It is an error to declare multiple function overloads that are considered identical (section 3.8.2) or differ only in their return types.

The following is an example of a function with overloads.

```
function attr(name: string): string;
function attr(name: string, value: string): Accessor;
function attr(map: any): Accessor;
function attr(nameOrMap: any, value?: string): any {
    if (nameOrMap && typeof nameOrMap === "string") {
        // handle string case
    }
    else {
        // handle map case
    }
}
```

Note that each overload and the final implementation specify the same identifier. The type of the local variable 'attr' introduced by this declaration is

```
var attr: {
    (name: string): string;
    (name: string, value: string): Accessor;
    (map: any): Accessor;
};
```

Note that the signature of the actual function implementation is not included in the type.

6.3 Function Implementations

A function implementation without a return type annotation is said to be an ***implicitly typed function***. The return type of an implicitly typed function *f* is inferred from its function body as follows:

- If there are no return statements with expressions in *f*'s function body, the inferred return type is Void.
- Otherwise, if *f*'s function body directly references *f* or references any implicitly typed functions that through this same analysis reference *f*, the inferred return type is Any.
- Otherwise, the inferred return type is the widened form (section 3.9) of the best common type (section 3.10) of the types of the return statement expression in the function body, ignoring return statements with no expressions. A compile-time error occurs if the best common type isn't one of the return statement expression types (i.e. if the best common type is an empty type).

TODO: Clarify exact meaning of "through this same analysis".

In the example