Automatic Simplification

The term *automatic* (or *default*) *simplification* refers to the mathematical simplification rules that are applied to an expression during the evaluation process. In computer algebra systems, this usually involves the "obvious" simplification rules from algebra and trigonometry that remove extraneous symbols from an expression and transform it to a standard form.

The MPL dialogue in Figure 1 illustrates some of these obvious simplifications. Example <1> shows a simplification that involves the sum of rational numbers. Example <2> shows that automatic simplification combines numerical coefficients of like terms. The next example <3> illustrates a similar simplification in which integer exponents of the common base x are combined. Example <4> illustrates some simplification rules that involve the integers 0 and 1. Notice that after evaluation, the x^3 term appears at the right end of the expression. This reordering, which is an application of the commutative law of addition, serves to put the result in a more readable form and, in some cases, contributes to the simplification process[1]. The next example <5> illustrates this point. To simplify this expression, the term $3 * y * x$ is first reordered (using the commutative law for multiplication) to $3 * x * y$ after which the coefficients of the two like terms are combined. Examples <6>, <7>, and <8> illustrate automatic simplification rules that involve known functions, while Examples <9> and <10> illustrate simplification rules that involve reserved symbols.

Examples <11> and <12> illustrate the automatic simplification rules that are applied in some systems to logical expressions as data objects[2]. In Example <12>, P and Q are unassigned identifiers and the simplification follows from the general logical rule P **and** $P \rightarrow P$.

The examples in Figure 1 are roughly similar to what happens in a real computer algebra system. However, since there is no consensus about which simplification rules should be included in automatic simplification, the process can vary somewhat from system to system.

Figure 2 shows an interactive dialogue with the Macsyma system that shows what happens when automatic simplification is suppressed. At the prompt `(c1)` we assign an expression to `u` and at `(c2)` turn off the automatic simplifier by assigning the value `false` to the variable `simp`. At `(c3)` we differentiate `u` and obtain an expression that is so involved it is

[1] The reordering process in Mathematica and MuPAD is similar to what is described here. The reordering process in Maple is handled in a different way (see page 71 in Section 3.1).

[2] Maple obtains <11> and <12>. Mathematica obtains <11>, but not <12>. MuPAD obtains <12>, but not <11>.

<1> $2 + 3/4 + 5/6;$

$$\rightarrow \quad \frac{43}{12}$$

<2> $x + y + 2 * x;$

$$\rightarrow \quad 3x + y$$

<3> $x * y * x^2;$

$$\rightarrow \quad x^3 y$$

<4> $1 * x^3 + a * x^0 + b * x^1 + 0 * x^2;$

$$\rightarrow \quad a + b\ x + x^3$$

<5> $x * y + 3 * y * x;$

$$\rightarrow \quad 4\,x\,y$$

<6> $\sin(\pi/2);$

$$\rightarrow \quad 1$$

<7> $\ln(e^2);$

$$\rightarrow \quad 2$$

<8> $\arctan(1);$

$$\rightarrow \quad \pi/4$$

<9> $\imath^2;$

$$\rightarrow \quad -1$$

<10> $e^{(-\imath * \pi)};$

$$\rightarrow \quad -1$$

<11> $0 \leq 1$ **and** $1 \leq 2;$

$$\rightarrow \quad \textbf{true}$$

<12> P **and** P **and** $Q;$

$$\rightarrow \quad P \textbf{ and } Q$$

Figure 1. An MPL dialogue that shows some examples of automatic simplification. (Implementation: Maple (mws), Mathematica (nb), MuPAD (mnb).)

difficult to interpret[3]. At `(c4)` we turn the automatic simplifier back on and at `(c5)` obtain a much more reasonable form for the derivative.

In MPL (as in a CAS), all expressions in dialogues and computer programs operate in the context of automatic simplification. This means:

- All input operands to mathematical operators are automatically simplified before the operators are applied.
- The result obtained by evaluating an expression is in automatically simplified form.

[3]Notice that Macsyma uses logarithmic differentiation to differentiate e^{x^2}. Logarithmic differentiation provides a way to differentiate general powers of the form $f(x)^{g(x)}$.

(c1) u : a*x + x*exp(x∧2);

(d1) $$x\,e^{x^2} + a\,x$$

(c2) simp : false;

(d2) false

(c3) diff(u,x);

(d3) $$1\;a + 0\,x + e^{x^2}\left(e^{-1}x^2\,0 + \log(e)\,(2x)\right)\;x + 1\,e^{x^2}$$

(c4) simp : true;

(d4) true

(c5) diff(u,x);

(d5) $$2\,x^2\,e^{x^2} + e^{x2} + a$$

Figure 2. An interactive dialogue with the Macsyma system that shows what happens when automatic simplification is suppressed.

Since automatic simplification is so central to the programming process, it is a good idea to understand which simplification rules are applied by the process and which are not. The algebraic component of the automatic simplification process is described in Chapter 3.

Return to Chapter 1, page 5