Using_typesystem

Using a typesystem

If you have defined a typesystem for a language, a typechecker will automatically use it in editors to highlight opened nodes with errors and warnings. You may additionally want also to use the information about types in queries, like editor actions, generator queries, etc. You may want to use the type of a node, or you may want to know whether a certain type is a subtype of another one, or you may want to find a supertype of a type which has a given form.

Type Operation

You may obtain a type of a node in your queries using the type operation. Just write <expr>.type, where <expr> is an expression which is evaluated to a node.

Do not use type operation inside inference rules and inference methods! Inference rules are used to compute types, and type operation returns an already computed type.

Is Subtype expression

To inspect whether one type is a subtype of another one, use the isSubtype expression. Write isSubtype(type1 :< type2) or is StrongSubtype(type1 :<< type2), it will return true if type1 is a subtype of type2, or if type1 is a strong subtype of type2, respectively.

Coerce expression

A result of a coerce expression is a boolean value, which says whether a certain type may be coerced to a given form, i.e. whether this type has a supertype, which has a given form (satisfies a certain condition). A condition could be written either as a reference to a concept declaration, which means that a sought-for supertype should be an instance of this concept; or as a pa ttern, which a sought-for supertype must match.
A coerce expression is written coerce(type :< condition) or coerceStrong(type :<< condition), where condition is what has just been discussed above.

Coerce Statement

A coerce statement consists of a list of statements, which are executed if a certain type can be coerced to a certain form. It is written as follows:

```
coerce ( type :< condition ) {
  ...
} else {
  ...
}
```

If a type can be coerced so as to satisfy a condition, the first (if) block will be executed, otherwise the else block will be executed. The supertype to which a type is coerced can be used inside the first block of a coerce statement. If the condition is a pattern and contains some pattern variables, which match parts of the supertype to which the type is coerced, such pattern variables can also be used inside the first block of the coerce statement.

Match Statement

A match statement compares a node against a pattern. The variables in that pattern will be assigned corresponding values from the matched node or its children.

```
public int foo( ) {
  match node  with {
      > <repeat $number times  < as r -> {
          if not wall ahead do
            step
          end
        end>
         return Integer.valueOf($number);
      }
      default -> return 0;
  }
}
```

Previous Next