Introduction to Mathematica patterns

Patterns are used to represent classes of expressions. For example, **f[_]** *stands for any expression of the form* **f[***anything***]**. *Pattern* **f[x_]** *also stands for* **f[***anything***]**, *but it gives name to the expression anything and allows to refer to it on the right-hand side of the transformation rule. For example:*

`f[n_]`	f with any argument, named n
`f[n_, m_]`	f with two arguments, named n and m
`x^n_`	x to any power, with the power named n
`x_^n_`	any expression to any power
`a_ + b_`	a sum of two expressions
`{a1_, a2_}`	a list of two expressions
`f[n_, n_]`	f with two *identical* arguments

A given pattern will match all expressions that can be obtained by filling in the named and unnamed blanks in any way.

```
In[1]:= f[x_] := g[x^2];
        f[a] + f[b] + f[Sin[θ]]
Out[2]= g[a^2] + g[b^2] + g[Sin[θ]^2]
```

```
In[1]:= Cases[{f[a], g[b], f[c]}, f[_]]
Out[1]= {f[a], f[c]}
```

Introduction to Mathematica patterns

Importantly, patterns represent classes of expressions with a given structure. In other words, while a pair of expressions may be mathematically equal, they might not match the same pattern.

Pattern	Expression	Match?
`(1+x_)²`	`(1+a)²`	Yes
`(1+x_)²`	$1+2a+a^2$	No
`x^_`	`x^2`	Yes
`x^_`	`1`	No

In all cases, the pattern matching in Mathematica is fundamentally structural rather than algebraic. This must always be kept in mind when designing patterns.

The internal representation of an expression may be obtained using the **`FullForm`** *command:*

```
In[1]:= FullForm[x^y]

Out[1]//FullForm=
        Power[x, y]
```

```
In[2]:= FullForm[a / b]

Out[2]//FullForm=
        Times[a, Power[b, -1]]
```

Introduction to Mathematica patterns

Real-life example: the error propagation law, a fairly tedious procedure, may be packed in full generality into one line of pattern-matching syntax.

$$\sigma_f = \sqrt{\sum_i \left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_{x_i}^2}$$

```
ErrEval[A_, x_, x0_, σ_] := {A /. Thread[x → x0],
   Sqrt[Total[(D[A, {x}]^2 /. Thread[x → x0]) σ^2]]};

ErrEval[Sin[x^y], {x, y}, {1.0, 2.0}, {0.1, 0.2}]

{0.841471, 0.10806}
```

Real-life example: Clebsch-Gordan expansions of products of spherical harmonics may be programmed in full generality with just two patterns (see Tutorial 1 for the details of those patterns).

$$Y_{2,0}(\theta,\varphi) = \frac{1}{4}\sqrt{\frac{5}{\pi}}\left(3\cos^2\theta - 1\right)$$

```
Y[2, 1] Y[4, -3] Y[6, 5] // ExpandAll
```

$$-\frac{9\sqrt{\frac{105}{286}}\,Y[4,3]}{17\pi} + \frac{9\sqrt{\frac{35}{22}}\,Y[6,3]}{646\pi} + \frac{9\sqrt{\frac{35}{221}}\,Y[8,3]}{38\pi} - \frac{99\sqrt{\frac{105}{2}}\,Y[10,3]}{14858\pi} + \frac{9\sqrt{42}\,Y[12,3]}{7429\pi}$$

Typical processing time: milliseconds… :)

Conditional patterns

Certain patterns should only be applied if certain conditions are met (e.g. a term can be taken out of integral only if it contains no integration variable). Mathematica provides a general way of putting conditions on patterns:

pattern /; *condition*	a pattern that matches only when a condition is satisfied
lhs :> *rhs* /; *condition*	a rule that applies only when a condition is satisfied
lhs := *rhs* /; *condition*	a definition that applies only when a condition is satisfied

An example of a conditional pattern for the complex conjugation operation:

```
Conjugate[A_] := A /; A ∈ Reals
Conjugate[A_] := -A /; (ⅈ A) ∈ Reals
```

Another example from the linearity definition of an integration operator:

```
Int[A_ + B_] := Int[A] + Int[B];
Int[A_ B_] := A Int[B] /; FreeQ[A, τ]
```

The '/;' symbol can be interpreted as 'whenever'. Conditions should be applied to the smallest possible parts of expressions – the sooner Mathematica encounters a violation, the sooner it can stop processing a given pattern.

More advanced patterns

Double blanks stand for sequences of one or more expressions. Triple blanks stand for sequences of zero or more expressions.

_	any single expression
*x*_	any single expression, to be named *x*
__	any sequence of one or more expressions
*x*__	sequence named *x*
*x*__*h*	sequence of expressions, all of whose heads are *h*
___	any sequence of zero or more expressions
*x*___	sequence of zero or more expressions named *x*
*x*___*h*	sequence of zero or more expressions, all of whose heads are *h*

Extending the linearity and threading of conjugation over an arbitrary number of arguments (will be used in BRW processor):

```
Conjugate[A_ B__] := Conjugate[A] Conjugate[Times[B]];
Conjugate[A_ + B__] := Conjugate[A] + Conjugate[Plus[B]];
```

Times and Plus operations return a sum or a product of elements in the list, so the rules above will operate repeatedly until the list of terms in the sum or product is exhausted.

Some frequently encountered patterns

Typical patterns for algebraic expressions:

`x_ + y_`	a sum of two or more terms
`x_ + y_.`	a single term or a sum of terms
`n_Integer x_`	an expression with an explicit integer multiplier
`a_. + b_. x_`	a linear expression $a + bx$
`x_ ^ n_`	x^n with $n \neq 0, 1$
`x_ ^ n_.`	x^n with $n \neq 0$
`a_. + b_. x_ + c_. x_^2`	a quadratic expression with non-zero linear term

Typical patterns for lists:

`x_List` or `x:{___}`	a list
`x_List /; VectorQ[x]`	a vector containing no sublists
`x_List /; VectorQ[x, NumberQ]`	a vector of numbers
`x:{___List}` or `x:{{___}...}`	a list of lists
`x_List /; MatrixQ[x]`	a matrix containing no sublists
`x_List /; MatrixQ[x, NumberQ]`	a matrix of numbers
`x:{{_, _}...}`	a list of pairs

General notes

• *Mathematica kernel has complete memory of past commands, which is retained between the worksheets. Restart the kernel (Menu > Evaluation > Quit Kernel > Local) to make it forget what you told it before.*

• *Floating point numbers disable many analytical ransformation routines in Mathematica. Always use analytical expressions (e.g.* **`1/2`** *instead of* **`0.5`***).*

• *A spacebar symbol is interpreted as multiplication. That can be a source of much frustration, so be careful. Never put a space anywhere unless you mean to multiply.*

• *Argument brackets are* **`[rectangular]`***, priority brackets are* **`(round)`** *and list brackets are* **`{curly}`***.*

The practical tutorial worksheets (~45 min and ~120 min respectively) on pattern-matching and its applications to relaxation theory can be downloaded from here:

http://spindynamics.org

Enjoy! :)