

Maple Lecture 7. Types, Attributes, Properties, and Units

The material in this lecture corresponds to [1, Sections 3.4-7]. See also [2].

7.1 Basic Data Types

Without explicit declarations, Maple knows how to figure out the data type of a variable.

```
[> a := 3; whattype(a);
[> b := 3.1415; whattype(b);
[> c := convert(b,fraction); whattype(c);
[> convert(b,fraction,3); # use only three digits
```

The fourth basic data type is a string:

```
[> d := convert(c,string); whattype(d);
[> parse(d); # convert back
```

Integers, floats, fractions, and strings are four basic data types, similar to what you may encounter in any programming language, except that Maple's arithmetic is software driven. To convert to hardware floats

```
[> b1 := convert(b,float[]); whattype(b1);
```

The difference is not directly noticeable, which may lead to errors as adding two hardware floats may lead to a sum more accurate than when computing in Maple's default 10-digits precision.

To get an overview of the basic types:

```
[> ?type,surface;
[> z := 1 + I; whattype(z);
```

The latter is new since Maple 7, earlier versions of Maple did not recognize `complex` as a data type.

Maple has types for expressions:

```
[> whattype(x+y); whattype(x/y); whattype(x^3);
```

We can get an accurate picture of the internal representations of expressions:

```
[> p := x*y + x;
```

We get access to the operands of an expression via the `op` command:

```
[> op(p); # select the operands
[> nops(p); # number of operands
[> p1 := op(1,p); # select first operand
[> op(p1); # operands of the first operand
[> op(2,p1); # second operand of p1
[> op([1,2],p); # is 2nd operand of 1st operand of p
[> op(0,p); # type of p
[> op(0,p1); # type of first operand of p
```

We can turn an expression sequence into a sum:

```
[> convert(p1,'+');
```

which does not affect the encapsulating variable `p`:

```
[> p;
```

With some key strokes we can build complicated expressions in Maple:

```
[> s := sum(x^i, 'i'=1..10);
[> op(s);
[> map(sin,s);          # apply the sine function to all operands
[> l := [op(s)];        # makes list of operands
[> ls := {op(s)};      # makes set of operands
```

Besides the order of elements, the main difference between lists and sets is that a set cannot contain any duplicates. To remove duplicates from a list, one can convert to a set, and then back to a list.

7.2 Attributes

Extra information can be added to a variable with `setAttribute`. This is similar to the struct data type in C. As example, we take one root of a polynomial as the variable we wish to set attributes for.

```
[> p := x^3 - 2;
[> a := evalf(RootOf(p,1));
```

Since `a` is just a floating-point number, we can add the meaning of the variable as a string attribute:

```
[> a := setattribute(a,"approximation for 2^(1/3)");
[> attributes(a);      # query the attributes of a variable
```

It is likely that in the near future we wish to increase the precision of the approximation. Therefore we must also add the original equation as attribute. Here is how we do it:

```
[> a := setattribute(a,attributes(a),'p' = x^3 - 2);
[> attributes(a);
```

Now you see that `attributes(a)` returns a sequence, we select the equation as follows :

```
[> equ := attributes(a)[2];
[> q := rhs(equ);
```

Here is how we achieve the increase of accuracy of the root :

```
[> b := setattribute(evalf(RootOf(q,1),30),attributes(a));
[> attributes(b);
```

7.3 Properties

Properties are similar to attributes, in the sense that they add extra information to the variable, but different as Maple takes properties into account during calculations:

```
[> assume(m,odd);      # impose an assumption
[> about(m);           # query the assumptions
[> n := (-1)^m;
```

In the output we see a flag after the `m` symbol, i.e.: m^{\sim} . This indicates there is an assumption on `m`:

```
[> simplify(n);
[> additionally(m>2); # add one more assumption
[> about(m);
```

7.4 Units

Mathematicians mostly calculate with dimensionless numbers. Calculations in chemistry, physics, and engineering are usually accompanied by units. Neglecting or confusing units (e.g., between inches and meters) can lead to disasters. Maple knows how to convert between units:

```
[> convert(1,units,miles,km);
[> evalf(%);
```

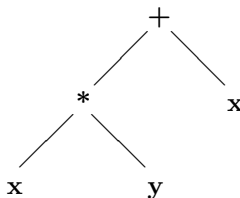
So Maple can tell us exactly how far one mile is in the metric system. We can calculate with units, adding three miles to two kilometers:

```
[> with(Units[Standard]):      # must load this package first
[> a := 3*Unit(miles); b := 2*Unit(km);
[> s := a + b;
```

We see that Maple automatically converts to the metric system.

7.5 Assignments

1. What is the difference between the type `rational` and the type `fraction`? In particular:
 - (a) Is every variable of type `fraction` a rational variable? If yes, justify your answer. If no, give an example of a variable of type `fraction` which is not of type `rational`.
 - (b) Is every rational variable a fraction? If yes, justify your answer. If no, give an example of a variable which is of type `rational`, but not of type `fraction`.
2. With the command `op` we can build an *expression tree*. For $\mathbf{x*y + x}$, this tree looks like



Draw the expression tree for the polynomial $34 + 18x^2 - 13xy$.

Give all Maple commands with the output to justify the position of the nodes in the tree.

3. Successively transform the expression $x + y + z$ into $x*y*z$, and $[x, y, z]$.
4. Give one (exactly one!) Maple command to make the expression

$$\sum_{i=1}^{20} \sin(x^i).$$

once as formal sum, and once as sum of sines. So your answer consists of two Maple commands: once the inert and once the actual form of the command.

5. Consider the expression $\sqrt{x^2}$. Under which assumption does this simplify to x ? Illustrate how you can add this assumption to let Maple do the simplification.

References

- [1] A. Heck. *Introduction to Maple*. Springer-Verlag, third edition, 2003.
- [2] D.I. Schwartz. *Introduction to Maple 8*. Prentice Hall, 2003.