

Projekt: Entwicklung einer Infsii-Turtle-Code-Programmierungsumgebung

Entwicklung eines Parsers

Wir wollen in Anlehnung an Eckart Modrows¹ Projektvorschlag „LOGO für Arme“ in Snap! (oder einer anderen aus dem Unterricht bekannten Programmierungsumgebung) eine textbasierte Bildbeschreibungssprache ähnlich zur Programmiersprache Logo entwickeln. Wir nennen diese selbst entwickelte Sprache *Infsii-Turtle-Code*.

Im Folgenden wird davon ausgegangen, dass das eigentliche Projekt sowie die einzelnen Bestandteile bereits in der Lerngruppe besprochen wurden (vgl. die zugehörigen Infsii-Materialien). Wir konzentrieren uns an dieser Stelle auf die schrittweise Entwicklung eines Parsers. Sie entscheiden selbst, wie umfangreich Ihr Parser gestaltet sein soll. Die Mindestanforderung ist, dass syntaktisch falsche Eingaben erkannt werden. Wenn Sie sich auf eine konkrete Sprachdefinition geeinigt haben, können Sie eigenständig mit der Entwicklung eines Parsers (d.h. Erstellung eines geeigneten Automatenmodells sowie zugehörige Implementierung in einer aus dem Unterricht bekannten Programmiersprache) beginnen. Alternativ können Sie sich an den folgenden Beispielen orientieren und diese jederzeit an Ihre eigene Sprachdefinition und Festlegung auf Art der Rückmeldungen anpassen bzw. erweitern.

Sprachdefinition:

Wir gehen im Folgenden von folgender Sprachdefinition aus:

Befehl	Bedeutung
F (x)	Die Turtle bewegt sich um x Schritte nach vorne (F orward).
U	Der Stift wird deaktiviert (U p).
D	Der Stift wird aktiviert (D own).
T (x)	Die Turtle dreht sich um den Winkel x im Uhrzeigersinn nach rechts (T urn).

Tabelle 1: Erste Version einer Turtle-Sprache

Aufgaben:

- Überprüfen Sie, ob die Eingabe F (20) UDT (332) F (22) syntaktisch korrekt ist.
- Geben Sie unabhängig von Aufgabenteil a) jeweils ein Beispiel für eine syntaktisch korrekte und eine syntaktisch falsche Eingabe an.

¹ vgl. Modrow, Eckart, Theoretische Informatik mit Delphi, emu-online, 2005 bzw. Modrow, Eckart, „Einführung in die Informatik – Teil VII Erkennende Automaten“, vlin-Material

Beschreibung eines Parsers

Beim Kompilieren eines Quelltextes überprüft ein Parser, ob dieser syntaktisch korrekt ist. Abhängig von der Art der Eingabe gibt der Parser unterschiedliche Rückmeldungen. In Abbildung 1 ist ein Übergangsgraph eines einfachen Parsers dargestellt. Zur besseren Lesbarkeit wurden alle Übergänge in den einzigen Fehlerzustand weggelassen, d.h. alle nicht aufgeführten Eingaben führen in den Fehlerzustand q_{-1} .

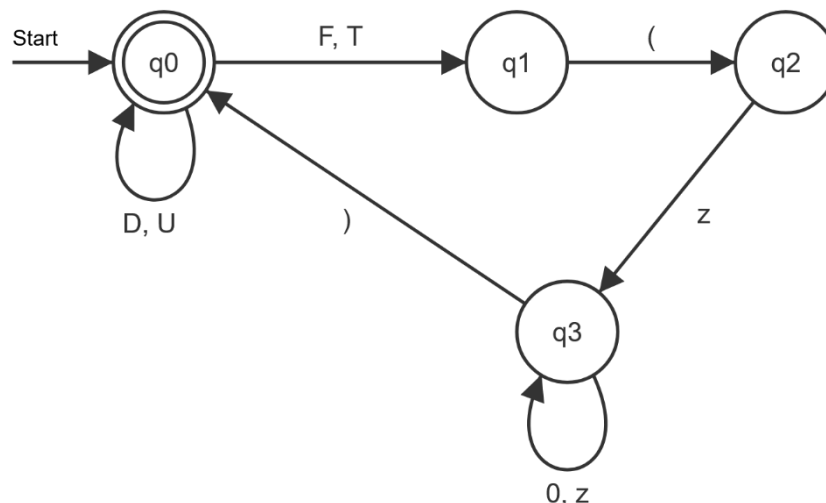


Abbildung 1: Übergangsgraph eines einfachen Parsers zum Eingabealphabet $\{F, T, D, U, 0, z\}$, hierbei steht z für eine Ziffer ungleich 0; alle nicht aufgeführten Eingaben führen in einen Fehlerzustand

Ausgehend von der Modellierung kann nun ein einfacher Parser implementiert werden, der für eine gegebene Eingabe entscheidet, ob sie syntaktisch korrekt ist oder nicht. Je komplexer der Parser wird, desto hilfreicher ist bei der Implementierung das „rezeptartige Vorgehen“ aus dem Material „Implementierung_Automat“:

- (1) Die einzelnen Zustände werden durch eine ganzzahlige Variable `zustand` kodiert. Diese hat zu Beginn den Wert `zustand=0`.
- (2) Die Eingabe wird mithilfe einer Schleife vollständig durchlaufen.
- (3) Abhängig vom aktuellen Zustand und dem aktuellen Eingabezeichen wird dabei jeweils der Folgezustand berechnet. Hierfür verwendet man eine geschachtelte Verzweigung.
- (4) Ist das Eingabewort vollständig durchlaufen, wird es als zur Sprache zugehörig akzeptiert, falls die Zustandsnummer `zustand` für einen Endzustand steht.

Unabhängig vom Automaten sind insbesondere die ersten beiden Aspekte (1) und (2) immer gleich. Die Methode (3) muss abhängig vom Automaten angepasst werden. In (4) muss jeweils die Liste der Endzustände aktualisiert werden.

Aufgaben

- Der durch den Übergangsgraphen in Abbildung 1 definierte Parser soll zwei Rückmeldungen erlauben: *Eingabe gehört zur Turtle-Code-Sprache* oder *Eingabe ungültig*. Testen Sie mithilfe des Automaten, ob die Eingaben F(03), DUT(1) und UT(11) jeweils syntaktisch korrekt sind und geben Sie die zugehörigen Rückmeldungen an.
- Implementieren Sie basierend auf dem Automatenmodell aus Abbildung 1 einen einfachen Parser. Sie können als Vorlage die Datei `Turtle-Parser-Vorlage.xml` verwenden und rezeptartig wie oben beschrieben vorgehen.

Bei der Implementierung eines Parsers sollen Sie selbst entscheiden, wie detailliert die Rückmeldungen sein sollen. Im Beispiel waren sie sehr einfach gehalten, es wurde nur zwischen syntaktisch korrekten und syntaktisch falschen Eingaben unterschieden. Alternativ können auch qualifiziertere Rückmeldungen erfolgen, wie beispielsweise „Fehler: Ziffer erwartet, stattdessen D gelesen“ oder „Fehler: Eingabe unvollständig“.

Aufgaben

- Ergänzen Sie den Automaten aus Abbildung 1 um vier Fehlerzustände q4 – q7. Anhand dieser soll zwischen unterschiedlichen Eingabefehlern unterschieden werden können. Beispiele solcher Fehler wären etwa „Fehler: Ziffer erwartet, stattdessen D gelesen“ oder „Fehler:) erwartet, stattdessen 4 gelesen“.
- Implementieren Sie basierend auf Ihrem Automaten aus a) einen erweiterten Parser mit qualifizierteren Rückmeldungen.

Mögliche Erweiterungen der Sprachdefinition

Die Infsii-Turtle hat zunächst nur wenige Befehle. In der folgenden Tabelle werden weitere mögliche Befehle vorgestellt. Diese Liste ist keineswegs vollständig und kann um eigene Ideen ergänzt werden.

Aufgabe: Wählen Sie mehrere neue Befehle aus und überlegen sich geeignete Codierungen. Erweitern Sie anschließend Ihr Automatenmodell des zugehörigen Parsers entsprechend und implementieren Sie es.

Befehl	Bedeutung
	Die Turtle bewegt sich um x Schritte rückwärts.
	Die Turtle dreht sich um den Winkel x entgegen dem Uhrzeigersinn.
	Die Stiftstärke wird geändert.
	Die Stiftfarbe wird geändert.
	Die Stiftfarbe wird auf den RGB-Wert (r, g, b) gesetzt.
	Die Turtle versteckt sich.
	Die Turtle wird sichtbar.
	Der Bildschirm wird gelöscht.
	Die Turtle geht zur Position (x, y) .
	Eine Befehlsfolge wird x-mal wiederholt.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#). Sie erlaubt Bearbeitungen und Weiterverteilung des Werks unter Nennung meines Namens und unter gleichen Bedingungen, jedoch keinerlei kommerzielle Nutzung.

Haftung Quellcode:

Für die korrekte Ausführbarkeit der Quelltexte in diesem Material wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.

Bildnachweis: Die Abbildungen zu endlichen Automaten wurden mit Flaci erstellt:

<https://flaci.com/home/> (Link vom 14.05.2020).