Compiler Construction

Dr. Michael Petter, Raphaela Palenta SS 2016

Exercise Sheet 10

Assignment 10.1. Type Checking vs. Type Inference

Explain the difference between *type checking* and *type inference.*

Assignment 10.2. Type Checking

This exercise is about checking the types of expressions given in our C-like language. Make sure to only use the rules given in the lecture and to write down every step in a tree structure.

1. Given the declarations $\Gamma \coloneqq \{\mathsf{int}\ x, \mathsf{int}\ a[]\}$, check whether the statement $\mathsf{int}\ y = x + a[42];$ is well-typed.

2. Given the declarations $\Gamma \coloneqq \{\mathsf{int}\ y,\ \mathsf{double}\ a[],\ \textit{struct}\ \{\mathsf{double}\ a[];\}\ g,\ \mathsf{int}\ (*f)(\mathsf{double})\}$, check whether the statement $\mathsf{int}\ x = f(g.a[y + 2]);$ is well-typed.

Assignment 10.3. Subtyping

Consider the following C structs:

```
struct A {
        A f(B, C);
        C g(C);
}

struct B {
        B f(A, D);
        A g(D);
}
```

```
struct C {
        C f(B, B);
        D g(A);
}

struct D {
        D f(B, B);
        D g(B);
        int a;
}
```

We are going to use the non-standard subtyping rules for C structures which have been introduced in the lecture. Let $\leq$ be the type comparison operator, that is, for two types A and B the following holds:

$$A \leq B \Leftrightarrow A \text{ is a subtype of } B \tag{1}$$

Now, proof the assertions below either right or wrong:

1. $A \leq B$

2. $A \leq C$