# Programming Language Design
# Assignment 3 2024

Torben Mogensen and Hans Hüttel

March 5, 2024

This assignment is a group assignment with groups of up to 3 people. Do not seek help from other groups, nor provide them with help.. This will be considered plagiarism. If you use material from the Internet or books, cite the sources. Plagiarism *will* be reported. If you use ChatGPT or similar large language models, you should include the entire dialogue with the AI in your report (otherwise, it is plagiarism), and you should discuss the relevance nd correctness of the answers you get (otherwise, you will get no points whatsoever).

Assignment 3 is the third of three mandatory assignments. You are required to get at least 40% of the possible score for every mandatory assignment, so you can not count on passing by the earlier assignments only. If you know what you are doing, you can solve the assignment in an afternoon, but if you have not actively participated in the plenary and classroom activities, you should expect to use *considerably* more time.

We advise you not to split the questions between the members of your group. You will get much more benefit if you work on all questions together, and you will need that for the exam assignment.

The deadline for the assignment is **Friday March 15 at 16:00 (4:00 PM)**. The exercises below are given percentages that provide a rough idea how much they count (and how much time you are expected to use on them).

The assignments consists of several exercises, some from the notes and some specified in the text below. You should hand in a single PDF file with your answers and a zip-file with your code. Hand-in is through Absalon. Your submission must be written in English.

## A3.1) Polymorphism – 20 %

Here are two functions in Haskell.

```
plip [] = []
plip ((x,y) : zs) = (x*y,x+y) : (plip zs)

dingo [] = []
dingo (x:xs) = (x,x) : (dingo xs)
```

One of these functions is ad hoc-polymorphic, another is parametric polymorphic.

a) Which one is which? Argue precisely why this is the case.

b) For the function that is parametric polymorphic, apply the Hindley-Milner type inference algorithm to find its type. Show each step of the algorithm as you apply it.

## A3.2) Object-oriented programs and subtyping – 20 %

Array types are covariant in Java and C#. Because of this, some array operations require a run-time type check.

1. Why is this? Give an example of code that will pass the compile-time type checker, but will give a run-time type error caused by an array operation.

We could also say that array types were *contravariant*.

2. Explain precisely what contravariance means.
3. Would such a solution require run-time type checks? If yes, explain in which cases you need these run-time type checks and give a concrete example that is well-typed in Java or C# and does not involve explicit downcasting but fails at run-time (same as above).

## A3.3) Prolog – 15 %

Labelled binary trees are defined to be the least set such that

- If $n$ is a label, then a leaf $\mathsf{Leaf}(n)$ is a tree
- If $x$ and $y$ are trees and $n$ is a label, then $\mathsf{Node}(n, x, y)$ is also a tree

Assume that labels can be arbitrary symbols and that labels in trees can occur in any order or multiplicity.

a) Express the above definition as a collection of predicates in Prolog.

b) Define a predicate Member that willl tell us if a tree contains a given label.

A3.4) **Unification – 15 %**

Here are four instances of Hindley-Milner type expressions $T_1$ and $T_2$ that we would like to unify. The $\alpha$'s denote type variables. For each of these four instances, find out if $T_1$ and $T_2$ can be unified and, in the cases where they can, provide a unifying substitution. In the cases where $T_1$ and $T_2$ cannot be unified, explain how and where unification fails.

$$\begin{aligned} T_1 &= \texttt{string} \to \alpha_1 && \text{and} && T_2 = \texttt{string} \to \alpha_1 \times \alpha_2 \\ T_1 &= \alpha_1 \times \alpha_2 \to \alpha_3 \times \alpha_4 && \text{and} && T_2 = \texttt{real} \times \alpha_5 \to \alpha_6 \times \alpha_1 \\ T_1 &= (\alpha_4 \to \alpha_5) \times \alpha_g && \text{and} && (\texttt{real} \to \alpha_3) \times \texttt{string} \\ T_1 &= \alpha_2 \to (\alpha_3 \to \alpha_4) && \text{and} && (\alpha_3 \to \alpha_3) \to \alpha_5 \end{aligned}$$

A3.5) **Objects and inheritance – 15 %**

Here is a piece of Java code.

```
class Pet {
void eat() {
    ...
  }
  float height;
}
class Pig extends Pet{
void jump(){
  ...}
int age;
}
class Kangaroo extends Pet{
void swim(){
  ...}
bool famous;
}
class Check{
public static void main(String args[]){
Kangaroo c=new Kangaroo();
c.swim();
c.eat();
}
}
```

Draw a pointer model to show how the classes mentioned are related.

A3.6) **Subtyping – 15 %**

Assume that $\preceq$ is a subtype ordering, i.e., that it fulfils the requirements for a partial order as described in Section 7.4.3 in *Programming Language Design and Implementation*, i.e., reflexivity, anti-symmetry, and transitivity.

We now define an ordering $\leq$ on pair types by $(p \times q) \leq (s \times t)$ if either $p \neq s \wedge p \preceq s$ or $p = s \wedge q \preceq t$.

Is $\leq$ a subtype ordering on pair types? Argue in terms of the requirements for partial orders.