CS 444 - Compiler Construction **Winter 2020**

Lecture 12: February 12th , 2020

Lecturer: Ondřej Lhoták *Notes By: Harsh Mistry*

12.1 Context-sensitive analysis continued

12.1.1 Name Resolution

- In Java : A2 and A3
  1. Build global environment
  2. Resolve type names
  3. Build/check class hierarchy (methods/files)
  4. Disambiguate ambiguous namespace
     - In Java you can't simply determine the namespace based on location of usage
  5. Resolve "expressions" (Variables, static fields)
  6. Type checking
  7. Resolve methods instance (non-static) fields

- (2) Resolving Name Types Continued : More Rules
  -
  $$\frac{\begin{array}{cccc} S \in super(T) & m \in contain(S) & abstract \notin mods(m) & nodecl(T,m) \\ S' \in super(T) & m' \in contain(S') & abstract \notin mods(m') & sig(m) = sig(m') \end{array}}{(m, m') \in replace}$$

  $$nodecl(T, m) = \forall m' \in declare(T) : sig(m) \neq sig(m')$$

  - An interface without super interfaces implicitly declares an abstract version of every public method in `java.lang.Object` (Refer to JLS 9.2)

- (3) Checking Hierarchy
  1. not $\exists T : T < T$ (Basically there should be no cycles)
  2. $\forall m, m' \in declare(T) : m \neq m' \implies sig(m) \neq sig(m^{p}rime)$
  3. $\forall m, m' \in contains(T) : sig(m) = sign(m') \implies type(m) = type(m')$
  4. $\forall m \in contains(T) : abstract \in mods(m) \implies abstract \in mods(T))$
  5. $\forall (m, m') \in replace : static \in mods(m') \iff static \in mods(m)$
  6. $\forall (m, m') \in replace : type(m) = type(m'))$
  7. $\forall (m, m') \in public \in mods(m') \implies public \in mods(m)$
  8. No 8, because the course is attempting to be consistent with numbering schemes in other offerings. 8 would normally involve exception handling

9. $\forall (m, m') \in replace : final \notin mods(m')$
10. $\forall f, f' \in declare(T) : f' \neq f \implies name(f) \neq name(f')$

- (4) Disambiguating ambiguous name spaces
  - Notation
    - P = package
    - T = Type
    - E = Expression
    - M = Method
  - (Refer to JLS 6.5.2) $(a_1 \cdot a_2 \cdot a_3 \cdot a_4 \cdot a_k)$
    1. If local variables/parameters named a_1 exists , resolve to it
    2. If field name $a_1 \in contain$(enclosing class), resolve to it
    3. For each k in increasing order
       - if $a_1 \cdot \ldots \cdot a_k$ resolves to a type, resolve to it
         - a_{k+1} is a static field
         - $a_{k+2\ldots}$ are non static fields
    4. Resolve instance fields, methods after type declaration
    5. General search outwards from inner most scope