Loose Ends: Evaluation, Mathematical Types, DLMF, Syntax and OCR

Lecture 20

Outline

- Evaluation (ref neweval.pdf)
- Mathematical Type Systems, esp Axiom
- Syntax Extension, esp Macsyma
- Nat'l Inst. of Stds and Tech: DLMF
- OCR

Evaluation and names

Too many contexts for a symbol like “x”

- Programming: int x=43. ...
- Programming: bound variable (lambda(x) (+ x 1))
- Programming declared name: double_float x; [methods]
- Symbolic: sin(2*x)=2*sin(x)*cos(x) .. for all x
- Symbolic: Solve $x^2-1=0$ for x
- Symbolic: Let F[x] be a polynomial domain consisting of the field F extended by the indeterminate x
- Symbolic/logical: assume x is real
- Symbolic bound variable: Sum(r_x,x=1..1)
- Type-symbolic class name x

Disambiguating a symbol’s interpretation

- Lisp has x, ‘x, also “x”
- Maple has x, uneval, evalf evalhf
- Mathematica has x, Hold[x], HoldAll[], HoldFirst[]
- No way of doing this naturally.

Evaluation and operators

- a+b simplifying to a+b.
- a+a simplifying to 2*a
- a+b+c simplifying to c+7 (if a=4, b=3)

- Application of functions vs. simplification
- sin(π) simplifies to 0?
- sin(3) vs sin(3.0)?

Evaluation and depth

- x:=y+1
- y:=z+1

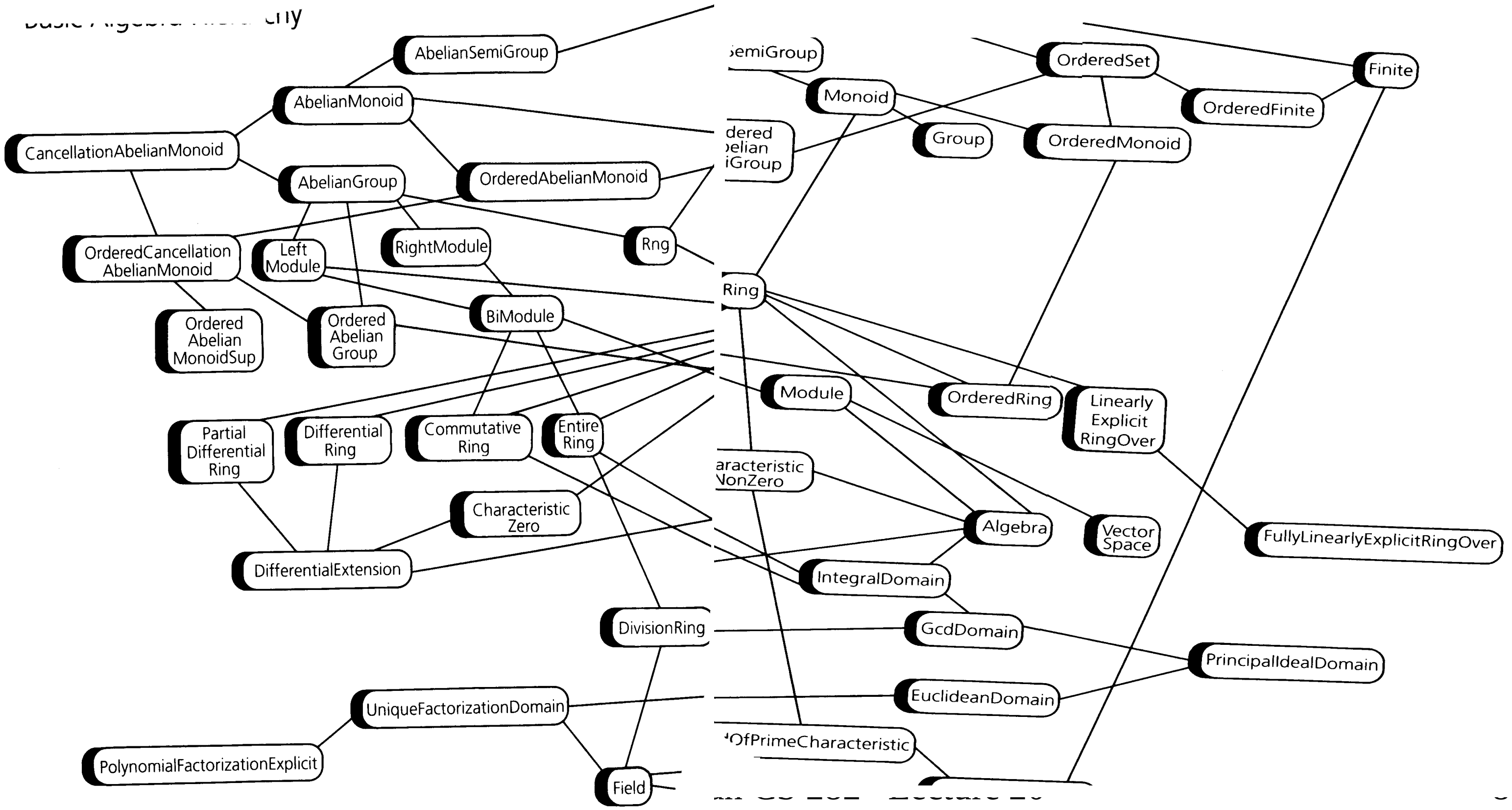
- what is x? z+2?
- evaluate 0,1,2, until stops changing?

Algebraic Type Systems

- Axiom
- Mupad
- ModeReduce (gone)
- Gauss / Maple
- Format/ Macsyma

The most thorough example is Axiom

- Basic Algebraic Hierarchy (scanned in 2 pieces)



How does this work

- Answers come out with tags like type:DistributedMultivariatePolynomial([x,y], Fraction Integer)
- and occasionally the user must assert domains like DMP([x,y],FRAC INT)
- More complicated items are “easily” constructed..

Axiom Categories and Domains

Categories define algebraic properties. Protect programs from doing meaningless operations, and allow all meaningful ones.

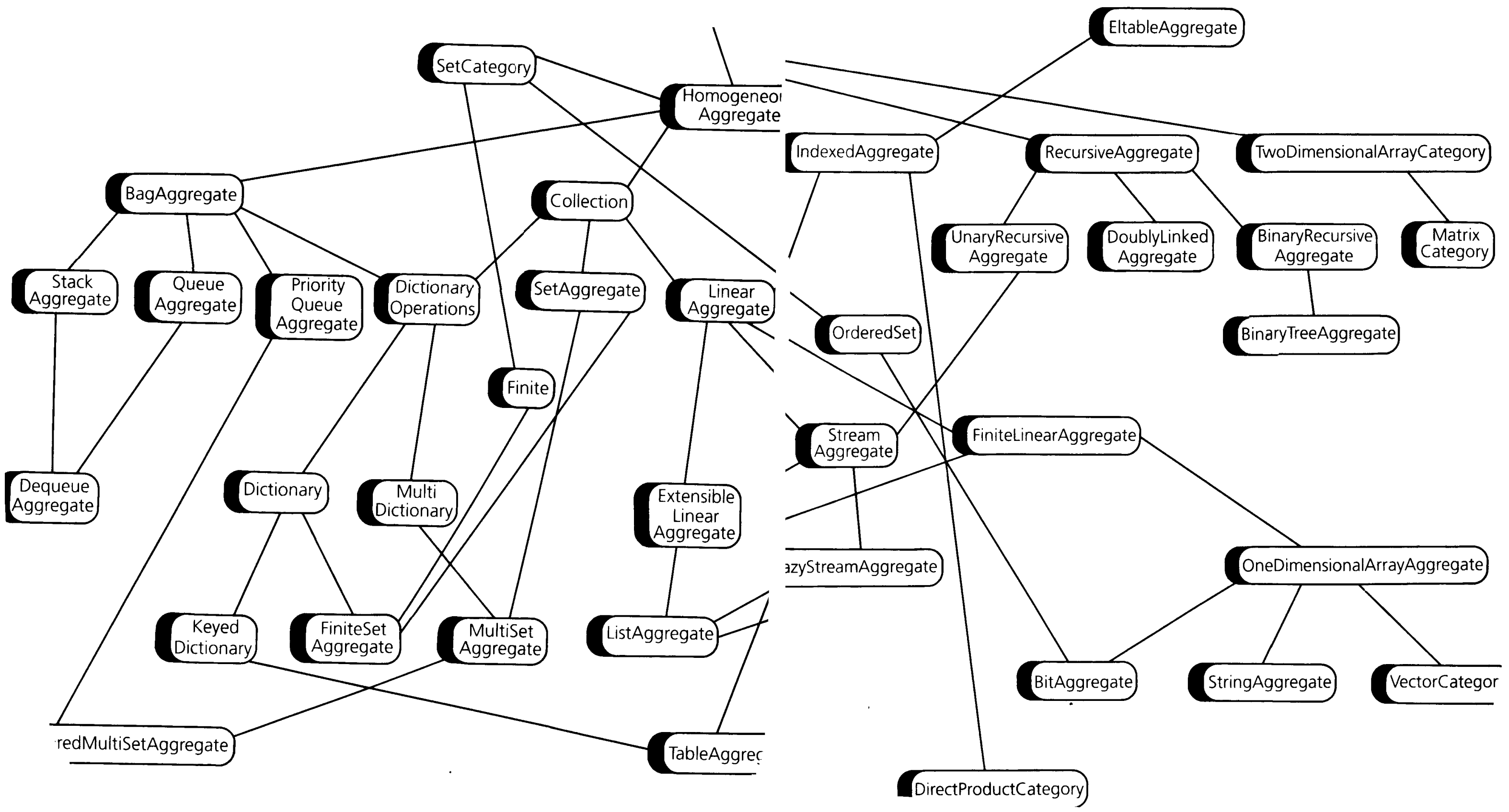
- name, e.g. Ring
- operations, e.g. "+", "*" "-"
- other categories that it extends (inherits from)
- Domains (of computation) denote classes of objects: Integer, Float, Polynomial(Integer), Matrix(Float)

Arguably Axiom allowed experts to define very general algebraic notions

- And programs to go along with them.
- But to specify programs you need data structures too, and these are categories.
- Here is what Axiom provides..

(from endpapers of Axiom reference...)

- Data Structure Hierarchy



How far can we push this?

- By allowing objects of type file, stream, color, point, ... there is not much that is excluded. It just isn't especially convenient to program for the novice. Nor is it especially compact or fast – although there is nothing to prevent it from being compiled to good code.
- Why did NAG drop Axiom (late 2001)? Perhaps
  - Not strategic: can use numerical NAG from Maple
  - Not profitable
  - Misunderstanding of the market?