Resolving of Intersection Types in Java 5.0

Martin Plümicke

In Java 5.0 the principal type of a given method could often be an intersection type. Only the restriction that in Java 5.0 the declaration of intersection types is not allowed, avoids these types. Therefore type inference as described in [Plü07] determines intersection types for some typeless methods.
There are different classes of intersection types respectively of overloading, which have to be treated different during compilation. For example a method, which multiples two matrices

```
class Matrix extends Vector<Vector<Integer>> {
    mul(m) { ret = new Matrix(); ...
        while(i <size()) { v1 = this.elementAt(i); v2 = new Vector<Integer>();...
            while(j < v1.size()) { ...
                while(k < v1.size()) {
                    erg = erg + v1.elementAt(k) * m.elementAt(k).elementAt(j);}
                v2.addElement(new Integer(erg)); j++; }
            ret.addElement(v2); i++; }
        return ret; }}
```

has a principal intersection type `mul:` $\&_{\beta,\alpha}(\beta \to \alpha)$, where β is subtype of `Vector<?extends Vector<?extends Integer>>` and α a supertype of `Matrix`. For all types of the parameter `m` in the call of `mul(m)` the same code is executed. Additionally, we consider

```
class OL {  m(Integer x) { return x + x; }
            m(Boolean x) { return x || x; }
            main(x) { OL ol = new OL(); return ol.m(x); } }
```

where the intersection type `main` : `Integer` → `Integer` & `Boolean` → `Boolean` is inferred. In this case for the call of `main(x)` different code, depending on the type of `x`, is executed.
During compilation the methods `mul` and `main` must be treated different. While for `mul` only one method for the reduced principal type

```
Matrix mul(Vector<? extends Vector<? extends Integer>> m) { ...}
```

is generated, for `main` two methods are necessary

```
Integer main(Integer x) { ...}
Boolean main(Boolean x) { ...}.
```

The differentiation during compilation is done by the call graph of the corresponding argument type.

References

[Plü07] Martin Plümicke. Typeless Programming in Java 5.0 with wildcards. In Vasco Amaral, Luís Veiga, Luís Marcelino, and H. Conrad Cunningham, editors, *5th International Conference on Principles and Practices of Programming in Java*, ACM International Conference Proceeding Series, pages 73–82, September 2007.