Advances in Programming Languages

APL18: Course Review

Ian Stark

School of Informatics
The University of Edinburgh

Monday 20 March 2008
Semester 2 Week 11

Course Review

The course has covered the following sample "advances in programming languages":

- Type-safe extensible records in OCaml
- JML and ESC/Java 2 for specifying and checking code
- LINQ and cross-language integration in .NET
- Concurrency in Java, Erlang, and Polyphonic C#
- Cyclone for pointer management in C

Plus additional material on:

- Spec# (Rustan Leino video)
- Typing units of measure in F# (Andrew Kennedy, Microsoft Research)
- Programming the Cell in Sieve C++ (Alasdair Donaldson, Codeplay)

as well as your own coursework topic.

OCaml Records

Extensible Records in OCaml

OCaml is a strongly-typed functional language with excellent performance across a range of platforms. It uses *structural typing*, where types are compatible if they contain the same components; this is in contrast to the *nominative typing* used by Java, C# and similar languages, where types are compatible only if they have been explicitly declared so by name.

The object system of OCaml uses *extensible records*, where a longer record can always be used in place of a shorter one:

```
let getfield p = p#m
val getfield : < m : 'a; .. > -> 'a = <fun>

let double p = p#height * 2;;
val double : < height : int; .. > -> int = <fun>
```

This means that OCaml can separate *method inheritance* from *subtyping*.

Reasoning and Specification

Hoare Logic

The logic of *Hoare triples* $\{P\}\ C\ \{Q\}$ allows us to make statements and reason about programs. This means we can *specify* desired behaviour, and then *verify* that programs meet this specification.

JML

The *Java Modeling Language* uses annotation comments in Java source to specify intended behaviour:

```
/*@  requires credit > amount;
     ensures credit == \old(credit) - amount; @*/
public int withdraw(int amount) { ... }
```

ESC/Java 2

The Extended Static Checker for Java version 2 carries out a range of checks on Java source, guided by JML annotations; some use an automatic theorem prover. It can work within the Eclipse IDE.

LINQ

Language-Integrated Query

The LINQ framework for Microsoft .NET maps the constructions of a *domain-specific language* into a general-purpose programming language. This maintains high-level abstractions and avoids the pitfalls of string manipulation. We saw this in two settings:

- In C#: writing SQL database queries
- In F#: writing SQL, runtime code generation, and GPU acceleration

In order to do this, LINQ provides several language extensions, including: lambda expressions, structural datatypes, anonymous types, and type inference.

Concurrency

Java and C#

Both languages provide shared-memory concurrency through explicit threads, with mutual exclusion for critical sections using **synchronized** (**lock**), and communication by wait/notify (wait/pulse).

Erlang

A declarative functional language with share-nothing concurrency, where all communication is by sending messages to mailboxes. There are no critical regions: this scales to high thread counts, and even "hot code upgrading".

Polyphonic C#

Focuses on communication and *asynchronous* computation rather than explicit concurrency. Importance of message patterns: a *chord* of methods together trigger appropriate actions. Raising the level of abstraction gives a compiler flexibility to optimise thread usage.

Safer C

Cyclone

Programming in C still remains attractive for many reasons, in particular the precise, transparent control over time and memory usage. Unfortunately this is also the source of many errors and vulnerabilities.

> *The design of the C programming language encourages programming at the edge of safety.*
>
> [Jim, Morrisett, et al.]

Cyclone is a dialect of C that has static and dynamic checking of pointer manipulation: including pointers into arrays or structures, and pointer arithmetic. Type annotations control these uses and how they are checked: fat pointers, non-null, memory regions,... The aim is to eliminate all unchecked memory errors.

The Importance of Language

Sapir-Whorf Hypothesis

We dissect nature along lines laid down by our native language

Boole: Language is an instrument of human reason, not merely a medium for the expression of thought [An Investigation of the Laws of Thought, 1854]

Wittgenstein: The limits of my language mean the limits of my world [Tractatus Logico-Philosophicus, 1922]

Languages frame the way we think, and the programs we can imagine.

Extending programming languages can give us new ways of programming

Significant Themes

Abstraction

Programmability means that computers can always do more. Best of all, you can program new ways to program. The concept of *abstraction* is an important way this appears in programming languages.

Abstractions build upon each other: bytes, arrays, pointers, lists, trees, files, sockets, databases, objects, procedures, threads, behaviours, ...

Abstraction frees up you to think about other things, and you should. Let the machine get on with its job.

Types

Several of the language extensions and applications in this course have made significant use of *types* for organisation and abstraction. Advanced type systems can help capture programmer intentions, while enforcing the constraints necessary to keep these meaningful and consistent.

Feedback

Thank you for you participation. Please fill out the feedback forms about the course. There are two of these:

- Standard Informatics anonymous course feedback. Either fill out on paper and leave here; hand in to the ITO; or fill out online.
- Specific APL topic feedback: of the five topics covered, which ones should come back next year? This uses *range voting*: choose any number 0–31 for each, higher means better. Also, make your own suggestions.

Programming languages continue to advance, and will continue to do so. Notice this as it happens, and keep learning new languages.

Further advances that take place between now and May 15 will not be considered examinable