John Longname Doe

Master of Science

A Very Long and Impressive Thesis Title with a Forced Line Break

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy in
<**Scientific Field of Study**>

Advisers:	Mary Doe Adviser Name, Full Professor, NOVA University of Lisbon
	Mary Doe other Adviser Name, Full Professor, NOVA University of Lisbon
Co-advisers:	John Doe Co-Adviser Name, Associate Professor, NOVA University of Lisbon
	John Doe other Co-Adviser Name, Full Professor, NOVA University of Lisbon

Examination Committee:

Chair:	Name of the committee chairperson, Full Professor, FCT-NOVA
Rapporteurs:	Name of a raporteur, Position, University
	Name of another raporteur, Position, University
Members:	Another member of the committee, Position, University
	Yet another member of the committee, Position, University

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

<**month**>, <**year**>

A Very Long and Impressive Thesis Title with a Forced Line Break

Dedicatory lorem ipsum.

Acknowledgements

The acknowledgements. You are free to write this section at your own will. However, usually it starts with the institutional acknowledgements (adviser, institution, grants, workmates, ...) and then comes the personal acknowledgements (friends, family, ...).

"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)

Abstract

The dissertation must contain two versions of the abstract, one in the same language as the main text, another in a different language. The package assumes that the two languages under consideration are always Portuguese and English.

The package will sort the abstracts in the appropriate order. This means that the first abstract will be in the same language as the main text, followed by the abstract in the other language, and then followed by the main text. For example, if the dissertation is written in Portuguese, first will come the summary in Portuguese and then in English, followed by the main text in Portuguese. If the dissertation is written in English, first will come the summary in English and then in Portuguese, followed by the main text in English.

The abstract shoul not exceed one page and should answer the following questions:

- What's the problem?
- Why is it interesting?
- What's the solution?
- What follows from the solution?

Keywords: Keyword 1, Keyword 2, Keyword 3, ...

Resumo

Independentemente da língua em que está escrita a dissertação, é necessário um resumo na língua do texto principal e um resumo noutra língua. Assume-se que as duas línguas em questão serão sempre o Português e o Inglês.

O *template* colocará automaticamente em primeiro lugar o resumo na língua do texto principal e depois o resumo na outra língua. Por exemplo, se a dissertação está escrita em Português, primeiro aparecerá o resumo em Português, depois em Inglês, seguido do texto principal em Português. Se a dissertação está escrita em Inglês, primeiro aparecerá o resumo em Inglês, depois em Português, seguido do texto principal em Inglês.

O resumo não deve exceder uma página e deve responder às seguintes questões:

- Qual é o problema?
- Porque é que ele é interessante?
- Qual é a solução?
- O que resulta (implicações) da solução?

E agora vamos fazer um teste com uma quebra de linha no hífen a ver se a LaTeX duplica o hífen na linha seguinte...

zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz zzz zzzz comentar-lhe zzz zzzz zzz zzzz

Sim! Funciona! :)

Palavras-chave: Palavra-chave 1, Palavra-chave 2, Palavra-chave 3, ...

Contents

List of Figures xvii

List of Tables xix

Listings xxi

Glossary xxiii

Acronyms xxv

Symbols xxvii

1 Introduction 1

1.1 Typestates . . . 1

1.1.1 What are Typestates? . . . 1

1.1.2 Why are Typestates useful? . . . 1

1.2 State Machines . . . 1

1.3 What is the relation between them? . . . 1

1.4 The Rust Language . . . 1

1.4.1 Why Rust? . . . 1

2 Related Work 3

2.1 Language Preprocessors . . . 3

2.1.1 OCaml . . . 3

2.1.2 Java . . . 4

2.1.3 Kotlin . . . 4

2.1.4 Rust . . . 4

List of Figures

List of Tables

Listings

Glossary

Acronyms

p4 Pre-Processor-Pretty-Printer 3

PPX PreProcessor eXtensions 3

Symbols

Chapter 1

Introduction

1.1 Typestates

1.1.1 What are Typestates?

1.1.2 Why are Typestates useful?

1.2 State Machines

1.3 What is the relation between them?

1.4 The Rust Language

1.4.1 Why Rust?

Chapter 2

Related Work

2.1 Language Preprocessors

Language preprocessors are a mechanism which runs during compilation, some languages will apply the preprocessor during different compilation stages while others will only apply the preprocessor in a single stage.

2.1.1 OCaml

The OCaml ecosystem currently uses OCaml PPX, however, previous to version 4.02, OCaml made use of p4.

We briefly review both p4 and PPX.

Camlp4

Camlp4 is a parsing library which provides extensible grammars, its main goal is to allow users to extend OCaml syntax, Camlp4 is also able to redefine the core syntax, OCaml even introduced a revised syntax[1] to enable Camlp4.

The library has been deprecated due to being confusing to users and tools alike. Users were required to learn the revised OCaml syntax which complicates the development process. These criticisms are found throughout documents which discuss Camlp4[2].

In a nutshell, the Camlp4 library would allow developers to develop an extension syntax, when the compiler would pass the source code as text to the preprocessor, which, in turn would generate valid OCaml source code.

[1] https://caml.inria.fr/pub/docs/manual-camlp4/manual007.html
[2] https://whitequark.org/blog/2014/04/16/a-guide-to-extension-points-in-ocaml/

PPX

2.1.2 Java

As other languages, Java is also capable of source code processing during compile time, we review two existing approaches, annotations and the ExtendJ compiler.

Java Annotation Processor

Java annotations were first introduced in Java 5 (JSR 269), they are a form of metadata which can be added to Java source code. Annotations can be used in conjunction with several components of the Java language, such as classes, interfaces, documentation and others. These are processed by build-time tools or by run-time libraries to achieve new semantic effects, a popular example of such library would be the compile-time dependency injection framework Dagger 2.

Annotation Syntax is simple: every annotation starts with an `@` and is followed by an identifier, optionally, the annotation can take parameters.

```
1 @Annotation
```

ExtendJ & JastAdd

2.1.3 Kotlin

Kotlin Compiler Plugins

2.1.4 Rust