

TM für eine Klammersprache

Aufgabenstellung

In dieser Aufgabe sollen Sie eine deterministische Turingmaschine bauen, die die Sprache aller korrekten Klammerungen mit $()$ und $[]$ erkennt, also z.B. $()()$ oder $([]())()$. Formal ausgedrückt ist dies die Sprache, die von der Grammatik $S \rightarrow SS \mid (S) \mid [S] \mid \varepsilon$ erzeugt wird. Für jedes Wort in dieser Sprache muss Ihre TM in einem Endzustand halten; für jedes Wort, das nicht in der Sprache ist, muss Ihre TM in einem Nicht-Endzustand “stecken bleiben”.

Im Judge sollen Sie dazu ein Programm einreichen, dass diese Turingmaschine auf stdout ausgibt.

Sie haben hierzu mehrere Möglichkeiten:

- Sie können das bereitgestellte Java-Template verwenden, welches eine Klasse `TuringMachine` zur Verfügung stellt. Damit können Sie ihre TM schrittweise aufbauen, indem Sie Zustände, Endzustände und Transitionen hinzufügen. Ein Beispiel für die Verwendung finden Sie ebenfalls in der `.tar.gz`-Datei auf der Webseite. **Hinweise zum Debuggen:** Die Templates beinhalten ebenfalls `run`-Methoden, mit denen Sie für eigene Tests die Turingmaschine auf einer bestimmten Eingabe ausführen können. Sie können sich eine TM auch mithilfe der Methode `TuringMachine.toDot` im Dot-Format ausgeben lassen, welches Sie dann mithilfe von Graphviz (oder diverser Webseite, die dieses Format ebenfalls unterstützen) in eine graphische Repräsentation der Maschine verwandeln können.
- Alternativ können Sie die Turingmaschine auf <https://wimmers.github.io/turing-machine-viz/> erstellen. Dort werden Ihre Lösungsversuche graphisch dargestellt und können simuliert werden. Beachte Sie dabei dass Sie auf der Website auch nichtdeterministische Turingmaschinen erstellen können, in dieser Aufgabe aber nach einer deterministischen Maschine gefragt ist. Außerdem visualisiert das Tool Endzustände nicht, sie können diese aber ähnlich wie den Startzustand definieren. Um zum Beispiel die Zustände *qf* und *final* als Endzustände zu markieren, fügen Sie dem Code einfach die Zeile `final states: [qf, final]` hinzu. Auf der Website werden die Maschinen in einem Yaml basierten Format definiert, dass Sie zuerst umwandeln müssen um es im Judge einreichen zu können. Nutzen Sie hierfür die Export Funktion der Website und wandeln Sie die resultierende Datei mit der in den Materialien bereitgestellten `convertYamlTM.jar` um. Liegt ihre exportierte Maschine z.B. in einer Datei `tm.yaml`, gibt Ihnen der Aufruf `java -jar convertYamlTM.jar tm.yaml` ein Python Programm aus dass Sie direkt im Judge einreichen können.
- Sie können die Turingmaschine auch direkt von Hand entwerfen. Dann müssen Sie lediglich ein Programm in einer der im Judge akzeptierten Sprachen schreiben, dass die Maschine im richtigen Format auf stdout ausgibt.

Bewertungshinweise

Um zu testen, ob Sie eine korrekte TM zurückgeben, wird Ihre TM für verschiedene Eingaben simuliert und geprüft. Im Falle einer Diskrepanz wird Ihnen angezeigt, welches Wort eine falsche Ausgabe verursacht.