W2 - IPS

Nikolaj Gade (qhp695)

May 2022

Task 1

1)

a)

The string must consist of any number of pairs of '`o`'s and '`g`'s. The pairs can be any combination of 2 characters, both of which are either '`o`' or '`g`'. Thus, each character can be written as, '`o|g`', and each pair as '`(o|g)(o|g)`'. The full string can be written as:

```
((o|g)(o|g))*
```

b)

The first character must be '`o`', and it must be followed by a character that is either '`o`' or '`g`'. Following that, the answer is the same as the previous problem:

```
o(o|g)((o|g)(o|g))*
```

c)

Keeping with the idea that an even number of characters can be split up in "pairs" of characters, an odd number of '`o`'s or '`g`'s *must* result in an odd number of both. Thus, the string will either consist of an even number of '`o`'s, followed by an even number of '`g`'s, **or** an even number of '`o`'s, followed by '`og`', followed by an even number of '`g`'s. The full string can be written as:

```
(oo)*(og)?(gg)*
```

2)

a)

'a's and 'b's are placed at the same time, using the T starting symbol. After all 'a's and 'b's are placed, the symbol turns to 'S', which places at least 1 'c'.

$$T = \begin{cases} aTb \\ S \end{cases}$$

$$S = \begin{cases} c \\ cS \end{cases}$$

b)

Like with the previous problem, both the 'a's and the 'b's are placed at the same time, but the 'b's are placed 2 at a time.

$$T = \begin{cases} aTbb \\ abb \end{cases}$$

c)

Once again, 'a's and 'b's are placed at the same time with the starting symbol T. Afterwards, any amount of 'a's are placed before the 'b's.

$$T = \begin{cases} aTb \\ S \end{cases}$$

$$S = \begin{cases} \\ aS \end{cases}$$

3)

a)

`%nonassoc letprec` designates the `let` token as being non-associative, which means ambiguous implementations will lead to a syntax error.

b)

The order of the associativity declarations provide the precedence for the operators. So in the current way the code in written, the line `let x = 10 in x + 10 > 15` will be parsed as `let x = 10 in (x < 15)`, but if , it would be parsed as `(let x = 10 in x) < 15`.

c)

The code `{ Let (Dec (fst $2, $4, $3), $6, $1) }` creates a `Let` instance, which containes the declared variable, the following expression, as well as the keyword.

Task 2

See code.

Task 3

a)

```
filter (('a -> bool) * ['a]) -> ['a]
```

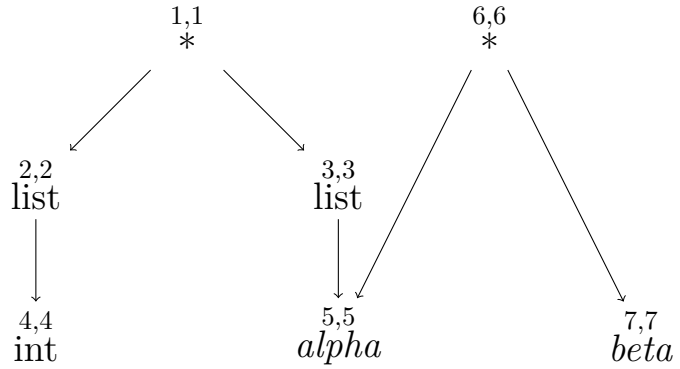
b)

```
CheckExp(exp, vtable, ftable) = case exp of
  filter(p, arr_exp) =>
    let array_type = CheckExp(arr_exp, vtable, ftable)
    let element_type = match array_type with
      | Array(type) -> type
      | _ -> Error()

    let function_type = lookup(ftable, name(p))
    match function_type with
      | unbound -> Error()
      | (input_type, output_type) ->
        if input_type == element_type && output_type == bool then
          Array(element_type)
        else Error()
      | _ -> Error()
```

Task 4

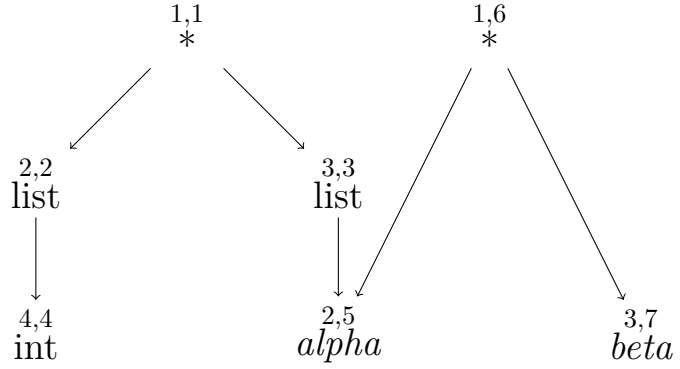
The unification graph with the initial R and ID values is as follows:



The R values can then be substituted in the following order:

- $6 \rightarrow 1$, Rule (IV)
- $5 \rightarrow 2$, Rule (III)
- $7 \rightarrow 3$, Rule (III)

So the final unification graph looks like this:



Which means the final type is `list(int) * list(list(int))`.