Aufgaben Theoretische Informatik

Elmar Eder

11. Juni 2019

Aufgabe 1 Im Prolog-Programm `ggT.pl` ist der euklidische Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen implementiert. Unter einem *Teiler* einer natürlichen Zahl a verstehen wir eine natürliche Zahl t derart, dass eine natürliche Zahl u existiert mit $t \cdot u = a$. Wir schreiben dann $t|a$. Unter einem *gemeinsamem Teiler (ggT)* zweier natürlicher Zahlen a und b verstehen wir eine Zahl t, die Teiler sowohl von a als auch von b ist, also so, dass $t|a \wedge t|b$ gilt. Jede natürliche Zahl t ist Teiler der Null 0. Jede natürliche Zahl $a \neq 0$ hat aber nur endlich viele Teiler. Daher gibt es zu je zwei Zahlen a und b – außer, wenn a und b beide gleich 0 sind, – stets einen größten gemeinsamen Teiler, den wir als den ggT von a und b bezeichnen. Wenn $a = b = 0$ ist, sind alle natürlichen Zahlen gemeinsame Teiler von a und b, und es gibt daher keinen größten gemeinsamen Teiler von a und b. Um trotzdem auch in diesem Fall ein wohldefiniertes Ergebnis zu bekommen, definieren wir den ggT von 0 und 0 als 0.

Probieren Sie das Programm zunächst mit der Anfrage

```
?- ggT(12,8,T).
```

aus (nicht abzugeben)! Wenn Sie zuvor mit

```
?- trace.
```

in den Trace-Modus umschalten, können Sie Prolog beim Beweisen der einzelnen Ziele zusehen. `Call` bedeutet, dass Prolog versucht, ein Ziel zu beweisen; `Exit` bedeutet, dass Prolog ein Ziel bewiesen hat. Prolog verwendet bei der Anzeige der Ziele interne Namen für die Variablen, die mit einem Underscore anfangen. So kann es etwa die Variable `T` als `_G2756` anzeigen und somit das Ziel `ggT(12,8,T)` als `ggT(12, 8, _G2756)` anzeigen. Mit der RETURN-Taste schalten Sie einen Schritt weiter. Beobachten Sie dabei die Zeilen, die einen `Call` von `ggT` beschreiben (nicht abzugeben)! Ich gebe hier diese Zeilen auf das Wesentliche verkürzt wieder:

```
Call: ggT(12, 8, _G2756)
Call: ggT(8, 12, _G2756)
Call: ggT(4, 8, _G2756)
Call: ggT(0, 4, _G2756)
```

Sie sehen, dass dabei das erste Argument von `ggT` jeweils von einem solchen `Call` zum nächsten immer kleiner wird (im obigen Beispiel: 12, 8, 4, 0).

Mit `?- nodebug.` können Sie Prolog wieder in den normalen Modus umschalten.

Die Definition des Prädikats `ggT` im Programm `ggT.pl` besteht aus zwei Klauseln.

(a) Formulieren Sie jede der beiden Klauseln des Programms `ggT.pl` als Aussage in deutscher Sprache (unter Verwendung der Abkürzung ggT)!

(b) Zeigen Sie, dass diese beiden Aussagen wahr sind! Wegen der Korrektheit von Prolog folgt daraus dann, dass das Programm keine falschen Antworten liefert.

(c) Wenn Prolog eine Anfrage abarbeitet, versucht es, sie zunächst mittels der ersten Klausel des Programms zu beweisen. Wenn das nicht geht, versucht es dies mit der zweiten Klausel des Programms. Dabei ruft es dasselbe Prädikat `ggT` wiederum mit anderen Argumenten (C,A,T) auf. Man nennt das einen *rekursiven Aufruf*. Dieser Aufruf kann wiederum zu einem rekursiven Aufruf von `ggT` führen, usw. Zeigen Sie, dass bei der Berechnung des ggT zweier natürlicher Zahlen mit diesem Programm die Kette der rekursiven Aufrufe stets nach endlich vielen Aufrufen enden muss und daher das Programm stets terminiert!

Aufgabe 2 Sei A die Aussage „2 ist ungerade“ und B die Aussage „3 ist eine Primzahl“ (Beispiel aus der Vorlesung). Geben Sie für jede der Aussagen $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$ und $A \Leftrightarrow B$ an, ob sie wahr oder falsch ist!

Aufgabe 3 In Prolog wird ein Prädikat durch seinen Namen gefolgt von einem Schrägstrich und seiner Stelligkeit bezeichnet. So gibt es zum Beispiel in SWI-Prolog (swipl) und in GNU-Prolog (gprolog) ein zweistelliges Prädikat `member` und ein dreistelliges Prädikat `append`. Diese werden bezeichnet mit `member/2` bzw. `append/3`. Machen Sie sich zunächst mit diesen beiden Prolog-Prädikaten vertraut, indem Sie unter anderen die Anfragen

```
?- member(X,[a,b,c]).
?- member(a,L).
?- append([a,b],[c,d,e],L).
?- append(L1,L2,[a,b,c,d]).
?- append(L1,L1,L).
?- append(L1,L2,L).
```

eingeben und indem Sie in SWI-Prolog `?- help.` aufrufen und in dem daraufhin erscheinenden Fenster in dem kleinen Eingabefeld `member/2` bzw. `append/3` eingeben bzw. in GNU-Prolog im Manual nachschlagen! Implementieren Sie dann diese beiden Prädikate als selbst definierte Prolog-Prädikate `element/2` bzw. `konkatenation/3` ohne Verwendung der eingebauten Prädikate!

Aufgabe 4 Bei dem Spiel *Nim* spielen zwei Spieler gegeneinander. Es liegen mehrere Haufen von Steinen auf einem Tisch. Die Spieler ziehen abwechselnd. Jeder Zug besteht darin, aus einem Haufen ein oder mehr Steine zu entfernen. Wer als erster nicht mehr ziehen kann, weil alle Haufen leer sind, hat verloren.

Geben Sie einen Algorithmus an, der für einen Spieler, der gerade am Zug ist, feststellen kann, ob es für ihn möglich ist, auch gegen einen guten Gegner zu gewinnen! Außerdem soll der Algorithmus in diesem Fall einen Zug ausgeben, der dem Spieler den Gewinn garantiert. Implementieren Sie Ihren Algorithmus

in Prolog! Bauen Sie damit ein Nim-Programm, das gegen einen menschlichen Gegner optimal spielen kann! Testen Sie Ihr Programm für einige Spielstellungen mit wenigen Steinen!

Hinweis: Wenn G ein Prolog-Ziel ist, dann ist `\+`G die *negation as failure* von G. Das Ziel `\+`G gelingt genau dann, wenn das Ziel G fehlschlägt. Bei einem Aufruf von `\+`G versucht Prolog zunächst G zu beweisen. Wenn dies gelingt, schlägt der Aufruf von `\+`G fehl. Andernfalls gelingt der Aufruf von `\+`G, ohne dass dabei irgendwelche Variablen ersetzt werden.

Aufgabe 5 Gibt es für das Spiel Schach einen Algorithmus, der ähnlich wie in der vorigen Aufgabe, in endlicher Zeit feststellen kann, ob es bei einer gegebenen Schachstellung einen Zug gibt, der dem ziehenden Spieler den Gewinn garantiert? Man sagt dann, es sei *entscheidbar*, ob es einen solchen Zug gibt. Begründen Sie Ihre Antwort!

Aufgabe 6 Geben Sie für $\neg((A \vee \neg B) \Leftrightarrow \neg(A \wedge C))$ die Wahrheitswerttafel an!

Aufgabe 7 Zeichnen Sie einen deterministischen endlichen Automaten, der die Menge aller Wörter über dem Alphabet $\{a, b\}$ von gerader Länge akzeptiert!

Aufgabe 8 Das dreistellige Prädikat P auf der Menge der natürlichen Zahlen sei definiert durch

$$P(x, y, z) \Leftrightarrow x \cdot y = z.$$

Geben Sie für jede der folgenden Aussageformen über dem Individuenbereich der natürlichen Zahlen an, für welche Werte der Variablen x, y und z diese Aussageform wahr ist und für welche Werte der Variablen x, y und z sie falsch ist!

$$\exists x\, P(x, 2, z)$$
$$\exists y\, P(x, y, z)$$
$$\exists z\, P(x, y, z)$$
$$\exists x \exists y \big(P(x, y, z) \wedge x > 1 \wedge y > 1\big)$$
$$\forall x \exists y \exists z\, P(x, y, z)$$
$$\forall x \exists y \forall z\, P(x, y, z)$$

Aufgabe 9 Schreiben Sie

$$\bigwedge_{\epsilon > 0} \bigvee_{\delta > 0} \delta < \epsilon$$

so um, dass darin keine eingeschränkte Quantifizierung mehr vorkommt. Stellt dies als Aussage über dem Individuenbereich der ganzen Zahlen betrachtet eine wahre oder eine falsche Aussage dar? Stellt es als Aussage über dem Individuenbereich der reellen Zahlen betrachtet eine wahre oder eine falsche Aussage dar?

Aufgabe 10 Sei $f\colon \mathbb{R} \to \mathbb{R}$ eine Funktion auf der Menge der reellen Zahlen. Drücken Sie die Aussage „f ist differenzierbar“ mittels eingeschränkter Quantifizierung aus! Wandeln Sie die so dargestellte Aussage um in eine Darstellung in der Logiksprache der Prädikatenlogik erster Stufe mit Quantoren ohne Einschränkungen auf Teilbereiche des Individuenbereichs!

Aufgabe 11 Zeigen Sie, dass aus den Gleichheitsaxiomen folgt, dass die Gleichheitsrelation $=$ eine Äquivalenzrelation ist:

$$\begin{array}{cc} \forall x\, x = x & \text{(Reflexivität)} \\ \forall x \forall y (x = y \Rightarrow y = x) & \text{(Symmetrie)} \\ \forall x \forall y \forall z (x = y \wedge y = z \Rightarrow x = z) & \text{(Transitivität)} \end{array}$$

Aufgabe 12 Wieviele n-stellige aussagenlogische Verknüpfungen gibt es? Geben Sie alle n-stelligen aussagenlogischen Verknüpfungen für $n = 0$, für $n = 1$ und für $n = 2$ an!

Aufgabe 13 Geben Sie alle einstelligen Prädikate auf der Menge $\{a, x, z\}$ an! Geben Sie alle zweistelligen Prädikate auf der Menge $\{3, 4\}$ an!

Aufgabe 14 Sei D ein Individuenbereich mit k Elementen. Wieviele n-stellige Prädikate gibt es auf D? Schreiben Sie ein Prolog-Programm, das alle n-stelligen Prädikate auf D ausgibt!

Aufgabe 15 Geben Sie einen Individuenbereich D und ein zweistelliges Prädikat P auf D an derart, dass die Aussage

$$\forall x \neg P(x, x) \wedge \forall x \forall y \forall z \big(P(x, y) \wedge P(y, z) \Rightarrow P(x, z)\big) \wedge \forall x \exists y\, P(x, y)$$

wahr ist! Wir sagen, durch D und P sei ein *Modell* dieser Aussage gegeben. Geht das mit einem endlichen Individuenbereich D?

Aufgabe 16 Beweisen Sie durch vollständige Induktion, dass

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$$

für alle natürlichen Zahlen n gilt!

Aufgabe 17 Mengen, die in endlich vielen Schritten aus einfachen Elementen aufgebaut sind, kann man in Prolog durch (eventuell geschachtelte) Listen darstellen. Implementieren Sie die Operationen und Begriffe der Mengenlehre, die wir in der Vorlesung kennengelernt haben, in Prolog!

Aufgabe 18 In Prolog lässt sich mit `op` ein Prolog-Funktor (also Funktionszeichen oder Prädikatszeichen) als Infix-, Präfix- oder Postfix-Operator deklarieren. Mit `current_op` können Sie sich alle als Infix-, Präfix- oder Postfix-Operatoren deklarierten Funktoren auflisten lassen. Schauen Sie sich die Dokumentation von `op` und `current_op` an!

Im Prolog-Programm-Fragment `wahrheitswert1.pl` sind die zwei Funktoren `nicht/1` und `und/2` als Präfix- bzw. Infix-Operator deklariert. Man kann dann z.B. `nicht A` statt `nicht(A)` schreiben und `A und B` statt `und(A,B)` schreiben. Das Programm soll den Wahrheitswert einer mittels Junktoren aus den Aussagen A und B von Aufgabe 2 zusammengesetzten Aussage berechnen können. Verbessern Sie das Programm-Fragment, indem Sie zwecks besserer Lesbarkeit für die Junktoren einheitlich Präfix- und Infix-Schreibweise im Programm

verwenden und indem Sie die fehlenden Fälle und Junktoren ergänzen! Das Programm soll die Wahrheitswerte der zusammengesetzten Aussagen der Aufgabe 2 berechnen können und darüberhinaus die Wahrheitswerte auch von beliebigen mit $\neg$, $\wedge$, $\vee$, $\Rightarrow$ und $\Leftrightarrow$ aus A und B zusammengesetzten Aussagen. Lassen Sie sich von Prolog den Wahrheitswert von $(A \Rightarrow B) \Rightarrow B$ und den Wahrheitswert von $A \Rightarrow A \wedge B$ berechnen!

Ändern Sie schließlich noch Ihr Programm, indem Sie den Namen des Prolog-Prädikats von `wahrheitswert` zu `hat_den_Wahrheitswert` ändern und dieses Prolog-Prädikat als Infix-Operator deklarieren! Verwenden Sie auch im Programmtext die Infixschreibweise! Geben Sie dann Anfragen zum Wahrheitswert jeweils einer zusammengesetzten Aussage in Infixnotation ein!

Aufgabe 19 Seien A, B und C Aussagen. Geben Sie für jede der folgenden zusammengesetzten Aussagen den entsprechenden Ausdruck unter möglichst weitgehender Weglassung von Klammern aufgrund der Vorrangregeln, sowie den Strukturbaum an!

$$A$$
$$\neg B$$
$$\neg A \vee (\neg B)$$
$$A \Leftrightarrow (\neg B \wedge (\neg\neg(B \wedge A)))$$

Aufgabe 20 Geben Sie alle Tripel (d.h. 3-Tupel) von Elementen der Menge $\{a, b\}$ an! Geben Sie alle Paare (d.h. 2-Tupel) von Elementen der Menge $\{3, 9, 11\}$ an!

Aufgabe 21 Geben Sie die Potenzmenge der Menge $\{2, 3, 4\}$ an! Geben Sie die Potenzmenge der Menge $\{(2, 3), (4, 5)\}$ an!

Aufgabe 22 Nehmen Sie die Logik mit Gleichheitsaxiomen sowie die Axiome der Mengenlehre an. Zeigen Sie damit, dass gilt: Seien A, B und C Mengen. Zeigen Sie

(a) Umkehrung des Extensionalitätsaxioms:
Wenn $A = B$ ist, dann gilt $\forall x(x \in A \Leftrightarrow x \in B)$.

(b) Wenn A eine Menge ist, dann ist auch $\{A\}$ eine Menge.

(c) Die Vereinigung zweier Mengen ist eine Menge.

(d) Wenn A, B und C Mengen sind, dann ist auch $\{A, B, C\}$ eine Menge.

Aufgabe 23 Geben Sie für jedes der folgenden Erzeugungssysteme an, welche Menge von Objekten es erzeugt!

(a) Objekte sind reelle Zahlen.

Axiom: 0

Regel: $\dfrac{x}{x+1}$

(b) Objekte sind reelle Zahlen.

$$\text{Axiom:} \qquad 1$$

$$\text{Regel:} \qquad \frac{x \qquad y}{x - y}$$

(c) Objekte sind reelle Zahlen.

$$\text{Axiom:} \qquad 1$$

$$\text{Regeln:} \qquad \frac{x \qquad y}{x - y}$$

$$\frac{x \qquad y}{x/y}\ , \quad \text{wenn } y \neq 0.$$

(d) Gegeben sei ein Würfel im dreidimensionalen euklidischen Raum. Objekte sind Punkte des Raumes.

$$\text{Axiome:} \qquad \text{Die 8 Eckpunkte des Würfels}$$

$$\text{Regel:} \qquad \frac{\vec{x} \qquad \vec{y}}{\alpha\vec{x} + (1-\alpha)\vec{y}}\ , \quad \text{wenn } 0 \leq \alpha \leq 1.$$

Die Regel sagt aus, dass man aus zwei Punkten des Raumes jeden Punkt auf der Verbindungsstrecke dieser beiden Punkte erzeugen kann.

(e) Gegeben seien zwei zueinander windschiefe Geraden g und h im dreidimensionalen euklidischen Raum. Objekte sind Punkte des Raumes.

$$\text{Axiome:} \qquad \text{Alle Punkte der Geraden } h$$

$$\text{Regel:} \qquad \frac{\vec{x}}{\vec{y}}, \text{wenn } \vec{y} \text{ aus } \vec{x} \text{ durch Drehung um die Achse } g \text{ entsteht}$$

Aufgabe 24 Betrachten wir als Objekte (Gegenstände) die natürlichen Zahlen. Dann wird durch die folgenden Regeln ein Erzeugungssystem definiert:

$$\text{Axiome:} \qquad 3$$

$$5$$

$$\text{Regel:} \qquad \frac{x \quad y}{x + y}.$$

Welche Zahlen sind in diesem Erzeugungssystem herleitbar?

Aufgabe 25 Gegeben sei das Alphabet $\Sigma = \{a, b\}$. Die Sprache L_1 über dem Alphabet Σ sei die Menge aller Wörter über Σ, die nur aus a's bestehen, d.h. die das Zeichen b nicht enthalten. Die Sprache L_2 über dem Alphabet Σ sei die Menge aller Wörter über Σ, die nur aus b's bestehen. Geben Sie für jede der Sprachen $\overline{L_1}$, $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$, $L_1 L_2$, $\{a, b\}^*$, $\{ab\}^*$, L_1^* und L_1^+ explizit an, aus welchen Wörtern sie besteht.

Aufgabe 26 Geben Sie einen regulären Ausdruck für die Sprache $\overline{L_{a^*}}$ über dem Alphabet $\{a, b\}$ an!

Aufgabe 27 Die Wörter

$$\varepsilon, a, ab, aba, abab, \ldots, b, ba, bab, baba, \ldots$$

sind dadurch charakterisiert, dass sie nur die Zeichen a und b enthalten, wobei sich a und b abwechseln. Die Menge

$$\{\varepsilon, a, ab, aba, abab, \ldots, b, ba, bab, baba, \ldots\}$$

dieser Wörter ist eine Sprache über dem Alphabet $\{a, b\}$. Geben Sie einen regulären Ausdruck an, der genau diese Sprache bezeichnet.

Aufgabe 28 Seien L und L' zwei reguläre Sprachen über demselben Alphabet Σ. Zeigen Sie, dass die Sprachen $L \cup L'$, LL' und L^* regulär sind!

Aufgabe 29 Sei Σ ein Alphabet und $\mathbb{L}$ eine endliche Menge von regulären Sprachen über Σ. Zeigen Sie, dass dann die Sprache $\bigcup \mathbb{L}$ ebenfalls regulär ist!

Aufgabe 30 Ist auch die Vereinigung einer unendlichen Menge von regulären Sprachen über Σ stets wieder regulär? Begründen Sie Ihre Antwort!

Aufgabe 31 Gegeben sei der deterministische endliche Automat (DEA) über dem Alphabet $\{a, b\}$ mit den Zuständen p, q und r und mit der folgendermaßen definierten Übergangsfunktion δ:

$$\begin{aligned}\delta(p, a) &= p\\ \delta(p, b) &= q\\ \delta(q, a) &= p\\ \delta(q, b) &= r\\ \delta(r, a) &= r\\ \delta(r, b) &= r\end{aligned}$$

Dabei sei p der Anfangszustand des Automaten und r der einzige Endzustand des Automaten. Die Dateien `dea_Simulation1.pl` und `dea_Simulation2.pl` enthalten zwei Varianten eines Prolog-Programms, das diesen deterministischen endlichen Automaten simuliert. Schauen Sie sich zunächst die Programme an und probieren Sie sie mit verschiedenen Eingabewörtern aus!

Nun zeichnen Sie den Graphen, der diesen endlichen Automaten darstellt!

Aufgabe 32 Welche Wörter werden von diesem endlichen Automaten akzeptiert? Geben Sie einen regulären Ausdruck an, der genau die Menge der Wörter bezeichnet, die von diesem endlichen Automaten akzeptiert werden!

Aufgabe 33 Geben Sie einen deterministischen endlichen Automaten an, der genau die Wörter der Sprache von Aufgabe 27 akzeptiert!

Aufgabe 34 Beweisen Sie, dass es keinen deterministischen endlichen Automaten gibt, der die Sprache $\{a^n b^n \mid n \in \mathbb{N}\}$ akzeptiert!

Aufgabe 35 Beweisen Sie, dass es keinen deterministischen endlichen Automaten gibt, der die Sprache $\{a^q \mid q \text{ ist eine Quadratzahl}\}$ akzeptiert!

Aufgabe 36 Im Prolog-Programm `nea_akz_Spr.pl` werden zwei Prolog-Prädikate `delta` (3-stellig) und `endzustand` (1-stellig) definiert, durch die die Übergangsrelation Δ und die Menge F der Endzustände eines nichtdeterministischen endlichen Automaten gegeben werden. Zeichnen Sie diesen Automaten! Geben Sie einen regulären Ausdruck an, der die durch diesen Automaten erzeugte Sprache bezeichnet! Lassen Sie sich vom Prolog-Programm nacheinander die vom Automaten akzeptierten Wörter ausgeben und verifizieren Sie, dass das genau die Wörter der von Ihrem regulären Ausdruck bezeichneten Sprache sind! Versuchen Sie zu verstehen, was das Prolog-Programm tut! Zeichnen Sie schließlich einen minimalen deterministischen endlichen Automaten, der diese Sprache akzeptiert!

Aufgabe 37 Sei Σ das Alphabet $\{\text{𝅘𝅥}, \text{𝅗𝅥}, \text{𝅝}, |\}$ aus der Musik. Die Noten 𝅘𝅥, 𝅗𝅥, 𝅝 haben *Notenwerte* von jeweils 1, 2 bzw. 4 Schlägen. Das Zeichen | ist der Taktstrich. Unter einem *Viervierteltakt* wollen wir ein Wort über dem Alphabet Σ verstehen, das aus Noten gefolgt von dem Taktstrich besteht, wobei die Summe der Notenwerte 4 Schläge beträgt. Unter einem *einfachen Viervierteltelrhythmus* wollen wir ein Wort über Σ verstehen, das eine Konkatenation von null oder mehr Viervierteltakten ist. Ein Beispiel für einen einfachen Viervierteltelrhythmus ist das Wort 𝅘𝅥 𝅗𝅥 𝅘𝅥 | 𝅝 | 𝅗𝅥 𝅗𝅥 | .

Zeichnen Sie einen deterministischen endlichen Automaten, der die Menge der einfachen Viervierteltelrhythmen akzeptiert! Geben Sie auch, angefangen beim Anfangszustand, nacheinander alle Zustände an, die der Automat beim Lesen des Wortes 𝅘𝅥 𝅗𝅥 𝅘𝅥 | 𝅝 | 𝅗𝅥 𝅗𝅥 | durchläuft!

Aufgabe 38 Sei $\Sigma = \{a\}$ und sei L die Sprache über Σ, die aus denjenigen Wörtern a^n besteht, für die die Zahl n nicht durch alle der Zahlen 5, 6 und 7 teilbar ist:

$$L = \{a^n \mid \neg 5|n \vee \neg 6|n \vee \neg 7|n\}.$$

Geben Sie einen nichtdeterministischen endlichen Automaten mit $5 + 6 + 7 + 1$ Zuständen an, der die Sprache L akzeptiert!

Aufgabe 39 Zeigen Sie, dass der minimale deterministische endliche Automat für die Sprache L der vorigen Aufgabe $5 \cdot 6 \cdot 7$ Zustände hat!

Aufgabe 40 Sei $\Sigma = \{a\}$. Zeigen Sie, dass es kein Polynom p gibt derart, dass zu jedem nichtdeterministischen endlichen Automaten mit n Zuständen, der eine Sprache über dem Alphabet Σ akzeptiert, ein deterministischer endlicher Automat mit höchstens $p(n)$ Zuständen existiert, der dieselbe Sprache akzeptiert!

Aufgabe 41 Durch die Regel

$$ab \to baa$$

wird ein sogenanntes *Semi-Thue-System* definiert. Die Regel bedeutet, dass man die Zeichenfolge ab durch die Zeichenfolge baa ersetzen darf. Man gibt nun eine

endliche Folge von Zeichen vor, z.B. die Zeichenfolge *abbb*. Diese Zeichenfolge wird schrittweise verändert. In jedem Schritt wird dazu ein Teilstück der Zeichenfolge gemäß der obigen Regel ersetzt. So wird zum Beispiel im ersten Schritt das Anfangsstück *ab* der Zeichenfolge *abbb* gemäß der Regel ersetzt durch *baa*, sodass man insgesamt die neue Zeichenfolge *baabb* erhält. Im zweiten Schritt wird darin wiederum das Teilstück *ab* durch *baa* ersetzt, sodass man die neue Zeichenfolge *babaab* erhält. Im dritten Schritt hat man nun die Wahl zwischen zwei Möglichkeiten. Eine Ableitung schaut z.B. so aus:

abbb
baabb
babaab
bababaa
...

Wenden Sie die Regel bitte solange an, bis sie nicht mehr anwendbar ist. Was ist das Ergebnis? Probieren Sie auch, mit der Zeichenfolge *a*, mit der Zeichenfolge *ab*, mit der Zeichenfolge *abb* oder mit der Zeichenfolge *abbbb* zu beginnen. Welche Funktion auf der Menge der natürlichen Zahlen kann man mit Hilfe dieses Semi-Thue-Systems berechnen? Semi-Thue-Systeme bilden die Grundlage für die Chomsky-Grammatiken, zu denen wir zu einem späteren Zeitpunkt in dieser Lehrveranstaltung kommen werden.

Kontextfreie Chomsky-Grammatiken In der Datei `grammar.pdf` wird ein Beispiel für eine kontextfreie Chomsky-Grammatik sowie für eine Herleitung eines terminalen Wortes (d.h. eines aus terminalen Zeichen bestehenden Wortes) in dieser Grammatik und der zugehörige Syntaxbaum gezeigt.

Aufgabe 42 Sei nun eine kontextfreie Chomsky-Grammatik folgendermaßen gegeben.

- Die *terminalen Zeichen* sind a, b, c, $+$, $-$, $*$, $/$, (und).
- Die *nichtterminalen Zeichen* sind S, T und U.
- Das *Startsymbol* ist S.
- Die *Produktionen* sind die folgenden.

$$
\begin{aligned}
&S \to T\\
&S \to S + T\\
&S \to S - T\\
&T \to U\\
&T \to T * U\\
&T \to T/U\\
&U \to a\\
&U \to b\\
&U \to c\\
&U \to (S)
\end{aligned}
$$

Leiten Sie in dieser Grammatik das terminale Wort $(a * b - c/a) * c$ her und zeichnen Sie den zu Ihrer Herleitung gehörigen Syntaxbaum!

Turing-Maschinen Eine *Turingmaschine* ist eine Maschine, die sich – ähnlich wie ein endlicher Automat – zu jedem Zeitpunkt in einem von endlich vielen Zuständen befindet. Ein Zustand S ist ausgezeichnet als der *Startzustand* und ein Zustand H als der Haltezustand.

Eine Turingmaschine hat aber zusätzlich ein nach links und rechts unbegrenztes in gleich große Felder eingeteiltes Band. Auf jedem Feld des Bandes steht ein Zeichen aus einem gegebenen Alphabet, dem *Bandalphabet*. Ein Zeichen des Bandalphabetes ist das *Leerzeichen*, das wir mit # bezeichnen. Weiters hat die Maschine einen Schreib-Lese-Kopf, der jeweils auf einem der Felder des Bandes, dem jeweiligen *Arbeitsfeld* sitzt. Der Schreib-Lese-Kopf kann die folgenden Aktionen ausführen.

(a) lesen, welches Zeichen auf dem Arbeitsfeld steht

(b) einen Elementarbefehl ausführen

Elementarbefehle sind die folgenden.

(a) Ersetzung des Zeichens auf dem aktuellen Arbeitsfeld durch ein gegebenes Zeichen a des Bandalphabets. Diesen Elementarbefehl bezeichnen wir einfach mit a.

(b) Verschieben des Schreib-Lese-Kopfes um ein Feld nach links. Diesen Elementarbefehl bezeichnen wir mit L.

(c) Verschieben des Schreib-Lese-Kopfes um ein Feld nach rechts. Diesen Elementarbefehl bezeichnen wir mit R.

Die Maschine verfügt außerdem über ein Programm, das aus Zeilen der Form

$$p \quad a \quad B \quad q$$

besteht. Dabei sind p und q zwei Zustände, a ist ein Zeichen des Bandalphabets und B ist ein Elementarbefehl. Für jeden Zustand p außer dem Haltezustand und für jedes Zeichen a des Bandalphabets gibt es genau eine solche Programmzeile.

Am Anfang einer *Berechnung* durch die Turingmaschine sind alle Felder auf dem Band außer endlich vielen Feldern mit dem Leerzeichen # beschriftet. Die Maschine befindet sich dabei im Startzustand S. Eine solche Konfiguration kann man z.B. folgendermaßen bildlich darstellen.

...	#	#	a	a	b	a	#	#	...
		$\uparrow$							
		S							

Der Pfeil symbolisiert den Schreib-Lese-Kopf. Unter dem Pfeil steht der momentane Zustand der Turingmaschine.

Ein Rechenschritt besteht darin, dass die Maschine das Zeichen auf dem aktuellen Arbeitsfeld, also auf dem Feld, auf dem der Schreib-Lese-Kopf im Moment sitzt, liest. Wenn zum momentanen Zeitpunkt der Zustand p ist und das Zeichen a auf dem Arbeitsfeld steht und eine Programmzeile

$$p \quad a \quad B \quad q$$

vorhanden ist, so führt die Maschine den Elementarbefehl B aus und geht in den neuen Zustand q über. So bedeutet zum Beispiel die Programmzeile

$$p \quad a \quad b \quad q$$

soviel wie „Wenn du im Zustand p bist und das Zeichen a auf dem Arbeitsfeld liest, dann ersetze auf dem Arbeitsfeld das Zeichen a durch das Zeichen b und gehe in den neuen Zustand q über“. Die Programmzeile

$$S \quad \# \quad R \quad p$$

bedeutet soviel wie „Wenn du im Startzustand S bist und das Leerzeichen $\#$ auf dem Arbeitsfeld liest, dann bewege den Schreib-Lese-Kopf um ein Feld nach rechts und gehe in den neuen Zustand p über“.

Sobald die Maschine sich im Haltezustand befindet, hält sie an.

Im Prolog-Programm `turingmaschine.pl` ist ein Beispiel für eine Turingmaschine gegeben. Lassen Sie das Programm laufen und schauen Sie es sich genau an!

Aufgabe 43 Geben Sie die Berechnung für die Turingmaschine mit der Zustandsmenge $\{S, p, q, H\}$, dem Startzustand S, dem Haltezustand H, dem Bandalphabet $\{a, b, \#\}$ und den Programmzeilen

$$\begin{array}{cccc} S & a & a & H \\ S & b & b & H \\ S & \# & R & p \\ p & a & b & q \\ p & b & a & q \\ p & \# & \# & H \\ q & a & R & p \\ q & b & R & p \\ q & \# & \# & H \end{array}$$

an, die mit der Konfiguration

$\ldots$	$\#$	$\#$	a	a	b	a	$\#$	$\#$	$\ldots$
		$\uparrow$							
		S							

startet. Was bewirkt diese Turingmaschine, wenn man sie mit einer beliebigen Startkonfiguration startet?

Aufgabe 44 Denken Sie sich selbst eine Turingmaschine aus, die interessante Berechnungen durchführen kann, geben Sie die Programmzeilen dieser Turingmaschine an, beschreiben Sie, was für Berechnungen diese Turingmaschine durchführen kann und geben Sie ein Beispiel für eine solche Berechnung!

Aufgabe 45 Geben Sie die Berechnung der Turingmaschine $R_{\#}$

S	a	R	q
S	b	R	q
S	c	R	q
S	$\#$	R	q
q	a	R	q
q	b	R	q
q	c	R	q
q	$\#$	$\#$	H

zur Startkonfiguration

$\ldots$	$\#$	$\#$	a	b	c	a	c	b	$\#$	a	$\#$	$\#$	$\ldots$
				$\uparrow$									
				S									

an! Was bewirkt diese Turingmaschine, wenn man sie mit einer beliebigen Startkonfiguration startet?

Aufgabe 46 Geben Sie die Berechnung der Turingmaschine

S	a	a	H
S	b	a	H
S	c	a	H
S	$\#$	a	H

zur Startkonfiguration

$\ldots$	$\#$	$\#$	a	b	c	a	c	b	$\#$	a	$\#$	$\#$	$\ldots$
				$\uparrow$									
				S									

an! Was bewirkt diese Turingmaschine, wenn man sie mit einer beliebigen Startkonfiguration startet?

Aufgabe 47 Geben Sie die Berechnung der Turingmaschine

S	a	L	H
S	b	L	H
S	c	L	H
S	$\#$	L	H

zur Startkonfiguration

$\ldots$	$\#$	$\#$	a	b	c	a	c	b	$\#$	a	$\#$	$\#$	$\ldots$
				$\uparrow$									
				S									

an! Was bewirkt diese Turingmaschine, wenn man sie mit einer beliebigen Startkonfiguration startet?

Aufgabe 48 Geben Sie die Berechnung der Turingmaschine

S	a	a	H
S	b	b	H
S	c	c	H
S	$\#$	L	q
q	a	$\#$	H
q	b	$\#$	H
q	c	$\#$	H
q	$\#$	R	H

zur Startkonfiguration

$\ldots$	$\#$	$\#$	a	b	c	a	c	b	$\#$	a	$\#$	$\#$	$\ldots$
									$\uparrow$				
									S				

an! Was bewirkt diese Turingmaschine, wenn man sie mit einer beliebigen Startkonfiguration startet?

Aufgabe 49 Betrachten wir die Kongruenzabbildungen (also Drehungen und Spiegelungen), die ein gegebenes Quadrat in der Ebene auf sich selbst abbilden. Die kann man sich veranschaulichen, indem man auf das Quadrat ein Bild zeichnet, etwa vom Buchstaben F, sodass das Quadrat dann so ausschaut: F. Mit a wollen wir eine 90°-Drehung gegen den Urzeigersinn bezeichnen. Die ergibt das Bild F. Mit b wollen wir eine Spiegelung um die senkrechte Achse bezeichnen. Die ergibt aus F das Bild F. Mit ab bezeichnen wir eine Drehung a gefolgt von einer Spiegelung b, was das Bild F ergibt. Allgemein wollen wir mit einem Wort $a_1 \ldots a_n \in \{a, b\}^*$ die Kongruenzabbildung bezeichnen, die sich durch Hintereinanderausführung der mit a_1,...,a_n bezeichneten Kongruenzabbildungen ergibt. Für die Kongruenzabbildungen gelten dann die Gleichungen $aaaa = e$, $bb = e$ und $ba = aaab$, wobei e für eine Drehung um 0° steht. In der Gruppentheorie zeigt man, dass diese Kongruenzabbildungen eine Gruppe bilden und dass e das neutrale Element dieser Gruppe ist.

Zeigen Sie, dass es genau 8 solche Kongruenzabbildungen gibt, und zwar e, a, aa, aaa, b, ab, aab, $aaab$! Zeichnen Sie dazu die entsprechenden Bilder (z.B. F für ab)! Zeigen Sie weiter, dass man in dem Semi-Thue-System mit den Regeln $aaaa \to \epsilon$, $bb \to \epsilon$ und $ba \to aaab$ aus jedem Wort $a_1 \ldots a_n \in \{a, b\}^*$ dasjenige der Wörter e, a, aa, aaa, b, ab, aab, $aaab$ ableiten kann, das die Kongruenzabbildung bezeichnet, die sich durch Hintereinanderausführung der mit a_1,...,a_n bezeichneten Kongruenzabbildungen ergibt!

Aufgabe 50 Sei $f\colon (\{a, b\}^*)^2 \to \{a, b\}^*$ die Konkatenationsfunktion auf $\{a, b\}^*$, also $f(u, v) = uv$ für alle $u, v \in \{a, b\}^*$. Geben Sie eine Turingmaschine für die Funktion f an!

Aufgabe 51 Geben Sie eine Grammatik an, die die Funktion f der vorigen Aufgabe berechnet!

Aufgabe 52 Sei $f\colon (\{a,b\}^*)^2 \to \{a,b\}^*$ die Spiegelungsfunktion auf $\{a,b\}^*$, sei also $f(u)$ das Spiegelbild des Wortes u für alle $u \in \{a,b\}^*$. Geben Sie eine Turingmaschine für die Funktion f an!

Aufgabe 53 Sei $f\colon \mathbb{N} \to \mathbb{N}$ definiert durch $f(x) = 2x$ für alle $x \in \mathbb{N}$. Geben Sie eine Turingmaschine für die Funktion f an!

Aufgabe 54 Geben Sie eine Grammatik an, die die Funktion f der vorigen Aufgabe berechnet!

Aufgabe 55 Die einstellige partielle zahlentheoretische Funktion f sei definiert durch

$$f(x) := \begin{cases} x, & \text{wenn } x \text{ gerade} \\ \text{undefiniert}, & \text{wenn } x \text{ ungerade} \end{cases}$$

für alle $x \in \mathbb{N}$. Geben Sie eine Turingmaschine für die Funktion f an!

Aufgabe 56 Die einstellige partielle zahlentheoretische Funktion f sei definiert durch

$$f(x) := \begin{cases} x/2, & \text{wenn } x \text{ gerade} \\ \text{undefiniert}, & \text{wenn } x \text{ ungerade} \end{cases}$$

für alle $x \in \mathbb{N}$. Geben Sie eine Turingmaschine für die Funktion f an!

Aufgabe 57 Ein *Palindrom* ist ein Wort, das gleich seinem eigenen Spiegelbild ist, z.B. das Wort abcba. Geben Sie eine Turingmaschine an, die für ein Wort $u \in \{a,b\}^*$, wenn man sie in der Konfiguration $u\underset{S}{\#}$ startet, genau dann nach endlich vielen Schritten hält, wenn u ein Palindrom ist!