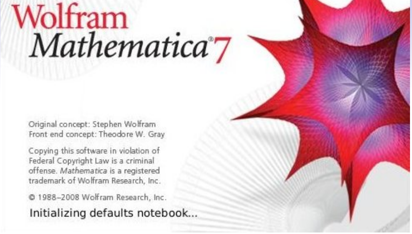
Mathematica

Christoph Koutschan

Research Institute for Symbolic Computation
Johannes Kepler Universität Linz, Austria

Computer Algebra Systems
11.10.2010



Wolfram Mathematica 7

Original concept: Stephen Wolfram
Front end concept: Theodore W. Gray

Copying this software in violation of Federal Copyright Law is a criminal offense. *Mathematica* is a registered trademark of Wolfram Research, Inc.

© 1988–2008 Wolfram Research, Inc.

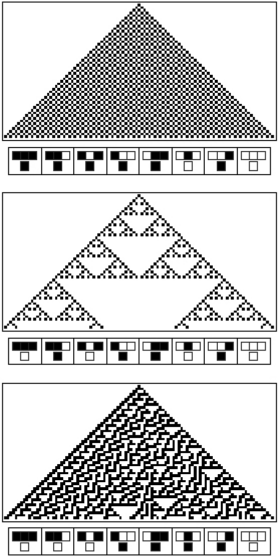
Initializing defaults notebook...

- developed by Wolfram Research (Urbana-Champaign, Illinois)
- first release in 1988
- proprietary license
- Professional Edition: 1.345 Euro
  Mathematica Home Edition: 295 Euro
  Standard Edition for Students: 150,40 Euro
- JKU has campus license!
- http://www.wolfram.com/products/mathematica/

Stephen Wolfram

- born 1959 (London, UK)
- received his PhD in particle physics in 1979 (Caltech)
- started developing SMP (Symbolic Manipulation Program) from 1979 on (together with Chris Cole)
- Mathematica 1.0 in 1988
- still the head of Wolfram Research

Stephen Wolfram



- born 1959 (London, UK)
- received his PhD in particle physics in 1979 (Caltech)
- started developing SMP (Symbolic Manipulation Program) from 1979 on (together with Chris Cole)
- Mathematica 1.0 in 1988
- still the head of Wolfram Research
- interest in cellular automata
- book *A New Kind of Science*
- Wolfram Alpha ("Computational Knowledge Engine")

Expressions in Mathematica

- everything is an expression (even programming constructs, graphics, notebooks, etc.)
- expressions are represented as trees
- internal representation of expressions is of the form
  Head[expr1,expr2,...] or Head
  only exceptions: atoms like
  - integers: ...−1, 0, 1, 2,...
  - floating point numbers: 3.141592653589793`
  - variables x, y, z, α, ...
  - strings: "this is a string"
- all (internally defined) heads start with a capital letter

Implications:

- function application is denoted with square brackets, e.g., Sin[x], Plus[3,7], etc.
- mathematical constants start with upper-case letter, e.g., Pi, E, I, etc.

Examples: Expressions in Mathematica

```
Head[{1,2,3}]

FullForm[x+y]

Head[x]

FullForm[2]

Head[2]

FullForm[ToString[2]]

TreeForm[a-2b^2]

SymmetricPolynomial[2,
          Symbol /@ CharacterRange["a", "z"]]
```

Naming Conventions in Mathematica

- words are usually written out, e.g., `Denominator`, `SingularValueDecomposition`, etc.
- for well-known mathematical functions, the common abbreviation is used, e.g., `Sin`, `Log`, `GCD`, `Det`, etc.
- functions which return a boolean value end with `Q`, e.g., `EvenQ`, `IntegerQ`, etc.
- special functions are given by the name of the person after whom the function is named plus the (capital) letter how it is usually denoted with, e.g., `LegendreP`, `BesselJ`, `KroneckerDelta`, etc.

Help in Mathematica

- `?Name` displays information about `Name`
- `??Name` displays information plus all definitions for `Name`
- `?*Name*` displays all symbols that contain `Name`
- `?*` displays all symbols that are known at this point
- `` ?Global`* `` displays all symbols in context `Global`
- `Options[Name]` displays all options that can be given to `Name` and their default values
- `Attributes[Name]` displays all attributes that are set for `Name`

Quick start into Mathematica (1)

Key combinations:

- ⟨Shift⟩ + ⟨Enter⟩ to execute a command
- ⟨Enter⟩ for new line
- ⟨Alt⟩+. to abort a computation

Bracketing:

- `(...)` for grouping of expressions, e.g., `a*(b+c)`
- `[...]` for function arguments, e.g., `f[x]`
- `{...}` for lists, e.g., `{1,2,3}`

Return values:

- semicolon at the end suppresses output of return value
- `%` refers to the last result
- `%n` refers to output `Out[n]`

Quick start into Mathematica (2)

Different equal signs:

- = for assignments
- := for delayed evaluation
- == for mathematical equations (negation: !=)
- === for checking syntactical equality (negation: =!=)

Example: What's the difference between

```
(a+b)^2 = a^2+2*a*b+b^2
(a+b)^2 == a^2+2*a*b+b^2
(a+b)^2 === a^2+2*a*b+b^2
```

Further examples:

```
test := Print["test"]
fib[n_] := (fib[n] = fib[n-1] + fib[n-2])
```

Quick start into Mathematica (3)

Iterators:

- `{n,0,5}`: `n` takes the values from 0 to 5
- `{n,5}`: `n` takes the values from 1 to 5
- `{n,1,15,2}`: `n` takes the values from $1, 3, 5, \ldots, 15$
- `{5}`: five times without assigning a variable

Programming constructs:

- conditional: `If[cond,t,f,u]`
- do loop: `Do[command,{i,a,b}]`
- while loop: `While[cond,command]`
- for loop: `For[init,cond,incr,command]`
- local variables: `Module[vars,command]`

Important Concepts in Mathematica

- separation of kernel and frontend (notebook)
- additional libraries (statistics, number theory, etc.)
- functional programming
- pattern matching
- visualization
- contexts
- list operations

Concepts: List Operations

- lists are used to represent vectors, matrices, and sets.
- commands for matrices: `Dot`, `Inverse`, `NullSpace`, `Eigenvalues`
- commands for sets: `Union`, `Intersection`, `Complement`

Structural operations for lists:

- many functions are `Listable`
- creating lists: `Table`, `Range`, `Join`
- standard list operations: `Append`, `Prepend`, `Rest`, `Most`,
- pick out some element: `Part`, `First`, `Last`, `Take`
- reordering: `Reverse`, `Sort`, `SortBy`
- many more operations: `Flatten`, `Transpose`, `PadLeft`, `PadRight`, `Riffle`, `Tally`, `Split`, etc.
- level specification

Examples: List Operations

```
{{1,2},{0,1}}.{2,-1}

Transpose[{{1,2},{0,1}}]

Table[2*n, {n,1,10}]

{1,2,3}*{7,8,9}

Range[10]^3

MapThread[Append, {{{1,2},{0,1}},{2,-1}}]

Riffle[{a,b,c}, {x,y,z}]

PadLeft[{a,b,c}, 10, {x,y,z}, 2]

Tally[Table[RandomInteger[9], {1000}]]
```

Structural operations for general expressions

Commands like `Append`, `Most`, `Map`, `Part`, etc. can be applied to any expression!

Example:

```
expr = a+b+c
Length[expr]
Append[expr, d]
Map[Sqrt, expr]
expr[[2]]
```

Concepts: Visualization

- plot functions: `Plot`, `Plot3D`, `ListPlot`, `ContourPlot`, etc.
- dynamic graphics
- all kinds of diagrams: `BarChart`, `PieChart`, `Histogram`, etc.
- compose pictures with `Graphics` and `Graphics3D`

Examples: Visualization

```
ListPlot[IntegerExponent[Table[Binomial[200, k],
{k,0,200}], 2]]
```

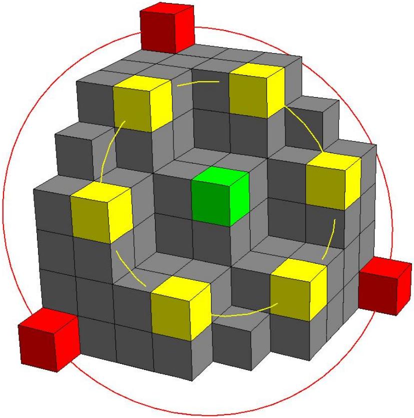
```
Plot[Table[BesselJ[n, x], {n,0,5}], {x,-10,10}]
```

```
Manipulate[ContourPlot[(x^2+y^2)^2 == x^2 - a*y^2,
{x,-1,1}, {y,-1,1}], {a,-1,1}]
```

```
Plot3D[x^2/y^2, {x,-1,1}, {y,-1,1}]
```

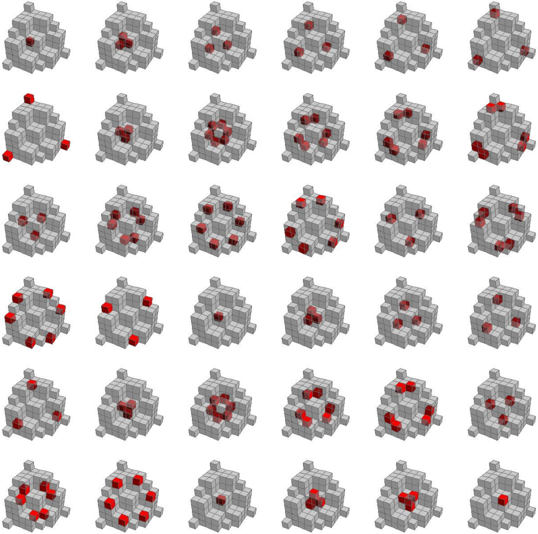
```
Graphics[Table[Circle[{x,0}, x^2], {x,-2,2,.1}]]
```

Examples: Visualization



A totally symmetric plane partition (TSPP)

Examples: Visualization



Concepts: Pattern Matching

Very useful for substitutions (`ReplaceAll`, etc.), but not only. Many other commands like `Count`, `FreeQ`, `Cases` accept patterns.

- `a_` matches any expression
- `a__` matches any non-empty sequence
- `a___` matches any (possibly empty) sequence
- `a_Head` matches any expression with head `Head`
- `a_?Test` matches any expression such that `Test[a]` is true
- `a_ /; cond` matches any expression such that `cond` is true
- `f_[_,_,_]` matches any expression of length 3
- `f[a_,a_]` matches any occurrence of `f` with two equal arguments

Examples: Pattern Matching

```
{f, f[x], f[x,y], f[2]} /.  f -> g

{f, f[x], f[x,y], f[2]} /.  f[x] -> g[x]

{f, f[x], f[x,y], f[2]} /.  f[x_] -> g[x]

{f, f[x], f[x,y], f[2]} /.  f[x_Integer] -> g[x]

Cases[Sin[x-2*y^2], a_[___] -> a, {0,Infinity}]

f[x_, opts:((_Rule|_RuleDelayed)...)]  :=

de = D[x^2*y^2*f[x, y], x, x, y, y, y];
de /.  f[x,y]->1 /.  Derivative[a__][f][x,y] :>
(Times@@({Dx,Dy}^{a}))
```

Concepts: Functional Programming

- pure functions: `Function`, `#`, `##`, `&`
- work with functions as symbolic objects: `D`, `InverseFunctions`
- `Map`, `MapAll`, `MapIndexed`, `Apply`
- `Nest`, `Fold`, `FixedPoint`

Examples: Functional Programming

```
f = Function[x, x^2]

f'[3]

Clear[f]

D[f[x]*g[x],x]

Solve[f[x] == y, x]

FixedPoint[Sin[#]+Cos[#]&, 0.1, SameTest ->
(Abs[#1-#2]<10^(-9)&)]

Nest[f, x, 5]

MapIndexed[{#1,#2-1}&, CoefficientList[x^3+y^2+2*x*y,
{x,y}], {2}]
DeleteCases[Flatten[%,1],{0,_}]
```

Concepts: Contexts

- contexts are namespaces
- notation: `` Context`Name ``
- usually the symbols of a package lie in a separate context

Some important contexts:

- `Global`
- `System`
- `Developer`

Some Exercises

1. Generate a list of all positive rationals with numerator and denominator not greater than 10!

Some Exercises

1. Generate a list of all positive rationals with numerator and denominator not greater than 10!

   ```
   Union[Flatten[Table[n/d, {n,0,10}, {d,10}]]]
   ```

Some Exercises

1. Generate a list of all positive rationals with numerator and denominator not greater than 10!

   ```
   Union[Flatten[Table[n/d, {n,0,10}, {d,10}]]]
   ```

2. All cubes of the smallest totally symmetric plane partition generated by some points (two different solutions):

Some Exercises

1. Generate a list of all positive rationals with numerator and denominator not greater than 10!

   ```
   Union[Flatten[Table[n/d, {n,0,10}, {d,10}]]]
   ```

2. All cubes of the smallest totally symmetric plane partition generated by some points (two different solutions):

   ```
   Union[Flatten[Table @@@
   (Prepend[Transpose[{{a,b,c}, #}], {a,b,c}]& /@
   Flatten[Permutations /@ p, 1]), 3]]
   ```

Some Exercises

1. Generate a list of all positive rationals with numerator and denominator not greater than 10!

   ```
   Union[Flatten[Table[n/d, {n,0,10}, {d,10}]]]
   ```

2. All cubes of the smallest totally symmetric plane partition generated by some points (two different solutions):

   ```
   Union[Flatten[Table @@@
   (Prepend[Transpose[{{a,b,c}, #}], {a,b,c}]& /@
   Flatten[Permutations /@ p, 1]), 3]]
   ```

   ```
   Select[Flatten[With[{m=Max[Flatten[p]]},
   Table[{a,b,c}, {a,m}, {b,m}, {c,m}]], 2],
   Function[point, Or@@((And@@Thread[point<=#])&
   /@ Flatten[Permutations /@ p, 1])]]
   ```