Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich (1989) *134*/2: 139–150

Die Logik oder Die Kunst des Programmierens

Ernst Specker, ETH Zürich

Die Grundgedanken der Logikprogrammierung werden am Beispiel eines Geduldspiels (Türme von Hanoi) erläutert. Die möglichen Stellungen dieses Spiels bilden einen Graphen, wenn zwei Stellungen durch eine Kante verbunden werden, falls sie durch einen legalen Zug auseinander hervorgehen. Dieser Graph wird auf zwei Arten realisiert: zum einen in unserer materiellen Welt mit Hilfe einer Vervielfältigungsmaschine, zum anderen in der idealen Welt von Herbrand mit Hilfe der Logikprogrammierung.

Logic or the Art of Programming

The fundamental ideas of logic programming are explained by considering a puzzle (the Towers of Hanoi). The possible positions of the puzzle form a graph, two positions being joined by an edge if they can be transformed into one another by a legal move. Two realizations of this graph are introduced: one in our material world with the help of a copying machine, the other in the ideal world of Herbrand with the help of logic programming.

1 Einleitung

Bis ins 19. Jahrhundert wurde an den Universitäten Logik gelehrt, um angehende Geistliche, Juristen, Ärzte mit der Kunst des richtigen Denkens vertraut zu machen. Eines der klassischen Werke für diesen Unterricht war «La logique ou l'art de penser» von A. Arnauld und P. Nicole, in erster Auflage 1662 erschienen. Inzwischen ist das Bemühen, mit Logikunterricht zu richtigem Denken anleiten zu wollen, aufgegeben worden, vermutlich weniger aufgrund des Wandels der klassischen Logik in eine mathematische Disziplin als aus der Überzeugung, dass richtiges Denken in einem Fachgebiet nur in ständiger Auseinandersetzung mit dem Fache selbst entwickelt werden kann und sich nicht einfach aufgrund allgemeiner Prinzipien von aussen herantragen lässt.

Die Logiker haben damit einen grossen Teil ihrer Studenten verloren. Um so erfreulicher ist es für sie, dass es gelungen ist, ein neues Zielpublikum zu finden. Es hat sich nämlich gezeigt, dass gewisse logische Sprachen ausgezeichnete Programmiersprachen sind und dass das, was in der Logik unter dem Begriff «Beweis» untersucht wird, auch als «Berechnung» oder «Datenverarbeitung» aufgefasst werden kann. Die bekannteste dieser neuen Programmiersprachen ist PROLOG (PROgrammierung in LOGic).

Die einfachsten Grundgedanken der Logikprogrammierung sollen im folgenden an einem konkreten Beispiel (den Türmen von Hanoi) erläutert werden. Die Türme von Hanoi sind von E. Lucas als Geduldspiel beschrieben worden. (Die Erfindung schreibt Lucas dem Mandarin N. Claus am siamesischen Kollegium Li-Sou-Stian zu. An welchem französischen Kollegium hat wohl Lucas unterrichtet?) Dabei ist unser Ziel nicht, die durch die Türme von

Hanoi gestellte Aufgabe zu lösen, sondern es soll vielmehr gezeigt werden, wie die durch sie definierte Struktur sich in einem «Herbrand-Universum» mit Logikprogrammierung definieren lässt. Dazu gehen wir in zwei Schritten vor. In einem ersten Schritt ordnen wir der Struktur einen Graphen zu und überlegen uns, wie sich dieser Graph mit Hilfe von Kopiergeräten und dem Zeichnen von neuen Kanten in unserer materiellen Welt realisieren lässt. Im zweiten Schritt werden diese Konstruktionen in der idealen Welt eines Herbrand-Universums (nach Jacques Herbrand, 1908–1931) mit logischen Mitteln nachvollzogen.

2 Die Türme von Hanoi als Graph

Als materielles Objekt bestehen die Türme von Hanoi aus einer Platte, in die drei Pflöcke eingelassen sind, und aus einigen (z. B. vier) in der Mitte gelochten Scheiben unterschiedlicher Grösse. In der Ausgangslage befinden sich alle Scheiben pyramidenförmig geordnet auf einem Pflock. Die Aufgabe besteht dann darin, diese Pyramide durch eine Folge von «legalen Zügen» auf einen anderen Pflock zu transferieren. Ein legaler Zug besteht dabei in folgendem: Von einem Pflock wird die oberste Scheibe abgehoben und (zuoberst) auf einen anderen Pflock gelegt; dabei darf sie nur auf die Platte oder auf eine grössere Scheibe zu liegen kommen (Bild 1).

Wir bezeichnen die Pflöcke mit a, b, c, die Scheiben der Grösse nach mit 1, 2, 3, 4 (die grösste).

Befinden sich etwa auf Pflock a die Scheiben 1, 2 (1 auf 2 liegend), auf b die Scheibe 4 und auf c die Scheibe 3, so gibt es drei mögliche legale Züge:

(I) Scheibe 1 von a nach b (sie kommt auf 4 zu liegen)
(II) Scheibe 1 von a nach c (sie kommt auf 3 zu liegen)
(III) Scheibe 3 von c nach b (sie kommt auf 4 zu liegen).

Andere Züge sind offenbar nicht zulässig; Scheibe 3 darf nicht auf Pflock a transferiert werden und Scheibe 4 weder auf a noch auf c. Man überlegt sich leicht, dass in jeder Situation, die aus der Ausgangsstellung durch eine Folge von legalen Zügen hervorgeht, die Scheiben auf jedem Pflock der Grösse nach geordnet sind; da nämlich bei jedem Zug eine Scheibe nur auf eine grössere oder die Platte gelegt wird, bleibt diese Eigenschaft erhalten. Solche Verteilungen der Scheiben auf die Pflöcke sollen «Stellungen» heissen. Durch einen legalen Zug geht eine Stellung in eine andere über, indem eine Scheibe, die auf einer grösseren (oder der Platte) liegt, auf eine grössere (oder die Platte) transferiert wird. Hieraus folgt, dass auch das Zurücknehmen eines legalen Zuges ein legaler Zug ist.

Wie viele Züge sind in einer Stellung möglich? Befinden sich alle Scheiben auf einem Pflock, so sind es deren zwei: die Scheibe 1 (in Spitzenposition) darf auf einen der beiden anderen Pflöcke transferiert werden. In jedem ande-

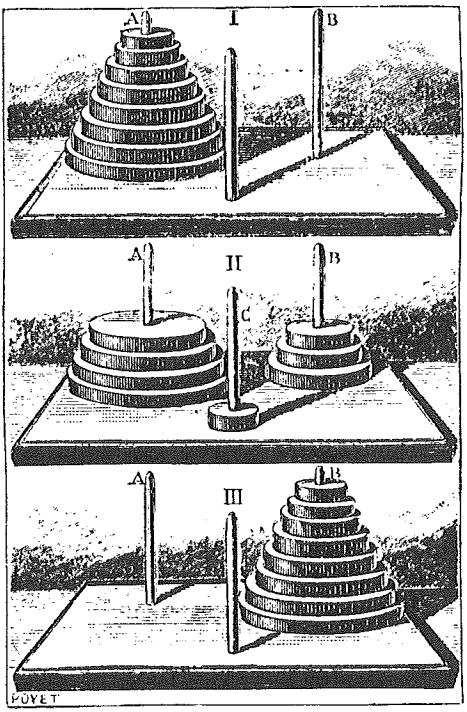

Bild 1 Türme von Hanoi nach Lucas
Fig. 1 Towers of Hanoi according to Lucas

ren Fall gibt es 3 legale Züge: Ausser der Scheibe 1 (wofür 2 Möglichkeiten bestehen) darf die zweitkleinste Scheibe in Spitzenposition auf einen anderen Pflock (wo sich nicht Scheibe 1 befindet) transferiert werden.

Wie viele Stellungen gibt es im ganzen, etwa im Fall von 4 Scheiben? Um diese Anzahl zu bestimmen, stellen wir uns vor, dass eine Stellung folgendermassen erzeugt wird (dies sind keine legalen Züge!):

Mit der grössten Scheibe beginnend, werden die 4 Scheiben der Reihe nach auf die (zunächst unbelegten) Pflöcke gelegt; jedesmal haben wir 3 Möglichkeiten der Wahl, im ganzen also $3 \cdot 3 \cdot 3 \cdot 3 = 81$ Möglichkeiten. Da auf diese Art offenbar jede Stellung genau einmal erzeugt wird, gibt es bei 4 Scheiben 81 Stellungen. (Und bei n Scheiben entsprechend deren 3^n.)

Gewisse Paare dieser 81 Stellungen besitzen die Eigenschaft, dass die eine aus der anderen durch einen legalen Zug hervorgeht. Denken wir uns die 81 Stellungen durch Punkte markiert und zwei Punkte durch eine Kante verbunden, falls sie einem solchen Paar von Stellungen zugeordnet sind, so erhalten wir einen Graphen. Wie sieht dieser Graph – er heisse G_4 – aus?

Wir fragen dazu zuerst nach den Graphen G_1 (Stellungen bei einer Scheibe) und G_2 (2 Scheiben). Bei einer Scheibe gibt es entsprechend den 3 Pflökken drei Stellungen, und jede geht durch einen legalen Zug in jede andere über: G_1 ist ein Dreieck.

Um G_2 zu erhalten, zerlegen wir die Menge S_2 der 9 Stellungen von zwei Scheiben entsprechend der Lage der grösseren Scheiben in 3 Teilmengen S_2^a, S_2^b, S_2^c. S_2^b etwa ist die Menge der Stellungen, bei denen die grössere Scheibe auf Pflock b liegt. Je zwei Stellungen von S_2^b sind durch eine Kante verbunden – wir dürfen uns die grössere Scheibe mit der Platte verschmolzen denken und befinden uns dann im Fall von G_1. Entsprechendes gilt von S_2^a, S_2^c. Wir können uns also den Graphen G_2 aufgebaut denken aus 3 Dreiecken, entsprechend der Lage der grösseren Scheibe.

Wie steht es mit Kanten einer Stellung aus S_2^a nach aussen, d. h. nach Stellungen in S_2^b, S_2^c?

Für die Stellung, bei der sich beide Scheiben auf Pflock a befinden, gibt es keine Verbindung nach aussen: Scheibe 2 kann nicht bewegt werden. Liegt aber etwa Scheibe 1 auf Pflock b, Scheibe 2 auf a, so kann Scheibe 2 auf Pflock c transferiert werden: Diese Stellung in S_2^a ist mit einer Stellung in S_2^c verbunden. Ganz entsprechend ist die dritte Stellung in S_2^a mit einer Stellung in S_2^b verbunden. Damit ist der Graph G_2 eindeutig bestimmt: Er besteht aus 3 Dreiecken G_2^a, G_2^b, G_2^c, die noch reihum durch je eine Kante verbunden sind (Bild 2). Auf ganz analoge Weise kann nun G_3 aufgrund von G_2, G_4 aufgrund von G_3 gebildet werden; man hat nur zu beachten, dass an die Stelle der Dreiecke G_2^a über S_2^a etc. Graphen des vorangehenden Typus treten.

Wir beschreiben den Übergang von G_3 nach G_4. Es sei G_4^a der Graph, welcher 4 Scheiben entspricht, bei dem Scheibe 4 sich auf Pflock a befindet. Scheibe 4 spielt dann für die Legalität von Zügen keine Rolle, d. h. G_4^a ist isomorph zu G_3 (derselbe Graph, nur über einer anderen Menge).

Entsprechendes gilt für G_4^b, G_4^c. G_4 ist also aufgebaut aus 3 Kopien von G_3. Es bleibt zu diskutieren, welche Kanten von G_4^a nach aussen führen.

In G_3 (und entsprechend in G_4^a) gibt es 3 Punkte, die nur mit zwei anderen durch eine Kante verbunden sind. Diese Punkte entsprechen den Stellungen, wo sich alle 3 kleineren Scheiben auf ein und demselben Pflock befinden; befindet sich auch die grösste Scheibe 4 auf diesem Pflock, so gibt es keine Kan-

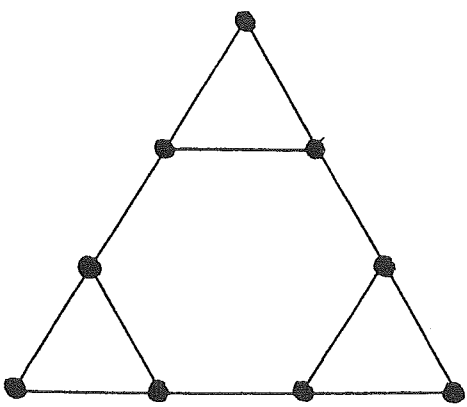

Bild 2: Der Graph, welcher Türmen mit 2 Scheiben entspricht.

Fig. 2: Graph associated to towers with 2 disks.

te nach aussen. Befinden sich aber alle kleineren Scheiben auf b, die grösste auf a, so kann diese nach c transferiert werden. Es gibt somit genau eine Kante von G_4^a nach G_4^c. Damit ist G_4 bestimmt. Wir wollen nun den Übergang noch etwas abstrakter (und damit einfacher) beschreiben. Es sei H ein Graph mit der Eigenschaft, dass alle Punkte bis auf drei mit genau drei anderen durch eine Kante verbunden sind. Die drei Ausnahmepunkte seien p, q, r; sie seien mit genau zwei anderen durch eine Kante verbunden.

Es werden nun drei isomorphe Kopien H^a, H^b, H^c von H erstellt. Der neue Graph G bestehe aus der Vereinigung der Graphen H^a, H^b, H^c, worin noch drei neue Kanten eingetragen sind. In der Vereinigung gibt es zunächst offenbar neun Ausnahmepunkte: Aus p wird p^a, p^b, p^c, aus q wird q^a, q^b, q^c, und aus r wird r^a, r^b, r^c.

Die drei neuen Kanten verbinden je

$$p^b \text{ mit } p^c,$$
$$q^a \text{ mit } q^c,$$
$$r^a \text{ mit } r^b.$$

Damit gehen auch von p^b, p^c, q^a, q^c, r^a und r^b drei Kanten aus, und neue Ausnahmepunkte sind p^a, q^b, r^c. Aufgrund dieser Konstruktion ist es nun sehr einfach, mit Hilfe eines Kopiergerätes den Graphen zu erzeugen.

Es liege eine Darstellung von H vor, und zwar seien die Ausnahmepunkte die Ecken eines gleichseitigen Dreiecks, und der Graph befinde sich innerhalb dieses Dreiecks. Es werden dann 3 Kopien von H erstellt, diese 3 Kopien entsprechend der Figur 1 auf ein Blatt geklebt und die 3 neuen Kanten eingetragen. Auch der neue Graph liegt dann auf einem gleichseitigen Dreieck, und das Verfahren kann iteriert werden. Figur 3 zeigt den so erzeugten Graphen G_4 entsprechend dem Fall von 4 Scheiben.

Durch die so beschriebene Konstruktion geht offenbar ein zusammenhängender Graph in einen ebensolchen über: Je zwei Stellungen der Hanoi-Türme können somit durch eine Folge von legalen Zügen ineinander übergeführt

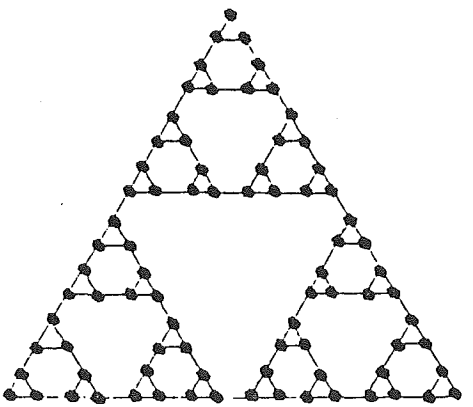

Bild 3: Der Graph, welcher Türmen mit 4 Scheiben entspricht.

Fig. 3: Graph associated to towers with 4 disks.

werden, insbesondere also eine Pyramide von einem Pflock auf einen anderen. Ebenso können viele andere Fragen leicht beantwortet werden, z. B. die Frage nach der Anzahl der benötigten Züge, nach der Eindeutigkeit des Weges bei minimaler Zugzahl, etc.

3 Die Türme von Hanoi im Herbrand-Universum

Für die Definition der Türme mit Hilfe der Logikprogrammierung verwenden wir «Listen». Alle diese Listen werden aus den Konstanten a, b, c, s aufgebaut sein. Dass wir sie Konstanten nennen, bedeutet, dass sie wie Eigennamen zu verstehen sind, allerdings nicht wie Namen von Personen der realen Welt, sondern eher wie Namen von Märchengestalten – Rumpelstilzchen existiert in seiner Geschichte und sonst nirgendwo.

Wir geben einige Beispiele von Listen: [a, b, c, a, s]. Diese Liste besteht aus vier Elementen; das Element a ist der Kopf der Liste, die Liste [b, c, a, s] der Rest. Um anzudeuten, dass [a, b, c, a, s] so aus Kopf und Rest gebildet ist, sagen wir auch

$$[a, b, c, a, s] \text{ ist } [a \mid [b, c, a, s]].$$

Listen können demnach aufgefasst werden als Terme, gebildet mit dem zweistelligen Funktionszeichen

$$[X \mid Y];$$

als Argument an zweiter Stelle ist nur eine Liste zugelassen.

Um mit dieser Bildung alle Listen zu erhalten, brauchen wir als Ausgangsliste die leere Liste [ ].

Die Liste [a] ist dann die Liste

$$[a \mid [\,]].$$

Die Liste [[a, b] | [a, a, b]]
besteht aus 4 Elementen; das erste Element ist selbst eine Liste, sie ist also die Liste

$$[[a, b], a, a, b].$$

Als Universum (sogenanntes Herbrand-Universum), worin wir unsere Strukturen definieren, wählen wir die Konstanten a, b, c, s und alle Listen, die – in iterierten Anwendungen der Listenbildung – aus ihnen gebildet werden können. Für unsere Zwecke der Definition der Hanoi-Türme müssen wir allerdings nur Listen von Listen von Listen heranziehen.

Ein Dreieck über a, b, c können wir uns durch folgende Liste von Listen definiert denken:

$$[[a, b], [b, c], [c, a]].$$

Die Liste [a, b] steht dabei für die Kante von a nach b. Bei Listen kommt es an und für sich auf die Reihenfolge an; so ist [a, b] nicht dieselbe Liste wie [b, a]. Trotzdem können wir verabreden, dass das Vorkommen von [a, b] die Kante von a nach b anzeigen soll. Es ist ja auch die Reihenfolge der Kanten in der Liste gleichgültig. Es stellt somit

[[c, a], [b, c], [b, a]]

ebenfalls das Dreieck über a, b, c dar.

Ein wesentlicher Schritt bei der Konstruktion der Hanoi-Graphen grösserer Scheibenzahl ist der Übergang von einem Graphen zu drei isomorphen und disjunkten Kopien.

Ein solcher Übergang ist am leichtesten zu bewerkstelligen, wenn Punkte des Graphen selbst schon Listen sind.

Ist nämlich [s] ein solcher Punkt, so können wir [s] verdreifachen, indem wir a, b, c davorsetzen und zu [a, s], [b, s], [c, s] übergehen. Aus diesem Grund wollen wir den Hanoi-Graphen mit einer Scheibe als Dreieck über [a], [b], [c] darstellen. Es ist dann der Graph

[[[a], [b]], [[b], [c]], [[c], [a]]].

Wir halten das fest durch das «Axiom»

(1) hanoi ([s], [[[a], [b]], [[b], [c]], [[c], [a]]]).

Damit ist eine Beziehung zwischen [s] – eine Scheibe – und dem entsprechenden Graphen stipuliert.

Wenn nun der Graph

[[[a], [b]], [[b], [c]], [[c], [a]]]

verdreifacht werden soll mit a, b, c, so können wir dies zunächst für eine Variable X (wofür später a, b, c zu setzen sein wird) andeuten.

Das isomorphe X-Bild soll sein

[[[X, a], [X, b]], [[X, b], [X, c]], [[X, c] [X, a]]].

(Allgemein verwenden wir die Konvention: Grossbuchstaben für Variable mit möglichen Werten im Herbrand-Universum, Kleinbuchstaben für Konstante.)

Für die Definition dieser Bildung im allgemeinen Fall betrachten wir zunächst die X-Markierung einer Liste von Listen.

Eine Liste von Listen wird dabei mit X markiert, indem bei jeder Liste aus L das Element X an erste Stelle gesetzt wird.

Die s-Markierung von

[[a, b], [c, b, a], [b], [ ]]

ist also [[s, a, b] [s, c, b, a], [s, b], [s]].

Wir führen dafür ein dreistelliges Prädikat mark(, ,) ein: mark(X, L, M) soll gelten, falls M die X-markierte Liste L ist.

Offenbar ist die X-markierte leere Liste die leere Liste, also

(2) $$\text{mark}(X,[\,],[\,]).$$

Für Menschen könnten wir weiter definieren

$$\text{mark}(X,[Y_1,\ldots,Y_m],[[X\mid Y_1],\ldots,[X\mid Y_m]]).$$

Für automatisches Bearbeiten ist dies kaum hinreichend. Wir legen daher fest, wie das Markieren einer Liste zurückzuführen ist auf das Markieren einer kürzeren Liste. Die zu markierende Liste sei $[Y\mid L]$, also erstens Liste Y, gefolgt von einer Liste L von Listen.

Die markierte Liste wird somit mit $[X\mid Y]$ beginnen, und es wird eine Liste M folgen. Was ist M? M ist offenbar die mit X markierte Liste L. Wir halten dies fest durch

(3) $$\text{mark}(X,[Y\mid L],[[X\mid Y]\mid M]) \text{ :- } \text{mark}(X,L,M).$$

Die Zeilen (2) und (3) bilden ein Prolog-Programm, mit dessen Hilfe Listen markiert werden. Das Zeichen «:–» steht dabei für «falls»; in der Logik ist es üblich, dies mit einer Implikation auszudrücken:

$$\text{mark}(X,L,M) \rightarrow \text{mark}(X,[Y\mid L],[[X\mid Y]\mid M]).$$

Die Variablen «laufen» dabei über alle Terme des Herbrand-Universums.

Wie kann nun ein solches Programm, enthaltend die beiden Zeilen (2), (3), benützt werden?

Wir können Fragen an das Programm stellen, etwa in der Form

$$?-\ \text{mark}(a,[[b],[c,a]],U).$$

Dies ist zu interpretieren als Aufforderung, einen Term U (d. h. eine Liste) zu finden, für welchen die Gültigkeit aus (2) und (3) folgt.

Die Frage, die am leichtesten zu beantworten ist, ist dabei

$$?-\ \text{mark}(a,[\,],U).$$

Aus (2) folgt unmittelbar die Antwort

$$U = [\,].$$

Wie steht es mit schwierigeren Fragen?

Wir wollen ausführlich den folgenden Fall betrachten:

$$?-\ \text{mark}(a,[[b],[c,a]],U).$$

Die Ausführungen sind vielleicht nicht ganz leicht zu verfolgen; so tröstet sich vielleicht mancher Leser damit, dass er die Antwort

$$U = [[a,b],[a,c,a]]$$

zu geben vermag.

Wie ist es denkbar, dass aus (2), (3) sich eine Antwort ergibt? (2) hilft offenbar zunächst nicht.

Falls wir dagegen

mark(a,[[b],[c,a]],U)

als die linke Seite von (3) auffassen können, so haben wir einen ersten Schritt getan. Es sei also

mark(a,[[b],[c,a]],U) identisch
mark(X,[Y|L],[[X|Y]|M]).

Dafür muss gelten

X identisch a;
[Y|L] identisch [[b],[c,a]]
[[X|Y]|M] identisch U.

Die Identität von [Y|L] mit [[b],[c,a]] bedeutet

Y identisch [b],
L identisch [[c,a]].

Um weiterzukommen, müssen wir die rechte Seite von (3) zur Verfügung haben, d. h.

mark(X,L,M).

Aufgrund der obigen Identitäten bedeutet dies

mark(a,[[c,a]],M).

Wir stellen also an das Programm die Frage (genauer: das Programm soll sich selber die Frage stellen!)

?– mark(a,[[c,a]],M).

Zur Beantwortung dieser Frage gehen wir genau gleich vor; (2) hilft nicht, und wir setzen eine Identifikation mit der linken Seite von (3) an.

Dabei muss verhindert werden, dass eine Variablenkollision auftritt. In jeder Verwendung von (3) sollen die Variablen neu bezeichnet werden.

mark(X1,[Y1|L1],[[X1|Y1]|M1]) :– mark(X1,L1,M1).

Durch Identifikation der linken Seite mit mark(a,[[c,a]],M) erhalten wir

X1 identisch a;
[Y1|L1] identisch [[c,a]]
[[X1|Y1]|M1] identisch M.

Ferner brauchen wir mark(X1,L1,M1). Aus den Identifikationen folgt:

Y1 ist identisch [c,a], L1 ist die leere Liste,
d. h. mark(X1,L1,M1) ist mark(a,[ ],M1).

Auf die Frage

?– mark(a,[ ],M1)

erhalten wir unmittelbar aus (2) die Antwort

M1 = [ ].

[[X1 | Y1] | M1] ist also

[[a | [c, a]] | [ ]], was nichts anderes als [[a, c, a]] ist.

Damit haben wir gefunden

mark(a,[[c, a]],[[a, c, a]]),

und wir können einen weiteren Schritt zurückgehen und finden die Antwort.

U = [[a, b],[a, c, a]].

Der Lösungsweg wird somit aufgrund einer Rückwärtsanalyse gefunden. Das Axiom (3) ist von der Art, dass ihm entnommen werden kann, aufgrund welcher Kenntnis eine Antwort möglich ist. Diese Kenntnis wird als neue Frage formuliert und das Verfahren iteriert.

Natürlich können nicht alle Probleme nach einem solchen einfachen Schema angegangen werden. Es ist aber überraschend, wie vielfältig die Anwendungsmöglichkeiten sind.

Das Schema ist übrigens verallgemeinerungsfähig: Es funktioniert auch in einem Fall wie

?– A :– K1, K2.

Für die Antwort A brauchen wir K1 und K2.

Unsere nächste Definition ist von dieser Art:

Aufgrund des Prädikates mark ist es nun leicht, isomorphe Kopien von Graphen zu definieren.

Graphen sind Listen von Listen von Listen; sie werden in isomorphe Kopien transformiert, indem den innersten Listen ein neuer Kopf vorangesetzt wird; dies ist gleichbedeutend mit der Markierung der zweitinnersten Listen.

isomorph(X, L, M) bedeute:

Durch Markierung der Elemente von L entsteht aus L die Liste M. Somit

(4) isomorph(X,[ ],[ ]).

(5) isomorph(X,[Y | L],[Z | M]) :– mark(X, Y, Z), isomorph(X, L, M).

Das Komma zwischen mark(X, Y, Z) und isomorph(X, L, M) bedeutet die Konjunktion.

Das X-Bild von [Y | L] wird erhalten, indem die erste Liste Y mit X markiert wird (was Z ergibt) und der Rest in das X-Bild transformiert wird.

Mit Hilfe von «isomorph» können von einem Graphen H drei isomorphe Kopien definiert werden:

isomorph(a, H, Ha),
isomorph(b, H, Hb) und isomorph(c, H, Hc).

(Verbindungen wie Ha gelten als eine Variable; die Notation dient nur der leichteren Lesbarkeit.)

Als nächstes soll nun das Aufkleben der drei Kopien auf einem Blatt im Herbrand-Universum nachgebildet werden. Diese Operation entspricht dem Verketten von drei Listen zu einer Liste. Es werde zunächst das Verketten von zwei Listen definiert:

verkettung(L, M, N)

gelte, wenn N die Liste ist, in der zuerst die Elemente von L, dann jene von M aufgeführt sind.

(6) verkettung([ ], L, L).
(Ist die erste Liste leer, so ist das Resultat die zweite.)

(7) verkettung([X | L], M, [X | N]) :– verkettung(L, M, N).
(Wird [X | L] mit M verkettet, so ist X das erste Glied der neuen Folge; der Rest ist die Verkettung von L und M.)

Für spätere Zwecke definieren wir die Verkettung von 4 Listen A, B, C, D zu einer neuen Liste L; dazu wird A, B zu H und C, D zu I verkettet; L ist dann die Verkettung von H und I. Somit

(8) verkettung(A, B, C, D, L) :– verkettung(A, B, H),
verkettung(C, D, I),
verkettung(H, I, L).

Als nächstes sind nun noch die drei Kanten zu definieren, welche die Kopien verbinden.

Wir nehmen an, dass wir den Fall betrachten, bei dem aus dem Graphen von 3 Scheiben jener von 4 konstruiert wird. Ferner seien die 3 Punkte in H, welche nur zu zwei Kanten gehören, die Tripel [a, a, a], [b, b, b], [c, c, c].

Durch Markierung werden daraus:

[a, a, a, a], [a, b, b, b], [a, c, c, c] (in Ha),
[b, a, a, a], [b, b, b, b], [b, c, c, c] (in Hb),
[c, a, a, a], [c, b, b, b], [c, c, c, c] (in Hc).

Wir definieren nun die Kanten

[b, a, a, a] nach [c, a, a, a],
[a, b, b, b] nach [c, b, b, b],
[a, c, c, c] nach [b, c, c, c].

Von den 9 obigen Punkten erhalten nur [a, a, a, a], [b, b, b, b] und [c, c, c, c] keine neuen Kanten.

Auch beim neuen Graphen gehören genau jene Punkte nur zu zwei Kanten, die den Listen mit nur einem Element entsprechen.

Die Scheibenzahl wollen wir im Herbrand-Universum durch eine Liste aus s darstellen; [s, s, s] soll also drei Scheiben bedeuten, [s, s, s, s] deren vier.

Wir definieren ein Prädikat konstant (, ,), das uns gestattet, von [s, s, s] überzugehen zu [a, a, a] etc.

(9) konstant(X, [ ], [ ]).

(10) konstant(X, [Y | L], [X | M]) :- konstant(X, L, M).

(konstant(X, L, M) gilt, falls die Listen L, M gleich lang sind und M nur aus dem Element X besteht.)

Damit können wir nun die Listen der 3 neuen Kanten definieren, die beim Übergang des Graphs, der zur s-Liste S gehört, neu einzuführen sind:

(11) neu(S, K) :- konstant(a, S, A),
konstant(b, S, B),
konstant(c, S, C),
K = [[[b | A], [c | A]], [[a | B], [c | B]], [[a | C], [b | C]]].

Ist S etwa die Liste [s, s, s], so ist A die Liste [a, a, a,], und es tritt als neue Kante [[b, a, a, a], [c, a, a, a]] auf, was dasselbe ist wie [[b | A], [c | A]].

Damit können wir die Hanoi-Graphen allgemein definieren:

(12) hanoi([s | S], G) :- hanoi(S, H),
isomorph(a, H, Ha),
isomorph(b, H, Hb),
isomorph(c, H, Hc),
neu(S, K),
verkettung(Ha, Hb, Hc, K, G).

Das kann so gelesen werden: G ist der Graph zu [s | S], wenn folgendes gilt: H ist der Graph zu S, Ha, Hb, Hc sind die 3 entsprechenden Kopien; K ist die Liste der 3 neuen Kanten; G ist die Verkettung von Ha, Hb, Hc, K.

Die Frage

?- hanoi([s, s, s, s], U)

wird beantwortet mit der Liste U der 120 Kanten des Graphen G_4.

4 Literatur

Arnauld A., Nicole P. (1622/1981), La logique ou l'art de penser, ed. crit.
Lloyd J. W. (1987), Foundations of Logic Programming, 2nd ed., Springer-Verlag, Berlin.
Lucas E. (1883), Récréations mathématiques, vol. 3, Gauthier-Villars, Paris.
Sterling L., Shapiro E. (1986), The Art of Prolog, MIT Presss, Cambridge Mass.

Prof. Dr. Ernst Specker, Steinbrüchelstrasse 45, 8053 Zürich.