BACK

Logische Funktionen mit DERIVE

von Karl Josef Fuchs, BG/BRG Hallein, Salzburg

I. Vorgeschichte und didaktische Zielsetzungen

Bereits während meines Studiums faszinierte mich das Studium der formalen Logik (Aussagenlogik, Prädikatenlogik, Modallogik) und Wissenschaftstheorie als Grundlagen- bzw. Metawissenschaften der unterschiedlichsten wissenschaftlichen Disziplinen. Umso mehr war ich über die Aufnahme formallogischer Bildungsinhalte in die Lehrpläne für Mathematik und Informatik an Allgemeinbildenden höheren Schulen begeistert, war damit doch eine bessere Einführung in den Gebrauch mathematischer Symbole bereits zu Beginn der Oberstufe möglich.
Ein weiterer zentraler Begriff des Lehrplanes für Mathematik in der Oberstufe ist der Funktionsbegriff. Das Definieren logischer Funktionen und das Untersuchen der Verkettungen dieser Funktionen schienen mir geeignete Unterrichtsthemen, um funktionale Zusammenhänge bereits früh zu diskutieren.
Zunehmend wurde in den didaktischen Grundsätzen der neuen Lehrpläne der Einsatz des Computers als Werkzeug im Mathematikunterricht gefordert. Zur Kodierung der logischen Funktionen boten sich die funktionalen Programmiersprachen LISP bzw. LOGO an. Mit geringem Aufwand ließen sich die logischen Funktionen für ein logisches Minimalsystem wie folgt definieren:

KONJUNKTION

```
(defun konj (a b)
 (((and (= a 1) (= b 1)) 1) 0)
)
```

```
TO KONJ :A :B
IF AND :A=1 :B=1 [OUTPUT 1][OUPUT 0]
END
```

DISJUNKTION

```
(defun disj (a b)
 (((and (= a 0) (= b 0)) 0) 1)
)

TO DISJ :A :B
IF AND :A=0 :B=0 [OUTPUT 0][OUTPUT 1]
END
```

NEGATION

```
(defun neg (a)
 (((= a 1) 0) 1)
)

TO NEG :A
IF :A=1 [OUTPUT 0][OUTPUT 1]
END
```

-> Hinweis: Man beachte, daß der Schüler bei der Eingabe der Werte für a bzw. b auf den Definitonsbereich {0,1} achten mußte, eine Überprüfung der Eingabewerte auf Zulässigkeit war in den Funktionsmodulen nicht vorgesehen. Die zusätzliche Aufnahme von Plausibilitätsabfragen in die Kodierungen der Funktionsmodule hätte wohl auch die Kernstruktur der logischen Funktionen gestört.

Mit DERIVE bot sich nun ein Softwarewerkzeug an, mit dem sich die Definitionen der logischen Funktionen, aber besonders das Studium der Verkettungen dieser Funktionen mit Hilfe von Wahrheitstabellen, noch besser realisieren ließen.

II. Lehrplanbezug

MATHEMATIK (5. KLASSE)

Arbeiten mit logischen Begriffen. Beweisen von Gesetzen der Aussagenlogik mit Wahrheitstafeln. Reflektieren über logische Begriffe und logische Beziehungen. Untersuchen von funktionalen Zusammenhängen.

INFORMATIK (7. KLASSE)

Einführung in die Schaltalgebra.

III. Schülerpopulation

REALISIERUNG MIT HILFE FUNKTIONALER PROGRAMMIERSPRACHEN

SCHULJAHR 1990/91

Mit LOGO: 5. Klasse (9. Schulstufe), Realgymnasium Hallein, gemischte Klasse.
Mit LISP: 6. Klasse (10. Schulstufe), Realgymnasium Hallein, gemischte Klasse.

SCHULJAHR 1991/92

Mit LOGO: 5. Klasse (9. Schulstufe), Gymnasium Hallein, gemischte Klasse.
7. Klasse (11. Schulstufe), Gymnasium Salzburg Nonntal und Akademisches Gymnasium Salzburg, gemischte Klassen.

REALISIERUNG MIT HILFE VON DERIVE

SCHULJAHR 1991/92

5. Klasse (9. Schulstufe), Gymnasium Hallein, gemischte Klasse.
7. Klasse (11. Schulstufe), Akademisches Gymnasium Salzburg, gemischte Klasse.

IV. Beschreibung des Lehrganges mit DERIVE

GLIEDERUNG NACH UNTERRICHTSEINHEITEN

Unterrichtseinheit 1: Definition der logischen Verknüpfungen KONJUNKTION, DISJUNKTION und NEGATION.
Unterrichtseinheit 2: Überprüfen der Belegungen mittels Wahrheitstabellen.

Unterrichtseinheit 3: Verketten von Funktionen (IMPLIKATION, ÄQUIVALENZ), Äquivalenzsätze und "Kürzungsregeln".

FÄHIGKEITEN IM UMGANG MIT DERIVE

WIEDERHOLUNG DER GRUNDBEFEHLE

Übertragen von Ausdrücken auf das Arbeitsblatt: <A>uthor
Vereinfachen von Ausdrücken: <S>implify
Substituieren von Ausdrücken: <M>anage <S>ubstitute

DEFINIEREN VON FUNKTIONEN

BENÜTZEN VORDEFINIERTER MODULE

Extrahieren mit ELEMENT.
Zusammenfügen mit VECTOR.
Bedingte Verzweigung mit IF.

UNTERRICHTSEINHEIT 1: DEFINITION DER LOGISCHEN VERKNÜPFUNGEN

Für die Kodierung der logischen Funktionen KONJUNKTION, DISJUNKTION und NEGATION betrachten wir zunächst die Wahrheitstabellen dieser Funktionen:

a	b	konj(a,b)
0	0	0
0	1	0
	0	0
	1	1

Problemanalyse: Betrachten wir die Wahrheitstabelle für die KONJUNKTION, so sehen wir, daß die Funktion nur für die Belegung von a und b mit 1 den Funktionswert 1 liefert, sonst aber den Funktionswert 0.
Daraus resultiert folgende Kodierung:

<A>utor konj(a,b):=if(a=1 and b=1,1,0)

a	b	disj(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

Problemanalyse: Betrachten wir die Wahrheitstabelle für die DISJUNKTION, so sehen wir, daß die Funktion nur für die Belegung von a und b mit 0 den Funktionswert 0 liefert, sonst aber den Funktionswert 1.

Daraus resultiert folgende Kodierung:

<A>utor disj(a,b):=if(a=0 and b=0,0,1)

Die NEGATION ist eine Funktion mit nur einem Argument, daher:

a	neg(a,b)
0	1
1	0

Daraus resultiert folgende Kodierung:

<A>uthor neg(a):=if(a=0,1,0)

UNTERRICHTSEINHEIT 2: ÜBERPRÜFEN DER BELEGUNGEN MITTELS WAHRHEITSTABELLEN.

Zunächst erstellen wir die Belegungsmatrix M für die KONJUNKTION und DISJUNKTION:

$$M: = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Kodierung:

<A>uthor M:=[[0,0],[0,1],[1,0],[1,1]]

Wollen wir die Wertetabellen für unsere logischen Funktionen herstellen, so definieren wir

α(i,j):=ELEMENT(ELEMENT(M,i),j)

i bezeichnet die Zeile der Belegungsmatrix,
j bezeichnet die Spalte der Belegungsmatrix.
Mit α(3,2) können wir also das 2. Element aus der 3. Zeile der Belegungsmatrix extrahieren.

Mit Hilfe des Moduls VECTOR können wir nun die Wertetabellen unserer logischen Funktionen erzeugen:

<A>uthor VECTOR([α(i,1),α(i,2),konj(α(i,1),α(i,2))],i,4]

<A>uthor VECTOR([α(i,1),α(i,2),disj(α(i,1),α(i,2))],i,4]

<A>uthor VECTOR([α(i,2),neg(α(i,2))],i,2]

Durch Anwenden von <S>implify auf die einzelnen Ausdrücke erhalten wir:

```
8:   VECTOR ([α (i, 1), α (i, 2), KONJ (α (i, 1), α (i, 2))], i, 4)

     ⎡ 0  0  0 ⎤
     ⎢ 0  1  0 ⎥
9:   ⎢ 1  0  0 ⎥
     ⎣ 1  1  1 ⎦

10:  VECTOR ([α (i, 1), α (i, 2), DISJ (α (i, 1), α (i, 2))], i, 4)

     ⎡ 0  0  0 ⎤
     ⎢ 0  1  1 ⎥
11:  ⎢ 1  0  1 ⎥
     ⎣ 1  1  1 ⎦

12:  VECTOR ([α (i, 2), NEG (α (i, 2))], i, 2)

13:  ⎡ 0  1 ⎤
     ⎣ 1  0 ⎦

COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer moVe Window approX
Enter option
User                B:\KREMSUEZ.MTH      Free:100%            Derive Algebra
```

UNTERRICHTSEINHEIT 3: VERKETTEN VON FUNKTIONEN (IMPLIKATION, ÄQUIVALENZ), ÄQUIVALENZSÄTZE UND "KÜRZUNGSREGELN".

Der letzte Abschnitt des Lehrgangs ist beispielorientiert in der Übungsphase auszuführen. Nachfolgend einige Beispiele aus der Übungsphase:

Beispiel 1: Definition der logischen Funktionen IMPLIKATION und ÄQUIVALENZ durch Verkettung unserer Grundfunktionen

Als Ausgangspunkt verwenden wir die beiden Äquivalenzsätze:

* impl(a,b) ≡ disj(neg(a),b)
* aequ(a,b) ≡ konj(impl(a,b),impl(b,a)).

Wir übertragen die beiden neuen logischen Funktionen auf das Arbeitsblatt des Computers:

<A>uthor impl(a,b):=disj(neg(a),b)
<A>uthor aequ(a,b):=konj(impl(a,b),impl(b,a))

Die Wertetabellen unserer neuen Funktionen erhalten wir durch Substitution.

Mit <M>anage <S>ubstitute ersetzen wir etwa im Term

VECTOR([α(i,1),α(i,2),konj(α(i,1),α(i,2))],i,4]

den Subterm *konj(α(i,1),α(i,2))* durch den Term

impl(α(i,1),α(i,2)) bzw. *aequ(α(i,1),α(i,2))*.

Mit <S>implify erhalten wir die Wertetabellen.

```
14:  IMPL (a, b) := DISJ (NEG (a), b)

15:  AEQU (a, b) := KONJ (IMPL (a, b), IMPL (b, a))

16:  VECTOR ([α (i, 1), α (i, 2), IMPL (α (i, 1), α (i, 2))], i, 4)

17:  [ 0 0 1 ]
     [ 0 1 1 ]
     [ 1 0 0 ]
     [ 1 1 1 ]

18:  VECTOR ([α (i, 1), α (i, 2), AEQU (α (i, 1), α (i, 2))], i, 4)

19:  [ 0 0 1 ]
     [ 0 1 0 ]
     [ 1 0 0 ]
     [ 1 1 1 ]

COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer moVe Window approX
Enter option
User                B:\KREMSUE2.MTH      Free:100%              Derive Algebra
```

Beispiel 2: Überprüfung der DE MORGANschen Regeln

* neg(konj(a,b)) ≡ disj(neg(a),neg(b))
* neg(disj(a,b)) ≡ konj(neg(a),neg(b))

Mit <M>anage <S>ubstitute ersetzen wir wieder etwa im Term

VECTOR([α(i,1),α(i,2),konj(α(i,1),α(i,2))],i,4]

den Subterm *konj(α(i,1),α(i,2))* durch

aequ(neg(konj(α(i,1),α(i,2))),disj(neg(α(i,1)),neg(α(i,2))))
bzw.
aequ(neg(disj(α(i,1),α(i,2))),konj(neg(α(i,1)),neg(α(i,2))))

Mit <S>implify erhalten wir Wertetabellen, die in der Spalte der Funktionswerte lauten Einser enthalten. Man kann nun mit den Schülern den Begriff der TAUTOLOGIE (für sämtliche Belegungen erhalten wir den Funktionswert 1), einen zentralen Begriff der formalen Logik, erarbeiten.

```
18:  VECTOR ([α (i, 1), α (i, 2), AEQU (α (i, 1), α (i, 2))], i, 4)

19:  [ 0 0 1 ]
     [ 0 1 0 ]
     [ 1 0 0 ]
     [ 1 1 1 ]

20:  VECTOR ([α (i, 1), α (i, 2), AEQU (NEG (KONJ (α (i, 1), α (i, 2))), DISJ (N

21:  [ 0 0 1 ]
     [ 0 1 1 ]
     [ 1 0 1 ]
     [ 1 1 1 ]

COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer moVe Window approX
Enter option
User                B:\KREMSUE2.MTH      Free:100%            Derive Algebra
```

Beispiel 3: Logische Ketten (MODUS PONENS)

Eine "klassische" Regel logischen Schießens ist die Regel MODUS PONENS:

Prämisse 1: a impliziert b
Prämisse 2: a
Konklusion: Daher: b

Die Regel müssen wir zunächst in eine Verkettung von logischen Funktionen übertragen

<A>uthor mopo(a,b):=impl(konj(impl(a,b),a),b)

und anschließend mit <M>anage <S>ubstitute als Subterm *mopo(α(i,1),α(i,2))* eintragen.
Mit <S>implify müssen wir wiederum die Wertetabelle einer TAUTOLOGIE erhalten.

Anmerkung: Anschließend könnten die Schüler überprüfen, ob auch

Prämisse 1: a impliziert b
Prämisse 2: Nicht a
Konklusion: Nicht b

eine gültige Schlußregel der formalen Logik ist.

Beispiel 4: Überprüfung der Verschmelzungsregeln als "Kürzungsregeln"

Als Subterm substituieren wir

konj(α(i,1),disj(α(i,1),α(i,2)))
disj(α(i,1),konj(α(i,1),α(i,2)))

und vergleichen die Funktionswerte in der Wertetabelle mit der ersten Spalte der Wertetabelle.

V. Schlußgedanke

Das große Engagement der Schüler läßt mich hoffen, daß sie durch diesen Lehrgang einen ersten Einblick in die Bedeutung mathematischer Begriffe (Logische Verknüpfungen, Funktionen und Verkettungen von Funktionen) erhalten haben.

Literaturhinweise

* Buchberger, B./Lichtenberger, F.: *Mathematik für Informatiker*; I. Springer Verlag, 1980, 1981
* Fuchs, K. J.: *Logic with DERIVE*; Angekündigt für Derive-News-Letter, #6, Würmla 1992
* Bürger u. a.: *Mathematik Oberstufe 1*; Verlag HPT, Wien 1989 (Schulbuch-Nr. 0618)
* Morscher, E./Stranzinger, E.: *LOGIK I: Übungsaufgaben*; Skriptum an der Philosophische Fakultät der Universität Salzburg, Salzburg, 1977
* Reichel u. a.: *Lehrbuch der Mathematik*, 1. Band; Verlag HPT, Wien 1989 (Schulbuch-Nr. 0620)
* Szirucsek u. a.: *Mathematik 5*; Verlag HPT, Wien 1989 (Schulbuch-Nr. 2508)
* Touretzky, D.S.: *A Gentle Introduction to Symbolic Computation*; Harper&Row, Publishers, N.Y., 1984