# MPL's Structure-based Operators

We describe below two MPL operators for which the operations are based only on the simplified structure of an expression. First, we introduce the terminology that is used in the definitions of these operators.

## Complete Sub-Expressions

**Definition 1** *Let $u$ be an automatically simplified expression. A* **complete sub-expression** *of $u$ is either the expression $u$ itself or an operand of some operator in $u$.*

In terms of expression trees, the complete sub-expressions of $u$ are either the expression tree for $u$ or one of its sub-trees.

**Example 1** Consider the expression

$$\sin(a) * (1 + b + c \wedge 2), \tag{1}$$

which has the expression tree shown Figure 1. This expression contains the following complete sub-expressions:

$$\begin{gathered}\sin(a) * (1 + b + c \wedge 2), \quad \sin(a), \quad a, \\ 1 + b + c \wedge 2, \quad 1, \quad b, \quad c \wedge 2, \quad c, \quad 2.\end{gathered}$$

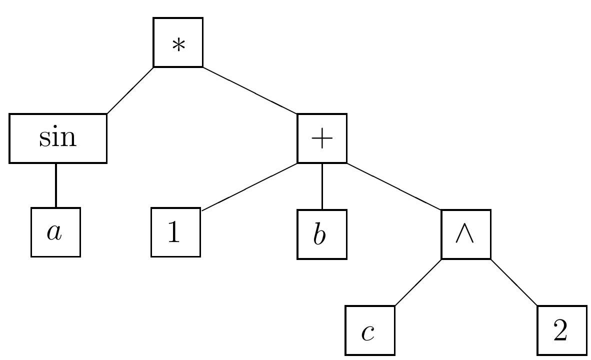


**Figure 1.** An expression tree for $\sin(a) * (1 + b + c \wedge 2)$.

| MPL | Maple | Mathematica | MuPAD |
|---|---|---|---|
| *Free_of*$(u,t)$ | `not(has(u,t))` | `FreeQ[u,x]` | `not(has(u,t))` |
| *Substitute*$(u, t=r)$ | `subs(t=r,u)` | `ReplaceAll[u,t->r]` or `u/.t->r` | `subs(u,t=r)` |

**Figure 2.** Structural operators in Maple, Mathematica, and MuPAD that correspond most closely to MPL's structural operators. (Implementation: Maple (mws), Mathematica (nb), MuPAD (mnb).)

There are some parts of an expression that are sub-expressions in a mathematical sense but are not complete sub-expressions. For example in Expression (1), $1 + b$ is not a complete sub-expression since it is not the operand of an operator.

## The MPL *Free_of* Operator

The MPL *Free_of* operator determines if an expression $u$ is free of an expression $t$ (or does not contain $t$).

**Definition 2** *Let $u$ and $t$ (for target) be mathematical expressions. The MPL operator*

$$\textit{Free\_of}\,(u,t)$$

*returns* **false** *when $t$ is identical to some complete sub-expression of $u$ and otherwise returns* **true**.

**Example 2**

$$\begin{aligned}
\textit{Free_of}\,(a+b,\ b) &\ \rightarrow\ \textbf{false}, \\
\textit{Free_of}\,(a+b,\ c) &\ \rightarrow\ \textbf{true}, \\
\textit{Free_of}\,((a+b)*c,\ a+b) &\ \rightarrow\ \textbf{false}, \\
\textit{Free_of}\,(\sin(x)+2*x,\ \sin(x)) &\ \rightarrow\ \textbf{false}, \\
\textit{Free_of}\,((a+b+c)*d,\ a+b) &\ \rightarrow\ \textbf{true}, && (2)\\
\textit{Free_of}\,((y+2*x-y)/x,\ x) &\ \rightarrow\ \textbf{true}, && (3)\\
\textit{Free_of}\,((x*y)^2,\ x*y) &\ \rightarrow\ \textbf{true}. && (4)
\end{aligned}$$

In Statement (2), $a+b$ is not a complete sub-expression of $(a+b+c)*d$ and so the operator returns **true**. In Statement (3), automatic simplification simplifies the first operand to 2 and so the expression no longer contains an $x$. In a similar way, in Statement (4) automatic simplification transforms $(x*y)^2$ to $x^2*y^2$ which gives the output **true**.

An operator similar to *Free_of* is available in most computer algebra systems (see Figure 2).

<1> $u := a + b;$

$$\rightarrow \quad u := a + b$$

<2> $v :=$ *Substitute*$(u,\ b = x);$

$$\rightarrow \quad v := a + x$$

<3> $u/v;$

$$\rightarrow \quad \frac{a+b}{a+x}$$

<4> *Substitute*$(1/a + a,\ a = x);$

$$\rightarrow \quad \frac{1}{x} + x$$

<5> *Substitute*$((a+b)^2 + 1,\ a + b = x);$

$$\rightarrow \quad x^2 + 1$$

<6> *Substitute*$(a + b + c,\ a + b = x);$

$$\rightarrow \quad a + b + c$$

<7> *Substitute*$(a + b + c,\ a = x - b);$

$$\rightarrow \quad x + c$$

**Figure 3.** An MPL dialogue that illustrates the use of the *Substitute* operator.

## The MPL *Substitute* Operator

Substitution is one of the essential operations used to manipulate and simplify mathematical expressions. The MPL *Substitute* operator performs a particularly simple form of substitution, called *structural substitution*, that is based solely on the tree structure of an expression.

**Definition 3** *Let $u$, $t$, and $r$ be mathematical expressions. The structural substitution operator has the form*

$$Substitute(u,\ t = r).$$

*It forms a new expression with each occurrence of the target expression $t$ in $u$ replaced by the replacement expression $r$. The substitution occurs whenever $t$ is structurally identical to a complete sub-expression of $u$.*

Keep in mind that *Substitute* does not change $u$, but instead creates an entirely new expression. Some examples of the use of the operator are given in the MPL dialogue in Figure 3.

The statements at <1>, <2>, and <3> illustrate that $u$ is not changed by the substitution operation. In <6>, the substitution does not occur since $a + b$ is not a complete sub-expression of $a + b + c$. However, in <7>, we obtain the substitution intended in <6> by modifying the form of the substitution.

Most computer algebra systems have a form of the *Substitute* operator (see Figure 2).

**Return to Chapter 1, page 10**