Type Expression in Compiler Design

Equivalence of Type Expressions

- If two type expressions are equal then return a certain type else return type_error.

Key Ideas:

- The main difficulty arises from the fact that most modern languages allow the naming of user-defined types.
- For instance, in C and C++ this is achieved by the typedef statement.
- When checking equivalence of named types, we have two possibilities.
    - Structural Equivalence
    - Names Equivalence

Structural Equivalence of Type Expressions

- Type expressions are built from basic types and constructors, a natural concept of equivalence between two type expressions is structural equivalence. i.e., two expressions are either the same basic type or formed by applying the same constructor to structurally equivalent types. That is, two type expressions are structurally equivalent if and only if they are identical.
- For example, the type expression integer is equivalent only to integer because they are the same basic type.
- Similarly, pointer (integer) is equivalent only to pointer (integer) because the two are formed by applying the same constructor pointer to equivalent types.

- The algorithm recursively compares the structure of type expressions without checking for cycles so it can be applied to a tree representation. It assumes that the only type constructors are for arrays , products, pointers, and functions.
- The constructed type array(n1 , t1) and array(n2 , t2) are equivalent if n1 = n2 and t1 =t2

Names for Type Expressions

- In some languages, types can be given names (Data type name). For example, in the Pascal program fragment.

```
Type  link   = ↑ cell;
Var   next   : link;
      last   : link;
      p      : ↑ cell;
      q, r   : ↑ cell;
```

- The identifier link is declared to be a name for the type cell. The variables next, last, p, q, r is not identical type, because the type depends on the implementation.

For More Details Click Here:

https://www.wikitechy.com/tutorials/compiler-design/type-expression-in-compiler-design

