
Substitution, subst_fun

CS 421
Revision 1.0

1 Change Log

1.0 Initial Release.

2 Objectives

Your objective for this assignment is to understand the details of the substitution.

3 Datatypes for Type Inference

Below is some of the code available for your use in the `Common` module. This module includes the following data types to represent the types of PicoML, which you should recognize from MP3:

```
type typeVar = int
```

```
type monoTy = TyVar of typeVar | TyConst of (string * monoTy list)
```

You can use `string_of_monoTy` in `Common` to convert your types into a readable concrete syntax for types.

4 Substitutions

In MP3, one of the things we returned was a substitution. Our substitutions have the type `(typeVar * monoTy) list`. The first component of a pair is the index (or “name”) of a type variable. The second is the type that should be substituted for that type variable.

In this ML, you are to implement a function `subst_fun` that takes a substitution and return a substitution *function*, a function that takes a type variable as input and returns the replacement type as given by the substitution. (Recall that we are using the type `int` for type variables, which we give the synonym `typeVar`.) When creating such a function from a substitution (i.e., a list of pairs as described above), if a given type variable does not have an entry in the list, the identity substitution is assumed for that type variable (i.e. the variable is substituted with itself). For instance, the substitution

```
# let phi = [(5, mk_fun_ty bool_ty (TyVar(2)))];;  
val phi : (int * monoTy) list =  
  [(5, TyConst (">", [TyConst ("bool", []); TyVar 2]))]
```

is considered to represent the substitution function

$$\phi(\tau_i) = \begin{cases} \text{bool} \rightarrow \tau_2 & \text{if } i = 5 \\ \tau_i & \text{otherwise} \end{cases}$$

Throughout this ML you may assume that substitutions we work on are always well-structured: there are no two pairs in a substitution list with the same index.

As described above, your function `subst_fun` should, given a substitution, return the function it represents. This should be a function that takes a `typeVar` and returns a `monoTy`.

1. Implement the `subst_fun` function as described above.

```
# let subst_fun s = ...
val subst_fun : (typeVar * monoTy) list -> typeVar -> monoTy = <fun>
# let subst = subst_fun [(5, mk_fun_ty bool_ty (TyVar(2)))];;
val subst : typeVar -> monoTy = <fun>
# subst 1;;
- : monoTy = TyVar 1
# subst 5;;
- : monoTy = TyConst ("->", [TyConst ("bool", []); TyVar 2])
```