.016 Bytes frei
```

Falls nicht die KleukerSEU genutzt wird und Java nicht in der PATH-Variablen steht, kann der PATH auch nur für diese Nutzung der Umgebung ergänzt werden. Dabei muss der Pfad natürlich zur vorhandenen Java-Installation passen.

```
set PATH=C:\kleukersSEU\java\bin;%PATH%
```

Jetzt muss zunächst die zur Verfügung gestellte Testdatei aus der Aufgabenstellung übersetzt werden. Dies erfolgt mit dem folgenden Befehl, der in einer Zeile steht.

```
javac -d out -cp junit-platform-console-standalone-6.0.3.jar;src
src\test\turingMaschine\TMcca_nb_nTest.java
```

Mit -d out wird ein Verzeichnis für die übersetzten Dateien angegeben. Werden verschiedene Tests ausgeführt, ist es sinnvoll, dieses Verzeichnis zwischenzeitlich wieder zu löschen. Mit -cp werden Bibliotheken, also Jar-Dateien und Ordner dem Class-Path von Java hinzugefügt, in dem Java die genutzten Klassen sucht. Die einzelnen Einträge in den Class-Path werden in Windows mit einem Semikolon getrennt.

Die eigentlichen Tests werden mit dem folgenden Befehl ausgeführt. Der letzte Parameter verhindert die farbige Ausgabe der Ergebnisse, da im Standard keine ANSI-Steuerzeichen unterstützt werden. Die Turing-Maschine muss in der in der Aufgabe geforderten Datei stehen. Im Beispiel befindet sich dort eine Turing-Maschine, die keine Schritte machen kann.

```
java -jar junit-platform-console-standalone-6.0.3.jar execute --class-path out
--scan-class-path --disable-ansi-colors
```

Der folgende Ausschnitt des Ergebnisses zeigt die Testergebnisse, weiterhin sind Ausgaben der theoriesammlung erahnbar, allerdings nicht klar lesbar. Zu jedem gescheiterten Test werden weitere Informationen ausgegeben.

```
C:\tmp\theoriesammlung>java -jar junit-platform-console-standalone-6.0.3.jar
execute --class-path out --scan-class-path --disable-ansi-colors

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

Start: (Start, #←[4m#←[0m)
Start: (Start, #cb←[4m#←[0m)
Start: (Start, #cab←[4m#←[0m)
Start: (Start, #cccab←[4m#←[0m)
Start: (Start, #ccaacbb←[4m#←[0m)
Start: (Start, #cccaabb←[4m#←[0m)
Start: (Start, #aabbc←[4m#←[0m)
Start: (Start, #ccab←[4m#←[0m)
Start: (Start, #cc←[4m#←[0m)
Start: (Start, #ccaaabbb←[4m#←[0m)
Start: (Start, #ccaaaaaaaaaabbbbbbbbbb←[4m#←[0m)
.
+-- JUnit Platform Suite [OK]
+-- JUnit Jupiter [OK]
| '-- TMcca_nb_nTest [OK]
|   +-- testNicht(String) [OK]
|   | +-- [1] "" [OK]
|   | +-- [2] "cb" [OK]
|   | +-- [3] "cab" [OK]
|   | +-- [4] "cccab" [OK]
|   | +-- [5] "ccaacbb" [OK]
|   | +-- [6] "cccaabb" [OK]
|   | '-- [7] "aabbc" [OK]
|   '-- testOK(String) [OK]
|     +-- [1] "ccab" [X] Das Wort ccab sollte akzeptiert werden, wurde es aber
nicht ==> expected: <true> but was: <false>
|     +-- [2] "cc" [X] Das Wort cc sollte akzeptiert werden, wurde es aber nicht
==> expected: <true> but was: <false>
|     +-- [3] "ccaaabbb" [X] Das Wort ccaaabbb sollte akzeptiert werden, wurde es
aber nicht ==> expected: <true> but was: <false>
|     '-- [4] "ccaaaaaaaaaabbbbbbbbbb" [X] Das Wort ccaaaaaaaaaabbbbbbbbbb sollte
akzeptiert werden, wurde es aber nicht ==> expected: <true> but was: <false>
```

```
'-- JUnit Vintage [OK]

Failures (4):
  JUnit Jupiter:TMcca_nb_nTest:testOK(String):[1] "ccab"
    MethodSource [className = 'test.turingMaschine.TMcca_nb_nTest', methodName =
'testOK', methodParameterTypes = 'java.lang.String']
    => org.opentest4j.AssertionFailedError: Das Wort ccab sollte akzeptiert
werden, wurde es aber nicht ==> expected: <true> but was: <false>
```

Um zu einer lesbareren Ausgabe zu kommen, wird eine Erweiterung der Windows Powershell genutzt, die unter https://github.com/microsoft/terminal/releases geladen werden kann. Im Beispiel wird https://github.com/microsoft/terminal/releases/download/v1.19.10821.0/Microsoft.WindowsTerminal_1.19.10821.0_x64.zip genutzt. Um Java temporär in der Shell zu nutzen, kann folgender Befehl genutzt werden, der zum Installationsverzeichnis zeigen muss.

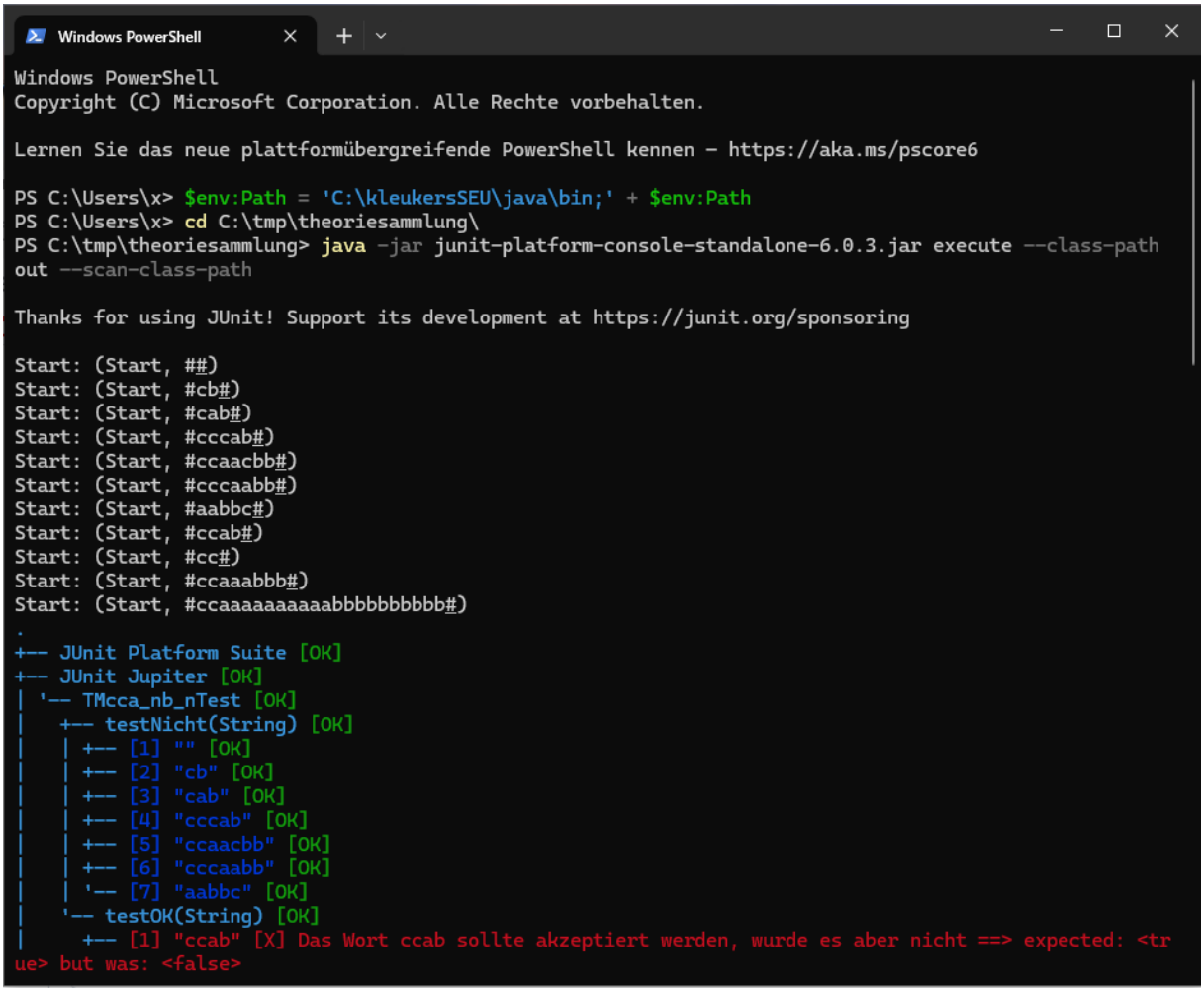
```
$env:Path = 'C:\kleukersSEU\java\bin;' + $env:Path
```

Es wird das Programm WindowsTerminal.exe gestartet. Generell werden fast die gleichen Befehle genutzt, allerdings der Class-Path (-cp) muss bei der Kompilierung in Anführungszeichen stehen und bei der Ausführung wird der letzte Parameter weggelassen.

```
javac -d out -cp 'junit-platform-console-standalone-6.0.3.jar;src'
src\test\turingMaschine\TMcca_nb_nTest.java


C:\tmp\theoriesammlung>java -jar junit-platform-console-standalone-6.0.3.jar
execute --class-path out --scan-class-path
```

Ein Ausschnitt des Ergebnisses sieht wie folgt aus:


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.
Lernen Sie das neue plattformübergreifende PowerShell kennen – https://aka.ms/pscore6
PS C:\Users\x> $env:Path = 'C:\kleukersSEU\java\bin;' + $env:Path
PS C:\Users\x> cd C:\tmp\theoriesammlung\
PS C:\tmp\theoriesammlung> java -jar junit-platform-console-standalone-6.0.3.jar execute --class-path
out --scan-class-path
Thanks for using JUnit! Support its development at https://junit.org/sponsoring
Start: (Start, ##)
Start: (Start, #cb#)
Start: (Start, #cab#)
Start: (Start, #cccab#)
Start: (Start, #ccaacbb#)
Start: (Start, #cccaabb#)
Start: (Start, #aabbc#)
Start: (Start, #ccab#)
Start: (Start, #cc#)
Start: (Start, #ccaaabbb#)
Start: (Start, #ccaaaaaaaaaabbbbbbbbbb#)
.
+-- JUnit Platform Suite [OK]
+-- JUnit Jupiter [OK]
| '-- TMcca_nb_nTest [OK]
| +-- testNicht(String) [OK]
| | +-- [1] "" [OK]
| | +-- [2] "cb" [OK]
| | +-- [3] "cab" [OK]
| | +-- [4] "cccab" [OK]
| | +-- [5] "ccaacbb" [OK]
| | +-- [6] "cccaabb" [OK]
| | '-- [7] "aabbc" [OK]
| '-- testOK(String) [OK]
| +-- [1] "ccab" [X] Das Wort ccab sollte akzeptiert werden, wurde es aber nicht ==> expected: <tr
ue> but was: <false>