Coq Mechanizations @ KAIST

ERC Workshop (25 June to 28 June, 2011)
Jieung Kim (KAIST) Sunggyeong Bae (KAIST) Sukyoung Ryu (KAIST)

FFMM Mechanization

- Defined a formal calculus that has symmetric multiple dispatch and symmetric multiple inheritance
- Mechanized the calculus and its type safety using a proof assistant tool, Coq

FFMM

Fortress is a programming language for scientists and engineers. Featherweight Fortress with Multiple Dispatch and Multiple Inheritance (FFMM) illustrates a core calculus for Fortress, which has multiple dispatch and multiple inheritance.

- Multiple dispatch:
  Allows method selection among overloaded methods at run time based on dynamic types of more than one method arguments
- Multiple inheritance:
  Allows a type to have more than one super type

Overloading Rules

Informal Description

The rules determine whether a set of overloaded declarations is valid by considering every pair of the declarations in the set independently. A pair is valid if it satisfies one of the following rules:

Exclusion Rule

If the parameter types of the declarations are disjoint types, then the pair is a valid overloading.

Syntax of FFMM

<table>
<thead>
<tr>
<th>τ</th>
<th>met name</th>
<th>O</th>
<th>method name</th>
<th>f</th>
<th>field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td></td>
</tr>
</tbody>
</table>

Subtype Rule

If the parameter type of one declaration is a strict subtype of the parameter type of the other declaration and the return type of the former is a subtype of the return type of the latter, then the pair is a valid overloading.

Meet Rule

If the parameter types of the declarations are not in the subtype relation, then the pair is a valid overloading if there is a declaration whose parameter type is an intersection type of the parameter types of the declarations.

Coq Implementation

<table>
<thead>
<tr>
<th>τ</th>
<th>met name</th>
<th>O</th>
<th>method name</th>
<th>f</th>
<th>field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td></td>
</tr>
</tbody>
</table>

Mathematical Representation

A graph G is called bipartite if it is possible to partition the vertex set of G into two subsets, say V₁ and V₂, so that every edge of G joins a vertex of V₁ with a vertex of V₂, and no vertex joins another vertex of its own set.

Example

We can construct a bigraph with two sets of vertices V₁ and V₂ and a set of edges E using one of the following rules:

1. An empty bigraph is constructed with two empty sets of vertices and an empty set of edges.
2. With a bigraph upon V₁, V₂, and E, by adding a new vertex (V₁ ∪ V₂) to V₁, a bigraph upon V₁ ∪ V₂ and E is constructed.
3. With a bigraph upon V₁, V₂, and E, by adding a new vertex (V₁ ∪ V₂) to V₂, a bigraph upon V₁, V₂ ∪ (V₁ ∩ V₂) and E is constructed.
4. With a bigraph upon V₁, V₂, and E, by adding a new edge such that x ∈ V₁, y ∈ V₁ and (x, y) ∉ E, a graph upon V₁, V₂ and E ∪ {(x, y)} is constructed.

Inductive Definition

Inductive type Bigraph := BG_empty | BG_vertex1 | BG_vertex2 | BG_edge

What Is Next?

- Mechanize proofs of the following theorems:
  - König’s theorem: In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.
  - Equivalences of König’s theorem with seven theorems: the Menger’s Theorem, the König’s theorem for matrices, the König-Egerváry theorem, the Hall’s marriage theorem, the Birkhoff-Von Neumann theorem, the Dilworth’s theorem, and the Max-Flow-Min Cut theorem.
- Implement a new Graph library
  For compatibility with the Sets library in the Coq standard library