Подстановки и связанные с ними операции

Подстановки являются наиболее часто встречающейся операцией, которую приходится выполнять в ходе символьных преобразований. Система Maple поддерживает несколько различных видов подстановок.

Простейшие подстановки можно выполнить с помощью команды *subs*. Первым аргументом этой команды является уравнение. В левой части этого уравнения можно указать имя некоторой переменной, а в правой – значение, которое следует вместо него подставить.

```
> subs(x=3,x/(x^2+1));
```

$$\frac{3}{10}$$

Данное действие равносильно выполнению команды *eval*. Если подстановка выполняется только для вычисления значения выражения при некотором значении аргумента, то целесообразно использовать именно *eval*.

```
> eval(x/(x^2+1),x=3);
```

$$\frac{3}{10}$$

В правой части уравнения может находиться любое выражение, допустимое синтаксисом системы Maple

```
> subs(a=X^2+1,sqrt(a-1));
```

$$\sqrt{X^2}$$

Отличительной особенностью команды subs является блокирование автоупрощений.

```
> subs(x=Pi,sin(x)+cos(2*x));
```

$$\sin(\pi) + \cos(2\,\pi)$$

Как уже отмечалось выше, если необходимо получить численное значение, то используйте команду *eval*.

```
> eval(sin(x)+cos(2*x),x=Pi);
```

$$1$$

Достоинством команд *subs* и *eval* является то, что с их помощью производятся расчёты, не изменяющие глобальный контекст.

```
> subs(x=2,(x+sqrt(3))/(sqrt(x)+sqrt(x+sqrt(3)))+(x-sqrt(3))/
          (sqrt(x)-sqrt(x-sqrt(3))));
```

$$\frac{2+\sqrt{3}}{\sqrt{2}+\sqrt{2+\sqrt{3}}} + \frac{2-\sqrt{3}}{\sqrt{2}-\sqrt{2-\sqrt{3}}}$$

При этом переменной *x* не присваивается никаких значений и она может использоваться в других символьных операциях.

```
> x;
```

$$x$$

Очевидно, что команда *subs* может являться аргументом других команд. Это позволяет приводить результаты к виду, который не достигается за счёт автоупрощений.

```
> normal(subs(y=(sqrt(3)-1)/2,(1-y)*(y+2)/(y^2*(y+1)^2)),
        expanded);
```

$$6$$

Благодаря блокированию автоупрощений в некоторых ситуациях удаётся привести результат к виду, который по умолчанию «не устраивает» Maple. В этом примере демонстрируется, как можно разместить два радикала под одним знаком корня.

```
> expr := 2*2^(1/6)*3^(1/3);
```

$$expr := 2\,2^{1/6}\,3^{1/3}$$

Команда *combine* умеет объединять только одинаковые корни. Мы достигаем этого

при помощи подстановки.

```
> combine(subs(3=9^(1/2),expr),radical);
```

$$2\,18^{1/6}$$

В левой части уравнения допускаются и более сложные языковые конструкции. Достаточно типичной является ситуация, когда известно значение некоторой функции. Предположим, что нам требуется вычислить значение выражения

```
> expr:=2-13*cos(2*alpha)+sin(2*alpha)^(-1);
```

$$expr := 2 - 13\cos(2\,\alpha) + \frac{1}{\sin(2\,\alpha)}$$

при условии, что известно значение

```
> equ:=cot(alpha)=-1/5;
```

$$equ := \cot(\alpha) = -\frac{1}{5}$$

Задача решается с помощью подстановки путём преобразования исходного выражения к функции котангенс.

```
> subs(equ,expand(convert(expr,cot)));
```

$$\frac{57}{5}$$

В ходе выполнения подстановки все вхождения левой части уравнения в исходном выражении заменяются его правой частью.

```
> expr:=(a+b+c)^2/(x+y)-cos(a+b+c)*exp(x-y);
```

$$expr := \frac{(a+b+c)^2}{x+y} - \cos(a+b+c)\,e^{x-y}$$

```
> subs(a+b+c=x-y,expr);
```

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

Следует, однако, отметить, что замена выполняется только для тех фрагментов выражения, которые являются операндами, то есть могут быть извлечены при помощи функции *op*. Подобное поведение указывает на то, что при выполнении команды структура синтаксического дерева не меняется. Замене подвергаются либо листья дерева, либо целые поддеревья.

В этом нетрудно убедиться путём незначительной модификации предыдущего примера.

```
> subs(a+b=x-y,expr);
```

$$\frac{(a+b+c)^2}{x+y} - \cos(a+b+c)\,e^{x-y}$$

Оказывается *expr* не содержит $a+b$ в качестве операнда!

В документации по системе Maple алгоритм работы команды *subs* принято называть «синтаксической» подстановкой. Этот алгоритм работает чрезвычайно быстро, но может вызывать множество вопросов у пользователей, не знакомых с логикой команды *op*. Для подстановок, способных трансформировать синтаксическое дерево, в системе имеются другие команды.

Команда *subs* способна выполнить сразу несколько подстановок. По этой причине у неё «плавающее» количество аргументов. Выражение, в котором будут выполнены подстановки, должно быть последним аргументом.

```
> subs(a=x,b=y,c=x+y,expr);
```

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

Если одна и та же переменная или же выражение несколько раз встречается в левых частях уравнений, то будет использована первая подстановка. В следующем примере вместо *a*

будет подставлена переменная *x*, а не *z*.

```
> subs(a=x,b=y,a=z,expr);
```

$$\frac{(x+y+c)^2}{x+y} - \cos(x+y+c)\, e^{x-y}$$

Если в команде *subs* указано несколько подстановок, то они выполняются последовательно слева направо. Данную закономерность удобно проиллюстрировать примером, в котором некоторая переменная появляется в правой части уравнения одной из начальных подстановок, а в последующих подстановках эта же переменная возникает уже в левых частях уравнений.

```
> subs(a=x-y,b=x+y,x=y+c,expr);
```

$$\frac{(2y+3c)^2}{2y+c} - \cos(2y+3c)\, e^{c}$$

В результате выполнения этого примера конечный результат оказался зависящим только от *y* и *c*. То же самое можно получить, если воспользоваться вложенной командой *subs*.

```
> subs(x=y+c,subs(a=x-y,b=x+y,expr));
```

$$\frac{(2y+3c)^2}{2y+c} - \cos(2y+3c)\, e^{c}$$

Несколько подстановок можно объединять в списки или же множества. В этом случае они выполняются одновременно за один просмотр дерева. Сравните результаты выполнения двух команд. Сначала последовательное выполнение подстановок,

```
> subs(a=b,b=a,x=y+c,expr);
```

$$\frac{(2a+c)^2}{2y+c} - \cos(2a+c)\, e^{c}$$

а затем параллельное.

```
> subs({a=b,b=a},x=y+c,expr);
```

$$\frac{(a+b+c)^2}{2y+c} - \cos(a+b+c)\, e^{c}$$

В документации ничего не сказано о том, есть ли какая-либо разница между использованием списка или множества.

Так как подстановка приводит к замене операнда сразу во всех частях выражения, то это может создавать проблемы при обработке особых случаев. Например, что может случиться если функция доопределена в особой точке.

```
> F:=piecewise(x=0,1,tan(x)/x);
```

$$F := \begin{cases} 1 & x = 0 \\ \dfrac{\tan(x)}{x} & otherwise \end{cases}$$

```
> subs(x=0,F);
Error, numeric exception: division by zero
```

Как уже отмечалось ранее, в таких ситуациях следует использовать команду *eval*, которая вернёт правильный результат.

```
> eval(F,x=0);
```

$$1$$

Рассмотренные выше примеры свидетельствуют о том, что после выполнения подстановки возникает новое выражение. Однако некоторые объекты, основанные на структуре данных rtable, могут быть модифицированы непосредственно. К числу таких объектов относятся часто используемые в приложениях экземпляры типов данных Array, Matrix или же Vector.

Предположим, что нам задан вектор

```
> vv:=Vector([seq(a^j,j=1..5)]);
```

$$vv := \begin{bmatrix} a \\ a^2 \\ a^3 \\ a^4 \\ a^5 \end{bmatrix}$$

Если команда *subs* выполняется в обычном режиме, то порождается новый вектор.

```
> subs(a=1,vv);
```

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Исходный вектор остаётся без изменений.

```
> vv;
```

$$\begin{bmatrix} a \\ a^2 \\ a^3 \\ a^4 \\ a^5 \end{bmatrix}$$

Добавим теперь к команде *subs* опцию *inplace*, которая должна быть помещена внутрь квадратных скобок.

```
> subs[inplace](a=1,vv);
```

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Отобразив на экран исходный вектор *vv*, мы видим, что его компоненты больше не содержат степеней переменной *a*.

```
> vv;
```

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Опция *inplace* не поддерживается для объектов, в основе которых лежит структура данных table. Объектами таких типов манипулирует популярная библиотека linalg.

```
> vvv:=vector([seq(a^j,j=1..5)]);
```

$$vvv := \left[\begin{array}{ccccc} a & a^2 & a^3 & a^4 & a^5 \end{array} \right]$$

Несмотря на то, что при вызове команды сообщение об ошибке не возникает

```
> subs[inplace](a=1,vvv);
```

$$vvv$$

исходный вектор не изменяется.

```
> evalm(vvv);
```

$$\left[\begin{array}{ccccc} a & a^2 & a^3 & a^4 & a^5 \end{array} \right]$$

В силу того, что объекты типа table подчиняются особым правилам вычисления, то вызов *subs* в стандартной форме также не выполняет никаких преобразований.

```
> subs(a=1,vvv);
```

$$vvv$$

Команда *eval* срабатывает и в этом случае.

```
> eval(vvv,a=1);
```

$$\left[\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array} \right]$$

Если всё-таки требуется выполнять подстановки в объектах типа vector или matrix, то соответствующее выражение следует сделать аргументом команды *evalm*.

```
> subs(a=1,evalm(vvv));
```

$$\left[\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array} \right]$$

Предположим, что задана следующая матрица

```
> M:=matrix(3,3,(i,j)->(i+j) mod 2);
```

$$M := \left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{array} \right]$$

и мы хотим заменить все её нулевые элементы переменной *x*. Тогда следует выполнить команду

```
> subs(0=x,evalm(M));
```

$$\left[\begin{array}{ccc} x & 1 & x \\ 1 & x & 1 \\ x & 1 & x \end{array} \right]$$