Logische Programmierung: Symbolisches Ableiten mit Prolog

Helge Janicke, Niels-Peter de Witt, Karsten Wolke

28. Mai 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Symbolisches Ableiten in Prolog

Die Ableitung wird durch ein 3 stelliges Prädikat repräsentiert:

```
derive(F,X,DF) :-
    ableitung(F,X,DX),
    zusammenfassen(DX,ZX),
    mache_Term(ZX,DF).
```

F ist eine Funktion als Prolog-Struktur.
z.B. `2+4*x+2*x^2`

X gibt die Variable an, nach der differenziert werden soll.

DF wird mit der Ableitung von **F** nach **X** belegt.
Der Rückgabewert ist auch wieder eine Funktion in Prolog-Struktur.

Die Prädikate `zusammenfassen(DX,ZX)` und `mache_Term(ZX,DF)` fassen die, evtl. sehr aufgeblähte, Ableitung wieder zusammen. Die Arbeitsweise von `zusammenfassen/2` wird in Abschnitt 2 näher erläutert. Das Prädikat `mache_Term/2` wandelt die Liste zurück in einen Term, wobei 0 Summanden und 1 Faktoren eliminiert werden, als kleiner Schmankerl wird das positive Vorzeichen des ersten Summanden des Gesamtterms weggelassen.

1.1 Implementierte Ableitungsregeln

1.1.1 Summenregel

$$\begin{aligned} y &= f_1(x) \pm f_2(x) \pm \ldots \pm f_n(x) \\ y' &= f_1'(x) \pm f_2'(x) \pm \ldots \pm f_n'(x) \end{aligned}$$

als Prolog-Regeln

```
ableitung(F+G,X,DF+DG):-
    ableitung(F,X,DF),
    ableitung(G,X,DG),
    !.

ableitung(F-G,X,DF-DG):-
    ableitung(F,X,DF),
    ableitung(G,X,DG),
    !.
```

1.1.2 Produktregel

$$\begin{aligned} y &= f(x) * g(x) \\ y' &= g(x) * f'(x) + f(x) * g'(x) \end{aligned}$$

als Prolog-Regeln

```
ableitung(F*G,X,G*DF+F*DG):-
    ableitung(F,X,DF),
    ableitung(G,X,DG),
    !.
```

1.1.3 Quotientenregel

$$\begin{aligned} y &= \frac{f(x)}{g(x)} \\ y' &= \frac{g(x) * f'(x) - f(x) * g'(x)}{g(x)^2} \end{aligned}$$

als Prolog-Regeln

```
ableitung(F/G,X,(G*DF-F*DG)/(G^2)):-
    ableitung(F,X,DF),
    ableitung(G,X,DG),
    !.
```

1.1.4 Kettenregel

Bei der Umsetzung der Kettenregel unterscheiden wir in einstellige und zweistellige Funktionen. Die Ableitungen der einzelnen Funktionen werde in Kapitel 1.2 näher beschrieben.

Einstellige Funktionen

$$\begin{aligned} y &= f(g(x)) \\ y' &= \frac{d\,y}{d\,g} * \frac{d\,g}{d\,x} \end{aligned}$$

als Prolog-Regeln

```
ableitung(F,X,DF*DG):-
    functor(F,Fname,Stelligkeit),
    Stelligkeit =:= 1,
    mache_liste(F,[Fname,G]),
    funktion(Fname,G,DF),
    ableitung(G,X,DG),
    !.
```

Der von uns behandelte Sonderfall für eine zweistellige Funktion bei der Verkettung ist die Potenzfunktion. Diese setzen wir wie folgend um:

Potenzfunktion

$$\begin{aligned} y &= g(x)^{h(x)} \\ &= e^{h(x)*ln(g(x))} \\ y' &= g(x)^{h(x)} * \left(h'(x) * ln(g(x)) + \frac{h(x)}{g(x)} * g'(x) \right) \\ &= g(x)^{h(x)-1} * \left(h'(x) * ln(g(x)) * g(x) + h(x) * g'(x) \right) \\ &= g(x)^{h(x)-1} * \left(\frac{h'(x) * ln(g(x)) * g(x)}{g'(x)} + h(x) \right) * g'(x) \end{aligned}$$

als Prolog-Regeln

```
ableitung(F,X,DF*((DH*ln(G)*G)/(DG*H)+1)*DG):-
    functor(F,Fname,Stelligkeit),
    Stelligkeit =:= 2,
    mache_liste(F,[Fname,G,H]),
    funktion(Fname,G,H,DF),
    ableitung(G,X,DG),
    ableitung(H,X,DH),
    !.
```

1.2 Ableitung elementarer Funktionen

Die Ableitung der elementaren Funktionen werden über das Prädikat `funktion/3` und `funktion/4` je nach Stelligkeit definiert. Die Prädikate befinden sich in der Datei *funktion.pro*.

Beispiel für eine einstellige Funktion:

$$\begin{aligned} y &= sin(x) \\ y' &= cos(x) \end{aligned}$$

als Prolog-Regeln

```
funktion(sin,X,cos(X)).
```

Beispiel für eine zweistellige Funktion:

			als Prolog-Regeln
y	$=$	$f(x)^n$	`funktion(^,X,N,N*X^N1) :-`
y'	$=$	$n * f(x)^{n-1}$	`  number(N),`
			`  N1 is N -1.`

Tabelle 1: Liste der bereits implementierten Funktionen

$sin(x)$	$cos(x)$	$tan(x)$	$cot(x)$
$arcsin(x)$	$arccos(x)$	$arctan(x)$	$arccot(x)$
$sinh(x)$	$cosh(x)$	$tanh(x)$	$coth(x)$
$arsinh(x)$	$arcosh(x)$	$artanh(x)$	$arcoth(x)$
$exp(x)$	$ln(x)$	$(x)^{\frac{z}{n}}$	$(x)^{g(x)}$

1.3 Beispiele

$\boxed{\sqrt{x} = \frac{1}{2\sqrt{x}}}$

Hier sind die einzelnen Schritte von `derive/3` ausführlich geschrieben. Wie man sofort erkennt kann das Prädikat `ableitung/3` bereits mit ganzrationalen Zahlen umgehen. Die folgenden Beispiele werden direkt das Prädikat `derive/3` verwenden.

```
?- ableitung(x^(1/2),x,D),zusammenfassen(D,Z),mache_Term(Z,T).

D = 1/2*x^ (-1/2)* ((2*0-1*0)/2^2*ln(x)*x/ (1* (1/2))+1)*1
Z = [[0.5, [[x^ -0.5]]], [0, [[x^0.5], [ln(x)^1]]]]
T = 0.5*x^ -0.5 ;
```

$\boxed{(sin(x))' = cos(x)}$

```
?- derive(sin(x),x,D).

D = cos(x)
```

$\boxed{(arctan(x))' = \frac{1}{1+x^2}}$

```
?- derive(arctan(x),x,D).

D = (1+x^2)^ -1
```

$\boxed{(x^2 + x + 3)' = 2x + 1}$

```
?- derive(x^2+x+3,x,D).

D = 1+2*x
```

$\boxed{((x^2 + x) * sin(x))' = (2x + 1)\, sin(x) + (x^2 + x)\, cos(x)}$

```
?- derive((x^2+x)*sin(x),x,D).

D = x*cos(x)+x^2*cos(x)+sin(x)+2* (x*sin(x))
```

$$\boxed{((x^2+x)/(x^2+1))' = \frac{(2x+1)*(x^2+1)-2x*(x^2+x)}{(x^2+1)^2}}$$

```
?- derive((x^2 + x) / (x^2+1),x,D).

D = -x^2* (1+x^2)^ -2+ (1+x^2)^ -2+2* (x* (1+x^2)^ -2)
```

$$\boxed{(sin(x^2))' = 2x\, cos(x)}$$

```
?- derive(sin(x^2),x,D).

D = 2* (x*cos(x^2))
```

$$\boxed{(sin(cos(x^2)))' = -2x\, cos(cos(x^2))\, sin(x^2)}$$

```
?- derive(sin(cos(x^2)),x,D).

D = -2* (x*cos(cos(x^2))*sin(x^2))
```

2 Zusammenfassen von Ausdrücken

Zum Zusammenfassen konvertieren wir einzelne Terme in Listen. Je nach Operation werden verschiedene Verknüpfungen vorgenommen, die später näher erläutert werden. Die Funktion wird rekursiv auf Konstanten (atomic) herruntergebrochen und Listen konvertiert. Beim Rekursionsrücklauf werden die einzelnen Listen miteinander verknüpft.
Aufbau einer solchen Liste:

$$[[F_1, [[x_{1,1}^{n_{1,1}}], [x_{1,2}^{n_{1,2}}], \ldots, [x_{1,k_1}^{n_{1,k_1}}]]], [F_2, \ldots], \ldots, [F_l, [[x_{l,1}^{n_{l,1}}], [x_{l,2}^{n_{l,2}}]]]]$$

die wie folgend interpretiert wird:

$$\sum_{i=1}^{l} \left(F_i * \prod_{j=1}^{k_i} x_{i,j}^{n_{i,j}} \right)$$

Für jede der Operationen `+, -, *, /` gibt es eine Regel `zusammenfassen/2`. Die Regeln für `+` und `*` verwenden die Prädikate `add_zusammen/2` bzw. `mul_zusammen/2`, welche die Summanden/ Faktoren kombinieren. Subtraktion und Division werden auf Addition und Multiplikation zurückgeführt:

$$\begin{aligned} a - b &= a + (-1) * b \\ a/b &= a * b^{(}-1) \end{aligned}$$

2.1 Ablauf der Addition

Im folgenden werden wir die Vorgang des Zusammenfassens für das Beispiel 1 erläutern.

```
a+b*(c+d)*b+a
```

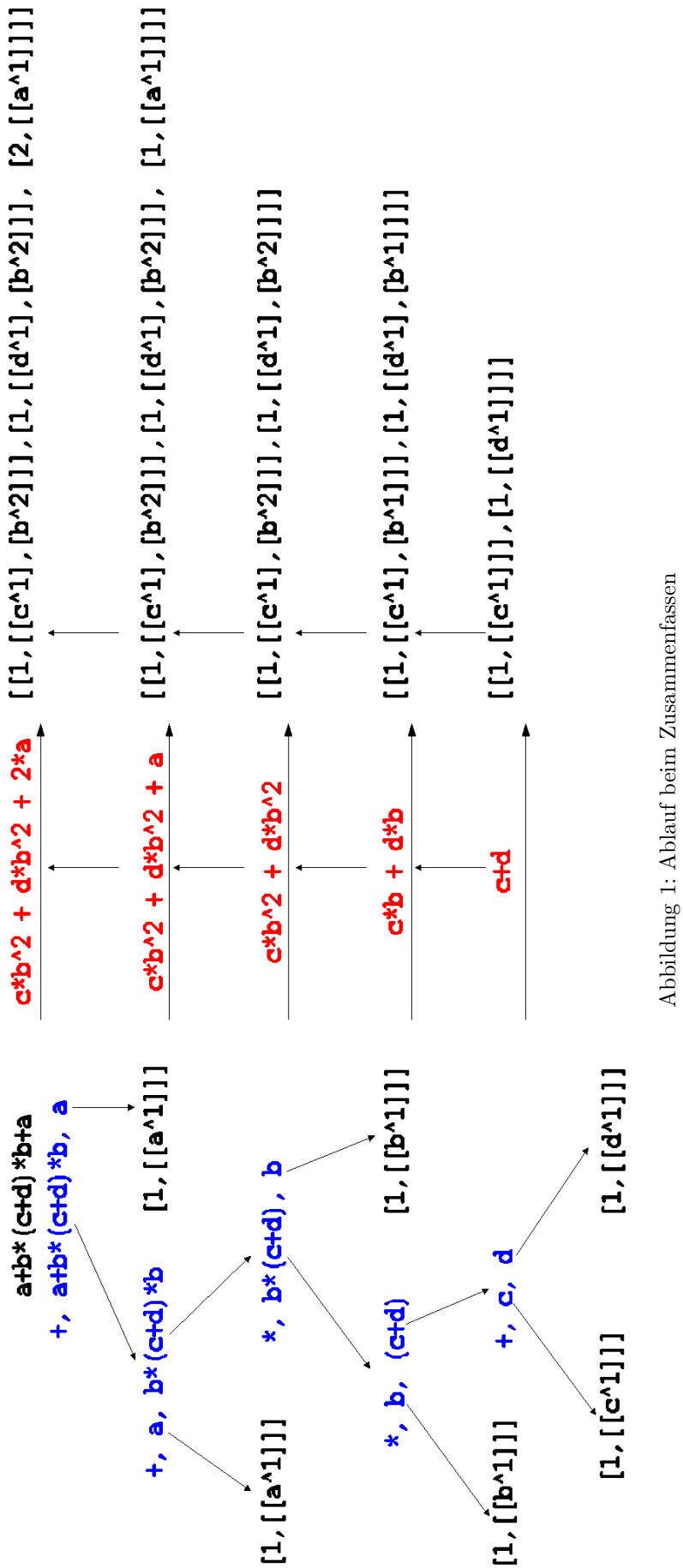

Abbildung 1: Ablauf beim Zusammenfassen

Die verwendeten Prädikate finden Sie in Abschnitt 2.1.1 abgedruckt.

Das Prädikat `zusammenfassen/2` (Beispielweise in Zeile 9 für die Addition) zerteilt den Eingangsterm rekursiv bis auf Konstanten (atomic). Der Rekursionsbaum ist im linken Teil von Abbildung 1 zu sehen.

Am Rekursionsende wird aus einer Konstanten eine Liste konstruiert. Siehe `zusammenfassen/2` in Zeile 1 und 5.
Beim Rekursionsrücklauf werden die beiden entstandenen Listen zusammengefasst. Beispielsweise wird in Zeile 15 für die Addition `add_zusammen/3` aufgerufen. Ganz zu Anfang des Rekursionsrücklaufs werden die beiden Listen `[1,[[c^1]]]` und `[1,[[d^1]]]` zu einer Liste verknüpft.
Dabei unterteilt `add_zusammen/3` in Zeile 21 die rechte Liste rekursiv in einzelne Summanden. Für jeden Summand wird `add_einzeln/3` gerufen. Hier wird überprüft, ob in der linken Liste bereits ein Summand mit gleichem $\prod_{j=1}^{k_i} x_{i,j}^{n_{i,j}}$ (siehe Beschreibung des Listenaufbaus in 2). Die Umsetzung finden Sie in Zeile 27. Existiert dort bereits ein solcher Summand erhöht das Prädikat `add/3` den Faktor (F_i) des Summanden der linken Liste um den Faktor des Summanden aus der rechten Liste. Im anderen Fall wird der Summand einfach der linken Liste hinzugefügt.

2.1.1 verwendete Prädikate

```
1  zusammenfassen(F,[[F,[[1^1]]]]) :-
2      atomic(F),
3      number(F).
4
5  zusammenfassen(F,[[1,[[F^1]]]]) :-
6      atomic(F),
7      \+ number(F).
8
9  zusammenfassen(F,ZF_List) :-
10     functor(F,+,Stelligkeit),
11     Stelligkeit =:= 2,
12     zerteile(F,[+,LOp,ROp]),
13     zusammenfassen(LOp ,ZLOp_List),
14     zusammenfassen(ROp,ZROp_List),
15     add_zusammen(ZLOp_List,ZROp_List,ZF_List).
16
17
18 add_zusammen(L_List,[],L_List_sort) :-
19     sort(L_List,L_List_sort).
20
21 add_zusammen(L_List,[Kopf|Rest],LR_List) :-
22     add_einzeln(L_List,Kopf,LList_neu),
23     add_zusammen(LList_neu,Rest,LR_List).
24
25
26 add_einzeln(L_List,[Faktor,Atome],LR_List) :-
27     member([_,Atome],L_List),
28     add(L_List,[Faktor,Atome],LR_List).
29
30 add_einzeln(L_List,[Faktor,Atome],LR_List) :-
31     \+ member([_,Atome],L_List),
32     append([[Faktor,Atome]],L_List,LR_List).
33
34
```

```
35 add([[LFaktor,Atome]|Rest],[RFaktor,Atome],[[LRFaktor,Atome]|Rest]) :-
36     LRFaktor is LFaktor + RFaktor.
37
38 add([[LFaktor,LAtome]|Rest],[RFaktor,RAtome],[[LFaktor,LAtome]|LList_neu]) :-
39     LAtome \== RAtome,
40     add(Rest,[RFaktor,RAtome],LList_neu).
```

2.2 Implementierte Vereinfachungen

Bei der Addition werden wenn möglich Faktoren (F_i) addiert (vgl.2.1). Bei der Multiplikation werden die bei gleichen Faktoren im Produkt die Potenzen addiert. Subtraktion und Division greifen auf diese Mechanismen zurück. Alle einstelligen Funktionen, wie z.B. `sin(f(x))` optimieren das Argument ($f(x)$) zu $g(x)$ und bilden eine neue Liste analog zu der am Rekursionsende gebildeten.

`[[1,[[sin(g(x))^1]]]]`

Bei Potenzen werden folgende Optimierungen durchgeführt:

$$\begin{aligned} 2^2 &= 4 \\ x^{2^2} &= x^4 \\ (x+x+x)^{(}y+y+y) &= (3*x)^{(3*y)} \end{aligned}$$

Präfix Plus wird ignoriert, Präfix Minus kehrt die Vorzeichen aller Summanden (F_i) um. Als Ergebnis liefert `zusammenfassen/2` eine Liste der oben genannten Struktur (Siehe 2).

Eine Liste dieser Struktur läßt sich mit dem Prädikat `mache_Term/2` wieder in ein Term umwandeln.

2.3 Mögliche Erweiterungen

- Bei der Multiplikation kann geprüft werden wie lang die Listen sind, und evtl. bei sehr langen Listen nicht ausmultiplizieren werden.
- Ausklammern von gleichen Faktoren aus Teilen der Summandenliste.
- Einfache Additionstheoreme können durch Analyse der Summanden erkannt werden. ($sin^2(x) + cos^2(x) = 1$)
- Komplexere Additionstheoreme lassen sich durch Schachtelung von Strukturen (siehe `sin(f(x))`) auflösen, oder durch einen nachgeschalteten Optimierungsdurchlauf bearbeiten.
- Brüche werden in Fließkommazahlen umgewandelt. Es ist aber durchaus möglich mit Brüchen weiterzurechnen. (siehe `ableitung/3`)
- Weitergehendes Kürzen läßt sich durch Ausklammern erledigen.

2.4 Beispiele

$\boxed{a + a = 2a}$ Zusammenfassen gleicher Summanden.

```
?- zusammenfassen(a+a,Z), mache_Term(Z,T).

Z = [[2, [[a^1]]]]
T = 2*a
```

$\boxed{a * a = a^2}$ Zusammenfassen gleicher Faktoren.

```
?- zusammenfassen(a*a,Z), mache_Term(Z,T).

Z = [[1, [[a^2]]]]
T = a^2
```

$\boxed{a*0+1=1}$ Eliminieren von 0* Termen.

```
?- zusammenfassen(a*0+1,Z), mache_Term(Z,T).

Z = [[0, [[a^1]]], [1, [[1^1]]]]
T = 1
```

$\boxed{1*a=a}$ Eliminieren von 1* Faktoren.

```
?- zusammenfassen(1*a,Z), mache_Term(Z,T).

Z = [[1, [[a^1]]]]
T = a
```

$\boxed{2^{2^3}=256}$ Berechnen von numerischen Potenzen.

```
?- zusammenfassen(2^2^3,Z), mache_Term(Z,T).

Z = [[256, [[1^1]]]]
T = 256
```

$\boxed{x^{2^3}=x^8}$ Berechnen von numerischen Exponenten.

```
?- zusammenfassen(x^2^3,Z), mache_Term(Z,T).

Z = [[1, [[x^8]]]]
T = x^8
```

$\boxed{(x+x+x)^{y+y+y}=3x^{3y}}$ Zusammenfassen von komplexen Potenzen.

```
?- zusammenfassen((x+x+x)^(y+y+y),Z), mache_Term(Z,T).

Z = [[1, [[ (3*x)^ (3*y)]]]]
T = (3*x)^ (3*y)
```

$\boxed{sin(a^2+a^2)+sin(a^2+a^2)=2sin(2a^2)}$ komplexere Summanden.

```
?- zusammenfassen(sin(a^2+a^2)+sin(a^2+a^2),Z), mache_Term(Z,T).

Z = [[2, [[sin(2*a^2)^1]]]]
T = 2*sin(2*a^2)
```

$\boxed{(4+(a*a*a*a-7)*a+(9*a))/a=\frac{4}{a}+a^4+2}$ komplexere Terme.

```
?- zusammenfassen((4+(a*a*a*a-7)*a+(9*a))/a,Z), mache_Term(Z,T).

Z = [[1, [[a^4]]], [2, [[1^1]]], [4, [[a^ -1]]]]
T = a^4+2+4*a^ -1
```

$\boxed{x^{a+x^{a+b}}*x^{a+x^{a+b}}+4*x=x^{2(a+x^{a+b})}+4x}$ komplexere Terme.

```
?- zusammenfassen(x^(a+x^(a+b))*x^(a+x^(a+b))+4*x,Z), mache_Term(Z,T).

Z = [[1, [[x^ (2*a+2*x^ (... +...))]]], [4, [[x^1]]]]
T = x^ (2*a+2*x^ (a+b))+4*x
```