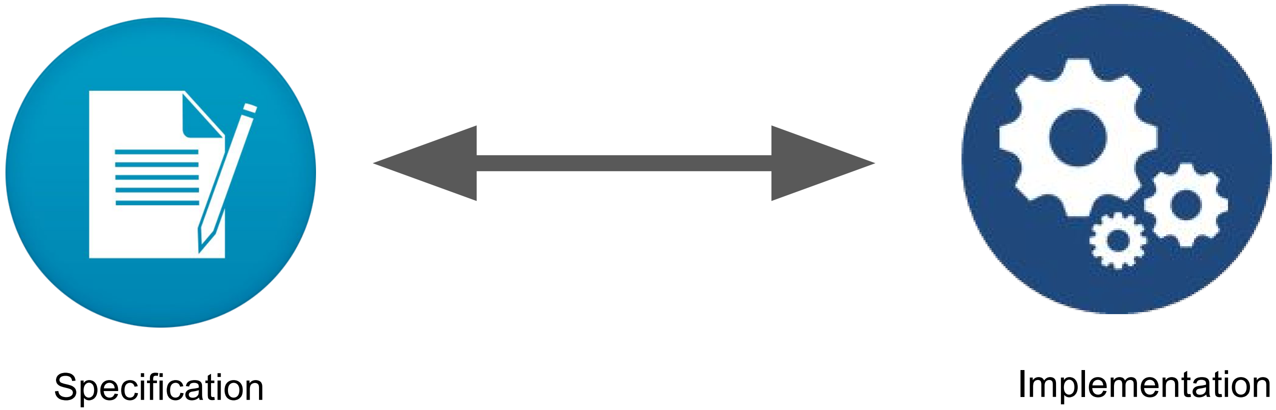
Institutions , Property-Aware Programming and Testing

Ali Alnajjar
Supervisor:Magne Haveraaen

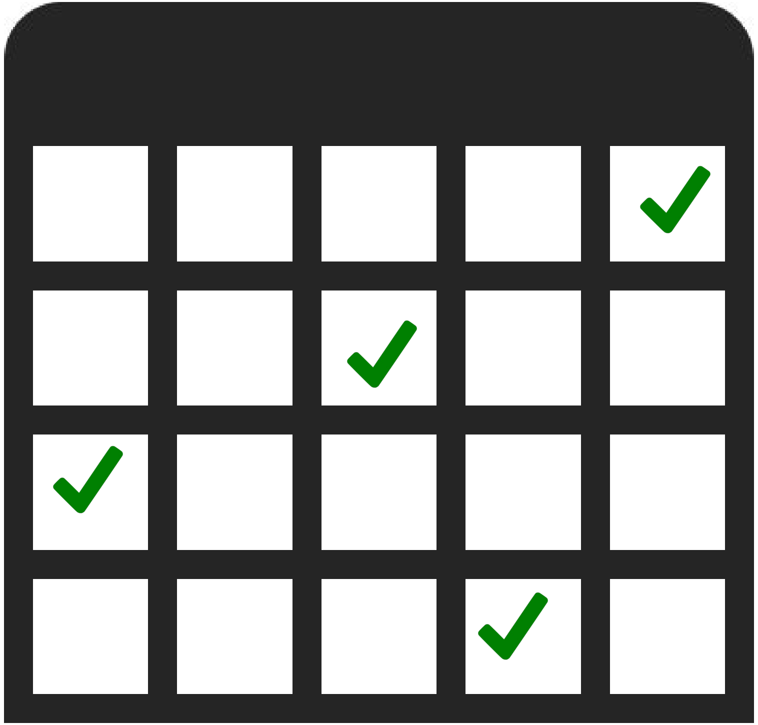
Institutions

Investigate the relationship between specifications and models at a general, theoretical level



Testing

Run the algorithms on **selected** data sets in order to increase our belief in their **correctness**.



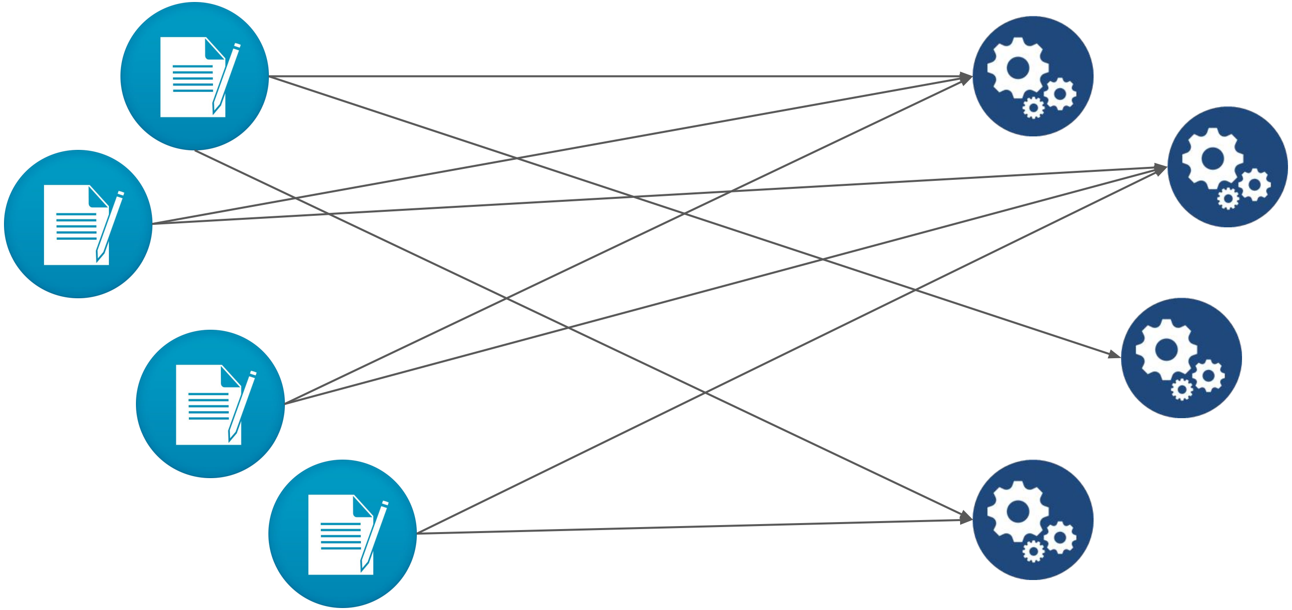
Property-Aware Programming (institutions)

Declaring syntactic and **semantic** properties on generic parameters.

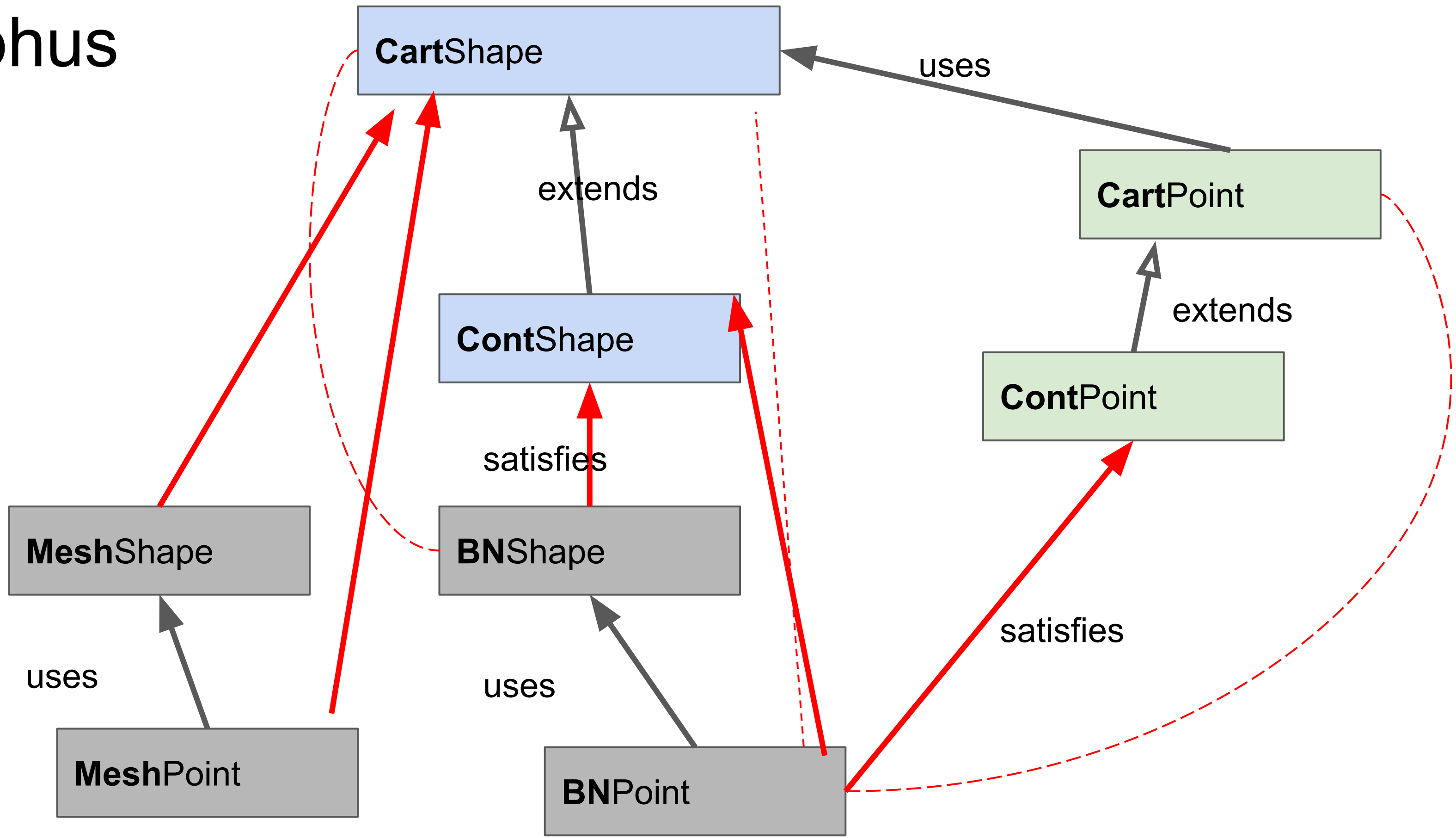
Sophus

- A medium-sized C++ software library developed for solving coordinate-free partial differential equations.
- Developed using algebraic specifications (with a focus on reusability).
- Axiomatic specification.
- Implementation were targeted to be as general as possible.

Sophus



Sophus



CartShape
uses
extends
CartPoint
ContShape
extends
ContPoint
satisfies
MeshShape
BNShape
satisfies
uses
uses
MeshPoint
BNPoint

Sophus

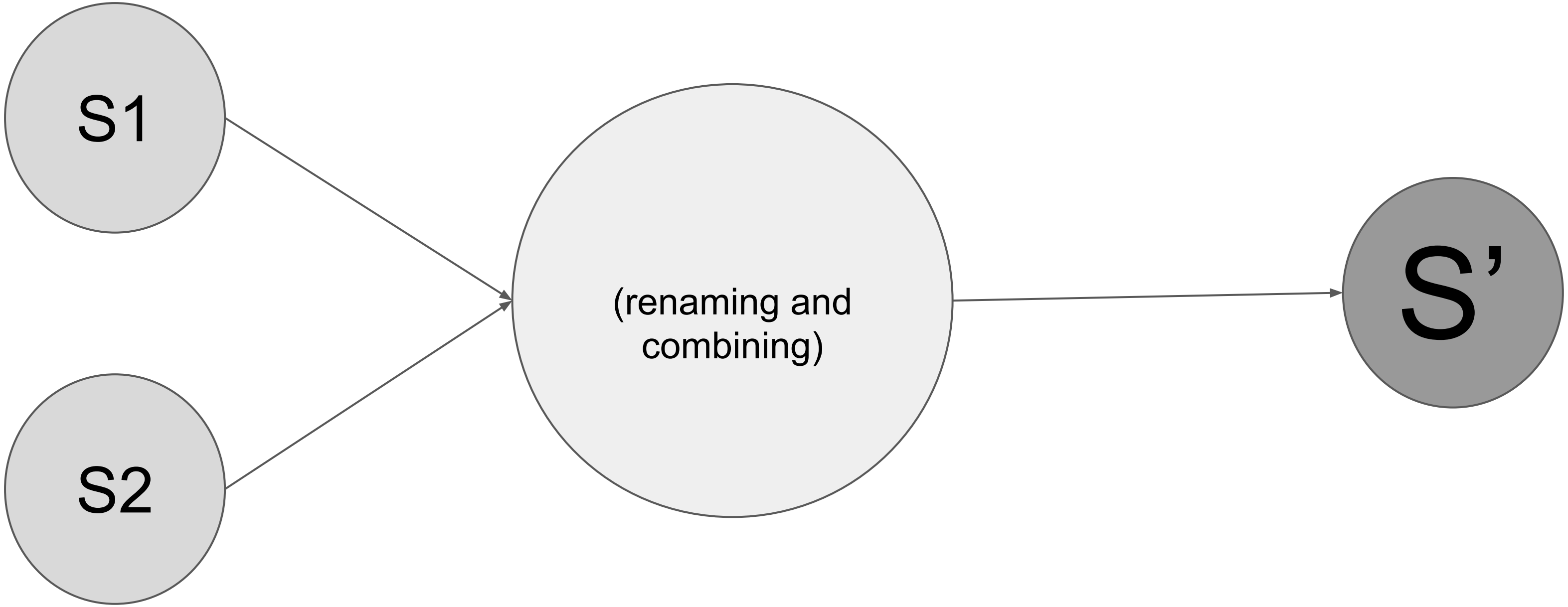
When a specification B in Sophus **uses** another specification A, it means that specification A defines operations and axioms on a sort-set and B **on another sort-set,** even though the sorts of A may be used by operations in B.

When a specification B in Sophus **extends** another specification A, it means that specification A defines operations and axioms on a sort-set and B provides more functiins and axioms **on the same set**.

Institutions: Signatures

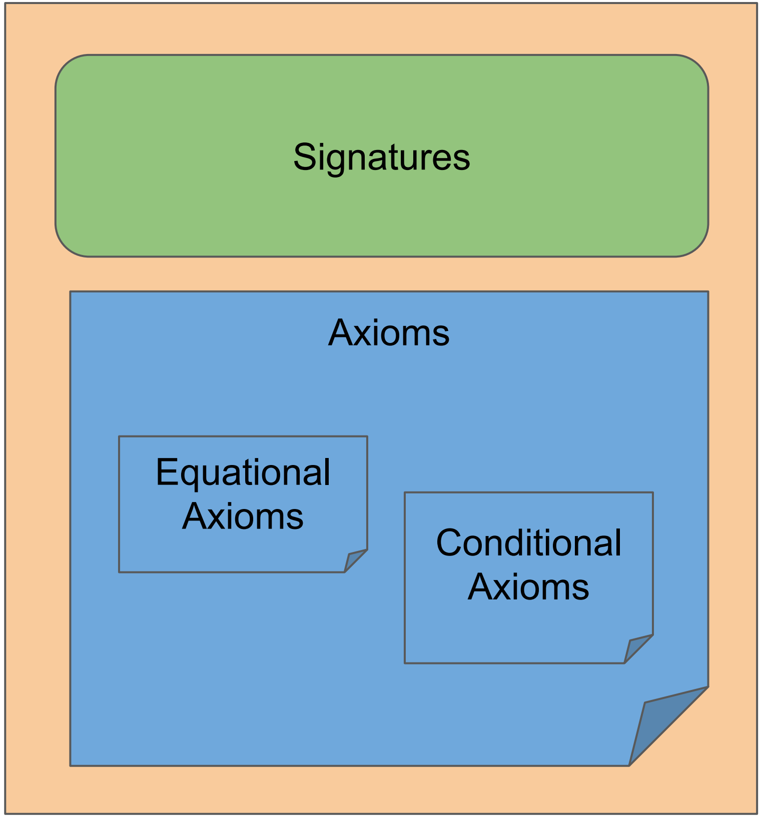
- **Sorts** (Types).
- **Operations** (functions,methods) + **arities** (arguments and return types).
- **Variables.**
- **Terms** (expressions).

Institutions: Signature Morphism



Institutions: Specification

- Can be combined and renamed.



Institutions: Models

- Provide the **semantic** for each signature.
- For each sort define a data structure.

$$S \longrightarrow \text{int}$$

- For each function define an algorithm.

Institutions: Satisfaction

- $M \models \mathbf{ax} \quad \Leftrightarrow \quad \forall(a : V \to M) : M \models_a \mathbf{ax}.$

Implementation

- Sorts → data structures (data invariants)
- Functions → Algorithms

Implementation

- Every algorithm must **preserve the data invariants**: if the input data satisfies the data invariant, so must the output data.
- Every algorithm must **preserve equality**

Testing

- Preservation of the data invariants
- Preservation of the equality. (provided data needed)
- Checking of axioms. (provided data needed)

Testing : Test Set

- $M \models \mathbf{ax} \quad \Leftrightarrow \quad \forall(a : V \to M) : M \models_a \mathbf{ax}.$

$$T \subseteq A(\mathbf{ax}, M)$$

$$M \models_T \mathbf{ax} \quad \Leftrightarrow \quad \forall a \in T : M \models_a \mathbf{ax}.$$

Testing : test reduction hypothesis.

- Random selection hypothesis
- Domain partitioning hypothesis (Discontinuity hypothesis)

Random selection hypothesis

```
boolean CartShapeAxiom1Case0()
{ return CartShapeAxiom1(0); }

boolean CartShapeAxiom1Case1()
{ return CartShapeAxiom1(4); }
```

Domain partitioning hypothesis (Discontinuity hypothesis)

```
boolean CartShapeAxiom2Case0lt0()
{ return CartShapeAxiom2(setDimensions(0),-10); }

boolean CartShapeAxiom2Case0m1()
{ return CartShapeAxiom2(setDimensions(0),-1); }

boolean CartShapeAxiom2Case0eq0()
{ return CartShapeAxiom2(setDimensions(0),0); }

boolean CartShapeAxiom2Case0gt0()
{ return CartShapeAxiom2(setDimensions(0),2); }
```

Domain partitioning hypothesis (Discontinuity hypothesis)

```
boolean CartShapeAxiom2CaseGTlt0()
{ return CartShapeAxiom2(setDimensions(9),-10); }

boolean CartShapeAxiom2CaseGTm1()
{ return CartShapeAxiom2(setDimensions(8),-1); }

boolean CartShapeAxiom2CaseGTeq0()
{ return CartShapeAxiom2(setDimensions(7),0); }

boolean CartShapeAxiom2CaseGTgt0ltGTm1()
{ return CartShapeAxiom2(setDimensions(6),2); }

boolean CartShapeAxiom2CaseGTGTm1()
{ return CartShapeAxiom2(setDimensions(5),4); }

boolean CartShapeAxiom2CaseGTeqGT()
{ return CartShapeAxiom2(setDimensions(4),4); }
```

Questions

Reference to specifications as models

Models Models provide the semantics for each signature. Models transform in the opposite direction of signatures. That is, one may think of a signature renaming as one signature pointing at compo- nents of another signature. Then the latter components are used as models for the former.

The equivalence of satisfaction relation in OO

What is the equivalence of satisfaction relation in OO??