University of Pisa - Scuola Superiore Sant'Anna
Department of Computer Science

An Imperative-Functional Programming Language

Technical Report

Module: Advanced Programming (301AA)

Lecturers: Gianluigi Ferrari Antonio Cisternino

MSc Computer Science and Networking
Academic Year 2014/2015

Table of Contents

Abstract 2

1 Introduction 3

2 Designing the Language 4

2.1 EBNF Grammar 4

2.2 Submission of Project Report 4

3 Designing the Grammar 5

3.1 Create a Report Structure 5

3.2 Typical Structure of your Report 5

3.3 Order of Writing 9

3.4 Other Advice 10

Abstract

The aim of this report is to offer an overview of the *funW@P* implementation and of the main design choices made during its development. Also, this document is intended both as a reference and a manual for the user wishing to use or extend our simple language.

All the information provided here will also be available at the official page of the project on GitHub (https://github.com/MCSN-project2014/APproject).

Chapter 1: **Introduction**

The report structure follows closely the project's one and focuses on the most important developing phases. After reporting the complete description of the *funW@P* grammar and a brief summary about the tools used for generating the tokenizer, the parser and the type-checker (see Chapter 2), a recapitulation about the chosen intermediate representation (see Chapter 3) is offered. Then, the `F#` code generator (see Chapter **??**) and the language interpreter (see Chapter **??**) are reviewed. To conclude, Chapter **??** and **??** are an overview of the testing phase and a quick user guide respectively.

Chapter 2: **Designing the Language**

The grammar design has been a complex, delicate and continuous task of the project, since the grammar represents the true definition of a language in formal terms and it revealed to be of crucial importance for all the subsequent phases. Lots of on the fly adjustments and corrections leaded to the definition reported in the following section.

2.1 EBNF Grammar

The whole grammar is described by means of the Extended Backus-Naur Form (EBNF, [2]) which is metasyntax, commonly adopted by parser generators nowadays. We first introduce the used tokens, which are the following ones:

TOKENS
```
ident  = letter {letter | digit}.
url = ap "http://" {UrlInLine} ap.
number = digit {digit}.
string = '"' {AnyButDoubleQuote | "\\\""} '"'.
```

Then we provide all the needed productions:

2.2 Submission of Project Report

Chapter 3: **Designing the Grammar**

Report writing is an important skill. No matter what field you are engaged in, you will almost certainly find it necessary to be able to write a clear report on your work. If you have a talent for technical writing, you will no doubt find it an easier task. However, it is a skill that can be acquired with practice and it is an essential part of your project work. Be sure to allow yourself enough time to write the report; the process generally takes **at least** two weeks.

The following sections suggest how to approach the structuring and writing of your report. It is not carved in stone; feel free to adjust this to suit your own particular project/style, but make sure your report is well written.

3.1 Create a Report Structure

The first step is to produce a draft table of contents, showing how the entire report is to be structured into chapters, sections, and even subsections. Annotate each item with the purpose it is to serve in the overall report, and its anticipated length in pages. When you have done this ask yourself the following questions:

- Is there a logical flowthrough my report? If it does not flowlogically at this high-level stage, it certainly will not flow well in the end either. Move sections around until you feel there is a logical thread running through the document.
- Have I written about all the important issues? Pull yourself back from the report and think about the project in general. You should not write about *everything* you did this is a report, not a diary but do not omit any vital sections either.
- Are the issues that I have written about important? You have probably written sections that should really be omitted. It is tough to cut out a section that you have laboured over, but dross in a report has a very negative effect on overall quality.

Once you have a sound overall structure, you can start writing sections knowing that they fit in to an overall plan. You will know how much preparatory material will have preceded each section, and you will know to what extent it is expected to lay the groundwork for later sections. You may find that you have to change the structure later in the writing of the report. As with software, the later you change the design, the more work it entails.

3.2 Typical Structure of your Report

The details of the structure of your report will of course depend on the content and nature of the particular project you are working on. Generally you should break down the report

into approximately six chapters. One possible template you could use is detailed below, but remember that this template is only for guidance. You may decide to merge some chapters, or have an extra core chapter. It all depends on your project and how you wish to present it.

3.2.1 Title page, Table of Contents, and Acknowledgements

The title page should state at least the project title, the name of your supervisors, and all names of the students. A Table of Contents is essential, but should be produced by the wordprocessing package you are using. In your Acknowledgments section, give credit to all the people who helped you in your project.

The Introduction chapter introduces the project and describes the general subject area of the project. Topics it may contain include:

- A discussion of the original aims of the project, and the modified aims if appropriate;
- The scope of the project and a general justification for the work undertaken, perhaps providing a brief background description;
- A description of the structure of the report, i.e., a road map for the reader.

3.2.2 The Abstract

The abstract should provide a short overview of your project that enables a reader to decide if your report is of interest to them or not. It should be concise, to-the-point and interesting. Avoid making it read like a verbose table of contents! Avoid references, jargon or acronyms, as the reader may not be familiar with them. An abstract usually contains a brief description of:

- The project and its context;
- How the project work was carried out;
- The major findings or results.

One paragraph is plenty! The main thing to remember is the principle that the abstract must be short, and a person reading it should be able to determine if they want to read more. For example, if your project involves building a compiler for Java, and a major section of your work is focussed on developing an efficient parser (rather than say code-generation), make this clear in the abstract. Then a reader who is interested in efficient parsing techniques knows that your report may be of interest to them.

3.2.3 Chapter 1: Introduction

This chapter introduces the project and describes the general subject area of the project. Topics it may contain include:

- A discussion of the original aims of the project, and the modified aims if appropriate;

- The scope of the project and a general justification for the work undertaken, perhaps providing a brief background description;
- A description of the structure of the report, i.e., a road map for the reader.

3.2.4 Chapter 2: Background Research

The contents of this chapter depend on the nature of your project. If you are working on a research-oriented project, then you will present the research landscape within which your project is being conducted and consider approaches that have been adopted by other researchers. In a development project, you may describe the domain in which you are working and the technologies and programming tools you are using. Tutorial-type descriptions are never appropriate, but if you are using a specialist tool, e.g., a parser generator, it is reasonable to provide a section that describes the tool at a high level.

3.2.5 Chapters 3 and 4: The Core Chapters

These are the principal chapters of your report and their structure will vary from project to project. The aspects of your project that you will describe in these chapters include:

- A detailed account of howy our approached your project, i.e. the strategy you employed. This should be at a high level, separate from design and implementation issues.
- A discussion of the design aspects of your project. Include here a discussion of interesting problems you encountered and the alternative solutions you considered.

Use the appropriate notations and formalisms in this chapter! Everything you have studied in your degree in relevant here. If there is a crucial algorithm at the centre of your project and its performance is important, attempt to provide an analysis of its complexity. If you are describing a complicated set of conditions, do not write it in English, use first-order logic. If you have performed an object-oriented design (as most of you will), use a UML Class Diagram to give a high level viewof your program. If you are describing how objects interact to perform some task, use a UML Sequence or Interaction diagram. Do not mindlessly produce "documentation", but think about what you want to communicate to the reader and dont be afraid to use the most appropriate method of doing so.

3.2.6 Chapter 5: Detailed Design and Implementation

In doing your project work, a lot of time will be spent on detailed design and implementation. The nature of programming is that it is a very time-consuming task, and even for experts a "silly" run-time error may take days to correct. In spite of this, this chapter should not be the main focus of your report. Make it clear what implementation technology you used and discuss any interesting implementation issues that arose. For example, if you were using a particular data structure that had to be optimised in a certain way to be suitable for your project, describe it in this chapter. On the other hand, if you used an obvious/standard approach, then there is no need to devote much space to it.

3.2.7 Chapter 6: Testing/Evaluation

You may decide to merge this chapter with another, but I have described it as a separate chapter as it is very important in its own right.

If the focus of your project is the development of a piece of software, then you should address the issue of howy ou demonstrate it to be correct. Formal proof is applicable in a small number of cases, but more commonly rigorous testing is required. You will not have time to really test your software in the way that industrial software is tested. However it is important to show that you have taken a methodological approach to testing and that you have tested your software in such a way that you are justified in having some confidence that it is correct. Any Software Engineering text will provide you with the basics of software testing; contact me if you want some notes on the topic.

Another type of project involves designing a heuristic or approximate solution to a challenging or ill-defined problem, e.g., to develop a junk mail filter or to mine a certain type of data from the web. In this case the precise desired behaviour of the software is hard to specify (what is junk mail anyway?), so it is more appropriate to describe howy ou evaluated the solution. This will involve running a number of experiments and presenting the results. As with testing, this is a complex area that you should spend some time coming to terms with. Consult [1] for some excellent advice on how to present the results of your experiments.

3.2.8 Chapter 7: Conclusions and Future Work

If you are writing this chapter bleary-eyed and caffeined-up on the day of submission, you are not going to do your project justice. This is a vital chapter in the assessment of your work. Academics, in getting the feeling for any type of report, will typically read the introduction and conclusions first. Your conclusions should not read like "I did all this stuff, it went great, and heres other stuff someone else might do." This chapter should cover the following areas:

- *Conclusions:* In a research-oriented project you will state the overall conclusions you have come to. In a development project there may not be a conclusion as such, so just state what has been achieved. Be very critical in this section. Describe the weaknesses of your approach and avoid making unwarranted conclusions.
- *Future Work:* Think carefully about howy our work might be extended or applied to another domain. There will probably be some obvious extensions. If you are able to propose some interesting ideas that are not immediately apparent, this demonstrates that you have a clear understanding of the field.

It is good scientific style to make strong statements. If a certain statement is warranted by the results of your project, dont be afraid to make it. Strange though it may seem, a strong statement that turns out to be wrong is better than one that is vague and wishy-washy. The former can lead to a lively debate where the truth may emerge, but the latter will produce meaningless agreement, because it ultimately says nothing.

3.2.9 References

Use one consistent system for citing works in the body of your report. Several such systems are in common use in textbooks and in conference and journal papers. Ensure that any works

you cite are listed in the references section, and vice versa. Word-processing packages will manage the referencing for you, and be sure to make use of this facility. It may take more time in the beginning, but at the end of the write-up it will certainly have saved you a lot of time.

You may instead opt for a bibliography, which is a list of material (books, papers, web resources) that you have read in preparing your project. The bibliography must be annotated, i.e., for each entry you must provide a paragraph summarising the work and stating why it is relevant to your project.

3.2.10 Appendices

Material that you want to include in your report, but that is not directly relevant to the main thread of your report, can be put in an appendix. Possible examples include program/code listings, detailed test results, user guides etc. In most cases, appendices are not necessary and it is only in an exceptional case that it is useful to provide a code listing.

Remember that material in the appendix counts towards report length, so do not exceed the limits defined earlier.

3.3 Order of Writing

The previous section dealt with one possible logical structure for your report. The order in which your write it all is another issue. There are no fixed rules here. Some people like to write notes throughout the project, so when it comes to writing the final report, they already have a lot of material prepared. This is a very valid idea, but avoid wasting time writing very polished notes during your project work. The notes/sections you write can be quite rough, and only in the final report do you bring them up to full report quality. The reason for this is that you may have to tailor them considerably to fit the context of the report, and this will mean that much of the polishing will have gone to waste.

Assuming you have created a report structure as described earlier, a good way to continue is to write the introduction in draft form. You already have an introduction from the interim report, so you can flesh this out. The reason why you write this in draft form is that you are not yet sure what you are introducing! Only when the later chapters are completed can you return and finish the introduction.

Now the Background chapter of the interim report can be revisited and improved for the final report. Again, you may find that when you write the core chapters later, that some of the background work becomes irrelevant and can be removed. This may seem like wasted effort, but if it results in a tighter Background chapter, do it.

Next are the Core chapters, followed by the design and testing chapters. When these are complete, you are in a position write your Conclusions chapter, and to return to the Introduction and Background chapters and bring them to completion. Finally, write the abstract.

The next step depends on how much time you have left. Ideally you will reach this point where you have a first full draft with at least a week to go. Proofread the report yourself, and pass it on to other people who can help you with the English grammar. Take a rest yourself, so you can return to it in a day or two and re-read the report with a fresh mind.

Note that at this late stage you can only make local improvements to the report. It is too late for major overhauls, so at this point the importance of creating a good overall structure becomes clear. If you have started with a good structure, you can aim to create an excellent finished product. However if your initial structure was awkward, the final report will not read well no matter how you tweak it.

3.4 Other Advice

This section contains a number of guidelines that are worth bearing in mind when writing.

3.4.1 Continuity

You may not realise this, but a good academic paper or report, like any good novel you have read, tells a story. It is valuable to keep this in mind when you are writing your report. There should be a storyline running through your report and you should make it easy for the reader to hang on to this storyline:

- At the end of the introduction provide a short description of the layout of the remainder of the report.
- Start every chapter with a brief recapitulation of the story so far, and an overview of what the chapter is going to add.
- Finish every chapter with a summary of the material in that chapter, and state how it relates to what follows.

In the core chapters, you should take care to make absolutely clear the logical connection between the overall project design and the detailed problems you discuss. If the reader is mired in a detailed description of your solution to some intricate problem, they will be encouraged to persevere if you have clearly indicated its place in the overall project.

This continuity material may sound unnecessary and redundant, but it is useful to the reader. It may help for you to imagine that the reader is coming back to your report after a break of a fewda ys: they will be greatly assisted by occasional reminders of what has already been said.

3.4.2 Presentation Issues

Focus on expressing your ideas clearly. Part of your report is of course its physical layout and use of diagrams. Try not to put too much time into this. If you use the provided LaTeX template, it will help you focus on the content as opposed to the form. Simple diagrams are fine, and avoid the use of colour unless it really contributes something in particular. Do not bother with tricks like adjusting spacing or margins or fonts in order to make your report bigger or smaller.

Most final reports will contain a mixture of figures and charts along with the main body of text. The figure caption should appear directly after the figure as seen in Figure 3.1 whereas

a table caption will appear directly above the table. Figures, charts and tables should always be centered horizontally.

Figure 3.1: Logo of the UCD Department of Computer Science displayed at various size.

If you wish to print a short excerpt of your source code, ensure that you are using a fixed-width sans-serif font such as the Courier font. Your code will be properly indented and will appear as follows:

```
static public void main(String[] args) {
  try  {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
  }
  catch(Exception e) {
    e.printStackTrace();
  }
  new WelcomeApp();
}
```

Bibliography

[1] Christian Dawson. *The Essence of Computing Projects – A Student's Guide*. 192 pages. ISBN: 013021972X. Pearson Education, 2000.

[2] *EBNF on Wikipedia*. http:en.wikipedia.orgwikiExtended_Backus%E2%80%93Naur_Form