Benjamin Mannal
Huixin Zhan
CSE 423: Compiler Writing

Homework 1: Kotlin, C, Python comparison

Kotlin is a programming language designed around the niceties of development, focused on features to make development life easier. At a surface level, Kotlin blends the strengths of both languages, but at a deeper lexical level, it poses challenges for compilation.

One of the most noticeable similarities between Kotlin and C is that both languages are statically typed. In both cases, types are checked at compile time, allowing many errors to be caught early in the development process. This contrasts with Python, which is dynamically typed, meaning type errors are typically discovered at runtime. However, Kotlin distinguishes itself by supporting type inference, a feature more commonly associated with Python. In Kotlin, the compiler can often infer the type of a variable based on its assigned value, allowing programmers to omit explicit type declarations without sacrificing the benefits of static typing. This creates a middle ground between the verbosity of C and the flexibility of Python.

From a syntax perspective, Kotlin also feels like a hybrid of the two languages. Like Python, Kotlin emphasizes readability and reduced boilerplate. For example, Kotlin does not require semicolons at the end of statements, which makes the code feel less cluttered and more natural to read. At the same time, Kotlin retains a structured, explicit style similar to C, particularly in how functions, variables, and control flow are defined. Compared to C, Kotlin removes many low-level concerns, such as manual memory management, while still preserving a sense of control and clarity. Overall, Kotlin's syntax feels like a refined blend: structured and explicit without being overly rigid.

When considering the task of writing a translator for Kotlin, several challenges arise even at a basic level. One primary challenge is handling Kotlin's type system, particularly type inference. While static typing simplifies certain aspects of translation, inferred types require the translator to correctly deduce and track types throughout the program. This is more complex than translating a language like C, where types are always explicitly stated, but also more structured than Python, where types may not be declared at all.

In conclusion, Kotlin occupies a unique position between C and Python, combining static typing with type inference and offering a syntax that is both structured and approachable. While Kotlin may appear simpler on the surface, writing a translator for it presents meaningful challenges, particularly in type handling and syntax interpretation. These challenges make Kotlin an interesting and educational language to explore at this stage of study.