Theorie der Informatik (CS 206)

Prof. Dr. M. Helmert, Prof. Dr. C. Tschudin
Dr. M. Wehrle
Frühjahrssemester 2013

Universität Basel
Fachbereich Informatik

Übungsblatt 5

Abgabe: 10. April

Bei diesem Übungsblatt handelt es sich um eine Programmieraufgabe. Sie dürfen Programmieraufgaben in den Sprachen C, C++, Java oder Python sowie weiteren Programmiersprachen nach Abspache mit Ihrem Tutor bearbeiten.
Wir erwarten, dass Sie Ihre Implementierung selbständig, das heisst ohne Anwendung von fremdem Code z.B. aus dem Internet erstellen. Die Standardbibliothek Ihrer gewählten Programmiersprache dürfen Sie selbstverständlich ohne Einschränkung verwenden.
Bei technischen Schwierigkeiten oder Verständnisproblemen helfen wir gerne weiter. Bitte wenden Sie sich dazu *mit genügend zeitlichem Abstand zum Abgabetermin* an Ihren Tutor.

Aufgabe 5.1 (Turing-Maschinen, 4+3+3+2 Punkte)

Im Rahmen dieser Aufgabe soll ein Simulator für Turing-Maschinen implementiert werden.
Geben Sie Ihre Lösung bitte nicht auf Papier ab, sondern laden Sie sie im Courses-Portal hoch. Falls Sie Testfälle schreiben, dann geben Sie sie bitte ebenfalls mit ab (dies kann sich nur positiv auf die Bewertung auswirken). Wählen Sie für Ihre Implementierung geeignete Datenstrukturen.

(a) Schreiben Sie eine Klasse `Tape`, die das beidseitig unendliche Band einer Turing-Maschine (inklusiv der Position des Schreib-/Lesekopfes) repräsentieren soll:

Der Konstruktor `Tape(word, blank)` bekommt die initiale Bandbelegung und das zu verwendende Blank-Symbol übergeben. Der Schreib-/Lesekopf soll sich initial auf dem ersten Zeichen der Eingabe befinden.

Eine Methode `read()` soll das Zeichen unter dem Kopf zurückgeben.

Eine Methode `write(symbol)` soll das Zeichen `symbol` an der Position des Kopfes auf das Band schreiben.

Die Methoden `move_right()` und `move_left()` sollen den Kopf um eine Position nach rechts oder nach links verschieben.

Die Methoden `dump_alpha()` und `dump_beta()` sollen das bisher verwendete Band links des Kopfes bzw. rechts des Kopfes (inkl. der Kopfposition) ausgeben.

Eine Methode `used_space()` soll die Grösse (die Anzahl der Bandfelder) des bisher verwendeten Bandes zurückgeben.

Hinweis: Wir verwenden hier den Python-Stil für Methodennamen. Wenn Sie Ihre Implementierung z.B. in Java machen, dürfen Sie auch gerne Camel Case verwenden (also `usedSpace()` statt `used_space()`).

(b) Schreiben Sie eine Klasse `TuringMachine` mit dem Konstruktor `TuringMachine(Z, Sigma, Gamma, delta, z0, blank, E)`, wobei die einzelnen Parameter den normalen Elementen einer Turing-Maschine entsprechen. Bei der Erstellung der Turing-Maschine soll sichergestellt werden, dass sie korrekt spezifiziert ist: Ist `delta` eine totale Funktion? Ist `Sigma` Teilmenge von `Gamma`? Ist das `blank` in `Gamma`, aber nicht in `Sigma`? Ist `E` eine Teilmenge von `Z`? Werfen Sie eine Exception, wenn die Spezifikation nicht in Ordnung ist.

(Bitte wenden)

(c) Erweitern Sie die Klasse `TuringMachine` um folgende Methoden:

`initialize(word)` soll die Turing-Maschine initialisieren (d.h. die initiale Konfiguration herstellen). Stellen Sie bitte sicher, dass nur Symbole aus dem Eingabealphabet in `word` vorkommen (sonst werfen Sie eine Exception).

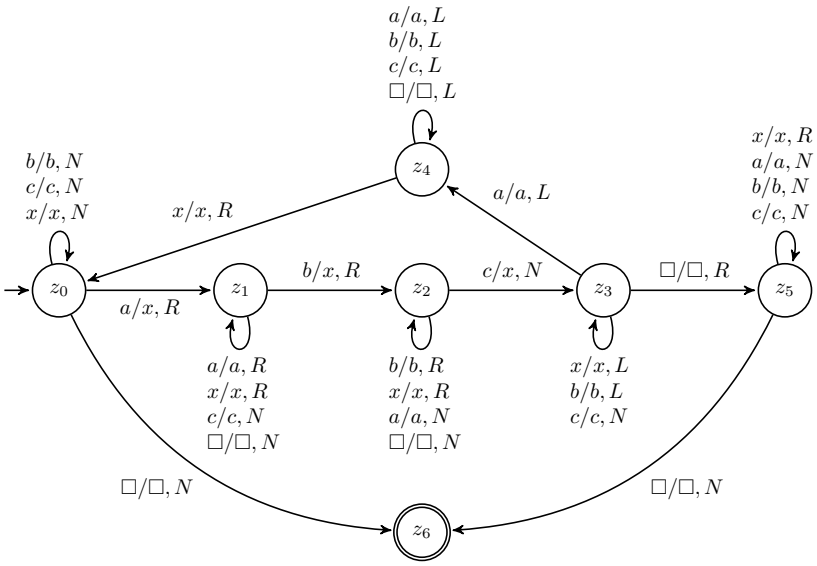
`step()` soll einen Schritt der Turing-Maschine ausführen. Werfen Sie eine Exception, wenn die Maschine nicht initialisiert wurde oder in einem Endzustand ist. Alternativ können Sie `step` auch privat machen und einfach Assertions verwenden.

`run(max_steps)` soll die Turing-Maschine laufen lassen. Der Parameter `max_steps` ist optional. Ist er gesetzt, wird die Maschine nach `max_steps` Schritten angehalten, auch wenn sie noch keinen Endzustand erreicht hat (Dies ist nützlich, um eine Maschine auf einem Wort testen zu können, auf dem sie nicht hält). Wird die Methode ohne Argumente aufgerufen, soll die Turing-Maschine laufen, bis sie einen Endzustand erreicht.

Eine Methode `dump_configuration()` soll die aktuelle Konfiguration der Turing-Maschine ausgeben.

Eine Methode `dump_statistics()` soll ausgeben, wieviele Schritte bereits gemacht wurden und wieviel Platz auf dem Band verwendet wurde.

(d) Betrachten Sie die folgende Turing-Maschine.



Schreiben Sie eine `main`-Funktion, in der die `TuringMachine` für die obige Turing-Maschine erstellt und auf verschiendenen Eingaben (darunter das leere Wort, ein Wort in der Sprache und ein Wort, das nicht in der Sprache liegt) getestet wird. Dabei soll bei jedem Schritt die Konfiguration ausgegeben werden (Sie können das standardmässig in der Methode `run` tun). Für Wörter, die nicht in der Sprache liegen, verwenden Sie bitte einen geeigneten Wert für `max_steps`, bei dem man sehen kann, dass die TM nicht mehr halten wird.

Zu Ihrer Kontrolle: Auf der Eingabe *aabbcc* benötigt die Turing-Maschine 29 Schritte und verwendet 8 Bandzellen.