I.1. Introduction

The goal of this module is to understand the interaction between languages and computation.

Questions to be considered

- What is a programming language?
- How to compile a programming language?
- How to parse a text written in a formal or semi-formal language?
- How to define the syntax and semantics of programming languages?
- What are the limits of computation?

Example 1: Java

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```
Observation

- Program is given as a file.
- The content of the file is a string of Unicode characters.
- When a programmer reads it he immediately parses it.
- The compiler needs to do the same in order to translate it into machine executable code.

Example 2: Haskell

data Bintree a = Leaf | Node a (Bintree a) (Bintree a)
deriving Show
type Table t = [t]

search :: Eq t => t -> Table t -> Int
search x t = head [i | (i,x') <- zip [0..] t, x == x']

numberNode :: Eq t => t -> (Table t) -> (Int, Table t)
numberNode x table = if elem x table
  then (search x table, table)
  else (length table, table ++ [x])

Observation

- Different paradigm.
- Semantics needs to be defined quite differently.
- Program again given essentially as a string which needs to be parsed.
Example 3: HTML

```html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <hr>
    <address><a href="mailto:a.g.setzer@swan.ac.uk">anton setzer</a></address>
  </body>
</html>
```

Example Structured Code

Swansea University results for 17 students as at 27/08/09

123456, SMITH. John (XCSCS G400 22/09/08-22/06/11), CSCI

,, (JONES M.W.), 10cr, 09 CS-106, 45% P,,
,, 20cr, 09 CS-108, 46% P,,
,, 20cr, 09 CS-116, 53% P,,
,, 10cr, 09 CS-124, 52% P,,
,, 10cr, 09 CS-126, 32% F (0S),
,, 20cr, 09 CS-141, 21% F (0S),
,, 10cr, 09 CS-191, 22% F (0S),
,, 20cr, 09 EG-060, 22% F (0S),
,, 120cr pursued. 60cr pass. 60cr fail/TF.

Progress: RFM,

Observation

- Above output was intended to be human readable.
- Not intended to be processed by a machine.
- Because it is computer generated, it follows a certain structure.
- This structure can be analysed and then the output can be processed by a machine.

- Structure changes when the program producing it is changed.
- Parsing it by hand is possible but difficult to maintain.
- Better to define a grammar for this output.
- Then generate a parse tree using tools from this grammar.
- This grammar can more easily be adapted when the format of the output changes.
Example: Google

**Observation**

- **Two levels:**
  - HTML encoding.
  - Structured output from the program encoded into HTML.
- **Structure rather complex.**
- **Parsing might be useful in order to find out strategies for obtaining higher Google ranks.**
- **Raw parsing possible (e.g. find out all links), but might miss out features like grouping of the links.**
- **Better to write a grammar.**
  - Parsing would usually use a standard parser from HTML and then parse the resulting output using a grammar.

Example: Email

From zhy0118@YAHOO.COM Sun Oct 18 08:50:56 2008
Received: from columba.its.uu.se (columba.its.uu.se [130.238.7.10]) by beurling.math.uu.se (8.8.6/8.8.6/Math_main_1.1) with ESMTP id IAA06630 for <setzer@math.uu.se>; Sun, 18 Oct 2008 08:50:56 +0200 (MET DST)
Received: from send102.yahoomail.com ([205.180.60.90]:1410 "HELO ; Message-ID: <20081018065323.18552.rocketmail@send102.yahoomail.com>
Received: from [202.99.126.131] by send102.yahoomail.com; Sat, 17 Oct 2008 23:53:23 +0200 (MET DST)
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Length: 3582
From: Yi Zhang <zhy0118@YAHOO.COM>
To: setzer@math.uu.se
Subject: Help Mssg.
Date: Sat, 17 Oct 2008 23:53:23 -0700 (PDT)

**Observation**

- Parsing of such files done by the email program.
- It might be useful for us to write our own parser.
- Again writing a grammar and using a tool for parsing is the solution.
3 Aspects of Programs

- Syntax
  - The text of the program.
- Semantics
  - The behaviour of the program.
- Data
  - The Information to be transformed by the program.

Administrative Issues

Course Coordinator:
Dr. A. Setzer
Dept. of Computer Science
University of Wales Swansea
Singleton Park
SA2 8PP
UK

Room: Room 211, Faraday Building
Tel.: (01792) 513368
Fax: (01792) 295651
Email: a.g.setzer@swansea.ac.uk
Home page: http://www.cs.swan.ac.uk/~csetzer/index.html

Assessment:

- 70% Exam.
- 20% Lab Sessions
- 10% Coursework

Home Page of the Module

- The homepage for the parts taught by Anton Setzer is located at http://www.cs.swan.ac.uk/~csetzer/lectures/languageComputation/10/index.html
- There is an open version, and a password protected version.
  - The password is __________.
- Errors in the notes will be corrected on the slides continously and noted on the list of errata.
- The homepage contains as well additional material for each section of the module.
- In order to reduce plagarism, coursework and solutions to coursework will not be made available in electronic form (e.g. on this web site).
Overview over the Module

I. Grammars for defining syntax.
II. Data.
III. Semantics of Programs.
IV. Limits of Computation.

In parallel there will be labsessions on tools for parsing syntax on Thursday and Friday, given by Chris Whyley. They are labelled as CSP236 in the timetable.

Literature

   ▶ Main Course Text.
   The slides used in this module will be heavily based on this text.

   ▶ Main additional book for this part.
   ▶ Classical text on theory of formal languages

Other Texts:

   ▶ The famous “Dragon Book” (because of the book cover), the bible of compiler construction.