2-1-Algebarske_manipulacije-S

Verzija 1.0, 2010-09-29, Krešimir Kumerički

Algebarske manipulacije

Moć paketa za simboličku matematiku, poput Sage-a, leži u sposobnosti manipulacije simboličkim izrazima. Kao prvo, potrebno je na slijedeći način deklarirati varijable koje namjeravamo koristiti u simboličkim izrazima. (Eksplicitno deklariranje simboličkih varijabli je moguće izbjeći pozivanjem funkcije automatic_names(True), ali to nećemo koristiti.)

```
var('a b c x y z t')
```

```
(a, b, c, x, y, z, t)
```

Pomoću ovih varijabli sad izgrađujemo simboličke izraze:

```
(b+a)^3
```

```
(a + b)^3
```

Sage ne provodi skoro nikakve operacije na izrazima dok ga eksplicitno ne instruiramo. Recimo da želimo razviti ("ekspandirati") gornji izraz koristeći binomni teorem. Za to služi funkcija `expand()`

```
expand((a+b)^3)
```

```
a^3 + 3*a^2*b + 3*a*b^2 + b^3
```

Postoji i alternativni način primjene ove funkcije (i većine drugih):

```
((a+b)^3).expand()
```

```
a^3 + 3*a^2*b + 3*a*b^2 + b^3
```

Na prvi pogled drugi način je možda zbunjujući pa je važno objasniti čemu služi. U prvom pristupu expand doživljavamo kao funkciju ili operaciju, dok je izraz (a+b)^3 njen argument odnosno operand. To je način razmišljanja svojstven standardnom proceduralnom ili pak tzv. funkcionalnom

svojstven standardnom proceduralnom ili pak tzv. funkcionalnom programiranju. (O raznim vrstama programiranja bit će više riječi kasnije.) U drugom pristupu izraz (a+b)^3 treba pak doživljavati kao *objekt*, u smislu tzv. objektno-orijentiranog (OO) programiranja, a `expand()` je tzv. *metoda* što je naziv za funkciju koja je pridružena tipu objekta na koji djeluje. To onda omogućuje da metode istog imena rade različite stvari s različitim objektima (tzv. *polimorfizam*). Kako je python OO jezik, takva sintaksa se obilato koristi u Sage-u i brojne funkcije se ni ne mogu koristiti na prvi način. Jedna od prednosti takvog pristupa je da elegantno možemo saznati popis svih funkcija koje rade nešto smisleno sa zadanim objektom, i to tako da nakon što stavimo točkicu "." poslije objekta stisnemo TAB tipku. Dobit ćemo popis svih metoda tog objekta. Ovo međutim ne funkcionira s netom upisanim izrazom u trenutnoj ćeliji, već samo s ranije definiranim izrazima (objektima) kojima smo pridjelili ime. Pridjeljivanje imena objektima se izvodi znakom jednakosti i korisno je ne samo zbog navedenog razloga već i inače radi lakšeg baratanja izrazima i kasnijeg referiranja na iste.

```
i1 = (b+a)^3
```

Da bi saznali što pojedina metoda radi, kliknemo na istu ili je upišemo, dodamo upitnik i stisnemo TAB. Pri upotrebi metode ne smije se zaboraviti na zagrade, koje su često prazne, ali nekad sadrže opcionalne argumente kojima modificiramo ponašanje metode. Ukoliko zaboravimo zagrade Sage ne poziva funkciju već samo ispisuje njeno puno ime poput "<*metoda expand pridružena objektu `Expression` koji je pohranjen na toj i toj adresi*>".

```
i1.expand
```

```
<built-in method expand of sage.symbolic.expression.Expression
object at 0x4a59200>
```

Tek zagrade daju zahtjev interpreteru da dotičnu metodu i pozove tj. izvrši:

```
i1.expand()
```

```
a^3 + 3*a^2*b + 3*a*b^2 + b^3
```

Naravno ovakve jednostavne izraze možemo razviti i na ruke, dok računalo blista kad radi s velikim izrazima (sve dok stanu u memoriju računala)

```
i2 = (a +2*b + 3*c)^3 * (x+y)^3
i2.expand()
```

```
a^3*x^3 + 3*a^3*x^2*y + 3*a^3*x*y^2 + a^3*y^3 + 6*a^2*b*x^3 +
18*a^2*b*x^2*y + 18*a^2*b*x*y^2 + 6*a^2*b*y^3 + 9*a^2*c*x^3 +
27*a^2*c*x^2*y + 27*a^2*c*x*y^2 + 9*a^2*c*y^3 + 12*a*b^2*x^3 +
36*a*b^2*x^2*y + 36*a*b^2*x*y^2 + 12*a*b^2*y^3 + 36*a*b*c*x^3 +
108*a*b*c*x^2*y + 108*a*b*c*x*y^2 + 36*a*b*c*y^3 + 27*a*c^2*x^3
+
81*a*c^2*x^2*y + 81*a*c^2*x*y^2 + 27*a*c^2*y^3 + 8*b^3*x^3 +
24*b^3*x^2*y + 24*b^3*x*y^2 + 8*b^3*y^3 + 36*b^2*c*x^3 +
108*b^2*c*x^2*y + 108*b^2*c*x*y^2 + 36*b^2*c*y^3 + 54*b*c^2*x^3
+
162*b*c^2*x^2*y + 162*b*c^2*x*y^2 + 54*b*c^2*y^3 + 27*c^3*x^3 +
81*c^3*x^2*y + 81*c^3*x*y^2 + 27*c^3*y^3
```

Potenciranjem i razvijanjem gornjeg izraza dobivamo izraz od 550 članova ...

```
i3 = expand(i2^3)
len(i3)
```

```
550
```

... kojeg Sage s lakoćom faktorizira:

```
factor(i3)
```

```
(x + y)^9*(a + 2*b + 3*c)^9
```

Često je korisno izraz organizirati kao polinom u nekoj varijabli. Za to služi funkcija `collect()`:

```
i4=i2.expand().collect(y)
i4
```

```
a^3*x^3 + 6*a^2*b*x^3 + 9*a^2*c*x^3 + 12*a*b^2*x^3 +
36*a*b*c*x^3 +
27*a*c^2*x^3 + 8*b^3*x^3 + 36*b^2*c*x^3 + 54*b*c^2*x^3 +
27*c^3*x^3
+ (a^3 + 6*a^2*b + 9*a^2*c + 12*a*b^2 + 36*a*b*c + 27*a*c^2 +
8*b^3
+ 36*b^2*c + 54*b*c^2 + 27*c^3)*y^3 + 3*(a^3*x + 6*a^2*b*x +
9*a^2*c*x + 12*a*b^2*x + 36*a*b*c*x + 27*a*c^2*x + 8*b^3*x +
36*b^2*c*x + 54*b*c^2*x + 27*c^3*x)*y^2 + 3*(a^3*x^2 +
6*a^2*b*x^2 +
9*a^2*c*x^2 + 12*a*b^2*x^2 + 36*a*b*c*x^2 + 27*a*c^2*x^2 +
8*b^3*x^2
+ 36*b^2*c*x^2 + 54*b*c^2*x^2 + 27*c^3*x^2)*y
```

```
len(i4)
```

```
13
```

Za dobiti koeficijent uz neku potenciju neke varijable koristi se funkcija `coefficient()`. Npr, koeficijent uz a^9 jest

```
i3.coefficient(a, 9)
```

```
x^9 + 9*x^8*y + 36*x^7*y^2 + 84*x^6*y^3 + 126*x^5*y^4 +
126*x^4*y^5
+ 84*x^3*y^6 + 36*x^2*y^7 + 9*x*y^8 + y^9
```

`collect()` ne pojednostavljuje koeficijente.

(*) To se može postići na jedan od dva slijedeća zaobilazna načina koja međutim uključuju naprednije koncepte pa ih privremeno zanemarite.

```
sum(i4.coefficient(y, a).factor()*y^a for a in (0..3))
```

```
(a + 2*b + 3*c)^3*x^3 + 3*(a + 2*b + 3*c)^3*x^2*y + 3*(a + 2*b +
3*c)^3*x*y^2 + (a + 2*b + 3*c)^3*y^3
```

```
sum([it.factor()*y^eksp for it,eksp in i4.coefficients(y)])
```

```
(a + 2*b + 3*c)^3*x^3 + 3*(a + 2*b + 3*c)^3*x^2*y + 3*(a + 2*b +
3*c)^3*x*y^2 + (a + 2*b + 3*c)^3*y^3
```

Najsveobuhvatnija funkcija za pojednostavljivanje simboličkih izraza je `simplify_full()`:

```
i4 = a/(1-a) + a/(1+a); i4
```

```
-a/(a - 1) + a/(a + 1)
```

```
i4.simplify_full()
```

```
-2*a/(a^2 - 1)
```

`simplify_full()` je kompozicija elementarnijih funkcija za pojednostavljivanje izraza. Jedna od tih elementarnijih funkcija je `simplify_trig()` koja pri pojednostavljivanju rabi samo trigonometrijske identitete.

```
(sin(x)^4 + 2*sin(x)^2*cos(x)^2 + cos(x)^4).simplify_trig()
```

```
1
```

Nažalost, ove funkcije se ponašaju suviše nonšalantno pri pojednostavljivanju izraza koji uključuju multifunkcije pa izrazi prije i poslije pojednostavljenja nisu doslovno ekvivalentni, kao na slijedećem primjeru, u kojem upoznajemo i važnu metodu subs() koja služi za uvrštavanje vrijednosti varijabli i druge supstitucije u izrazima

```
i5 = log(a) + log(b); print i5
i5.subs(a=-1,b=-1)
```

```
log(a) + log(b)
2*I*pi
```

```
print i5.simplify_full()
i5.simplify_full().subs(a=-1,b=-1)
```

```
log(a*b)
0
```

Zadatak 2-1.1: Uzmite izraz $(a+b)((c+yx)x+tx^2)$ i algebarskim manipulacijama natjerate Sage da ga prikaže u slijedećim oblicima:

1. $(a+b)(tx+xy+c)x$
2. $atx^2+ax^2y+btx^2+bx^2y+acx+bcx$
3. $(t+y)(a+b)x^2+(a+b)cx$ (ovo preskočiti!)

Zadatak 2-1.2: Koristeći algebarske manipulacije pokažite da vrijedi

$$\frac{\sin^3 x + \cos^3 x}{\sin x \ + \cos x} = 1 - \sin x$$

Pazite na sintaksu: $\sin^3 x \cos x$ se unosi kao sin(x)^3!