

7. Basic Data Types; Attributes and Properties

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 := 3.4; whattype(b);
[> c := convert(b,fraction); whattype(c);
[> d := convert(c,string); whattype(d);
```

Integers, floats, fractions, and strings are four basic data types, similar to what you may encounter in a programming language. But Maple offers far more elaborate structured data 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;
[> dismantle(p);
```

The outcome of this last command is displayed like a tree, but notice: *Maple stores every object only once!* So the data structure for the internal representation of a Maple expression is a Directed Acyclic Graph (DAG), see Figure 1.

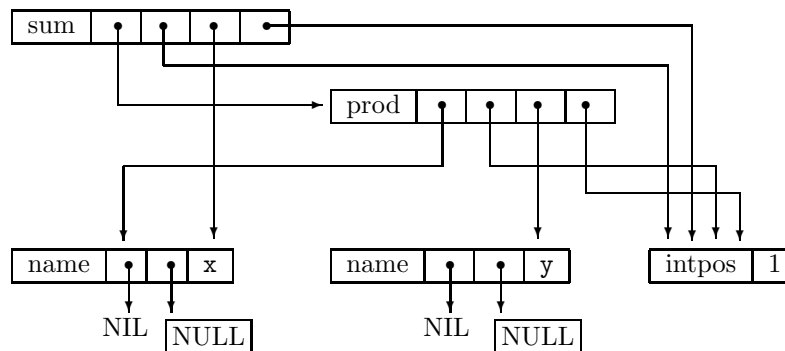


Figure 1: The Directed Acyclic Graph represents the expression $x*y + x$.

We will study the creation of DAGs for general expressions in the second part of the course. While the command `dismantle` gives us a view to the full low-level representation of an expression, we get access to the operands of an expression via the `op` command:

```
[> op(p); # select the operands
[> p1 := op(1,p);
[> op(p1);
[> whattype(%);
```

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);
[> 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 Assignments

1. Successively transform the expression $x + y + z$ into $x*y*z$, and $[x,y,z]$.
2. 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.

3. 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.