Kurs ▮ Cours

11. Problemsammlung zur Musterentsprechung (Mustererkennen, musterkonformes Abarbeiten)
▮ Collection de problèmes concernant les correspondances et patrons (reconnaître les patrons, travailler conformement aux patrons)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach.... Hinweis: Kapitel 11 lesen!
▮ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 11.

Run mit WIN+*Mathematica* Version 5.2
▮ Testé avec *Mathematica* version 5.2+WIN

11.1. Mustererkennung bei einer Sequenz
▮ Reconnaître un patron lors d'une séquence

Ein Beispiel
▮ Un exemple

Bisher bekannte Variablen mit "_": Z.B. in f[x_]:=x wird bei der Anwendung f[y] x durch y ersetzt. Oder bei f[3] x durch 3. Oder bei f[x+2y-Sin[z]] wird x durch x+2y-Sin[z] ersetzt. *Mathematica* erkennt hier das "Muster" des zu ersetzenden Ausdrucks. Wie lässt sich damit z.B. der Mittelwert einer
beliebig grossen Zahlenmenge berechnen? Beispiel:
▮ Variables avec "_" déjà connues: P.ex. dans f[x_]:=x on remplace f[y] x par y à l'application.Ou dans f[3] x par 3. Ou dans f[x+2y-Sin[z]] on remplace x par x+2y-Sin[z]. *Mathematica* reconnaît ici le "patron" de l'expression à remplacer. Comment peut-on calculer p.ex. la valeur moyenne d'un ensemble d'une grandeur quelconque de nombres? Exemple:

```
In[1]:= mittel[a_]:= a;
        mittel[a_, b_]:= (a+b)/2;
        mittel[a_, b_, c_]:= (a+b+c)/3;  (*etc. ....*)
```

In[4]:= **mittel[4,6]**

Out[4]= 5

In[5]:= **mittel[9]**

Out[5]= 9

In[6]:= **mittel[1,2,3]**

Out[6]= 2

In[7]:= **mittel[1,2,3,4]**

Out[7]= mittel[1, 2, 3, 4]

Was tun? - Wir können auch das Symbol "__" (doppelter Unterstrich) verwenden resp. "___" (dreifach).
■ Que faire? Nous pouvons employer aussi le symbole "__" (double sous-ligne) resp. "___" (triple).

In[8]:= **??__**

__ (two _ characters) or BlankSequence[ ] is a pattern object that can stand for any sequence of one or more Mathematica expressions. __h or BlankSequence[h] can stand for any sequence of one or more expressions, all of which have head h. Mehr...

Attributes[BlankSequence] = {Protected}

In[9]:= **?___**

___ (three _ characters) or BlankNullSequence[ ] is a pattern object that can stand for any sequence of zero or more Mathematica expressions. ___h or BlankNullSequence[h] can stand for any sequence of expressions, all of which have head h. Mehr...

Beispiel: ■ Exemple:

In[10]:=
```
Remove[mittel];
mittel[x__] := Plus[x]/Length[{x}]
```

In[12]:= **mittel[1,2,3,4,5,6,7,8,9]**

Out[12]= 5

In[13]:= **mittel[a,b,c,d,e,f,g]**

Out[13]= $\frac{1}{7}(a + b + c + d + e + f + g)$

In[14]:= **mittel[{a,b,c,d,e,f,g}]**

Out[14]= {a, b, c, d, e, f, g}

Um die Funktion auch auf eine Liste anwendbar zu machen, definieren wir:
■ Pour rendre la fonction aussi applicable à une liste, nous définissons:

In[15]:= **mittel[{x__}] := mittel[x]**

In[16]:= **mittel[{a,b,c,d,e,f,g}]**

Out[16]= $\frac{1}{7}(a + b + c + d + e + f + g)$

Nun klappt es!
■ Voilà, ça marche!

11.2. Nachbau von Funktionen
■ Copier (imiter) des fonctions

11.2.1. "First" (herauspicken des 1. Elements einer Liste)
■ "First" (choisir le 1er élément d'une liste)

"Von Hand":
■ "A la main":

Probiere: ■ Essaie:

```
In[17]:= {a, b, c, d, e, f, g} /. {x_, y___} :> x
         (* 3 Unterstriche! *)

Out[17]= a
```

Definition einer Funktion:
■ Définition d'une fonction:

```
In[19]:= nimmErstes[{x_, y___}] := x
```

Ausprobieren: ■ Essayer:

```
In[20]:= nimmErstes[{a, b, c, d, e, f, g}]

Out[20]= a
```

11.2.2. "Last" (herauspicken des letzten Elements einer Liste)
■ "Last" (choisir le dernier élément d'une liste)

"Von Hand":
■ "A la main":

Probiere: ■ Essaie:

```
In[21]:= {a, b, c, d, e, f, g} /. {x___, y_} :> y

Out[21]= g
```

Definition einer Funktion:
■ Définition d'une fonction:

```
In[22]:= nimmLetztes[{x___, y_}] := y
```

Ausprobieren: ▮ Essayer:

```
In[23]:= nimmLetztes[{a, b, c, d, e, f, g}]

Out[23]= g
```

11.2.3. "Head"
▮ "Head"

"Von Hand":
▮ "A la main":

Probiere: ▮ Essaie:

```
In[24]:= {a, b, c, d, e, f, g} /. h_[x___] :> h
         (* 3 Unterstriche! *)

Out[24]= List
```

Definition einer Funktion:
▮ Définition d'une fonction:

```
In[26]:= nimmKopf[h_[x___]] := h
```

Ausprobieren: ▮ Essayer:

```
In[27]:= nimmKopf[{a, b, c, d, e, f, g}]

Out[27]= List
```

11.2.4. Neuordnung von Elementen
▮ Mettre des éléments dans un nouvel ordre

Zum Beispiel Vertauschung der Reihenfolge
▮ Par exemple renverser l'ordre

Probiere: ▮ Essaie:

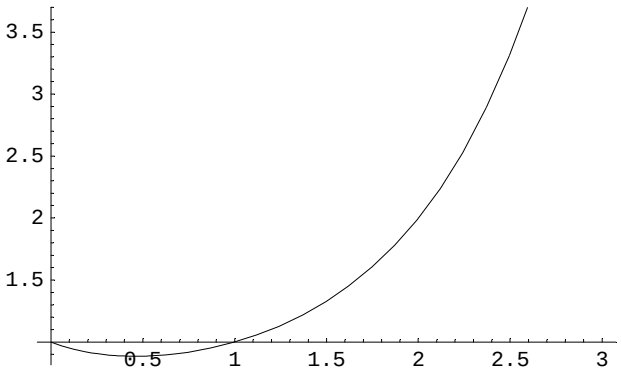
```
In[28]:= {{1,3},{2,4},{3,5},{4,6}} /. {x_, y_} :> {y, x}

Out[28]= {{3, 1}, {4, 2}, {5, 3}, {6, 4}}
```

11.3. "Polymorphe" Definitionen
▮ Définitions "polymorphes"

Hier geht es um datenabhängige Definitionen!
▮ Il s'agit ici de définitions dépendantes de données!

Gemeint sind da Definitionen, die ihre Form ändern je nach den Daten, auf die sie angewandt werden. Beispiel:
▮ Il s'agit ici de définitions qui changent leur forme selon les données auxquelles on les applique. Exemple:

```
In[29]:= Remove[pasc];
         pasc[n_Integer] := Table[Binomial[n,i],
                                  {i,0,n}];
         pasc[n_Real   ] := n!;

In[32]:= pasc[5]

Out[32]= {1, 5, 10, 10, 5, 1}

In[33]:= pasc[5.5]

Out[33]= 287.885

In[34]:= Map[pasc,{1,1.2,1.4,1.6,1.8,2,2.2,2.4,2.6,2.8,
                                              3,3.2}]

Out[34]= {{1, 1}, 1.1018, 1.24217, 1.42962, 1.67649, {1, 2, 1},
          2.42397, 2.98121, 3.71702, 4.69417, {1, 3, 3, 1}, 7.75669}

In[35]:= Plot[n!,{n,0,3}];
```



11.4. Benennung von Ausdrücken
▮ Nommer des expressions

Das Muster "x:Muster"
▮ Le patron "x:Patron"

Komplette Ausdrücke können mit Namen versehen werden. Beispiel:
▮ Les expressions complètes peuvent être pourvues de noms. Exemple:

```
In[36]:= 2 + 3 Cos[x] /.
           a:(x_ +y_ Cos[z_]) :>
               {{"a", a},{"x", x},{"y",y},{"z",z}}

Out[36]= {{a, 2 + 3 Cos[x]}, {x, 2}, {y, 3}, {z, x}}
```

Identifiziere die Namen!
▮ Identifier les noms!

11.5. Ausdrücke auffinden, die mit Mustern übereinstimmen
▮ Chercher des expressions qui correspondent à des patrons

11.5.1. Beispiele mit "Cases"
▮ Exemples avec "Cases"

Probiere aus: ▮ Essaie:

```
In[37]:= ??Cases
```

```
Cases[{e1, e2, ... }, pattern] gives a list of the ei that match the pattern. Cases[
   {e1, ... }, pattern -> rhs] gives a list of the values of rhs corresponding to
   the ei that match the pattern. Cases[expr, pattern, levspec] gives a list of all
   parts of expr on levels specified by levspec which match the pattern. Cases[expr,
   pattern -> rhs, levspec] gives the values of rhs which match the pattern. Cases[expr,
   pattern, levspec, n] gives the first n parts in expr which match the pattern. Mehr...

Attributes[Cases] = {Protected}

Options[Cases] = {Heads → False}
```

Studieren: ▮ Etudier:

```
In[38]:= Cases[{1, 2.3, {a, b}, x^2 +7, x+y+z},  x_]

Out[38]= {1, 2.3, {a, b}, 7 + x^2, x + y + z}
```

Einschränkung auf Integers oder Reals:

■ Se limiter à des Integers ou Reals:

```
In[39]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                        x_Integer]

Out[39]= {1}

In[40]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                        x_Real]

Out[40]= {2.3, 3.}

In[41]:= ?|

        p1 | p2 | ...  is a pattern object which represents any of the patterns pi. Mehr...
```

Einschränkung auf eine Alternative von Integers oder Reals:

■ Se limiter à une alternative de Integers ou de Reals:

```
In[42]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                 (x_Integer | x_Real)]

Out[42]= {1, 2.3, 3.}
```

Einschränkung auf Listen:

■ Se limiter à des listes:

```
In[43]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                           x_List]

Out[43]= {{a, b}}
```

Einschränkung auf "Numbers":

■ Se limiter à "Numbers":

```
In[44]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                        x_?NumberQ]

Out[44]= {1, 2.3, 3.}
```

11.5.2. Beispiel mit "Select"
■ Exemple avec "Select"

Nochmals die letzte Operation, aber mit "Select":

■ Encore une fois la dernière opération, mais avec "Select":

```
In[45]:= Select[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                        NumberQ]

Out[45]= {1, 2.3, 3.}
```

11.5.3. Zusammengesetzte Muster
▮ Patrons composés

Z.B. stimmt der Ausdruck "a+b+c+d" überein mit dem Muster "x_ + y__":
▮ P.ex. l'expression "a+b+c+d" correspond au patron "x_ + y__":

```
In[46]:= Cases[{1, 2.3, 3., {a, b}, x^2 +7, x+y+z},
                                          x_+y__]
```

```
Out[46]= {7 + x^2, x + y + z}
```

Oder: ▮ Ou:

```
In[47]:= Cases[{1, a + I 5, 3., {a, b}, 2 - 3I, x+y+z},
                 x_+I y_]
```

```
Out[47]= {}
```

```
In[48]:= Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
                 x_+I y_]
```

```
Out[48]= {}
```

Wieso hat das jetzt nicht "funktioniert"? - Studiere:
▮ Pourquoi cela n'a-t-il pas fonctionné? - Etudie:

```
In[49]:= FullForm[a + 5I]
```

```
Out[49]//FullForm=
       Plus[Complex[0, 5], a]
```

```
In[50]:= FullForm[a + I 5]
```

```
Out[50]//FullForm=
       Plus[Complex[0, 5], a]
```

("Complex" ist atomar. Man kann also diesen "Head" suchen.)
▮ ("Complex" est atomique. On peut donc chercher ce "Head".)

```
In[51]:= Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
                 x_Complex]
```

```
Out[51]= {2 - 3 i}
```

```
In[52]:= Cases[{1, a + 5I, 3., {a, b}, 2 - 3I, x+y+z},
                Plus[x_Complex,y_]]
```

```
Out[52]= {5 i + a}
```

11.6. Das Attribut "Orderless"
▮ L'attribut "Orderless"

Studiere das folgende Beispiel:
▮ Etudie l'exemple suivant:

```
In[53]:= f[a_Integer, b_Complex, c_Real] := {a, b, c}

In[54]:= f[5, 3-8I, 3.346]

Out[54]= {5, 3 - 8 i, 3.346}

In[55]:= f[3-8I, 3.346, 5]

Out[55]= f[3 - 8 i, 3.346, 5]
```

Offenbar spielt die Reihenfolge der Argumente eine Rolle. Das kann aber umgangen werden! Dazu muss allerdings für f das Attribut "Orderless" gesetzt und f neu definiert werden! Probiere aus:

■ Evidemment l'ordre des arguments joue un rôle. Cependant on peut éviter cela! Pour cela il faut remplacer f par l'attribut "Orderless" et définir à nouveau f ! Essaie:

```
In[56]:= Attributes[f]

Out[56]= {}

In[57]:= ClearAll[f];
         SetAttributes[f, Orderless];
         f[a_Integer, b_Complex, c_Real] := {a, b, c}

In[60]:= f[3-8I, 3.346, 5]

Out[60]= {5, 3 - 8 i, 3.346}
```

Achtung! Die Attribute Orderless, Flat, HoldAll, FoldFirst, HoldRest und Listable müssen vor der Definition der Funktion gesetzt werden!

■ Attention! Les attributs Orderless, Flat, HoldAll, FoldFirst, HoldRest et Listable doivent être placés avant la définition de la fonction!

11.7. Beispiele mit Mustererkennung
■ Exemples avec reconnaissance de patron

11.7.1. Selektives Ausmultiplizieren
■ Multiplications sélectives

Beispiel

■ Exemple

Es soll eine Funktion geschrieben werden, die es erlaubt, ausschliesslich nur die Terme auszumultiplizieren, die unter einer Logarithmus-Funktion stehen. Eine Lösung:

■ Qu'on écrive une fonction qui permette de multiplier exclusivement les termes qui sont placés sous une fonction de logarithme. Voici une solution:

```
In[61]:= Clear[logAusmult];
         logAusmult[x_] := x /. Log[y_] :> Log[Expand[y]]
```

```
In[63]:= logAusmult[(4-3x)^5 Log[-(1-x)^3] /(x-1)^3]
```

$$\text{Out[63]}= \frac{(4-3x)^5\,\text{Log}[-1+3x-3x^2+x^3]}{(-1+x)^3}$$

11.7.2. Manipulation von Klammern
▮ Manipulation de parenthèses

Beispiele
▮ Exemples

Eine Liste von Listen soll in eine Liste verwandelt werden:

▮ Qu'on transforme une liste de listes en une liste:

```
In[64]:= {{a, b, c, d},{e, f, g}} /.
                        {{x__},{y__}} :> {x, y}

Out[64]= {a, b, c, d, e, f, g}

In[65]:= {{a, b, c, d},{e, f, g}} /.
                        {x__} :> x

Out[65]= Sequence[{a, b, c, d}, {e, f, g}]

In[66]:= Remove[f];
         Map[f,{{a, b, c, d},{e, f, g}}]

Out[67]= {f[{a, b, c, d}], f[{e, f, g}]}

In[68]:= Map[f,{{a, b, c, d},{e, f, g}}] /.
                        f[{x__}] :> f[x]

Out[68]= {f[a, b, c, d], f[e, f, g]}
```

11.7.3. Eine eigene Definition von "Map"
▮ Une propre définition de "Map"

Studiere das folgend Beispiel:

▮ Etudie l'exemple suivant:

```
In[69]:= fktMap[f_, {}] := {};
         fktMap[f_, {a_, b___}] :=
                     Prepend[fktMap[f, {b}], f[a]];


In[71]:= fktMap[Sin,{a, 2, 3, 4, 5, 6, 7}]

Out[71]= {Sin[a], Sin[2], Sin[3], Sin[4], Sin[5], Sin[6], Sin[7]}
```

11.7.4. Ziffern zählen
▮ Compter des Chiffres

Beispiel
▮ Exemple

Es soll die Frage beantwortet werden, ob es sechs aufeinanderfolgende Ziffern "9" gibt unter den ersten 1000 Dezimalstellen von Pi. Falls ja, so soll die Anzahl Dezimalstellen zwischen Komma und dieser Sechsergruppe ausgegeben werden. Studiere die Lösung:
▮ Il faut répondre à la question si le chiffre "9" apparaît 6 fois de suite parmi les 1000 premières unités décimales de Pi. Si cela est le cas, il faut sortir le nombre d'unités décimales situées entre la virgule et ce groupe de six "9". Etudie la situation:

```
In[72]:= Characters[ToString[1234567]]
```

```
Out[72]= {1, 2, 3, 4, 5, 6, 7}
```

```
In[73]:= charListe = Characters[ToString[N[Pi, 10]]]
```

```
Out[73]= {3, ., 1, 4, 1, 5, 9, 2, 6, 5, 4}
```

Ohne Output (1000....)!
Sans output (1000...)!

```
In[74]:= charListe = Characters[ToString[N[Pi, 1000]]];
```

```
In[75]:= findeNeun[{"3", ".", x___,
                "9","9","9","9","9","9",___}] :=
                           Length[{x}];
```

```
In[76]:= findeNeun[charListe]
```

```
Out[76]= 761
```

11.7.5. Uebergabe von Optionen (Options)
▮ Transmission d'options

Beispiel
▮ Exemple

Studiere: ▮ Etudie

```
In[77]:= Needs["Graphics`Graphics`"]
```

In[78]:= **??Graphics**

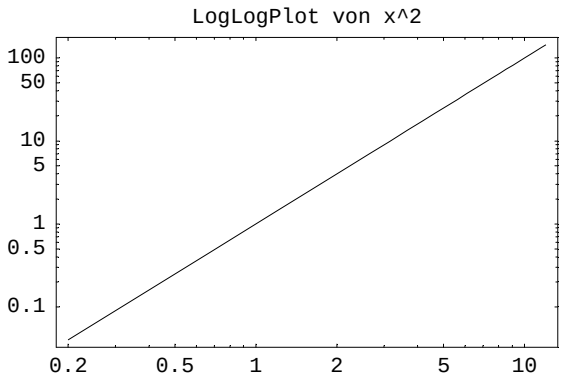
```
Graphics[primitives, options] represents a two-dimensional graphical image. Mehr...

Attributes[Graphics] = {Protected, ReadProtected}

Options[Graphics] = {AspectRatio → 1/GoldenRatio, Axes → False, AxesLabel → None,
  AxesOrigin → Automatic, AxesStyle → Automatic, Background → Automatic,
  ColorOutput → Automatic, DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, FormatType :→ $FormatType, Frame → False,
  FrameLabel → None, FrameStyle → Automatic, FrameTicks → Automatic, GridLines → None,
  ImageSize → Automatic, PlotLabel → None, PlotRange → Automatic, PlotRegion → Automatic,
  Prolog → {}, RotateLabel → True, TextStyle :→ $TextStyle, Ticks → Automatic}
```

Beachte: ■ Observe:

Der zu verwendende Modul sieht etwa so aus
(- man sieht sofort, was übergeben werden kann...):
■ Le module à utiliser a environ cet aspect
(- on voit tout de suite ce qui peut être transmis...):

```
LogLogPlot[f_, {x_, xmin_, xmax_}, opts___] :=
      Module[{r,
        g = ParametricPlot[{Log[10,x], Log[10,f]}, {x, xmin, xmax},
            Ticks->{LogScale, LogScale},
            FrameTicks -> {LogScale, LogScale,
                  LogScale, LogScale},
            DisplayFunction -> Identity, opts]},
    r = PlotRange[g];
    Show[g, DisplayFunction -> $DisplayFunction,
            PlotRange -> r, AxesOrigin -> Map[#[[1]]&,r]]]
```

In[79]:=
```
LogLogPlot[x^2, {x, 0.2, 12},
           PlotLabel -> "LogLogPlot von x^2",
           Frame -> True];
```



Versuche dieses Beispiel mit anderen Optionen!
■ Essaie cet exemple avec d'autres options!

"Putzmaschine" einsetzen
▮ Employer la "machine de nettoyage"

```
In[80]:= (* Old Form: Remove["Global`@*"] *)

In[81]:= Remove["Global`*"]
```