Advances in Programming Languages

APL2: Types and type systems

Ian Stark

School of Informatics
The University of Edinburgh

Thursday 15 January 2009
Semester 2 Week 1

Some types

A selection of types from some languages.

C/C++

int, **long**, **float**, **unsigned int**, **char**
int [], **char**∗, **char**&, **int**(∗)(**float**,**char**)

OCaml

int, int64, bool, char, string, unit
string∗string, int list , bool array
int−>int, int−>string−>char, 'a list −> 'a list

Java

Object, **byte**[], **boolean**
StringBuffer, LinkedList, TreeSet, ArrayList<String>
IllegalPathStateException, BeanContextServiceRevokedListener

What do people do with types?

- Type checking
- Static type checking
- Dynamic type checking
- Type annotation
- Type inference

- Subtyping
- Structural typing
- Nominative typing
- Duck typing
- Effect types

What is a type system?

A *type system* is a syntactically defined subset T of programs such that:

$$\mathrm{P} \in \mathrm{T} \quad \Longrightarrow \quad \text{Compile}(\mathrm{P}) \models \phi$$

(read: "if P is in T then Compile(P) satisfies ϕ")

where Compile(P) is the object code corresponding to P and ϕ is some desired property of its execution.

For example,

$$\begin{aligned} \mathrm{T} &= \text{"well-typed Java programs"} \\ \phi &= \text{"methods are always correctly invoked"} \end{aligned}$$

Slogan: *Well-typed programs cannot go wrong.* [Robin Milner, 1978]

Java

Java is serious about abstraction

Java works almost entirely through class-based object-oriented programming; it encourages the use of abstract classes through inheritance and interfaces; and it does not expose the private workings of classes and packages.

Java is serious about typing

Java has strong static typing: all programs are checked for type-correctness at compile-time. Bytecode is checked again when classes are loaded, by the *bytecode verifier*, before execution. The recent introduction of *generics* extends the power of the type system.

Even so, things do not always go as well as one might hope...

Subtyping arrays in Java

Java has subtyping: a value of one type may be used at any more general type. So String $\leqslant$ Object, and every String is an Object.

Not all is well with Java types

```
String[] a = { "Hello" };              // A small string array
Object[] b = a;                        // Now a and b are the same array
b[0] = Boolean.FALSE;                  // Drop in a Boolean object
String s = a[0];                       // Oh, dear
System.out.println(s.toUpperCase());   // This isn't going to be pretty
```

This compiles without error or warning: in Java, if $S \leqslant T$ then $S[] \leqslant T[]$.

Except that it isn't. So every array assignment gets a runtime check.

Subtype variance

The issue here is with *parameterized types* like String[] and List<Object>; or in OCaml ('a list −> 'a list) and ('a ∗ 'b).

Suppose some type A<X> depends on type X, and types $S \leqslant T$. Then the dependency is:

Covariant	if $A{<}S{>} \leqslant A{<}T{>}$	e.g. pair $A{<}X{>} = X * X$
Contravariant	if $A{<}S{>} \geqslant A{<}T{>}$	e.g. test $A{<}X{>} = X \rightarrow \mathrm{bool}$
Invariant	if neither of these holds.	e.g. array $A{<}X{>} = X[]$

For example, in the Scala language, type parameters can be annotated with variance information: List[+T], Function[−S,+T].

In Java, arrays are typed as if they were covariant. But they aren't. We shall revisit this later. . .

see also *parameter covariance* in Eiffel

Homework

By the next lecture, on Monday:

- Test out the Java array subtyping example, and confirm that (a) it compiles, and (b) there is a type error when run.
- Read the Java fable *Execution in the Kingdom of Nouns.*

If you are uncertain about OCaml programming, try these online guides:

- Chapter 1 of *OCaml for Scientists*
- *The Objective Caml Tutorial*
- *Developing Applications with Objective Caml*
- For those who already know Standard ML, Andreas Rossberg has written a handy conversion guide.

Summary

- Languages use types and type systems for several reasons.
- A *type system* is a syntactically defined subset of programs which are certain to have some desired property.
- Java has covariance subtyping of arrays, which can cause runtime type errors.