0.1 Type Ambiguity Error Handler

0.1.1 Handler Algorithm

Algorithm 1: Handle

Input: L, B, R, M

```
1  ID_L ← {n ∈ A_L | n.type = IMPORTDECL};
2  ID_R ← {n ∈ A_R | n.type = IMPORTDECL};
3  if ID_L = ∅ ∨ ID_R = ∅ then return;
4  M_U ← textualMerge(treeToText(L), treeToText(B), treeToText(R));
5  cs ← extractConflicts(M_U);
6  c ← compile(M_U);
7  ps ← problems(c);
8  foreach l ∈ ID_L do
9      m_l ← extractPackageMember(l.body);
10     foreach r ∈ ID_R do
11         m_r ← extractPackageMember(r.body);
12         if m_l = m_r then
13             p ← importDeclarationsProblem(l, r, ps);
14             if p ≠ null then
15                 m ← find(m ∈ M → m.body = l.body);
16                 m.body ← conflict(l.body, ε, r.body);
17                 m ← find(m ∈ M → m.body = r.body);
18                 removeNode(m, M);
19                 ps ← ps − p;
20                 break;
21             end
22         else if m_l = ∗ ∨ m_r = ∗ then
23             if importDeclarationsConflict(l, r, cs) then
24             end
25         end
26     end
27 end
```

Algorithm 2: Import Declarations Problem

Input: l, r, ps
Output: compilation problem in ps concerning l and r import declarations, if there is one

```
1  foreach p ∈ ps do
2      if p.type = COLLISION then
3          foreach a ∈ p.arguments do
4              if a ∈ l.body ∨ a ∈ r.body then return p ;
5          end
6      else if p.type = AMBIGUITY then return p ;
7  end
8  return null;
```

Algorithm 3: Import Declarations Conflict

Input: l, r, cs
Output: wether there is a unstructured conflict in cs concerning l and r import declarations

```
1  foreach c ∈ cs do
2      if l.body ∈ c.left ∧ r.body ∈ c.right then return TRUE ;
3  end
4  return FALSE;
```