

Final Lecture

George Giorgidze

Universität Tübingen

Advanced Functional Programming (Lecture 15)
2012 JAN 30, Tübingen, Germany

This lecture

- ▶ Topics covered in this course
- ▶ Relevance to other languages
- ▶ Where do we go from here?

Topics (1)

- ▶ Purely functional programming
- ▶ Algebraic data types
- ▶ Pattern matching
- ▶ Polymorphism
- ▶ Overloading with type classes
- ▶ Strict and lazy evaluations

Topics (2)

- ▶ Higher-order functions
- ▶ Higher-order combinator libraries
- ▶ Type inference
- ▶ Type system to separate pure functions from effectful computations (*IO monad*)
- ▶ Abstract data types
- ▶ Structuring code with modules and packages

Topics (3)

- ▶ Automated property-based testing
- ▶ Functors, applicative functors, monoids, monads and monad transformers
- ▶ Embedded domain-specific languages
- ▶ Metaprogramming
- ▶ Purely functional data structures
- ▶ Parallelism and concurrency
- ▶ Datatype-generic programming

Relevance to other languages

- ▶ Functional languages: ML family (SML, OCaml and F#), Erlang, Lisp family (e.g., Common Lisp, Scheme, Racket and Clojure)
- ▶ Theorem provers: Coq, Isabelle and Agda
- ▶ Imperative programming languages: C++, C#, and Java
- ▶ Multi-paradigm programming languages: Scala
- ▶ Domain-specific languages: Excel, LINQ and SQL
- ▶ Programming techniques: Parallel loops and MapReduce

ML family (SML, OCaml and F#) (1)

- ▶ Functional
- ▶ Higher order
- ▶ Statically typed with type inference
- ▶ Impure
- ▶ Strict
- ▶ Advanced module system

ML family (SML, OCaml and F#) (2)

- ▶ SML has been formally specified
- ▶ OCaml has full-blown object-oriented layer.
- ▶ F# is fully supported language in Microsoft's .NET Framework and Visual Studio since 2010.

Erlang

- ▶ Functional
- ▶ Higher order
- ▶ Dynamically typed
- ▶ Impure
- ▶ Strict
- ▶ Concurrency with message passing
- ▶ Fault tolerant
- ▶ Hot swapping

Lisp family (e.g., Common Lisp, Scheme, Racket and Clojure)

- ▶ Functional
- ▶ Higher order
- ▶ Dynamically typed
- ▶ Impure
- ▶ Strict
- ▶ Purely functional, lazy and statically-typed dialects exist

Theorem provers

- ▶ Coq
- ▶ Isabelle
- ▶ Agda

Lambdas are everywhere

- ▶ C#: `x => x + x`
- ▶ C++: `[] (int x) { return x + x; }`
- ▶ JavaScript: `function(x) {return x + x;}`
- ▶ Scala, Ruby, Python, Perl, PHP, ...

Functional programming inspired features

- ▶ Template metaprogramming in C++
- ▶ Concepts in C++
- ▶ Generics in Java
- ▶ Language Integrated Query (LINQ) in .NET languages
- ▶ Type inference in C# and Scala
- ▶ Scala is a prominent example of integrating object-oriented and functional programming paradigms

Domain-specific languages (Excel)

Improving the world's most popular functional language:
user-defined functions in Excel

Simon Peyton Jones, Margaret Burnett, Alan Blackwell.

<http://research.microsoft.com/en-us/um/people/simonpj/papers/excel/index.htm>

Domain-specific languages (LINQ)

- ▶ Language Integrated Query (LINQ) adds data querying capabilities to .NET languages.
- ▶ Based on monad comprehensions
- ▶ Can query a number of data sources (e.g., in-memory collections, XML documents and SQL databases)
- ▶ Programmable

```
from n in numbers
where n < 42
select n
```

Domain-specific languages (SQL)

- ▶ SQL queries do not feature mutable variables

```
SELECT n
FROM   numbers
WHERE  n < 42
```

Programming techniques

- ▶ Parallel loops
- ▶ MapReduce

Parallel for loops

Here's an example of a sequential **for** loop in C#.

```
int n = ...
for (int i = 0; i < n; i++)
{
    // ...
}
```

To take advantage of multiple cores, replace the **for** keyword with a call to the **Parallel.For** method and convert the body of the loop into a lambda expression.

 **Note:**

Don't forget that the steps of the loop body must be independent of one another if you want to use a parallel loop. The steps must not communicate by writing to shared variables.

```
int n = ...
Parallel.For(0, n, i =>
{
    // ...
});
```

<http://msdn.microsoft.com/en-us/library/ff963552.aspx>

Parallel for loop notes (1)

 **Note:**

Don't forget that the steps of the loop body must be independent of one another if you want to use a parallel loop. The steps must not communicate by writing to shared variables.

 **Note:**

The **Parallel.For** method does not guarantee any particular order of execution. Unlike a sequential loop, some higher-valued indices may be processed before some lower-valued indices.

 **Note:**

If you're unfamiliar with the syntax for lambda expressions, see "Further Reading" at the end of this chapter. After you use lambda expressions, you'll wonder how you ever lived without them.

<http://msdn.microsoft.com/en-us/library/ff963552.aspx>

Parallel for loop notes (2)

 **Note:**

Don't forget that iterations need to be independent. The loop body must only make updates to fields of the particular instance that's passed to it.

 **Note:**

You must be extremely cautious when getting data from properties and methods. Large object models are known for sharing mutable state in unbelievably devious ways.

 **Note:**

Don't allow duplicate instances in parallel loops. If an object appears more than once in the input to a loop, it's possible that two parallel threads could update the object at the same time.

<http://msdn.microsoft.com/en-us/library/ff963552.aspx>

Google's MapReduce

mapReduce :: (*k1* → *v1* → [(*k2*, *v2*)]) -- MAP
 → (*k2* → [*v2*] → *Maybe v3*) -- REDUCE
 → *Map k1 v1* -- Input
 → *Map k2 v3* -- Output

Ralf Lämmel, Google's MapReduce programming model - Revisited,
Science of Computer Programming, Volume 70, Issue 1, 1 January 2008.

Where do we go from here?

- ▶ Functional programming in industry
- ▶ Functional programming in academia

Functional programming in industry

- ▶ High assurance software/hardware systems
- ▶ Financial industry
- ▶ (Embedded) domain-specific languages
- ▶ Consultancies
- ▶ Startups

- ▶ Update your CV
- ▶ Contribute to open source projects
- ▶ Consider opportunities such as Google Summer of Code (GSoC)

Functional programming in academia

- ▶ MSc/Diploma thesis
- ▶ Dozens of good research groups and labs with PhD programs are doing functional programming related research (both practice and theory)

MSc/Diploma thesis and Studienarbeit topics (1)

- ▶ Implementation approaches for nested data-parallelism in Haskell
ALREADY TAKEN!
- ▶ GHC extension for overloading the list notation and allowing for rebinding of SQL-like monad comprehensions
ALREADY TAKEN!
- ▶ Optimiser and code generator for database queries formulated using relational algebra operators (suitable for MSc/Diploma thesis)
Available!

MSc/Diploma thesis and Studienarbeit topics (2)

- ▶ Other topics possible
 - ▶ Compilers, optimisers and code generators
 - ▶ Domain-specific languages
 - ▶ Databases
 - ▶ New Haskell libraries and tools
 - ▶ Contributions to Haskell projects (GHC, Cabal, etc.)
- ▶ Your own suggestions are also very much welcome
- ▶ Mentoring for GSoC projects may also be possible

Thanks for sharing the fun!