

	\vee <code>\vee</code>	\cdot <code>\cdot</code>	\triangleleft <code>\triangleleft</code>	1
	\wedge <code>\wedge</code>	\diamond <code>\diamond</code>	\triangleright <code>\triangleright</code>	
	\amalg <code>\amalg</code>	\bullet <code>\bullet</code>	∇ <code>\nabla</code>	
	\cap <code>\cap</code>	\circ <code>\circ</code>	\triangleup <code>\triangleup</code>	
	\cup <code>\cup</code>	\bigcirc <code>\bigcirc</code>	$*$ <code>*</code>	
	\uplus <code>\uplus</code>	\odot <code>\odot</code>	\star <code>\star</code>	
	\sqcap <code>\sqcap</code>	\ominus <code>\ominus</code>	\times <code>\times</code>	
	\sqcup <code>\sqcup</code>	\oplus <code>\oplus</code>	\div <code>\div</code>	
	\dagger <code>\dagger</code>	\oslash <code>\oslash</code>	\backslash <code>\setminus</code>	
	\ddagger <code>\ddagger</code>	\otimes <code>\otimes</code>	\wr <code>\wr</code>	
	\land <code>\land</code>	\pm <code>\pm</code>		
	\lor <code>\lor</code>	\mp <code>\mp</code>		

These commands produce the symbols for various binary operations. Binary operations are one of T_EX's classes of math symbols. T_EX puts different amounts of space around different classes of math symbols. When T_EX needs to break a line of text within a math formula, it will consider placing the break after a binary operation—but only if the operation is at the outermost level of the formula, i.e., not enclosed in a group.

In addition to these commands, T_EX also treats ‘+’ and ‘−’ as binary operations. It considers ‘/’ to be an ordinary symbol, despite the fact that mathematically it is a binary operation, because it looks better with less space around it.

Example:

```


$$z = x \div y \quad \text{if and only if} \quad z \times y = x \text{ and } y \neq 0$$


```

produces:

$$z = x \div y \quad \text{if and only if} \quad z \times y = x \text{ and } y \neq 0$$