A Cluster of Languages for Mathematical Computing

Stephen M. Watt

Department of Computer Science
University of Western Ontario
London Ontario, Canada

Heriot-Watt University
22 September 2012
Solving Too Easy Problems

- Many languages tested on handling GUIs
  - Moving windows around…
  - Add a border…
  - Add a scroll bar…
  - Respond to a button ….

- We have harder problems now.
Declaration of Prejudices

- Key problem: How to cascade efficient, expressive abstractions.

Theorem 1. Let $A$, $B$ be hybrid sets over $U$, $S$ an arbitrary set, and $f : U \rightarrow S$ a total function. Then

1. $R(f^B)$ is the empty function,
2. $f^A \odot f^A = f^{2A}$
3. $f^A \odot f^B = f^{A \oplus B}$, and thus a hybrid function,
4. For $g : U \rightarrow S$ another total function, then $f^A \odot g^B = (f \odot g)^{A \oplus B}$ if and only if $A \oplus B = \emptyset$ (where $f \odot g$ is the join of regular functions).
5. Let $H_1, H_2$ be hybrid sets, with $\text{supp} \ H_1$ and $\text{supp} \ H_2$ disjoint, $f_1 : \text{supp} \ H_1 \rightarrow S$ and $f_2 : \text{supp} \ H_2 \rightarrow S$, then $f_1^{H_1} \odot f_2^{H_2} = (f_1 \odot f_2)^{H_1 \oplus H_2}$
Mathematics as a Programming Language Canary
Why?

- Complex problems with many parts
- Complex interactions among the parts
- Many different levels of abstraction
- Precise definition
- Can tell if an answer is right or wrong
Examples

- Garbage collection
  - Lisp ➔ ⋮ ⋮ ➔ Java etc
- Algebraic expressions
  - Fortran
- Big integer
  - Crypto
- Generics
  - ➔ Java, C++, ***
Computer Algebra

- A couple of research problems of personal interest
  - Symbolic-numeric algorithms
  - Symbolic exponents
Approximate Polynomials

\[ f = y^2 - x^4 = (y - x^2)(y + x^2) \]

\[ f^* = y^2 - x^4 + .01x^2 \]

\[ \approx (y - x^2 + .00500)(y + x^2 - .00504) \]
Symbolic Exponents

\[ p = 8x^{n^2+6n+4+m^2-m} - 2x^{2n^2+7n+2mn}y^{n^2+3n} \]
\[ - 3x^{n^2+3n+2mn}y^{n^2+3n} + 12x^{4+m^2-m+2n} \]
\[ = x^{2n} \times \left( 2x^{n^2+4n} + 3 \right) \]
\[ \times \left( 2x^{1/2 m^2-1/2 m+2} - x^{1/2 n^2+mn+1/2 n}y^{1/2 n^2+3/2 n} \right) \]
\[ \times \left( 2x^{1/2 m^2-1/2 m+2} + x^{1/2 n^2+mn+1/2 n}y^{1/2 n^2+3/2 n} \right) \]
Some Languages

- Maple
- Axiom
- Aldor
- OpenMath, MathML
- InkML
Language 1: Maple

- Init by Keith Geddes & Gaston Gonnet.
- University, then company. Collaboration.
- Dynamically typed, interpreted language for scripting computer algebra programs.
\[ p := \left( x^2 + 39 \cdot x + 2 \right) \cdot \left( x^4 + x^3 - 1 \right) \cdot \left( x + 1 \right) \]

\[ p := (x^2 + 39x + 2)(x^4 + x^3 - 1)(x + 1) \]

\[ \text{expand}(p) \]
\[ x^7 + 41x^6 + 81x^5 + x^3 - 40x^2 + 43x^4 - 41x - 2 \]

\[ q := \sum(x^k, k=0..15) \]
\[ q := 1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14} + x^{15} \]

\[ \text{factor}(q) \]
\[ (x + 1)(1 + x^2)(1 + x^4)(1 + x^8) \]

\[ \int \left( \frac{\sin(a \cdot x + b)}{x^2}, x \right) \]
\[ a \left( -\frac{\sin(ax + b)}{ax} - \text{Si}(ax) \sin(b) + \text{Ci}(ax) \cos(b) \right) \]
# Find the n-th un-normalized LS polynomial, assuming we have 0..n-1.

\[
\text{findNext} := \text{proc}(n)
\]
\[
\text{local } bn, Bn, i, co, eqn, N, \text{desired};
\]
\[
Bn := \text{subs}(bn[n] = 1, \text{sum}(bn[i] \cdot t^i, i = 0..n));
\]
\[
\text{for } i \text{ from } n - 1 \text{ to } 0 \text{ by } -1 \text{ do}
\]
\[
eqn := \text{ip}(Bn, B[i]);
\]
\[
co := \text{solve}(\%, bn[i]);
\]
\[
Bn := \text{subs}(bn[i] = co, Bn)
\]
\[
\text{od};
\]
\[
N := \text{ip}(Bn, Bn);
\]
\[
desired := \frac{2}{(2 \cdot n + 1)};
\]
\[
isimplifyLS\left(Bn \cdot \text{sqrt}\left(\frac{\text{desired}}{N}\right)\right)
\]
\[
\text{end}:
\]
\[
\text{unormalLS} := \text{proc}(p) \text{ simplifyLS}\left(\frac{p}{lcoeff(p, t)}\right) \text{ end}:
\]
\[
\text{for } d \text{ from } 0 \text{ to } \text{LIMIT} \text{ do}
\]
\[
B[d] := \text{unormalLS}(\text{findNext}(d))
\]
\[
\text{od;}
\]
\[
\frac{1}{t}
\]
\[
- \frac{1}{3} + r^2
\]
\[
- \frac{3}{5} \frac{(1 + 5 \mu) t}{1 + 3 \mu} + r^3
\]
\[
\frac{3}{35} \frac{1 + 35 \mu}{1 + 15 \mu} - \frac{6}{7} \frac{(1 + 21 \mu) r^2}{1 + 15 \mu} + r^4
\]
Maple Architecture

- Compiled kernel, interpreted library
- What was compiled was hand-chosen
- Support many students on shared 1980s hw

- Commercially viable project
- Company focus education and CAE
Maple Language

• Easy to lay down code, quick library growth

• Language limitations
  • Experimental language ideas mixed success
    • Call by “evaluated name”, three-valued logic, …
  • Uninitialized variables symbolic => subtle bugs
  • Simple name space => difficulty organizing, conflicts
  • Many interactions => structuring large libraries hard
  • Kernel architecture => can’t make new things fast
    • evalhf and other oddities
    • Maple clones for other areas of mathematics

• Each of these has a good side and bad side.
Example 2: Axiom

- 1984 moved from Waterloo to IBM Research

- Scratchpad II in-house research project

- Initiated by Richard Jenks
  (unrelated follow-on to Scratchpad by Griesmer, Jenks, Yun)

- 1991 released commercially by NAG
Axiom

- Main idea:
  Generic algorithms based on structures of modern algebra (groups, rings, algebras, ...)

- Compiled programming language for writing libraries “in the large”

- Syntactically similar, dynamically typed interpreted language for scripting.
Type Inference in Interpreter

\[ p := r^{\ast\ast}2 + \frac{2}{3} \]

\[ r^2 + \frac{2}{3} \]

Type: Polynomial Fraction Integer

\[ p :: Fraction \ Polynomial \ Integer \]

\[ \frac{3 r^2 + 2}{3} \]

Type: Fraction Polynomial Integer
More Complicated Types

\[ PZ := \text{UnivariatePolynomial}(x, \text{Integer}) \]

\[ \text{UnivariatePolynomial}(x, \text{Integer}) \]

Type: Domain

\[ x: PZ := 'x \]

\[ x \]

Type: \text{UnivariatePolynomial}(x, \text{Integer})

\[ \text{Mat} := \text{SquareMatrix}(3, PZ) \]

\[ \text{SquareMatrix}(3, \text{UnivariatePolynomial}(x, \text{Integer})) \]

Type: Domain
The operators act on the vectors considered as a Mat-module.

\[ \text{Vect} := \text{DPMM}(3, \text{PZ}, \text{Mat}, \text{PZ}) \]

\[
\text{DirectProductMatrixModule}(3, \\
\text{UnivariatePolynomial}(x, \text{Integer}), \\
\text{SquareMatrix}(3, \text{UnivariatePolynomial}(x, \text{Integer})), \\
\text{UnivariatePolynomial}(x, \text{Integer}))
\]

\[ \text{Type: Domain} \]

\[ \text{Modo} := \text{LODO2} (\text{Mat}, \text{Vect}) \]

\[
\text{LinearOrdinaryDifferentialOperator2}( \\
\text{SquareMatrix}(3, \text{UnivariatePolynomial}(x, \text{Integer})), \\
\text{DirectProductMatrixModule}(3, \\
\text{UnivariatePolynomial}(x, \text{Integer}), \\
\text{SquareMatrix}(3, \text{UnivariatePolynomial}(x, \text{Integer})), \\
\text{UnivariatePolynomial}(x, \text{Integer})))
\]

\[ \text{Type: Domain} \]
a :  Modo := Dx + m

\[ D + \begin{bmatrix} x^2 & 1 & 0 \\ 1 & x^4 & 0 \\ 0 & 0 & 4x^2 \end{bmatrix} \]

Type: LinearOrdinaryDifferentialOperator2(
  SquareMatrix(3, UnivariatePolynomial(x, Integer)),
  DirectProductMatrixModule(3, UnivariatePolynomial(x, Integer),
  SquareMatrix(3, UnivariatePolynomial(x, Integer)),
  UnivariatePolynomial(x, Integer)))

b :  Modo := m*Dx + 1

\[ \begin{bmatrix} x^2 & 1 & 0 \\ 1 & x^4 & 0 \\ 0 & 0 & 4x^2 \end{bmatrix} D + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Type: LinearOrdinaryDifferentialOperator2( SquareMatrix(3, UnivariatePolynomial(x, Integer)), DirectProductMatrixModule(3, UnivariatePolynomial(x, Integer), SquareMatrix(3, UnivariatePolynomial(x, Integer)), UnivariatePolynomial(x, Integer))))
Axiom

• Great concept for building well-structured and flexible libraries.

• Not enough “dogfooding.”
• Top-level tried to hide types from user, but was not sufficiently successful at doing that.
• Powerful and flexible, but too complex for most users.

• Now open source.
Example 3: Aldor

- Re-design of Axiom language 1984 on.
- Initiated by W.

- Efficiency, elegance, take no prisoners
- Nothing special about built-in types

- Dependent types everywhere
- Interoperability with C, Fortran and Lisp
Aldor and Its Type System

- Types and functions are values
  - May be created dynamically

- The type system has two levels
  - Each value belongs to a unique type, its domain, which is known statically and gives the representation.
  - The domains are values with domain Domain.
  - Each value may belong to any number of subtypes of its domain.
  - Subtypes of Domain are categories.

- Categories
  - Specify the exports (operations, constants) a domain must provide.
  - Fill the role of interfaces or abstract base classes in OO languages.
Why Two Levels?

- **OO problem with multi-argument functions:**

```java
class SG { "*": (SG, SG) -> SG; }
DoubleFloat extends SG ...
Permutation extends SG ...

x, y ∈ DoubleFloat ⊂ SG
p, q ∈ Permutation ⊂ SG

x * y ✓
p * q ✓
p * y ✓ ??? Bad, Bad, Bad
```
Why Two Levels?

- **OO problem with multi-argument functions:**

\[
\text{SG} \equiv \ldots \{ \text{"*"} : (\%,\%) \rightarrow \%; \}\]

\text{DoubleFloat: SG ...}

\text{Permutation: SG ...}

\[x, y \in \text{DoubleFloat} \in \text{SG}\]

\[p, q \in \text{Permutation} \in \text{SG}\]

\[x \ast y \checkmark\]

\[p \ast q \checkmark\]

\[p \ast y \times\]
Dependent Types

- Give dynamic typing, e.g.

\[ f: (n: \text{Integer}, R: \text{Ring}, m: \text{IntegerMod}(n)) \rightarrow \text{SqMatrix}(n, R) \]

- Recover OO through dependent products:

\[ \text{prodl}: \text{List Record}(S: \text{Semigroup}, s: S) = [ \begin{array}{c} \text{DoubleFloat}, x \\ \text{Permutation}, p \\ \text{DoubleFloat}, y \end{array} ] \]

- With categories, guarantee required operations available:

\[ f(R: \text{Ring})(a: R, b: R): R = a \times b + b \times a \]
Multi-sorted Algebras

- Category signature as a dependent product type.

```haskell
RationalModel: Category == with {
  Nat: IntegralDomain;
  Rat: Field;
  (/): (Nat, Nat) -> Rat;
}
```
Parametric Polymorphism

- PP is via category- and domain-producing functions.

```plaintext
factorial(n: Integer): Integer == if n = 0 then 1 else n*factorial(n-1)

Module(R: Ring): Category == Ring with { *: (R, %) -> % }

Complex(R: Ring): Module(R) with {
  complex: (%,%)-gt R; real: %->R; imag: %->R; conj: % -> %; ...
} == add {
  Rep == Record(real: R, imag: R);
  0: % == ...
  1: % == ...
  (x: %) + (y: %): % == ...
}
```
Using Genericity

LinearOrdinaryDifferentialOperator(
    A: DifferentialRing,
    M: LeftModule(A) with differentiate: % -> %
) : ...
== SUP(A) add {
    ...
    if A has Field then {
        Op == OppositeOperator(% , A);
        DODiv == NonCommutativeOperatorDivision(%, A);
        OPdiv == NonCommutativeOperatorDivision(Op, A);

        leftDivide (a,b) == leftDivide(a, b)$DODiv;
        rightDivide(a,b) == leftDivide(a, b)$OPdiv;
    }
    ...
}
Conditional Categories

DirectProduct(n: Integer, S: Set): Set with {
    component: (Integer, %) -> S;
    new: Tuple S -> %;
    if S has Semigroup then Semigroup;
    if S has Monoid then Monoid;
    if S has Group then Group;
    ...
    if S has Ring then Join(Ring, Module(S));
    if S has Field then Join(Ring, VectorField(S));
    ...
    if S has DifferentialRing then DifferentialRing;
    if S has Ordered then Ordered;
    ...
} == add { ... }
Better with *Post Facto* Extension

DirectProduct\((n: \text{Integer}, S: \text{Set})\): Set with {
    component: (Integer, \%) \to S;
    new: \text{Tuple } S \to \%
}\} \equiv \text{add } \{ \ldots \}

\begin{align*}
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Semigroup}): \text{Semigroup} \equiv & \ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Monoid}): \text{Monoid} \equiv & \ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Group}): \text{Group} \equiv & \ldots \\
\ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Ring}): \text{Join(Ring, Module}(S)) \equiv & \ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Field}): \text{Join(Ring, VectorField}(S)) \equiv & \ldots \\
\ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Field}): \text{Join(Ring, VectorField}(S)) \equiv & \ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{DifferentialRing}): \text{DifferentialRing} \equiv & \ldots \\
\text{extend } \text{DirectProduct}(n: \text{Integer}, S: \text{Ordered}): \text{Ordered} \equiv & \ldots \\
\ldots \\
\end{align*}

- Normally these extensions would all be in separate files.
Higher Order Operations

- E.g. Reorganizing constructions
  \[ \text{Polynomial}(x) \text{ Matrix}(n) \text{ Complex } \mathbb{R} \approx \text{Complex} \text{ Matrix}(n) \text{ Polynomial}(x) \text{ R} \]

- Simpler example
  \[ \text{List} \text{ Array} \text{ String} \text{ R} \approx \text{String} \text{ Array} \text{ List} \text{ R} \]
Higher Order Operations

Ag ==> (S: BasicType) -> LinearAggregate S;

swap(X:Ag, Y:Ag)(S:BasicType)(x:X Y S):Y X S ==
[[s for s in y] for y in x];

al: Array List Integer :=
array(list(i+j-1 for i in 1..3) for j in 1..3);

la: List Array Integer :=
swap(Array, List)(Integer)(al);
Design Principles

- Language-defined types should have no privilege whatsoever over application-defined types.
  - Syntax, semantics (e.g. in type exprs), optimization (e.g. constant folding)

- Language semantics should be independent of type.
  - E.g. named constants overloaded, not functions

- Combining libraries should be easy, $O(n)$, not $O(n^2)$.
  - Should be able to extend existing things with new concepts without touching old files or recompiling.
  - Full support for inter-language communication: C, Fortran, Lisp

- Safety through optimization removing run-time checks, not by leaving off the checks in the first place.
The Compiler as an Artefact

- Written primarily in C (C++ too immature in 1990)
- 1550 files, 295 K loc C + 65 K loc Aldor

Intermediate code (FOAM):
- Primitive types: booleans, bytes, chars, numeric, arrays, closures
- Primitive operations: data access, control, data operations

Runtime system:
- Memory management
- Big integers
- Stack unwinding
- Export lookup from domains
- Dynamic linking
- Written in C and Aldor
Example

generator(seg:Segment Int):Generator Int
== generate {
  a := lo seg; b := hi seg;
  while a <= b repeat { yield a; a := a + 1 }
}

client() == {
  ar := array(...);
  s := 0;
  for i in 1..#ar repeat s := s + a.i;
  stdout << s
}
How Generators Work

generator(seg:Segment Int):Generator Int
  ==
  generate {
    a := lo seg; b := hi seg;
    while a <= b repeat { yield a; a := a + 1 }
  }

client() == {
  ar := array(...);
  s := 0;
  for i in 1..#ar repeat s := s + a.i;
  stdout << s
}

- Generator(T) exports
  empty?: % -> Boolean
  next!: % -> T
  step!: % -> ()

- These are used by generate/yield and for/repeat.
- Essentially computed gotos.
Parallel Iteration

From the domain Segment(E: OrderedAbelianMonoid)
generator(seg:Segment E):Generator E == generate {
  (a, b) := (low seg, hi seg);
  while a <= b repeat { yield a; a := a + 1 }
}

From the domain List(S: Set)
generator(l: List S): Generator S == generate {
  while not null? l repeat { yield first l; l := rest l }
}

Client code
client() == {
  ar := array(...); li := list(...);
  s := 0; -- NOTE PARALLEL TRAVERSAL.
  for i in 1..#ar for e in l repeat { s := s + ar.i + e }
  stdout << s
}
Inlined

B0:  ar := array(...);
     l := list(...);
     segment := 1..#ar;
     lab1 := B2;
     l2 := l;
     lab2 := B9;
     s := 0;
     goto B1;
B1:  goto @lab1;
B2:  a := segment.lo;
     b := segment.hi;
     goto B3;
B3:  if a > b then goto B6; else goto B4;
B4:  lab1 := B5;
     val1 := a;
     goto B7;
B5:  a := a + 1
     goto B3;
B6:  lab1 := B7;
     goto B7;
B7:  if lab1 == B7 then goto B16; else goto B8
B8:  i := val1;
     goto @lab2;
B9:  goto B10
B10: if null? l2 then goto B13; else goto B11
B11: lab2 := B12
     val2 := first l2;
     goto B14;
B12: l2 := rest l2
     goto B10
B13: lab2 := B14
     goto B14
B14: if lab2 == B14 then goto B16; else goto
B15:  e := val2;
     s := s + ar.i + e
     goto B1;
B16: stdout << s
Clone Blocks for 1st Iterator
### Dataflow Analysis

<table>
<thead>
<tr>
<th>Block</th>
<th>Preds</th>
<th>Succs</th>
<th>Gen</th>
<th>Kill</th>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>B1a</td>
<td>B1a</td>
<td>1..11</td>
<td></td>
<td>1..</td>
<td>1..</td>
</tr>
<tr>
<td>B1a</td>
<td>B0</td>
<td>B2</td>
<td>B5</td>
<td>B7a</td>
<td>1..1</td>
<td>1..1</td>
</tr>
<tr>
<td>B1b</td>
<td>B15</td>
<td>B2</td>
<td>B5</td>
<td>B7a</td>
<td>1..11</td>
<td>11.11</td>
</tr>
<tr>
<td>B2</td>
<td>B1a B1b</td>
<td>B3</td>
<td>11.11</td>
<td></td>
<td></td>
<td>1..</td>
</tr>
<tr>
<td>B3</td>
<td>B2</td>
<td>B5</td>
<td>B6</td>
<td>B4</td>
<td>11.11</td>
<td>11.11</td>
</tr>
<tr>
<td>B4</td>
<td>B3</td>
<td>B7b</td>
<td>1..11</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B5</td>
<td>B1a B1b</td>
<td>B3</td>
<td>11.11</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B6</td>
<td>B3</td>
<td>B7c</td>
<td>11.11</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B7a</td>
<td>B1a B1b</td>
<td>B8</td>
<td>B16</td>
<td>11.11</td>
<td>11.11</td>
<td></td>
</tr>
<tr>
<td>B7b</td>
<td>B4</td>
<td>B8</td>
<td>B16</td>
<td>11.11</td>
<td>11.11</td>
<td></td>
</tr>
<tr>
<td>B7c</td>
<td>B6</td>
<td>B8</td>
<td>B16</td>
<td>11.11</td>
<td>11.11</td>
<td></td>
</tr>
<tr>
<td>B8</td>
<td>B7a B7b B7c B9 B12 B14</td>
<td>111111</td>
<td></td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B9</td>
<td>B8</td>
<td>B10</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B10</td>
<td>B9 B12</td>
<td>B11 B13</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B11</td>
<td>B10</td>
<td>B14</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B12</td>
<td>B8</td>
<td>B10</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B13</td>
<td>B10</td>
<td>B14</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B14</td>
<td>B8 B11 B13 B17 B15</td>
<td>11111</td>
<td></td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B15</td>
<td>B14</td>
<td>B1b</td>
<td>11111</td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B16</td>
<td>B7a B7b B7c B17</td>
<td>11111</td>
<td></td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
<tr>
<td>B17</td>
<td>B16 B14</td>
<td>111111</td>
<td></td>
<td></td>
<td></td>
<td>1..1</td>
</tr>
</tbody>
</table>

[lab1 == B2, lab1 == B5, lab1 == B7]
Resolution of 1st Iterator

\[
\text{lab2} := \text{B10}
\]

\[
\text{a > b}
\]

\[
\text{null?}
\]

\[
\text{lab2} := \text{B12}
\]

\[
\text{lab2} := \text{B14}
\]

\[
\text{lab2} = \text{B14}
\]

\[
\text{done}
\]

\[
a := a + 1
\]
Resolution of 2\textsuperscript{nd} Iterator

```c
client() == {
    ar := array(...);
    l := list(...);
    l2 := l;
    s := 0;
    a := 1;
    b := #ar;
    if a > b then goto L2
L1: if null? l2 then goto L2
    e := first l2;
    s := s + ar.a + e
    a := a + 1
    if a > b then goto L2
    l2 := rest l2
    goto L1
L2: stdout << s
}
```
Aldor vs C

(Aldor = Red, C = Green)

Geometric Mean

Q2/00  Q3/01  Q4/02  Q5/03

Optimisation Level
Aldor Lessons

- It is possible to be elegant, abstract and high-level without sacrificing significant efficiency.
- Well-known optimization techniques can be effectively adapted to the symbolic setting.
- Optimization of generated C code is not enough.

- Procedural integration, dataflow analysis, subexpression elimination and constant folding are the primary wins.
- Compile-time memory optimization, including data structure elimination, is important.
  - Removes boxing/unboxing, closure creation, dynamic allocation of local objects, etc. Can move hot fields into registers.
Further Lessons

- Language design 20+ years old.
  - In the mean time, many of the ideas now mainstream.
  - Some still being adopted.

- Mathematics is a valuable canary in the coal mine of general purpose software.
  - The general world lags in recognizing needs.

- It has to be free.
  - Free$^1$ is the standard price.
  - Free$^2$ is required for engagement.
The Frontier

- Identities (for humans, provers and compilers)
  
  \[ a+b \iff b+a \]
  
  \[ \text{length}(l) = n \implies \text{isLength}(l, n) \]

- More tools for homomorphisms
  - Embeddings, representations, …

- More tools for families of related types

- Functors to move between views
  - \( \implies \) eliminate exponential type expression size
Example 4: OpenMath/MathML

OpenMath

- Initiated at Maplesoft in late 1980s for communication between algebra engine and interface.
- Broader participation invited in early 1990s, leading to OpenMath society.
- European projects: OpenMath, MONET, …. 

- Data language
- Binary and text formats (later XML)
- Primarily used in special projects.
MathML

- Unfulfilled `<math>` element in HTML 3.2 Jan 1997.

- Initial, unchartered Math WG defining microsyntax for `<math>`. 

- Internecine rivalry between syntax and semantics camps coming from TeX, Mathematica and SGML.

- Whatever it was to be, it was destined to have broad impact.
\[
\int_C \, d\omega = \int_{\partial C} \omega
\]
\[
\left( \frac{p}{q} \right) \left( \frac{q}{p} \right) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}
\]
\[
G(E/F) = G(K/F) / G(K/E)
\]
\[
\nabla^\mu \nabla_\mu A^\nu - \nabla^\nu \nabla_\mu A^\mu = j^\nu
\]
\[
\partial_{n-1} \partial_n \ c = 0
\]

</math>
MathML

- W. convened “HTML-native” math group at Yorktown Heights in Jan 1997 to form unified proposal.
- First ever XML application.
- “Presentation” and “Content” aspects.
- Swallowed OpenMath in V3
- Supported in major browsers, computer algebra systems, incorporated in HTML 5.
MathML Lessons

- Play well with, and leverage, select broadly applicable technologies
- Find a niche in the ecosystem
- Catch the wave
Example 5: InkML

- Ink Messaging
- Annotation
- Archival
Pen-Based Math

- Input for CAS and document processing.
- 2D editing.
- Computer-mediated collaboration.
Pen-Based Math

- Different than natural language recognition:
  - 2-D layout is a combination of writing and drawing.
  - No fixed dictionary.
  - Many similar few-stroke characters.
  - Well segmented.
  - Highly ambiguous

\[
\sum_{i} z^2 \quad i + z = \sin \omega t
\]
Digital Ink

- Collected by surface digitizer or camera
- Sequence of \((x, y)\) points sampled at some known frequency
- Possibly other info (angles, pressure, etc)
- Grouping into traces, letters, words + labelling
Ink Markup Language (InkML)

W3C Recommendation 20 September 2011

This version:  
http://www.w3.org/TR/2011/REC-InkML-20110920/

Latest version:  
http://www.w3.org/TR/InkML

Previous version:  
http://www.w3.org/TR/2011/PR-InkML-20110510/

Editors:  
Stephen M. Watt, University of Western Ontario and Maplesoft  
Tom Underhill, Microsoft

Authors:  
Yi-Min Chee (until 2006 while at IBM)  
Katrin Franke (until 2004 while at Fraunhofer Gesellschaft)  
Max Froumentin (until 2006 while at W3C)  
Sriganesh Madhvanath (until 2009 while at HP)  
Jose-Antonio Magaña (until 2006 while at HP)  
Grégory Pakosz (until 2007 while at Vision Objects)  
Gregory Russell (until 2005 while at IBM)  
Muthuselvam Selvaraj (until 2009 while at HP)  
Giovanni Seni (until 2003 while at Motorola)  
Christopher Tremblay (until 2003 while at Corel)  
Larry Yaeger (until 2004 while at Apple)
InkML Concepts

- Traces, trace groups
- Device information: sampling rate, resolution, etc.
- Pre-defined and application defined channels
- Trace formats, coordinate transformations
- Streaming and archival
- Annotation text and XML
InkML Evolution

- Started as low-level language for traces and hardware description. Explicitly disavowed semantics.

- Wanted base language sufficiently rich to support full range of digital ink applications. Semantic grouping added, annotation, etc.

- W3C Standard
- Built in to Microsoft Office 2010
Research: Symbol Recognition

- **Main idea:** Represent coordinate curves as truncated orthogonal series.

- **Advantages:**
  - *Compact* – few coefficients needed
  - *Geometric*
    - the truncation order is a property of the character set
    - gives a natural metric on the space of characters
  - *Algebraic*
    - properties of curves can be computed algebraically
      (instead of numerically using heuristic parameters)
  - *Device independent*
    - resolution of the device is not important
Choose a functional inner product, e.g.

\[ \langle f, g \rangle = \int_{-1}^{1} f(t)g(t)dt + \mu_1 \int_{-1}^{1} f'(t)g'(t)dt + \mu_2 \int_{-1}^{1} f''(t)g''(t)dt + \cdots \]

This determines an orthonormal basis in the subspace of polynomials of degree \( d \).
Determine \( \phi_i \) using GS on \{1, t, t^2, t^3, \ldots \}.

Can then approximate functions in subspaces

\[ A(t) \approx \sum_{i=0}^{d} \alpha_i \phi_i(t) \quad \alpha_i = \langle A(t), \phi_i(t) \rangle \]
Like Symbols form Clusters
A Problem

- In handwriting recognition, the human and the computer take turns thinking and sitting idle.

- We ask:
  Can the computer do useful work while the user is writing and thereby get the answer faster after the user stops writing?

- We show:
  The answer is “Yes”!
On-Line Series Coefficients

- If we choose the right basis functions, then the series coefficients can be computed on line. [Golubitsky+SMW CASCON 2008, ICFHR 2008]

- The series coefficients are linear combinations of the moments, which can be computed by numerical integration as the points are received.

- This is the Hausdorff moment problem (1921), shown to be unstable by Talenti (1987).

- It is just fine, however, for the orders we need.
Distance Between Curves

- Matching based on distance between curves
- Usually:
  - Approximate the variation between curves by some fn of distances between points.
  - May be coordinate curves or curves in a jet space.
  - Sequence alignment
  - Interpolation (“resampling”)

- Why not just calculate the area?
- This is very fast in ortho series representation.
Better Distance Between Curves

$$\bar{x}(t) = x(t) + \xi(t) \quad \xi(t) = \sum_{i=0}^{\infty} \xi_i B_i(t)$$

$$\bar{y}(t) = y(t) + \eta(t) \quad \eta(t) = \sum_{i=0}^{\infty} \eta_i B_i(t)$$

$B_i$ orthonormal on $[a, b]$ with $w(t) = 1$.

$$\rho^2(C, \bar{C}) = \int_{a}^{b} \left[ (x(t) - \bar{x}(t))^2 + (y(t) - \bar{y}(t))^2 \right] \, dt$$

$$= \int_{a}^{b} \left[ (\xi(t))^2 + (\eta(t))^2 \right] \, dt$$

$$\approx \int_{a}^{b} \left[ \sum_{i=0}^{d} \xi_i^2 B_i^2(t) + \text{cross terms} + \sum_{i=0}^{d} \eta_i^2 B_i^2(t) + \text{cross terms} \right] \, dt$$

$$= \sum_{i=0}^{d} \xi_i^2 + \sum_{i=0}^{d} \eta_i^2$$
Comparison of Candidate to Models

- Use Euclidean distance in the coefficient space.

- *Just as accurate* as elastic matching.

- *Much less expensive.*

- Linear in $d$, the degree of the approximation.
  
  $< 3d$ machine instructions (30ns) *vs* several thousand!

- Can trace through SVM-induced cells incrementally.

- Normed space for characters gives other advantages.
Distance-Based Classification

Red class or blue class?
Distance-Based Classification

The nearest $k$ samples are blue.
The Joy of Convexity

- Classes 99% linearly separable
- Linear homotopies lie within a class

\[ C = (1 - t) A + t B \]

- Can compute distance of a sample to this line
- Convex hull of a set of models
Distance-Based Classification

The nearest convex hull of neighbors is red.
Error Rates as Fn of Distance

- Error rate as fn of distance gives confidence measure for classifiers [MKM – Golubitsky + SMW]
Recognition Summary

- Database of samples => set of LS points
- Character to recognize =>
  - Integrate moments as being written
  - Lin. trans. to obtain one point in LS space
  - Classify by distance to convex hull of k-NN.

- InkML allows natural representation of annotated database and real-time input.
Overall Conclusions

- Mathematical problems provide excellent challenges for language design.
  - Rich, complex, hard
  - Well-defined
  - Performance matters – a lot!

- Don’t be put off by the loud, confident proclamations of mass-market language designers.

- Lots left to do!