17 - La fonction convert

Il est souvent utile de convertir une expression de Maple en *autre chose*.

Il y a essentiellement deux types de conversions :

- Les *conversions de type*, qui consistent le plus souvent à garder les opérandes de l'expression, mais à modifier sa racine : on peut ainsi transformer une liste en un ensemble, ou un ensemble en une somme.
- Les *conversions de forme*, qui consistent, tout en conservant le type de l'expression, à en modifier l'apparence ou les composants.

Dans les deux cas, l'instruction `convert`, qui possède un grand nombre d'options, est l'outil principal. On verra ici une liste de conversions possibles, en se limitant aux plus utiles.

1. Conversions de type

- Voici comment convertir un objet en une somme ou en un produit.

```
> convert({a,b,c},`+`), convert([a,b,c],`*`), convert(f(a,b,c),`+`) ;
```

$$a+b+c,\ a\,b\,c,\ a+b+c$$

Rappelons que le quotient a/b est codé en mémoire $a*b^{-1}$.

```
> convert(a/b,`+`), convert(a+b+c,`*`) ;
```

$$a+\frac{1}{b},\ a\,b\,c$$

- On peut aussi convertir un objet (table, tableau, expression) en un ensemble.

```
> convert(a*b*c,set), convert(a*b+c,set), convert([a,b,c,a,c],set) ;
```

$$\{a,\ b,\ c\},\ \{a\,b,\ c\},\ \{a,\ b,\ c\}$$

```
> t :=array([[2,5,6],[5,7,2]]) : convert(t,set) ;
```

$$\{5,\ 6,\ 7,\ 2\}$$

```
> t :=table([a=x+1,b=y-2,3=x+1]) : convert(t,set) ;
```

$$\{x+1,\ y-2\}$$

- Voici comment transformer un objet (table, vecteur, expression) en une liste.

```
> convert(a+b*c-d,list), convert({a,b,a,c,d,b,a},list) ;
```

$$[a,\ b\,c,\ -d],\ [a,\ b,\ d,\ c]$$

```
> t :=table([a=x+1,b=y-2,3=x+1]) : convert(t,list) ;
```

$$[x+1,\ y-2,\ x+1]$$

```
> t :=array([2,5,6,7,2]) : convert(t,list) ;
```

$$[2,\ 5,\ 6,\ 7,\ 2]$$

Un tableau de dimension supérieure à 1 ne peut pas être converti en une liste.

```
> t :=array([[2,7,1],[3,8,6]]) : convert(t,list) ;
Error, (in convert/list) can't convert array of dimension > 1
```

– On peut convertir un tableau de dimension supérieure à 1 en une liste de listes.

```
> t :=array([[2,7,1],[3,8,6]]) : convert(t,listlist) ;
```

$$[[2,\ 7,\ 1],\ [3,\ 8,\ 6]]$$

– Inversement, on peut convertir une liste en un tableau (avec des indices débutant à 1).

```
> convert([1,5,8,1,3],array), convert([[2,5,7],[8,4,1]],array) ;
```

$$[1,\ 5,\ 8,\ 1,\ 3],\ \begin{bmatrix} 2 & 5 & 7 \\ 8 & 4 & 1 \end{bmatrix}$$

– Voici comment convertir une liste en une liste de listes [*val*,*mult*] où *val* est une entrée de la liste et *mult* sa "multiplicité".

```
> convert([a,b,c,a,b,d,a,b,c,a,a],multiset) ;
```

$$[[a,\ 5],\ [b,\ 3],\ [d,\ 1],\ [c,\ 2]]$$

– La même syntaxe permet de convertir une expression de type produit en une liste de listes [*fact*,*expo*] où *fact* est un facteur de l'expression et *expo* son exposant.

```
> convert(a^3*b*(c+1)^5*sqrt(d)*a,multiset) ;
```

$$[[a,\ 4],\ [b,\ 1],\ [c+1,\ 5],\ [d,\ \frac{1}{2}]]$$

– Voici comment convertir une relation en une égalité, ou une relation du type < ou <=.

```
> restart : convert(x^2+x=2,lessthan), convert(x^2+x=2,lessequal),
  convert(x^2-y^2<z^3,equality) ;
```

$$x^2+x<2,\ x^2+x\leq 2,\ x^2-y^2=z^3$$