Domain Engineering of the PDE Domain

Magne Haveraaen

Bergen Language Design Laboratory (BLDL)
Department of Informatics, University of Bergen, Norway

INF329 Selected Topics in Programming Technology
Bergen, 2012-03-07

Defining the Core Assets of a Domain

Must fit the language of software

Algorithms + Data Structures = Programs

Niklaus Wirth 1976

- A **Data Structure** *abstracts to* a **type**
  - Values of a type can be compared for **equality**
- An **Algorithm** *abstracts to* a **function**
  - Input argument list
  - Result type
- Properties of a type are defined by **predicates** on expressions

```
T a,b,c;
assert ( (a+b)+c == a+(b+c) );
```

Magne Haveraaen • Bergen Langauge Design Laboratory • http://bldl.ii.uib.no

Questions to ask of a Domain

- What are the **types**
- What are the **functions**
- What are the **axioms**

("Jim " + "J ") + "Horning"
== "Jim J Horning"
== "Jim " + ("J " + "Horning");

"Guttag" + "" == "Guttag" == "" + "Guttag"

That is, what are the (C++) **concepts**

```
template<typename m>
concept monoid (binary<m> bin, nullary<m> unit) {
  axiom associative (m a, m b, m c) {
    assert bin(bin(a,b),c) == bin(a,bin(b,c));
  }
  axiom neutral (m a) {
    assert bin(a,unit()) == a;
    assert bin(unit(),a) == a;
  }
}
```


Data Structure Algebra

Isomorphisms

- The same information content for different declarations

```
struct {
    int a[100];
    int b[100];
} d1;
```

```
struct D {
    int a;
    int b;
};
D d2[100];
```

- Alternative data structures
  - Different access patterns
  - Different *abstractions*

Concepts for Arithmetic Operations - 1

```
template<typename r>
concept unit_ring(binary<r> plus, unary<r> minus, binary<r> mult) {
  axiom abelian_group(r a, r b, r c) {
    assert plus(plus(a,b),c) == plus(a,plus(b,c));
    assert plus(a,b) == plus(b,a);
    assert plus(a,r(0)) == a;
    assert plus(a, minus(a)) == r(0);
  }
  axiom monoid(r a, r b, r c) {
    assert mult(mult(a,b),c) == mult(a,mult(b,c));
    assert mult(a,r(1)) == a;
    assert mult(r(1),a) == a;
  }
  axiom distributive(r a, r b, r c) {
    assert mult(a,plus(b,c)) == plus(mult(a,b),mult(a,c));
    assert mult(plus(a,b),c) == plus(mult(a,c),mult(b,c));
  }
}
```


Some Examples of Unit Rings

- The integers with +,-,*
- The reals with +,-,*
- The rational numbers with +,-,*
- The complex numbers C<r> with +,-,* where r is a unit ring
- Polynomials P<r> with +,-,* where r is a unit ring
- Matrices M<r> with +,-,∘ where r is a unit ring
- Arrays A<r> with pointwise +,-,*, where r is a unit ring

Concepts for Arithmetic Operations - 2

```
template<typename r>
concept commutative_unit_ring
    (binary<r> plus, unary<r> minus, binary<r> mult) {
  require unit_ring(plus,minus,mult);
  axiom commutative(r a, r b) {
    assert mult(a,b) == mult(b,a);
  }
}

template<typename r>
concept field
    (binary<r> plus, unary<r> minus, binary<r> mult, unary<r> inv) {
  require commutative_unit_ring(plus,minus,mult);
  axiom nontrivial(r a) {
    assert !(r(0) == r(1));
  }
  axiom inverse(r a) {
    assert !(a==r(0)) => mult(a,inv(a)) == r(1);
    assert !(a==r(0)) => mult(inv(a),a) == r(1);
  }
}
```


Linear Algebra Types

- Scalars (0-indexed)
  - Real numbers, complex numbers
  - *Temperature*
  - *Pressure*
- Vector (1-indexed)
  - Describes direction and magnitude
  - *Velocity*
  - *Displacement*
- Matrix (2-indexed)
  - Linear mapping from vector to vector
  - *Change of coordinate system*
- Tensor (k-indexed), rank k for $0 \leq k$

Magne Haveraaen • Bergen Langauge Design Laboratory • http://bldl.ii.uib.no

Data Fields

A value at every point in a spatial and/or temporal domain

- Scalar field
  - Scalar value at every point
  - *Temperature and pressure in a room*
- Vector field
  - Vector value at every point
  - *Air flow at every point in a room*
- Matrix field
  - Matrix value at every point
  - *Stress and strain distributed in a material, e.g., earth's crust*
- Tensor field
  - *Stiffness: rank 4 tensor at every point in a material*

Derivatives on Data Fields

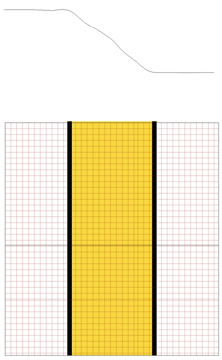
Measures rates of change in a data field

- Partial derivatives: scalar field to scalar field
  - $\partial/\partial t$ – partial derivative in time
  - $\partial/\partial x_i$ – partial derivative in spatial direction i
- Gradient ∇, a rank increasing spatial derivative
  - Scalar field to vector field
  - Vector field to matrix field
  - Computed by partial derivatives on tensor components
- Divergence ∇·, a rank decreasing spatial derivative
  - Vector field to scalar field
  - Matrix field to vector field
  - Computed by partial derivatives on tensor components

The Heat Equation

Temperature across the wall

warm



cold

Grid for the wall and surroundings

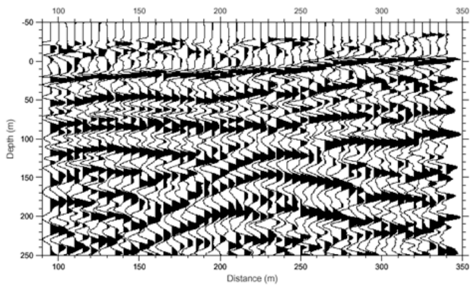
$$\frac{\partial}{\partial t} u = \alpha * (\nabla \cdot (\nabla u)) + f$$

Variables, in space and time
u – temperature, scalar field
α – thermal diffusivity, scalar field
f – heat source, scalar field

Derivatives
∂/∂t – partial derivative in time
∇ – gradient, scalar field to vector field
∇· – divergence, vector field to scalar field

Operations
* – scalar field multiplication
+ – scalar field addition

Seismic Waves



$$\rho \frac{\partial}{\partial t} \frac{\partial}{\partial t} u = \nabla \cdot \sigma + f,$$
$$\sigma = \Lambda \circ e,$$
$$e = L(u, g)$$

Elastic wave equation

Variables
ρ – density, scalar field
u – displacement, vector field
σ – stress, matrix field
f – external force, vector field
Λ – stiffness, tensor field
e – strain, matrix field
g – metric, matrix field

Derivatives
∂/∂t – partial derivative in time
∇· – divergence, matrix field to vector field
L – Lie derivative, matrix field to matris field

Operations
∘ – tensor application, returns matrix field
+ – vector field addition

Engineering the PDE domain

- Data field df<r>: a value of type r at every point in space-time
  - Scalar field sf<real>, ring with pointwise +,-,* and ∂/∂t, ∂/∂x, ..
- Tensor tensor<k,r> with +,-,∘ from any ring r and rank k
- Tensor field with ∇·, ∇, alternative data structures
  - df<tensor<k,real>>
  - tensor<k,sf<real>>

Choosing matrix field format: consider the derivation operations

- Derivatives require access to neighbouring data
- Scalar field has partial derivatives ∂/∂t, ∂/∂x, ..
  - The derivations can be defined from partial derivatives

tensor<k,sf<real>> will give more reuse than df<tensor<k,real>>

Dot Product Problem

```
template<typename r> r dot(vector<r> a, vector<r> b) {
  return ∑_i a[i] * b[i];
}
template<typename r> vector<r> new_coordinate( matrix<r> m, vector<r> v) {
  return mm(m,v);
}

template<typename r>
concept dot_properties () {
  axiom coordinate_system_invariance(matrix<r> m, vector<r> u, vector<r> v) {
    assert dot(u,v) == dot(new_coordinate(m,u),new_coordinate(m,v));
  }
  // ...
}
```

- Dot algorithm is wrong? Take coordinate system into account
- Typing is wrong? Vector and covector
- Change of coordinate algorithm is wrong? Covectors are different

Conclusions

- Domain engineering
  - Defines the core assets of a software domain
  - Essential for software product lines
  - Precedes application engineering
- C++ style concepts for core asset development
  - Libraries
    - Declares types, declares functions, defines axioms
    - Drives towards a comprehensive API
  - Architectural considerations
  - Testing
    - Axioms as test oracles
  - Tools: refactoring and optimisation
    - Equational axioms as refactoring rules

Magne Haveraaen • Bergen Langauge Design Laboratory • http://bldl.ii.uib.no