The algorithm

```
unify(a: MonoType, b: MonoType) → Substitution:
  if a is a type variable:
    if b is the same type variable:
      return {}
    if b contains a:
      throw error "occurs check failed, cannot create infinite type"
    return { a ↦ b }

  if b is a type variable:
    return unify(b, a)

  if a and b are both type function applications:
    if a and b have different type functions:
      throw error "failed to unify, different type functions"
    let S = {}
    for i in range(number of type function arguments):
      S = combine(S, unify(S(a.arguments[i]), S(b.arguments[i])))
    return S
```

Output:

```
main()
```

```
Enter the first expression
knows(f(x),y)
Enter the second expression
knows(J,John)
The substitutions are:
['J / f(x)', 'John / y']
```

```
main()
```

```
Enter the first expression
Student(x)
Enter the second expression
Teacher(Rose)
Cannot be unified as the predicates do not match!
The substitutions are:
[]
```

```
main()
```

```
Enter the first expression
knows(John,x)
Enter the second expression
knows(y,Mother(y))
The substitutions are:
['John / y', 'Mother(y) / x']
```

```
main()
```

```
Enter the first expression
like(A,y)
Enter the second expression
like(K,g(x))
A and K are constants. Cannot be unified
The substitutions are:
[]
```