

Einführung in Sage

Einheit 2

Jochen Schulz

Georg-August Universität Göttingen



14. Januar 2010

- 1 Grundlagen von Sage
- 2 Symbolisches Rechnen I
- 3 Gleichungen

1 Grundlagen von Sage

2 Symbolisches Rechnen I

3 Gleichungen

Ein erstes Beispiel

```
>> f = x^2-3*x-18; solve(f==0,x)
```

```
[x == 6, x == -3]
```

```
>> solve(f<=0,x) ??
```

```
[-3, 6] union  
{I x + 3/2 | x in (-infinity, infinity)}
```

```
>> assume(x,Type::Real): solve(f<=0,x) ??
```

```
[-3, 6]
```

```
>> x:=1: f ??
```

```
-20
```

Betrachte:

```
>> f := x^2-3*x-18:
```

- Wie geht MuPAD mit der Unbekannten x um?
- Welchen Datentyp hat f ?
- Was kann ich mit f machen?

- **Bezeichner** sind Namen, wie z.B. x oder f . Sie können im mathematischen Kontext sowohl Variablen als auch Unbestimmte repräsentieren.
- Bezeichner sind aus Buchstaben, Ziffern und Unterstrich `_` zusammengesetzt.
- Sage unterscheidet zwischen Groß- und Kleinschreibung.
- Bezeichner dürfen nicht mit einer Ziffer beginnen

Beispiele für Bezeichner

- zulässige Bezeichner: x , f , $x23$, $_x_1$
- unzulässige Bezeichner: $12x$, $p\sim$, $x > y$, Das System

- Der **Wert** eines Bezeichners ist ein **Objekt** eines bestimmten **Datentyps**.
- Ein **Datentyp** ist durch seine Eigenschaften gegeben (in MuPAD **Domain-Typ**).
Beispiel: Natürliche Zahlen, rationale Zahlen, Bezeichner, Zeichenketten, ...
- Ein **Objekt** ist eine Instanz (Einheit) eines Datentyps.

Zuweisungsoperator :=

- Die Operation `bez:=wert` weist dem Bezeichner `bez` den Wert `wert` zu.
- Beispiele: `N:=5`, `f:= x^2-3*x-18`
- `:=` ist verkürzte Eingabe von `_assign`. `bez:=wert` ist äquivalent zu `_assign(bez,wert)`.
- Rückgabeparameter ist die rechte Seite.
- Warnung: Unterscheiden Sie stets zwischen dem Zuweisungsoperator `:=` und dem logischen Operator `=`.

Beispiele

```
>> N:=6
```

```
6
```

```
>> _assign(N,6)
```

```
6
```

```
>> f:=x+2*x*x-y
```

```
      2  
x - y + 2 x
```

```
>> y:= cos( (x:=PI)): x,y
```

```
PI, -1
```

Beispiele für Datentypen

```
>> domtype(5)
```

```
DOM_INT
```

```
>> f := x^2-3*x-18: domtype(f)
```

```
DOM_EXPR
```

```
>> domtype(x)
```

```
DOM_IDENT
```

```
>> f+f
```

```
      2  
2 x  - 6 x - 36
```

Einige Datentypen

Domain-Typ	Bedeutung	Beispiel
DOM_INT	ganze Zahlen	$-3, 0, 100$
DOM_RAT	rationale Zahlen	$7/11$
DOM_FLOAT	Gleitpunktzahl	0.123
DOM_COMPLEX	komplexe Zahlen	$1 + 3 * i$
DOM_IDENT	symbolische Bezeichner	x, f, y
DOM_EXPR	symbolische Ausdrücke	$x + y$
DOM_BOOL	logische Werte: TRUE FALSE, UNKNOWN	

Befehle im Umgang mit :=

- Viele Bezeichner sind vordefiniert und gegen Zuweisungen geschützt, z.B. `sin`, `limit`,...
- Aufheben des Schutzes: `unprotect(bezeichner)`
- Schützen von Bezeichnern: `protect(bezeichner,Error)`
- Liste aller definierten Bezeichner: `anames(All)`
- Liste aller vom Benutzer definierten Bezeichner: `anames(All,User)`
- Löschen von Zuweisungen: `delete bezeichner`

Beispiele

```
>> sin:=2
```

```
Error: Identifier 'sin' is protected [_assign]
```

```
>> N:=3
```

```
3
```

```
>> protect(N,Error): N:=5
```

```
Error: Identifier 'N' is protected [_assign]
```

```
>> unprotect(N): N:=3
```

```
3
```

Beispiel: Auswertung

```
>> f:=x*x-3*x-a
```

$$x^2 - 3x - a$$

```
>> a:=1: f
```

$$x^2 - 3x - 1$$

```
>> x:=1: f
```

$$-3$$

Beispiel: Auswertung

```
>> delete a: f
```

```
- a - 2
```

```
>> x:=5 :f
```

```
10 - a
```


- Der *Bezeichner* ist der Name einer Unbekannten.
- Die *Auswertung* eines Bezeichners benutzt alle aktuell bekannten Zuweisungen.
- Der *Wert* bezeichnet die Auswertung zum Zeitpunkt der Zuweisung.

- Auf interaktiver Ebene wertet MuPAD in der Regel vollständig aus.
- Maximale Auswertungstiefe wird durch die Konstanten **LEVEL** und **MAXLEVEL** gesteuert.
(Default: **LEVEL**= 100, **MAXLEVEL**=100)
- Zuerst wird **MAXLEVEL** geprüft; erst dann **LEVEL**.
- Ist **MAXLEVEL** erreicht, wird eine Fehlermeldung zurückgegeben.
- Die Auswertung wird bei **LEVEL** gestoppt, d.h. es wird keine Fehlermeldung zurückgegeben.

- Bei Aufrufen `%n` wird nicht ausgewertet. Ähnliches gilt bei Einträgen von Matrizen und Tabellen.
- Im Zusammenhang mit dem `$`-Operator wird nicht ausgewertet.
- Auswertungen können durch `eval` erzwungen werden.
- Auswertungen können durch `hold` unterbunden werden.
- Die Auswertungstiefe eines Bezeichners `a` kann gezielt durch den Befehl `level(a,n)` gesteuert werden (n Auswertungstiefe).

Beispiele I

```
>> a:=sin(b)
```

```
sin(b)
```

```
>> b:=0
```

```
0
```

```
>> a
```

```
0
```

Beispiele II

```
>> %3,eval(%3)
```

```
sin(b), 0
```

```
>> a,hold(a)
```

```
0, a
```

- Typische Operatoren sind $+$, $-$, $*$, $!$,...
- In MuPAD werden Objekte immer durch Funktionen miteinander verbunden.
- Operatoren sind in MUPAD als Funktionen realisiert.
- Für wichtige Operatoren gibt es die gewohnte Kurzschreibweise.
- Bei Kombination verschiedener Operatoren gelten die üblichen Regeln der Bindungsstärke (Punktrechnung vor Strichrechnung); Die Ordnung kann durch Klammersetzung geändert werden.

Wichtige mathematische Operatoren

Not.	Funktion	Erklärung
+	<code>_plus(a,b,c)</code>	Addition
—	<code>_subtract(a,b)</code>	Subtraktion
*	<code>_mult(a,b,c)</code>	Multiplikation
/	<code>_divide(a,b)</code>	Division
^	<code>_power(a,b)</code>	Potenz
!	<code>_fact(n)</code>	Fakultät
<i>div</i>	<code>_div(a,b)</code>	Quotient ohne Rest
<i>mod</i>	<code>_mod(a,b)</code>	Rest bei Division

Zerlegen von Objekten

- Viele Objekte sind zusammengesetzt. Ihre Bausteine heißen **Operanden**.
- Durch `nops(Objekt)` erhält man die Anzahl der Operanden.
- Durch `op(Objekt)` bzw. `op(Objekt,i)` erhält man alle Operanden bzw. den i -ten Operand.
- Mittels `has(Objekt,a)` kann untersucht werden, ob a ein Operand von `Objekt` ist.
- Die Befehle beziehen sich jeweils auf die automatisch vereinfachten Objekte.

Beispiele I

```
>> f := _plus(a,b,c)
```

```
a + b + c
```

```
>> nops(f), op(f), op(f,2)
```

```
3, a, b, c, b
```

```
>> op(f,0)
```

```
_plus
```

```
>> has(f,a), has(f,a+b)
```

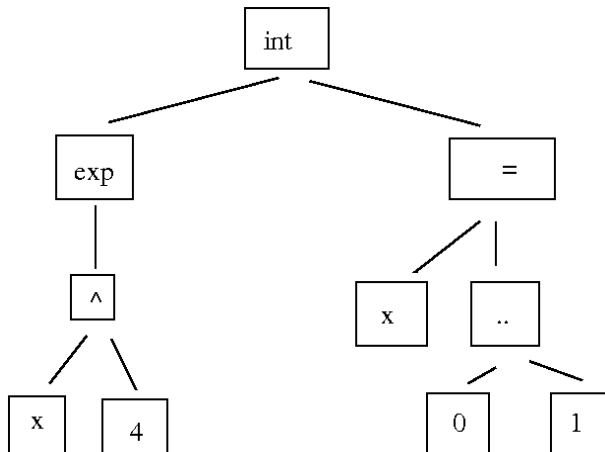
```
TRUE, FALSE
```

```
>> f:=x*z+3*x+sqrt(y):  
>> op(f), nops(op(f,2))
```

```
          1/2  
3 x, x z, y  , 2
```

Darstellungsbaum

```
>> a:=int(exp(x^4),x=0..1)
```



Beispiel

```
>> a:=int(exp(x^4),x=0..1):  
>> op(a)
```

$$\exp(x^4), x = 0..1$$

```
>> op(a,0)
```

int

```
>> op(op(a,1))
```

$$x^4$$

```
>> prog::expree(a)
```

Automatische Vereinfachung

MuPAD führt oft automatische Vereinfachungen durch. Ansonsten muß der Benutzer gezielt Vereinfachungen anfordern.

Beispiele:

```
>> sin(15*PI), exp(0)
```

0, 1

```
>> 2*infinity-5
```

infinity

```
>> y:=(-4*x+x^2+4)*(7*x+x^2+12)
```

$$(x^2 - 4x + 4)(7x^2 + 12x + 12)$$

1 Grundlagen von Sage

2 **Symbolisches Rechnen I**

3 Gleichungen

Manipulation von Ausdrücken

- Verbinden von Ausdrücken
- Vereinfachen
- Umformen
- Einsetzen der Unbekannten

Verbinden von Ausdrücken

Ausdrücke können beliebig addiert, subtrahiert, multipliziert und dividiert werden.

- Definition

```
>> f := x*x+3*x+y: g:=x-y:
```

- Potenz

```
>> f^g
```

$$(3x^2 + y + x)^{x-y}$$

Verbinden von Ausdrücken II

- Addition / Subtraktion

```
>> f+g, f-g
```

$$4x^2 + x^2, 2x^2 + 2y^2 + x^2$$

- Multiplikation/ Division

```
>> f*g, f/g
```

$$(x^2 - y^2) (3x^2 + y^2 + x^2), \frac{3x^2 + y^2 + x^2}{x^2 - y^2}$$

Durch `collect(Ausdruck, Unbestimmte)` wird der Ausdruck bzgl. der Unbestimmten sortiert.

```
>> f:=a*x^2+a*x+x^3+sin(x)+b*x+4*x+x*sin(x):  
>> collect(f,x)
```

$$x^3 + a x^2 + (a + b + \sin(x) + 4) x + \sin(x)$$

```
>> collect(f,[x,sin(x)])
```

$$x^3 + a x^2 + x \sin(x) + (a + b + 4) x + \sin(x)$$

Durch `combine(Ausdruck,Option)` wird der Ausdruck zusammengefaßt. Dabei werden mathematische Identitäten benutzt, die durch `Option` angegeben werden. Optionen sind `arctan`, `exp`, `ln`, `sincos`, `sinhcosh`. Ohne Angabe der Option werden nur die Potenzgesetze benutzt.

```
>> g:= sin(a)*cos(b):  
>> g=combine(g, sincos)
```

$$\cos(b) \sin(a) = \frac{\sin(a + b)}{2} + \frac{\sin(a - b)}{2}$$

expand

Ausmultiplizieren von Ausdrücken erfolgt durch

`expand(Ausdruck, f_1, f_2, ...)`.

`f_1`, `f_2` sind Ausdrücke, die **nicht** expandiert werden sollen.

```
>> expand((x+2)^4)
```

$$x^4 + 8x^3 + 24x^2 + 32x + 16$$

```
>> expand(sin(x+y))
```

$$\cos(x) \sin(y) + \cos(y) \sin(x)$$

Beispiele zu expand

```
>> f:=(x-y)^2+(x+y)^2: expand(f)
```

$$x^2 + 2xy + y^2$$

```
>> expand(f,x-y)
```

$$x^2 + y^2 + (x - y)^2 + 2xy$$

Der Befehl `factor(Ausdruck)` faktorisiert Polynome und Ausdrücke.

- MuPAD faktorisiert nur, wenn die resultierenden Koeffizienten rationale Zahlen sind.
- Auch anwendbar auf rationale Funktionen. Es wird ein gemeinsamer Hauptnenner gesucht.

Beispiel: factor

```
>> factor(x^2-2), factor(x^2-9/4)
```

$$x^2 - 2, \frac{(2x - 3)(2x + 3)}{4}$$

```
>> factor(2 - 2/(x^2-1))
```

$$\frac{2(x^2 - 2)}{(x - 1)(x + 1)}$$

Durch `normal(f)` wird eine 'Normalform' eines rationalen Ausdrucks f erzeugt.

```
>> normal(2 - 2/(x^2-1))
```

$$\frac{2x^2 - 4}{x^2 - 1}$$

Durch `partfrac(f)` wird ein rationaler Ausdruck in eine Summe rationaler Terme zerlegt, in denen jeweils der Zählergrad kleiner als der Nennergrad ist. (Partialbruchzerlegung)

```
>> f:=x^2/(x^2-1): f=partfrac(f)
```

$$\frac{x^2}{x^2 - 1} = \frac{1}{2(x - 1)} - \frac{1}{2(x + 1)} + 1$$

Durch `rewrite(Ausdruck, Option)` wird versucht, den Ausdruck so umzuformen, dass gewisse Funktionen aus dem Ausdruck eliminiert werden.

- Beispielsweise können `sin` und `cos` immer durch `tan` ausgedrückt werden (Option: `tan`).
- Optionen sind `diff`, `exp`, `fact`, `gamma`, `heavyside`, `ln`, `sign`, `sincos`, `sinhcosh`, `tan`.
- Man versucht die Ausdrücke mit Hilfe der in der Option genannten Funktion(en) auszudrücken.

Beispiele - rewrite I

```
>> rewrite(tan(x), sincos)
```

$\sin(x)$

$\cos(x)$

```
>> rewrite(tan(x), exp)
```

$$- \frac{I \exp(I x)^2 - I}{\exp(I x)^2 + 1}$$

Beispiele - rewrite II

```
>> rewrite(tan(x),sinhcosh)
```

$$\frac{i \sinh(-i x)}{\cosh(-i x)}$$

- Durch `simplify(f,target)` wird versucht den Ausdruck f zu vereinfachen. Optional können durch `target` spezielle Vereinfachungen angefordert werden.
- Mögliche `targets` sind `exp`, `ln`, `cos`, `sin`, `sqrt`, `logic` und `relation`.
- Die Optionen `logic` und `relation` dienen zur Vereinfachung von logischen Ausdrücken bzw. von Gleichungen und Ungleichungen.
- Alternativ zu `simplify(f,sqrt)` kann auch die Funktion `radsimp` verwendet werden.

Beispiele: Simplify I

```
>> f:=x/(x+y)+y/(x+y)-sin(x)^2-cos(x)^2:  
>> f=simplify(f)
```

$$\frac{x}{x+y} + \frac{y}{x+y} - \cos(x)^2 - \sin(x)^2 = 0$$

Beispiele: Simplify II

```
>> g:= sqrt(4+2*sqrt(3)):
```

```
>> g=simplify(g,sin)
```

$$2^{\frac{1}{2}} (3^{\frac{1}{2}} + 2)^{\frac{1}{2}} = 2^{\frac{1}{2}} (3^{\frac{1}{2}} + 2)^{\frac{1}{2}}$$

```
>> g=simplify(g)
```

$$(2^{\frac{1}{2}} (3^{\frac{1}{2}} + 2)^{\frac{1}{2}})^{\frac{1}{2}} = 3^{\frac{1}{2}} + 1 = 3^{\frac{1}{2}} + 1$$

- 1 Grundlagen von Sage
- 2 Symbolisches Rechnen I
- 3 Gleichungen**

- lineares Beispiel

```
>> Gleichungen := {x+y = 1, x-y = 1}:  
>> solve(Gleichungen)
```

```
{[x = 1, y = 0]}
```

- nichtlineares Beispiel

```
>> Gleichungen1:={x+y=1,(x-y)^2=1}:  
>> solve(Gleichungen1)
```

```
{[x = 0, y = 1], [x = 1, y = 0]}
```

- Der Operator `=` vergleicht zwei Objekte.
- `a=b` ist wahr (richtig), wenn `a` und `b` die gleichen Auswertungen besitzen (und vom gleichen Typ sind).
- Zur Überprüfung von Aussagen gibt es die Funktion `bool(Ausdruck)`. Sie liefert als Ergebnis `TRUE` oder `FALSE`.
- Die inverse Operation zu `'='` ist `'<>'`, also `a<>b` ist `TRUE`, falls `a` nicht gleich `b` ist.

Beispiele: Vergleiche I

```
>> bool(4-3=1)
```

```
TRUE
```

```
>> bool(4*x=x); x:=0: bool(4*x=x)
```

```
FALSE TRUE
```

```
>> bool(x=0); bool(x<>0)
```

```
TRUE FALSE
```

Beispiele: Vergleiche II

```
>> bool(0.5=1/2)
```

```
FALSE
```

```
>> domtype(1.0), domtype(1)
```

```
DOM_FLOAT, DOM_INT
```

- Solve ist der universelle Befehl zum Lösen von Gleichungen und Ungleichungen und auch Differentialgleichungen.
- Der Befehl ist von der Form `solve(Gleichungen, Variablen)`.
- Gleichungen kann ein System von Gleichungen sein.
- Variablen gibt an, wonach aufgelöst wird.
- Bei einzelnen Gleichungen wird der Lösungswert zurückgegeben. Bei mehreren Gleichungen wird ein System äquivalenter Gleichungen zurückgegeben.
- Weitere Optionen werden später erklärt.

Beispiele - Solve I

```
>> solve(x^2+x=y/4,x)
```

$$\left\{ \begin{array}{l} \frac{(y+1)^{1/2}}{2} - \frac{1}{2}, \frac{(y+1)^{1/2}}{2} - \frac{1}{2} \end{array} \right\}$$

Beispiele - Solve II

```
>> assume(x>0): solve(x^2+x=y/4,y)
```

$$\{4x^2 + 4x\}$$

```
>> solve({x^2-y^2=0},{x,y})
```

$$\{[x = z, y = z], [x = -z, y = z]\}$$

```
>> solve({x^2-y^2=0,x+y=1},{x,y})
```

$$\{[x = 1/2, y = 1/2]\}$$