Aufgaben Theoretische Informatik

Elmar Eder

21. Januar 2015

Lösungen der Aufgaben bitte abgeben auf dem Abgabesystem von Dominik Kaaser auf `https://ti.cosy.sbg.ac.at/` als ASCII- oder UTF-8-Dateien mit der unter Unix/Linux üblichen Codierung der Zeilenenden als LF (linefeed) oder als PDF- oder JPEG-Dateien! Dateinamen wie dort angegeben. Bearbeitung in Teams von maximal 2 Studenten gestattet. In diesem Fall bitte die Namen beider Autoren an den Anfang der Datei schreiben, und ich bitte jeden der beteiligten Autoren, dieselbe Datei unter seinem Account abzugeben! Die Aufgaben sind dann im Proseminar zu präsentieren (Kreuzerlliste zur Präsentationsbereitschaft liegt jeweils am Anfang der Doppelstunde aus).

Aufgabe 1 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 2 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 3 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!

Aufgabe 4 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 5 Geben Sie für $\neg((A \vee \neg B) \Leftrightarrow \neg(A \wedge C))$ die Wahrheitswerttafel an!

Aufgabe 6 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 7 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 8 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 9 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 1 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 1 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 10 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 11 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 12 Geben Sie die Potenzmenge der Menge $\{2, 3, 4\}$ an! Geben Sie die Potenzmenge der Menge $\{(2, 3), (4, 5)\}$ 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 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 15 Beweisen Sie durch vollständige Induktion, dass

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

für alle natürlichen Zahlen n gilt!

Aufgabe 16 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 17 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 18 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 19 Betrachten wir als Objekte (Gegenstände) die natürlichen Zahlen. Dann wird durch die folgenden Regeln ein Erzeugungssystem definiert:

$$\begin{array}{ll} \text{Axiome:} & 3 \\ & \\ & 5 \\ & \\ \text{Regel:} & \dfrac{x \quad y}{x+y}. \end{array}$$

Welche Zahlen sind in diesem Erzeugungssystem herleitbar?

Aufgabe 20 Geben Sie für jedes der folgenden Erzeugungssysteme an, welche Menge von Objekten es erzeugt!

1. Objekte sind reelle Zahlen.

$$\begin{array}{ll} \text{Axiom:} & 0 \\ \text{Regel:} & \dfrac{x}{x+1} \end{array}$$

2. Objekte sind reelle Zahlen.

$$\begin{array}{ll} \text{Axiom:} & 1 \\ \text{Regel:} & \dfrac{x \qquad y}{x-y} \end{array}$$

3. Objekte sind reelle Zahlen.

$$\begin{array}{ll} \text{Axiom:} & 1 \\ \text{Regeln:} & \dfrac{x \qquad y}{x-y} \\ & \dfrac{x \qquad y}{x/y}\ , \quad \text{wenn } y \neq 0. \end{array}$$

4. Gegeben sei ein Würfel im dreidimensionalen euklidischen Raum. Objekte sind Punkte des Raumes.

   Axiome: Die 8 Eckpunkte des Würfels

   Regel: $\dfrac{\vec{x} \qquad \vec{y}}{\alpha\vec{x} + (1-\alpha)\vec{y}}$, 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.

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

   Axiome: Alle Punkte der Geraden h

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

Aufgabe 21 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 22 Geben Sie einen regulären Ausdruck für die Sprache $\overline{L_{a^*}}$ über dem Alphabet $\{a, b\}$ an!

Aufgabe 23 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 24 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 25 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 26 Geben Sie einen deterministischen endlichen Automaten an, der genau die Wörter der Sprache von Aufgabe 23 akzeptiert!

Aufgabe 27 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 28 Die zweistellige zahlentheoretische Funktion $\mathrm{add}\colon \mathbb{N}^2 \to \mathbb{N}$ sei definiert durch die Rekursionsgleichungen

$$\begin{aligned} \mathrm{add}(x, 0) &= x \\ \mathrm{add}(x, y') &= \mathrm{add}(x, y)'. \end{aligned}$$

Berechnen Sie $\mathrm{add}(0'', 0''')$! Stellen Sie die Funktion add durch primitive Rekusion, also in der Form (Rfg) dar!

Aufgabe 29 Die zweistellige zahlentheoretische Funktion $\mathrm{mul}\colon \mathbb{N}^2 \to \mathbb{N}$ sei definiert durch die Rekursionsgleichungen

$$\begin{aligned} \mathrm{mul}(x, 0) &= 0 \\ \mathrm{mul}(x, y') &= \mathrm{add}(\mathrm{mul}(x, y), x). \end{aligned}$$

Berechnen Sie $\mathrm{mul}(0'', 0''')$! Stellen Sie die Funktion mul in der Form (Rfg) dar!

Aufgabe 30 Definieren Sie in ähnlicher Weise die Potenzierungsfunktion $\mathrm{pot}\colon \mathbb{N}^2 \to \mathbb{N}$ mit $\mathrm{pot}(x, y) = x^y$ durch Rekursionsgleichungen! Stellen Sie die Funktion pot als (Rfg) dar!

Aufgabe 31 Definieren Sie die Fakultätsfunktion durch Rekursionsgleichungen! Stellen Sie sie als (Rfg) dar!

Aufgabe 32 Geben Sie die Rekursionsgleichungen für die Vorgängerfunktion V an und berechnen Sie $V(0''')$! Stellen Sie die Funktion V als (Rfg) dar!

Aufgabe 33 Geben Sie die Rekursionsgleichungen für die modifizierte Differenz diff an und berechnen Sie $\mathrm{diff}(0'', 0''')$ und $\mathrm{diff}(0''', 0'')$!

Aufgabe 34 Eine zweistellige zahlentheoretische Funktion $h\colon \mathbb{N}^2 \to \mathbb{N}$ sei definiert durch $h = (\mathrm{add}\ \mathrm{add}\ \mathrm{mul})$. Was ist $h(3,4)$ und was ist allgemein $h(x,y)$?

Aufgabe 35 Die dreistellige zahlentheoretische Funktion $h\colon \mathbb{N}^3 \to \mathbb{N}$ sei definiert durch $h = (NP_1^3)$. Was ist $h(5,3,9)$ und was ist allgemein $h(x_1,x_2,x_3)$?

Aufgabe 36 Was ist $P_1^3(8,4,5)$? Was ist $K_2^3(8,4,5)$?

Die dreistellige zahlentheoretische Funktion $f\colon \mathbb{N}^3 \to \mathbb{N}$ sei definiert durch

$$f = (\mathrm{add}(\mathrm{add}(\mathrm{mul}(\mathrm{mul}\ K_2^3 P_1^3) P_3^3)(\mathrm{mul}\ K_3^3 P_2^3)) K_7^3).$$

Was ist $f(8,4,5)$ und was ist allgemein $f(x_1,x_2,x_3)$?

Aufgabe 37 Sei die einstellige zahlentheoretische Funktion $f\colon \mathbb{N} \to \mathbb{N}$ definiert durch $f = (ROP_1^2)$. Berechnen Sie $f(0'')$! Welche Funktion ist f? Welche Funktion ist $(N(NO))$? Welche Funktion ist $(N(Nf))$? Welche Funktion ist (fP_1^3)? Welche Funktion ist $(N(N(fP_1^3)))$? Welche Funktion ist $((N(Nf))P_1^3)$?

Aufgabe 38 Welche der folgenden Wörter sind primitivrekursive Funktionale? Geben Sie gegebenenfalls die Stelligkeit des Funktionals an und die zahlentheoretische Funktion, die durch das Funktional bezeichnet wird!

`O N (OO) (ON) (NO) (NN) (NNN) ((NN)N)` $(\mathtt{P}_2^2\mathtt{NN})$ $(\mathtt{ROP}_1^2)$

Aufgabe 39 Lesen Sie sich das Prolog-Programm `mypartrek.pl` und die Beschreibung in `mypartrek.pdf` dazu genau durch! Geben Sie nun ein μ-partiellrekursives Funktional an, das eine einstellige partielle zahlentheoretische Funktion h beschreibt, deren Definitionsbereich weder die leere Menge noch ganz $\mathbb{N}$ ist! Lassen Sie das Programm `mypartrek.pl` damit laufen sowohl für Argumente im Definitionsbereich von h als auch für Argumente außerhalb des Definitionsbereichs von h! Wie verhält sich das Programm in jedem der beiden Fälle?

Aufgabe 40 Die Additionsfunktion add ist die zweistellige primitivrekursive Funktion $(RP_1^1(NP_1^3))$. Sie ist charakterisiert durch die Rekursionsgleichungen

$$\begin{aligned} \mathrm{add}(x,0) &= x \\ \mathrm{add}(x,y') &= add(x,y)' \,. \end{aligned}$$

Man kann diese Gleichungen auch in der Sprache der Logik ausdrücken, indem man ein Funktionszeichen f für die Nachfolgerfunktion auf den natürlichen Zahlen und ein Prädikatszeichen A einführt, wobei $A(x,y,z)$ bedeuten soll, dass $\mathrm{add}(x,y) = z$ ist:

$$\begin{gathered} \forall x\, A(x,0,x) \wedge \\ \forall x \forall y \forall z \big(A(x,y,z) \Rightarrow A(x, f(y), f(z)) \big) \,. \end{gathered}$$

In Prologsyntax müssen Funktionszeichen und Prädikatszeichen klein und Variablen groß geschrieben werden. Wir schreiben also `add` für A, `f` für f, `0` für 0, `X` für x, `Y` für y und `Z` für z. Außerdem verwendet man in Prolog die umgekehrte Implikation $\Leftarrow$ (in Prolog geschrieben als `:-`) statt der Implikation $\Rightarrow$, und die Allquantorpräfixe $\forall x$, $\forall y$ und $\forall z$ werden weggelassen. Wir erhalten so das Prologprogramm `add.pl`:

```
add(X,0,X).
add(X,f(Y),f(Z)):- add(X,Y,Z).
```

Starten Sie Prolog und lassen Sie dieses Programm mit der Anfrage

```
?- add(f(f(0)),f(f(f(0))),f(f(f(f(f(0))))))).
```

laufen. Prolog sollte darauf mit `yes` (oder mit `true` oder etwas ähnlichem) antworten. Prolog hat soeben bewiesen, dass die in der Anfrage angegebene Formel logisch aus dem Programm folgt. Schreiben Sie die Formel, die Prolog bewiesen hat, hin. Man kann die Sache aber auch so sehen: Prolog hat bewiesen, dass das Programm zusammen mit der Negation der Anfrage logisch widersprüchlich ist. Schreiben Sie die Formel hin, die Prolog widerlegt, d.h. als logisch widersprüchlich (wir sagen auch *unerfüllbar*) nachgewiesen hat. Stellen Sie nun die Anfrage

```
?- add(f(f(0)),f(f(f(0))),Z).
```

Sie sehen, dass man mit unserem Prologprogramm nicht nur nachweisen kann, dass $2+3=5$ ist, sondern $2+3$ auch berechnen kann, wenn man das Ergebnis 5 nicht schon vorher kennt.

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:

$$abbb$$
$$baabb$$
$$babaab$$
$$bababaa$$
$$\dots$$

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.

Lesen Sie das Programm `semithue.pl` und lassen Sie es mit der Anfrage

```
?- maximale_ableitungen.
```

laufen!

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.

1. lesen, welches Zeichen auf dem Arbeitsfeld steht
2. einen Elementarbefehl ausführen

Elementarbefehle sind die folgenden.

1. Ersetzung des Zeichens auf dem aktuellen Arbeitsfeld durch ein gegebenes Zeichen a des Bandalphabets. Diesen Elementarbefehl bezeichnen wir einfach mit a.
2. Verschieben des Schreib-Lese-Kopfes um ein Feld nach links. Diesen Elementarbefehl bezeichnen wir mit L.
3. 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.

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

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

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!