SE580: Lecture 4

Static semantics
Type system
Type context
Base context
Type inference
Implementation
Homework
Static semantics

What is static semantics? How can we specify the static semantics of a programming language?

What is a type checking? What is type inference?
Type system

Hobbes is specified using a *type system*.

The semantics is given as a collection of rules such as:

\[
\begin{align*}
\text{If } &\Gamma \vdash V : \text{Boolean} \\
&\text{and } \Gamma \vdash B : T \\
&\text{and } \Gamma \vdash C : T \\
&\text{then } \Gamma \vdash \text{if } (V) \text{ then } \{B\} \text{ else } \{C\} : T
\end{align*}
\]

Read this as:

In a *type context* \(\Gamma\)

if value \(V\) has type Boolean

and blocks \(B\) and \(C\) have type \(T\)

then block if \((V)\) then \(\{B\}\) else \(\{C\}\) has type \(T\).

What is a type context? Why do we need it?
Type system

What rules do we need to add to the Hobbes specification to typecheck this block?

    if (True) { return 1; } else { return 2; }
Type system

What rules do we need to add to the Hobbes specification to typecheck this block?

```plaintext
let x : Integer = 5;
let y : Integer = x;
return Nothing;
```
**Type system**

What rules do we need to add to the Hobbes specification to typecheck this program?

```plaintext
thread Main {
    let x : Integer = 5;
    let y : Integer = x;
    return Nothing;
}
```
Type system

What rules do we need to add to the Hobbes specification to typecheck this program?

thread Main {
    let x : String = "hello";
    let y : Thread = Main;
    return Nothing;
}
Type system

What rules do we need to add to the Hobbes specification to typecheck this program?

class Foo { }
object F : Foo { }
thread Main {
    let x : Foo = F;
}
Type system

What rules do we need to add to the Hobbes specification to typecheck this program?

class Foo {
    mutable field bar : Integer;
}
object F : Foo { bar = 5; }
thread Main {
    F.bar := 37;
    let x : Integer = F.bar;
}
Type system

We have (almost) dealt with all Hobbes code.

To-do list:

What is a type context $\Gamma$?

What about base types like Integer?

What about type inference?
Type context

A type context gives the type information for:

a) Free variables foo : Integer;

b) Classes class Bar { field x : Integer; }

c) Objects object B : Bar;

d) Thread thread Main;

For example, what is the type context at point HERE?

```plaintext
class Foo {
   field baz : Integer;
}
object F : Foo { baz = 37; }
thread Main {
   let x : Integer = F.baz;
      HERE
}
native class Boolean {
}
native object True : Boolean { 0 };
native object False : Boolean { 1 };
native class Integer {
    method prefix - () : Integer { ... }
    ... lots of stuff!...
}
native class String {
    method infix + (x : String) : String { ... }
    ... more stuff!...
}
native class Thread { ... }
native class Unit { ... }
object Nothing : Unit { 0 };
...
Type inference

What is type inference? Why does Hobbes need a type inference step? How can we specify type inference?
Implementation

How can we implement this?
Homework

Complete the type checker for Hobbes (currently missing most of the cases for Block).

Same procedure as before!
Next week

Object oriented features (including methods!).