Testing C++ Generic Libraries

Ali Alnajjar
Supervisor:Magne Haveraaen

intro

Ad hoc style tests:

- written against simple concrete inputs e.g arrays of int
- in response to specific defect reports
- exercise only a few specific cases

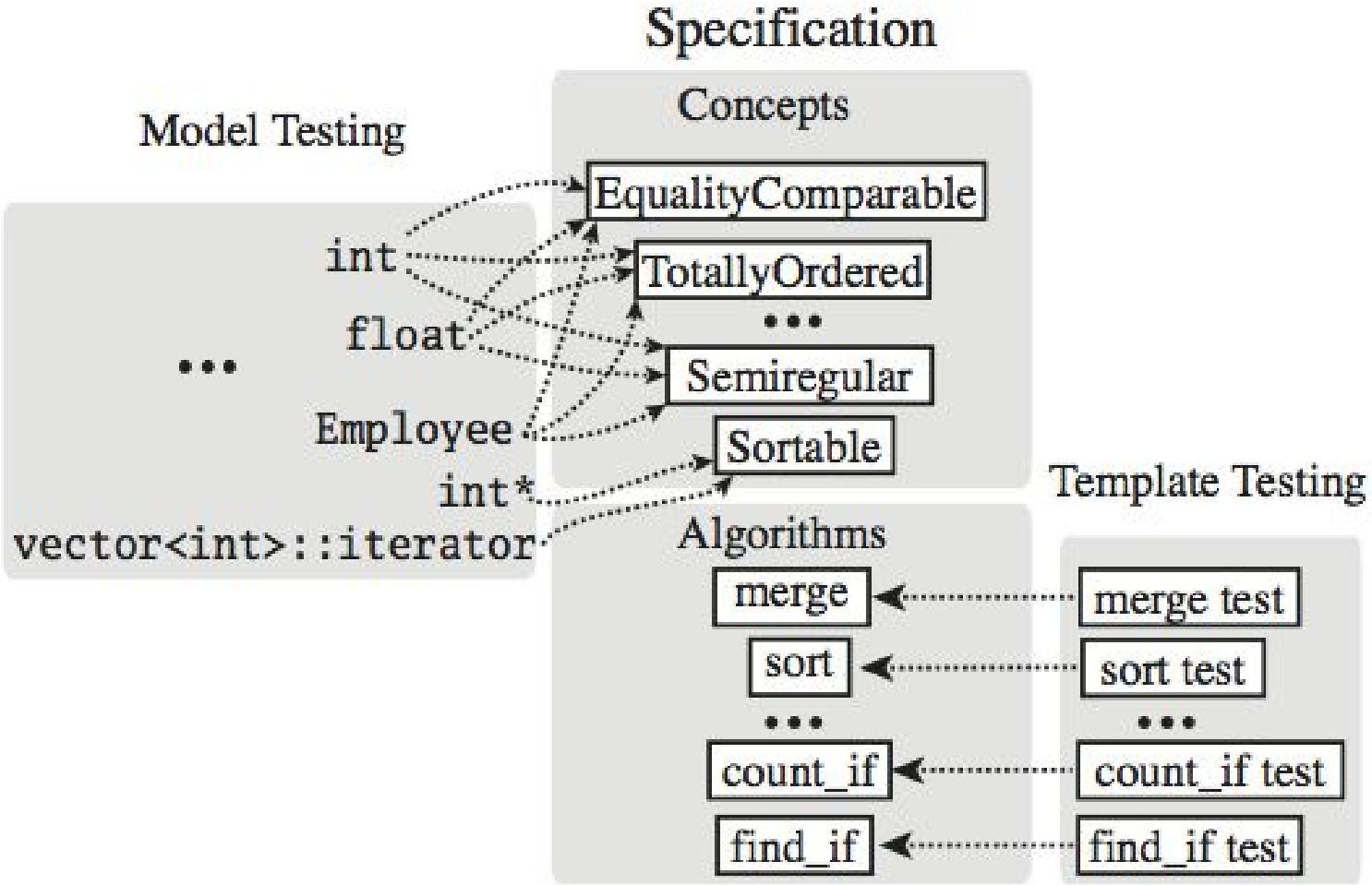
intro

Generic programming:

- concepts.
- templates.
- specifications.

The Contract

the generic interface is the **contract** between the library **designer** and the library **user** that, if kept, guarantees final **correctness**.


Specification
Concepts
Model Testing
EqualityComparable
int
Ttotally Ordered
float
Semiregular
Employee
Sortable
int*
vector<int>::iterator
Algorithms
Template Testing
merge
merge test
sort
sort test
count_if
count_if test
find_if
find_if test

Model Testing

type must :

- provide the interface required by the concept’s type constraints,
- implement the behavior specified by its axioms.

Template Testing

1. Translate a specification into a set of testable properties.
2. Analyze testable properties and implement prototypes
3. Write unit tests using prototypes wrapped by archetypes.

Template Testing - Prototype Testing

representative type : **minimum set of values** needed test a property.

Avoiding isomorphic test values:

<5, 7, 0, 6, 6, 1>, find first x where x == 6

<0, 0, 0, 1, 0, 0>, find first x where x == 1

<1, 2, 3, 4, 4, 9>, find first x where x == 4

Template Testing - Prototype Testing

<0,1,1>, <0,1>, and <0,1,0,1> **are all equivalent**

- $<0^*>$
- $<0^*,1>$

Template Testing - Archetype Testing

Testing of generic algorithms requires the selection of appropriate test values that:

- meet the preconditions
- allow the checking of postconditions and invariants.

An archetype is a class that provides an interface that **exactly** matches template requirements.

Template Testing - Archetype Testing

```
template <typename T>
  struct eq_arch
  {
    eq_arch() = delete;
    eq_arch(const eq_arch&) = delete;
    eq_arch& operator=(const eq_arch&) = delete;

    template <typename... Args>
      eq_arch(dummy_t, Args&&... args) : value(args...) { }

    bool operator==(const eq_arch& x) const
    { return value == x.value; }

    bool operator!=(const eq_arch& x) const
    { return value != x.value; }

    T value;
  }
```