Mathematica

Numeric or Symbolic Processing

Mma computes symbolically unless either
1. no efficient symbolic technique is known, or
2. processing efficiency is far more important than coding efficiency.

Otherwise, it uses optimized numeric techniques.

SYMBOLIC MODEL

meaning
--written as--> symbol structures
--reduced by--> symbolic transformation rules
--into--> simpler symbolic structures
--interpreted for--> meaning

NUMERIC MODEL

meaning
--exemplified by--> selected instances
--substituted into--> symbolic structures
--reduced by--> numeric evaluation rules
--into--> approximate results
--read for--> meaning

The Mathematica Program

A general purpose computational engine for

numerical calculations	(arithmetic)
symbolic transformations	(algebra)
graphic display	(geometry)

A modern programming language with multiple styles
- procedural
- functional
- logical
- object-oriented
- rule-based
- mathematical

An integrated tool
C, TeX, UNIX, Postscript. MathLink

The Philosophy

The programmer's time is more valuable than the processor's time.

Thus, the architecture is
interpreted (interactive)
real-time
goal-oriented

"Programs you write in Mathematica may nevertheless end up being faster than those you write in compiled languages" p.506

- Processing speed depends on the exact implementation algorithm
- Mathematica algorithms are both sophisticated and optimized
- The internal data form is optimized and compiled for efficiency

The Limits

- Out-of-memory can crash the program
- Some algorithms require large searches and exponential processing time

"The internal code of Mathematica uses polynomial time algorithms whenever they are known." p.63

- In a second, you can do
  arithmetic with thousand digit numbers
  factoring a hundred term polynomial
  apply a recursive rule a few thousand times
  find the numerical inverse of a 50x50 matrix
  format a few pages of output
  draw a few thousand lines

Everything is an Expression

```
x + y           Plus[x,y]
120             Integer[120]
2ab             Times[2,a,b]
{a,b,c}         List[a,b,c]
i = 3           Set[i,3]
x^2+2x+1        Plus[ Power[x,2], Times[2,x], 1]
```

An undefined symbol is *itself*, providing functional transparency and WYSIWYG debugging.

The Meaning of Expressions

`F[x,y]`	`F` is the *head*. `x,y` are the *contents*. Apply *function* `F` to arguments `x` and `y`. Do *action* `F` to objects `x` and `y`. The *label* `F` points to elements `x` and `y`. The *object-type* `F` has parts `x` and `y`.

The head can both act on its contents (as a function) and maintain the structure of its contents (as an object), depending on context (the location of the expression, the presence of a definition).

Lists

The boundary labeled List maintains its internal structure as a database.
Lists are used for all collections:

data record	`{John, 555-1234, j@mma.com}`
vector	`{x, y, z}`
matrix	`{ {11,12},{21,22} }`
set	`Union[{a,b},{a,c}] ==>  {a,b,c}`
graphics spec	`Line[ {{0,1}, {1,1}, {1,0}} ]`
stream	`{1,2,3, ...}`
structure template	`{_, {_,_}, { {_,_},...} }`

The Fundamental Principle of Computation

Take any expression and apply transformation rules until the result no longer changes.

1. Reduce head	
2. Reduce each element	base case arithmetic
3. Standardize	
4. Apply user defined rules	inductive case algebra
5. Apply built-in rules.	
6. Reduce the result.	recursion

The Internal Mechanism

All expressions are stored in an *augmented binary transform table* (?)

we see	*internal*	*table entry*
`x=3`	`11001...`	`11`
`a[1]=x`	`1000[1]`	`10110`
`f[n_]=n^2`	`1011[#]`	`10110[#1110]`

- The input expression is matched (using a linear-time algorithm) to the internal table:

  `E = 00110[0100011]001010[11101]`

- Each match generates a substitution.
- No match causes no change.
- The structure of the expression is not a string but a network.

Patterns

A *pattern* is a class of expressions with the same structure.

`_`	"blank", underline means *any* expression
`x_`	any expression locally named x
`x__`	any sequence of expressions
`x___`	any sequence, including none
`x_h`	any expression with head = h.

Examples:

`f[n_]`	the function f with a parameter named n
`2^n_`	2 raised to any power
`a_ + b_`	the sum of two arbitrary expressions
`{a__}`	a list with at least one element

Data Types

"At a fundamental level, there are no data types in Mathematica. Every object you use is an expression, and every function can take any expression as an argument." p.496

The head of an expression can be interpreted as a *type constraint*.

`Integer[3]` means `Type[3] = Integer`

Unity of Programming Paradigms

Mathematica accepts code in all of the modern programming paradigms.

"All the approaches are in a sense ultimately equivalent, but one of them may be vastly more efficient for a particular problem, or may simply fit better with your way of thinking about the problem." p.487

"As a matter of principle, it is not difficult to prove that *any* Mathematica program can in fact be implemented using transformation rules alone." p.503

Object-oriented Organization

```
square/:    perimeter[ square[n_] ] := 4*n
square/:    area[ square[n_] ]      := n^2
circle/:    area[ circle[r_] ]      := Pi*r^2
```

The outer "function" transforms the inner "argument".
The inner "object" contains a private outer "message handler".
The outer "matrix" is indexed by the inner "accessors".