Turingmaschine

Begriffe

Menge

Eine Menge ist eine Zusammenfassung von **unterscheidbaren** (unterschiedlichen) Elementen.

Beliebige Art von Elementen (Zahlen, Zeichen, Formen…)

Beliebige Anzahl

Leere Menge: $\emptyset = \{\ \}$

Unendlich Große Mengen, z. B. $\mathbb{N}$, $\mathbb{R}$

Element von: $1 \in \mathbb{N}$ -> 1 ist Element von $\mathbb{N}$ -> 1 ist in der Menge $\mathbb{N}$ enthalten

Teilmenge: $\{\ 1, 2\ \} \subseteq \mathbb{N}$ -> Die Menge { 1, 2 } ist Teilmenge von $\mathbb{N}$ -> $\mathbb{N}$ ist Obermenge von { 1, 2 }

Alphabet

Ein Alphabet ist eine Menge an Zeichen (bzw. Symbolen/Buchstaben).

Nicht unendlich groß

Nicht leer

z. B. { *a*, *b* }, { *0*, *1* }, { *a*, …, *z* }, { *1* }

Üblicherweise mit dem Buchstaben Σ bezeichnet

Wort

Ein Wort ist eine endlich lange **Folge von Zeichen** aus einem Alphabet.

Ein Wort hat immer ein Zusammenhang zu einem Alphabet. Man sagt z. B. *abab* ist ein Wort über das Alphabet { *a*, *b* }

Ein Wort kann leer sein (die Länge 0 haben) -> ε

Σ* ist die Menge aller Wörter über ein Alphabet

Wenn z. B. Σ = { *a*, *b* } dann ist Σ* = { ε, *a*, *b*, *aa*, *ab*, *ba*, *bb*, *aaa*, … }

Sprache

Eine Sprache über ein Alphabet ist eine Teilmenge aller möglichen Wörter über ein Alphabet.

L ⊆ Σ* -> L ist eine Sprache über Σ

Darf auch leer sein

Darf aber muss nicht unendlich groß sein

Beispiele (hier noch keine formale Schreibweise):

L = { *aa*, *ab*, *ba*, *bb* } -> Alle Wörter über { *a*, *b* } der Länge 2

L = { ε, *a*, *aa*, *ab*, *aaa*, ...} -> Alle Wörter über { *a*, *b* } die nicht mit *b* beginnen

L = Σ* -> Alle Wörter über dem Alphabet

Turingmaschine

Die Turingmaschine wurde 1936/37 von Alan Turing entwickelt und kann (unter anderem) genutzt werden, um zu entscheiden, ob ein Wort zu einer Sprache gehört oder nicht. Nicht für jede Sprache existiert zwangsweise auch eine passende Turingmaschine.

Funktionsweise

Die Turingmaschine hat ein beliebig großes Band, auf welchem ein endlich langes Wort geschrieben werden kann. Vor und hinter dem Wort wird mit einem “Blank” Symbol aufgefüllt. Ein **Leseschreibkopf** liest das Symbol ein und führt eine **Überführungsfunktion** aus, welche, anhand des eingelesenen Symbols und dem aktuellen Zustand, ein oder mehrere der folgenden Dinge entscheidet:

- Der Kopf bewegt sich nach rechts oder links
- Ein neues Symbol wird geschreiben
    - Das neue Symbol muss nicht aus dem Alphabet Σ sein, es gibt ein extra **Bandalphabet** mit $\Sigma \subset \Gamma$, welches mindestens noch ein “Blank” Symbol (*#*) enthält
- Die Turingmaschine wechselt in einen anderen **Zustand**
    - Abhängig vom neuen Zustand **hält** die Turingmaschine -> sie stoppt und akzeptiert das Eingabewort oder akzeptiert es nicht
    - Es gibt einen Anfangszustand und eine endliche Menge an möglichen Zuständen

Formale Bestandteile

Eine Turingmaschine kann formal und mathematisch definiert werden und setzt sich aus folgenden Mengen/Elementen zusammen (sie ist ein Tupel aus diesen Bestandteilen).

- Eingabealphabet; z. B. $\Sigma = \{ a, b \}$
- Bandalphabet; z. B. $\Gamma = \{ a, b, A, \# \}$ -> $\Sigma \subset \Gamma$ und # ist das “Blank” Zeichen
- Das “Blank” Symbol; $\# \in \Gamma$
- Endliche, nichtleere Menge an Zuständen Q
- Startzustand; $q_0 \in Q$
- Menge an Endzuständen; $F \subseteq Q$; $F = \{ q_{accept}, q_{reject} \}$
- Eine Überführungsfunktion

Entscheidbarkeit

Berechenbar

Ein Problem ist berechenbar -> Es gibt eine Turingmaschine, welche das Problem Lösen kann

Entscheidbar

Ein Problem ist entscheidbar -> Die Turingmaschiene, welche das Problem löst, wird immer (egal bei welcher Eingabe) irgendwann stoppen, also in einen Endzustand übergehen.

Semi-Entscheibar

Ein Problem ist semi-entscheidbar -> Es gibt eine Turingmaschine, welche das Problem löst und bei akzeptierten Eingaben immer irgendwann stoppt, aber bei nicht akzeptierten Eingaben eventuell unendlich lange weiterläuft.

Halteproblem

Beim Halteproblem wird nach einem Algorithmus gesucht, welcher für ein beliebiges Programm entscheidet, ob dieses Programm irgendwann stoppt oder beliebig lang weiterläuft. Das Halteproblem ist ein Beispiel für ein semi-entscheidbares Problem: Es gibt eine Turingmaschine, welche stoppende Programme erkennt und akzeptiert, aber eventuell für unendlich lange laufende Programme selbst unendlich lang weiterläuft.

Turing-Vollständigkeit

Ein System ist Turing-vollständig, wenn es alles berechnen kann, was auch eine Turingmaschine berechnen kann. Einschränkungen wie z. B. Speicher werden hier nicht mit einberechnet -> die meisten Programmiersprachen sind Turing-vollständig.