# CM50260 - FOUNDATIONS OF COMPUTATION – PROF N.VOROBJOV
## COURSEWORK 3: CONSTRUCT A PARSER – DR P.BRUSCOLI

ISSUED ON 1 NOVEMBER 2019, DUE BY 13 DECEMBER 2019

ABSTRACT. This coursework contributes 40% of the final mark for CM50260. This practical coursework focusses on the construction of a parser for a given context-free grammar. This document contains important information that you will need in order to complete this assignment. Full documentation, support material, and link to the development environment for this coursework is available on the Moodle page for CM50260 - Foundations of Computation, and includes also the location for the electronic submission of your working.
*Moodle page:* `https://moodle.bath.ac.uk/course/view.php?id=57451`

## MOTIVATIONS AND OVERVIEW OF THE PROPOSED ACTIVITY

This coursework will be completed over the next weeks, until the end of the unit. It contributes 40% of the final mark for this course.

Context-free grammars are used extensively in modelling the syntax of programming languages. A parser is the first stage of a compiler for these languages. A parser is the implementation of an algorithm to determine whether a given string is generated by the production rules of a given context-free grammar and, if so, it constructs a parse tree of the string. The compiler will then continue its processing by translating the parse tree into low-level machine code, performing various steps that we will not see (and are beyond the scope of this course). More than one algorithm exist that can be used for parsing.

This coursework consists in implementing a specific algorithm (that we provide) for parsing that determines whether a specific grammar (that we provide) can generate some strings (or words) and, if so, it produces the parse tree for those strings.

The implementation will be in Java (which you may know from the Programming course running in parallel to this one), and your practical work is well supported and guided by the documentation available on Moodle and the skeleton code that allows you to focus implementing just some specific parts. It is recommended to start familiarising with the system interface soon, as well as with the documentation that is available to you from the Moodle page of the course CM50260. The scheduled tutorials are venues to discuss your progress.

This coursework is part of an individual assessment. General discussions at a conceptual level will be held during the tutorials. Please be aware of the University regulations about plagiarism and other academic offences:
*University of Bath – Examination and Assessment Offences:*
`https://www.bath.ac.uk/publications/qa53-examination-and-assessment-offences/`

As a final note, please consider that the purpose of this coursework is not testing your abilities and sophistication in programming. Our priority is for you to learn how to apply the acquired formal knowledge and understanding in a more practical environment.

*Date*: 1 November 2019.

## The Assignment

The aim of this coursework is to construct one version of a parser for a given context-free language. It includes also to run and test such implementation on a few input strings, and eventually to adapt the code so to deliver parse trees when this is indeed possible. In practice,

**You will be given:**

- A context-free grammar $G$ which generates the language $L$
- Some input strings in the alphabet $\Sigma$ of terminals of $G$
- Informal description of the algorithm to decide whether or not a grammar generates a string
- Java code which will include:
  - Classes for modelling context-free grammars and their associated components;
  - Classes for producing and displaying parse trees;
  - An interface `IParser` to which your code must conform;
  - A skeleton class `Parser` which you can start modifying with your own code;
  - An interactive script which demos the code.

These resources are accessible on the Moodle pages for CM50260:
*Coursework 3: Parser – The Data*
`https://moodle.bath.ac.uk/mod/page/view.php?id=793076`

**You will have undertaken the following tasks in the given period:**

- Transform $G$ into Chomsky Normal Form;
- Implement the algorithm that we provide to you to determine whether a given string $w$ is generated by $G$;
- Run your implementation on the example strings as evidence for testing the input;
- Write code to adapt your implementation of the algorithm to produce a parse tree for the given strings.

## Guidelines

**Submission Requirements**
Your submission will be a `.zip` file, named as `Surname-Name`, containing the following documentation:

(1) Your working to transform the given grammar in Chomsky Normal Form. This elaboration should be a PDF, preferably typeset in LaTeX or Word (if this is not possible, we can accept very neatly handwritten own elaborations that you will have to scan).
(2) The source code of a single Java class implementation of the algorithm to determine whether a given string is in the language generated by a given grammar. The class must implement (in the Java sense) the interface `IParser`. This will be subject to automatic code testing so it must perform correctly. There are some sample tests built into the demo script that you will have to perform.
(3) Screenshots of your code that produces correct parse trees (or declares that the string is not in the language) for the provided examples.
(4) Your declaration, dated and signed, for academic integrity (text provided)
`‘‘I declare that I have been made aware of the regulations concerning plagiarism and other academic offences at University of Bath, and`

```
I have been warned about penalties for late submissions.  I declare
that all material in this assignment is my own work, that any re-use
of my work, or use of work of others, is referenced appropriately''.
```

Please refer to the Moodle page for CM50260 for the electronic submission (deadline is 13 December at 23:59):
*Coursework 3: Submission*
`https://moodle.bath.ac.uk/mod/assign/view.php?id=793086`

**Grading**
Parts(1)–(3), listed above, respectively carry 9, 25 and 16 marks, for a total of 50 marks of this coursework. The final mark obtained contributes the 40% of the grading for this unit.

We assess in a spirit of positively marking your learning experience for a course of this level. In Part (2), *if your code is not compatible with the tests included with the sample code, marks cannot be awarded.*

## The Data: Grammar, Algorithm and Strings for Testing

The following specification for the coursework is available on Moodle:
*Coursework 3: Parser - The Data*
`https://moodle.bath.ac.uk/mod/page/view.php?id=793076`

**Grammar**
We consider a non-ambiguous grammar $G$ that generates the language of syntactically correct boolean expressions involving disjunction, conjunction, negation and only two identifiers for propositions.

Let $G = (V, \Sigma, R, E)$ be a grammar: $V$ is the *set of non-terminal symbols* (or Variables) that includes the *start symbol* $E$, $\Sigma$ is the *set of terminal symbols*, and $R$ is the *set of production rules*. The grammar $G$ is specified as follows:

$$V = \{E, T, F\} \quad \Sigma = \{+, \&, -, (, ), p, r\}$$

$$\begin{aligned} R = \{ \quad & E \to E + T \\ & E \to T \\ & T \to T\&F \\ & T \to F \\ & F \to (E) \\ & F \to -F \\ & F \to p \\ & F \to r \\ \} \quad & \end{aligned}$$

*Note* The symbols $E, T, F$ are abbreviations for (boolean) expression, term, and factor respectively. The symbols $+, \&$ and $-$ stand for disjunction (OR), conjunction (AND) and negation (NOT), respectively. The symbols $p, r$ are identifiers for propositions. The left and right parentheses are terminals in $\Sigma$ (and not surrounding the ‘,’).

**Strings**
Here are example strings to test your implementation of the algorithm given the

grammar:

$$\begin{array}{l} p + r \\ pp \\ () \\ p\&(r + p) \\ - - p \\ -(p\&r) \\ (p + r)) \\ (p\&r) - p \end{array}$$

**Algorithm**

The following property holds: if the grammar $G$ is in Chomsky Normal Form (CNF), then any derivation of a non-empty string $w \in L(G)$ will have exactly $2n - 1$ steps, where $n = |w|$. The property is the basis for a parsing algorithm, which is specified as follows:

On an input $w$ for grammar $G$:

- List all derivations with $2n - 1$ steps, where $n = |w|$ if $n$ is strictly positive, or list all derivations with 1 step otherwise;
- If any of the derivations so obtained generate $w$, then accept. Otherwise, reject.

The final task of this coursework asks you to modify the algorithm (and hence your implementation) so that you can also output a *parse tree.*

## Underlining Technologies and System Resources

Your code will be integrated in an underlining technology (a Skeleton Code) that provides an interface with a development environment (`repl.it`): this allows you to focus on implementing the separate tasks without having to build the full parser from scratch.

We provide to you an interface to these environments so that you can work on your implementations by logging on Moodle. Alternatively, you can make a direct download of the system on your own computers (and we do not provide assistance in installation and trouble-shooting).

A video is available, giving some hints about how you can configure your workplace, and getting started with the code: this is highly recommended. As a note, the video mentions a platform called Engage (which we do not use, so please ignore that), and it explains the initial steps needed for the construction of a similar grammar.

Moreover, the Skeleton Code documentation provides a list of pre-fabricated classes and methods (in the Java sense) that you may want (or have) to use: this documentation includes some descriptions on how these methods operate.

All these resources are available and accessible from the course Moodle page in relation to Coursework 3:

- The explanatory video:
  *Coursework 3: Getting Started with the Code*
  `https://moodle.bath.ac.uk/mod/page/view.php?id=793078`
- Accessing the Skeleton Code via Moodle (interface at the end of the page):
  *Coursework 3: Parser - The Data*
  `https://moodle.bath.ac.uk/mod/page/view.php?id=793076`
- Alternative direct download of the system:
  *Coursework 3: Parser - Skeleton Code Direct Download*
  `https://moodle.bath.ac.uk/mod/resource/view.php?id=793079`

- Java classes and methods:
  *Coursework 3: Parser - Skeleton Code Documentation*
  `https://moodle.bath.ac.uk/mod/page/view.php?id=793077`

(My gratitude to Andrew Chinery and Michael Wright for sharing resources).

University of Bath (UK)
Claverton Down - Bath BA2 7AY - UK
P.Bruscoli@Bath.Ac.UK
`http://cs.bath.ac.uk/pb`