A generalized dynamical systems approach to adaptive compilation on the rust compiler

Emanuel Lima de Sousa

Report presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
for the Master of Science
qualifying examination

Program: Computer Science
Advisor: Prof. Dr. Alfredo Goldman vel Lejbman

São Paulo
October, 2023

A generalized dynamical systems approach to adaptive compilation on the rust compiler

Emanuel Lima de Sousa

This is the original version of the
qualifying text prepared by candidate
Emanuel Lima de Sousa, as submitted
to the Examining Committee.

O God, I could be bounded in a nut shell and count myself
a king of infinite space, were it not that I have bad dreams.
– William Shakespeare

Acknowledgments

If I have seen further than others, it is by standing on the shoulders of giants.

— Isaac Newton

I have much to give thanks for. First and foremost, I would like to thank my parents, Sônia and Agnaldo, who have always supported me in my academic career. When life was at it's toughest and I was at my lowest, they were there to help me get back up. I would also like to thank my brother, Elohim, for all the times we spent cooking and playing video games together, indispensible as it was for me to avoid burning out.

I would also like to thank my friends at the IFUSP Hackerspace, who have always been there for me. Carlos Henrique, Thiago Badaró, Victor Tsutsumiuchi, Priscila Rodrigues, Cristhian Talacimon and many others, thank you for all the laghter and fun times we had together and for helping me during the many surgeries I had to do this year.

In addition, I would like to thank my advisor, Alfredo Goldman, for his guidance and patience. I wouldn't be able to progress on this work without his help. I would also like to thank my undergraduate research co-advisor Pedro Bruel for his help and guidance during that work that funneled directly into this one.

I own a big thanks to Michael Zargham and Jamsheed Shorish from Block Science for their help and guidance on the theoretical aspects of this work. The discussions we had, both in New York and online were instrumental for me to see adaptive compilation as a generalized dynamical system.

And finally, I would like to thank the many open source engineers who made LLVM, Rust and Cargo possible. Without their work, many times thankless, this research would not be possible.

Resumo

Emanuel Lima de Sousa. **Uma abordagem de sistemas dinâmicos generalizados para a compilação adaptativa no compilador rust**. Exame de Qualificação (Mestrado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento. Texto texto. Texto texto.

Palavras-chave: LLVM IR. Rust. Cargo. Adaptive Compilation. GDS.

Abstract

Emanuel Lima de Sousa. **A generalized dynamical systems approach to adaptive compilation on the rust compiler**. Qualifying Exam (Master's). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Keywords: LLVM IR. Rust. Cargo. Adaptive Compilation. GDS.

Lista de abreviaturas

LLVM	*Low Level Virtual Machine*
IR	*Intermediate Representation*
HIR	*High-level Intermediate Representation*
MIR	*Middle-level Intermediate Representation*
GDS	*Generalized Dynamical System*
HPC	*High Performance Computing*

List of Figures

List of Tables

List of Programs

Contents

Introduction

As defined by Matsakis and Klock (2014) the Rust programming language is a modern systems programming language focused on safety, speed and concurrency. It compares to C++ in terms of mapping directly to hardware, but it is safer to use thanks to the guarantees it's static type system provides. The language has been successfully used on a number of different domains, including astrophysics (Blanco-Cuaresma and Bolmont, 2016) and operating system kernels development (Levy *et al.*, 2017).

The Rust compiler is a complex piece of software, with many different components. It is written in Rust itself, making it a self-hosted language, and it is composed of a Rust front-end, a LLVM middle-end and a LLVM back-end.

As described by the The Rust Project Developers (2018) the front-end is responsible for parsing the source code, desugaring, checking and validating it, and generating a first intermediate representation of the code, called High-level Intermediate Representation (HIR). After this, the middle-end is responsible for lowering the HIR to a lower-level representation called Middle-level Intermediate Representation (MIR) by doing type checking. On the next step, the compiler enforces the borrow checking rules, which are one of the main features of the Rust language, and executes some initial optimizations, finally lowering the MIR to an IR called LLVM IR. From here on, the LLVM middle-end enters and starts to transform the LLVM IR into a more optimized version of itself. Finally, the LLVM back-end is responsible for lowering the LLVM IR to the target machine assembly code. It's important to notice that the Rust compiler is not tied to the LLVM compiler framework, but it is the most used back-end. A high-level overview of the Rust compilation pipeline can be seen on Figure 1.

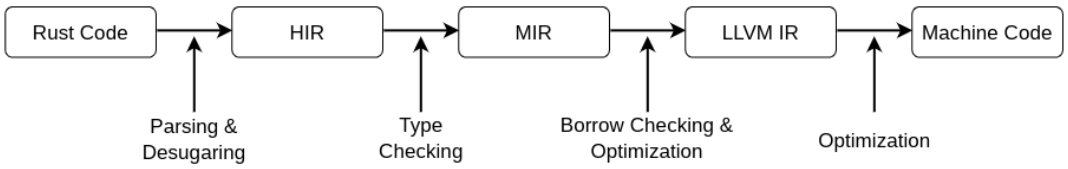


Figure 1: *Overview of the Rust compilation pipeline*

LLVM Optimizations

As Lattner and Adve (2004) describe, the LLVM compiler framework is a collection of modular and reusable compiler and toolchain technologies. It is designed to be used as

a back-end for a wide variety of programming languages, and it is used as the middle-end and back-end for the Rust compiler. One of the central features of LLVM is it's intermediate representation (IR), which is a low-level programming language similar to assembly. The IR is designed to represent programs in a form that is independent of the source programming language, and it is used as the common language between the different components of the compiler. The IR is also designed to be a target of both static and dynamic compilation techniques, and it is suited for optimization and analysis.

The LLVM IR is a typed, static single assignment (SSA) representation of a program, designed to be used in three different forms: in memory, as an on-disk bitcode representation (suitable for fast loading by a Just-In-Time compiler), and as a human readable assembly language representation. The LLVM IR is defined in three different levels of abstraction: the high-level virtual instruction set, the low-level virtual instruction set, and the concrete representation. The high-level virtual instruction set is the most abstract representation, and it is used by the front-end of the compiler. The low-level virtual instruction set is a more concrete representation, and it is used by the middle-end of the compiler. The concrete representation is the most concrete representation, and it is used by the back-end of the compiler. The LLVM IR is a static single assignment representation, which means that each value is assigned only once, and it is defined before it is used. This property simplifies the analysis and transformation of the code, and it is a requirement for some optimizations.

There is a set of passes that can be applied to the LLVM IR, and they are divided into two categories: analysis passes and transformation passes. Analysis passes are used to obtain information about the code, without changing it. That information can be queried by a transformation pass or by other analysis passes. The result of an analysis pass is always cached and will only be invalidated if the unit of IR used by it has changed. Transformation passes are used to change the code in some way. The operation is done in-place and always leaves the IR in a valid state. A transformation pass can depend on the result of an analysis pass and will only run after querying it.

Table A.1 describe all LLVM IR analysis passes, and Table B.1 the transformation passes, that are available in the LLVM 16.0.6 release.

Adaptive Compilation

As described by Cooper *et al.* (2005) adaptive compilation is a technique that allows a compiler to increase the performance of a program by monitoring it's execution and recompiling it with different optimizations. The compiler can use the information gathered during the execution to decide which optimizations to apply, and it can also use the information to decide when to recompile the program. The compiler can also use the information to decide when to stop recompiling the program, and it can also decide to recompile the program with less optimizations if it detects that the program is not benefiting from them. As such, an adaptive compiler can be seen as a compiler that is able to adapt to the behavior of the program through a compile-execute-analyze feedback loop, based on some well defined performance metric, such as binary size or execution time. The main task of an adaptive compiler becomes then to find the best set of optimizations

for a given program.

The main advantage of adaptive compilation is that it allows the compiler to make decisions based on the actual behavior of the program, instead of making decisions based on the static analysis of the code. This allows the compiler to make better decisions and to generate better code. The main disadvantage of adaptive compilation is the large amount of time it takes to walk over the search space of all optimization passes.

Chapter 1

Theoretical Underpinnings

On this chapter we present the theoretical underpinnings of this work, including a non exhaustive bibliographical review about adaptive compilation and generalized dynamical systems. We start by presenting the main concepts of the Rust language compilation process and how the LLVM compiler framework fits in the whole picture. After that, we present the concept of adaptive compilation and how it can be used to improve the performance of a program. Finally, we present the concept of generalized dynamical systems, as a way to model the adaptive compilation process.

Chapter 2

Usando este modelo

Não é necessário que o texto seja redigido usando LaTeX ou este modelo, mas seu uso é fortemente recomendado, pois ele facilita diversas etapas do trabalho e o resultado final é muito bom[1]. Este modelo é distribuído com uma "colinha" dos principais comandos LaTeX e inclui comentários explicativos para auxiliá-lo com ele, sendo composto dos arquivos:

- `tese.tex`, `artigo.tex`, `apresentacao.tex` e `poster.tex` (exemplos de cada um desses tipos de documento);
- Capítulos, apêndices, imagens etc. deste texto de exemplo, nos diretórios `conteudo`, `figuras` e `logos` (procure os comandos `\input` e `\graphicspath` nos arquivos de exemplo mencionados acima para modificar o nome desses diretórios);
- `bibliografia.bib` (exemplo de banco de dados bibliográficos; procure o comando `\addbibresource` nos arquivos de exemplo mencionados acima para modificar o nome desse arquivo ou acrescentar outros);
- Arquivos com as configurações e *packages* usadas, no diretório `extras` (você só precisa mexer neles se quiser aprender mais sobre LaTeX ou modificar/acrescentar algo ao modelo).

Para compilar o documento, basta executar o comando latexmk[2]. Talvez seu editor ofereça uma opção de menu para compilar o documento; sempre que possível, configure-o para utilizar o latexmk ao selecioná-la. LaTeX gera diversos arquivos auxiliares durante a compilação que, em algumas raras situações, podem ficar inconsistentes (causando erros de compilação ou erros no PDF gerado, como referências faltando ou numeração de páginas incorreta no sumário). Nesse caso, é só usar o comando latexmk -C, que apaga todos esses arquivos auxiliares gerados, e em seguida rodar latexmk novamente.

Você pode mudar a língua do documento para o inglês no início de cada arquivo .tex de exemplo, na linha \documentclass. No caso do arquivo tese.tex, isso muda todos os textos padrão da capa e folhas de rosto.

---

[1] O uso de um sistema de controle de versões, como mercurial (mercurial-scm.org) ou git (git-scm.com), também é altamente recomendado.

[2] Você também pode usar latexmk poster, latexmk apresentacao etc.

Os arquivos deste modelo, incluindo os do diretório `extras`, incluem vários comentários com dicas e explicações; se o que você precisa não está mencionado diretamente, é provável que haja pelo menos a indicação da *package* relacionada ao que você precisa.

Se você encontrar algum problema com o modelo, ajude a melhorá-lo! Envie um relatório de erro ou entre em contato em gitlab.com/ccsl-usp/modelo-latex.

Chapter 3

Instalação do LaTeX

LaTeX é, na verdade, um conjunto de programas. Ao invés de procurar e baixar cada um deles, o mais comum é baixar uma coleção com todos eles juntos. Há duas coleções desse tipo disponíveis: MiKTeX (miktex.org) e TeXLive (www.tug.org/texlive). Ambos funcionam em Linux, Windows e macOS. Em Linux, TeXLive costuma estar disponível para instalação junto com os demais opcionais do sistema. Em macOS, o mais popular é o MacTeX (www.tug.org/mactex/), a versão do TeXLive para macOS. Em Windows, o mais comumente usado é o MiKTeX.

Por padrão, eles não instalam tudo que está disponível, mas sim apenas os componentes mais usados, e oferecem um gestor de pacotes que permite adicionar outros. Embora uma instalação completa do LaTeX seja relativamente grande (perto de 5GB), em geral vale a pena instalar a maior parte dos componentes. Se você preferir uma instalação mais "enxuta", não deixe de incluir tudo que é necessário para este modelo, como indicado no arquivo README.md.

Também é muito importante ter o latexmk. No Linux, a instalação é similar à de outros programas. No macOS e no Windows, latexmk pode ser instalado pelo gestor de pacotes do MiKTeX ou TeXLive. Observe que ele depende da linguagem perl. No macOS, perl já faz parte do sistema; no Windows, TeXLive inclui uma versão básica de perl, mas se você estiver usando MiKTeX será preciso instalar perl manualmente (www.perl.org/get.html).

3.1 Documentação sobre LaTeX

Há muito material sobre LaTeX na Internet, mas também há muita informação obsoleta (incluindo trechos da própria documentação oficial!). Em particular, você pode ignorar explicações sobre como converter arquivos no formato DVI gerados por LaTeX em PDF: as versões atualmente recomendadas de LaTeX (cf. Seção 4.11) geram arquivos PDF diretamente. Quanto a imagens, os formatos de arquivo PS/EPS (PostScript e Encapsulated PostScript) não são adequados para essas novas versões de LaTeX; elas trabalham com arquivos de imagem nos formatos PDF, PNG e JPEG. Finalmente, recursos gráficos normalmente não usam mais *packages* como pstricks, eepic ou outras tradicionalmente citadas; ao invés disso, PGF/TikZ é a ferramenta mais comum.

Como dito anteriormente, LaTeX é, na verdade, um conjunto de programas e, em geral, instalamos coleções pré-prontas com todos eles. Essas coleções (TeXLive e MiKTeX) contêm também a documentação das *packages* incluídas: Basta digitar texdoc nome-da-package (TeXLive) ou mthelp nome-da-package (MiKTeX) para ter acesso à documentação correspondente. texdoc/mthelp incluem também alguns tutoriais e textos introdutórios.

Um possível caminho para o aprendizado é começar com o Capítulo 4 deste modelo e o conteúdo em overleaf.com/learn, que tem escopo similar mas também inclui várias páginas sobre como utilizar recursos específicos. Após esse contato inicial, o tutorial em tug.org/twg/mactex/tutorials/ltxprimer-1.0.pdf é bastante abrangente e detalhado. Não deixe de ver também o Capítulo 5 deste modelo (e seu código-fonte), que inclui várias dicas úteis. Para os principais comandos do modo matemático, veja texdoc undergradmath e, para aprender a criar apresentações, veja texdoc beamer.

Depois que você estiver razoavelmente familiarizado com a linguagem, utilize o manual de referência que pode ser acessado em latexref.xyz ou com texdoc latex2e (disponível também em francês, com texdoc latex2e-fr.pdf, e em espanhol, com texdoc latex2e-es.pdf).

A documentação de referência mais importante sobre os recursos matemáticos é acessível com texdoc amsmath, texdoc amsthm e texdoc mathtools; texdoc maths-symbols agrega os símbolos matemáticos disponíveis. Para uma lista completa de todos os símbolos disponíveis com LaTeX, use texdoc symbols-a4 (esse documento tem mais de 300 páginas!).

Existem também diversos bons livros sobre LaTeX (embora em geral um tanto antigos), dos quais destacamos dois:

1. A quarta edição de "A Guide to LaTeX", de Helmut Kopka e Patrick W. Daly (publicada em 2003), além de uma ótima introdução, aborda vários tópicos relativamente avançados e úteis[1].
2. A segunda edição de "The LaTeX Companion" (publicada em 2004) é um livro quase obrigatório, pois discute em detalhes praticamente todos os recursos e *packages* importantes de LaTeX, servindo tanto para o aprendizado quanto como material de referência.

Para dúvidas pontuais, o sítio tex.stackexchange.com é um fórum de perguntas e respostas sobre LaTeX muito útil, pois os principais desenvolvedores do sistema participam das discussões, e o sítio texfaq.org é bastante abrangente e atualizado.

Existem inúmeras alternativas aos materiais citados acima; outros exemplos de textos introdutórios são www.maths.tcd.ie/~dwilkins/LaTeXPrimer/GSWLaTeX.pdf e www.andy-roberts.net/writing/latex. Em português, você pode consultar polignu.org/sites/polignu.org/files/latex/latex-fflch.pdf e git.febrace.org.br/material-latex/material-latex

[1] Uma versão não-final está disponível em www2.mps.mpg.de/homes/daly/GTL/gtl_20030512.pdf.

(este precisa ser baixado e compilado). O canal youtube.com/c/anteroneves tem vários vídeos instrutivos em português. texdoc/mthelp incluem ainda opções como "The Not So Short Introduction to LaTeX 2_ε" (texdoc lshort-eng; há uma versão em português, mas não está em dia com o original) e "A Simplified Introduction to LaTeX" (texdoc simplified-intro). Versões recentes do LaTeX incluem também o "LaTeX 2_ε via exemplos" (texdoc latex-via-exemplos), em português.

3.1.1 Outros recursos (avançados)

O sítio ctan.org é o repositório semi-oficial das *packages* LaTeX e sua documentação; TeXLive e MiKTeX são construídas a partir do que está nesse site, então a última versão estável de qualquer *package* (e da documentação acessível com texdoc/mthelp) em geral está ali.

texdoc fntguide explica como funciona a gestão de fontes de LaTeX, e você pode ver exemplos de fontes disponíveis para LaTeX em tug.org/FontCatalogue. LuaLaTeX e XeLaTeX funcionam de outra maneira, permitindo também o uso das fontes comuns instaladas no seu sistema operacional (veja texdoc fontspec).

Minúcias sobre o funcionamento interno do sistema estão descritas em texdoc source2e e, sobre as classes padrão (article, book etc.), em texdoc classes. Você normalmente não vai usar esses documentos, mas eles podem servir para esclarecer algum detalhe. texdoc macros2e, texdoc xparse e texdoc interface3 apresentam a linguagem de programação usada por LaTeX, enquanto texdoc clsguide é um guia para a criação de novas classes e *packages*.

Quando você se tornar um usuário avançado, pode se interessar em conhecer melhor a linguagem TeX, que está na base do LaTeX. "The TeX book", de Donald Knuth (o criador do TeX), é amplamente recomendado, mas há três livros completos a respeito que são instalados com LaTeX: "A gentle introduction to TeX" (texdoc gentle), "TeX for the impatient" (texdoc impatient) e "TeX by topic" (texdoc texbytopic).

Chapter 4

Do zero ao mínimo com LaTeX

Neste capítulo, apresentamos uma visão geral sobre LaTeX para quem nunca trabalhou com ele antes. Se você já tem conhecimento básico ou intermediário sobre o sistema, sinta-se à vontade para ir diretamente ao Capítulo 5, que inclui diversos exemplos e dicas úteis. A intenção deste capítulo não é propriamente ensinar a usar LaTeX, mas sim expor seus princípios de funcionamento e principais recursos, de maneira que o leitor esteja melhor capacitado a compreender outros documentos e exemplos.

4.1 Por que LaTeX?

Preparar um texto para impressão envolve duas coisas:

Escrever: digitar, recortar/colar trechos, revisar etc.

Formatar: definir o tamanho da fonte, o espaçamento entre parágrafos etc.

Hoje é comum fazer essas duas coisas ao mesmo tempo, graças à visualização imediata que o computador oferece. No entanto, imagine como era o processo de produção de um livro nos anos 1970: o autor escrevia seu texto em uma máquina de escrever e enviava esse material para o editor, que era responsável pela tarefa de formatá-lo para impressão. O autor muitas vezes inseria anotações para o editor explicando coisas como "este parágrafo é uma citação", e o editor criava algum mecanismo visual para representar isso.

Não é de se surpreender que, com o surgimento do microcomputador, os primeiros programas para criação de textos seguissem um funcionamento similar: o autor digitava e editava seu texto sem formatá-lo visualmente, apenas inserindo alguns comandos correspondentes a aspectos da formatação que ele depois revisava na versão impressa. LaTeX é uma ferramenta baseada nesse processo: você prepara seu texto no editor de sua preferência, insere comandos no texto que indicam a estrutura do documento e o processa com o LaTeX, que gera um arquivo PDF formatado. Embora seja um estilo "antigo" de trabalhar, ele é muito eficiente em vários casos. Ou seja, dependendo da situação, pode ser mais adequado trabalhar fazendo tudo ao mesmo tempo ou dividindo o trabalho nessas duas fases. De maneira geral:

- Se você precisa criar páginas diferentes entre si com *layout* definido manualmente, é melhor usar uma ferramenta que permita trabalhar visualmente, como LibreOffice Writer, MS-Word, Google Docs etc.;
- Se você precisa fazer um documento relativamente longo com estrutura regular (capítulos, seções etc.), é melhor usar ferramentas que formalizam essa estrutura (como LaTeX) ao invés de ferramentas visuais;
- Se você precisa fazer um documento envolvendo referências cruzadas, bibliografia relativamente extensa ou fórmulas matemáticas, é difícil encontrar outra ferramenta tão eficiente quanto LaTeX;
- Se você precisa criar um documento simples, ambas as abordagens funcionam bem; cada um escolhe esta ou aquela em função da familiaridade com as ferramentas;
- Se você quer que a qualidade tipográfica do resultado seja realmente excelente, é necessário usar uma ferramenta profissional, como LaTeX, Scribus, Adobe InDesign ou outras; processadores de texto convencionais não oferecem o mesmo nível de qualidade dessas ferramentas[1].

4.2 Visão geral

Com LaTeX, você prepara o texto (incluindo as indicações de estrutura) em um editor de textos qualquer, salva como arquivo de texto puro (".txt", mas é comum usar a extensão ".tex" ao invés de ".txt") e processa esse arquivo com o comando "latexmk" ("compila" o documento) para obter o PDF correspondente. Qualquer editor capaz de salvar arquivos em formato texto puro, como o bloco de notas do windows, vim, emacs etc. pode ser usado. Programas como LibreOffice Writer, MS-Word etc. também funcionam, mas possivelmente vão gerar dores de cabeça porque vão tentar formatar algumas coisas automaticamente (e de maneira incompatível com LaTeX).

Em geral, é recomendável usar editores projetados especificamente para trabalhar com LaTeX; eles utilizam cores para distinguir o texto dos comandos de formatação, automatizam o processo de compilação do documento (veja a Seção 4.4) e oferecem outras comodidades. O mais usado atualmente é o TeXstudio, que é software livre e funciona em Windows, macOS e Linux. O editor Visual Studio Code (code.visualstudio.com) é voltado para programadores e tem uma interface às vezes peculiar para outros usuários, mas em conjunto com a *package* LaTeX Workshop (do editor, não do LaTeX), é uma boa opção. O mesmo vale para o editor emacs (www.gnu.org/software/emacs) e sua package AUCTeX. Ainda outra possibilidade são os editores *online*; dentre eles, o overleaf (www.overleaf.com) é o mais usado.

Um documento LaTeX é dividido em duas partes: o *preâmbulo*, onde você coloca comandos de configuração para o documento, e o *corpo* do documento em si, que contém o texto propriamente dito. O preâmbulo é onde você define as características do resultado tipográfico esperado para o documento como um todo: tipo e tamanho da fonte a usar,

[1] A maior diferença (mas não a única) é o algoritmo que divide cada parágrafo em uma série de linhas: TeX (desde 1982) e Adobe InDesign (desde 1999) analisam cada parágrafo como um todo, ao invés de uma linha por vez, para obter espaçamentos mais homogêneos e menos palavras hifenizadas.

posição dos títulos e subtítulos na página etc. O corpo, por sua vez, consiste no texto e em alguns comandos indicativos da estrutura.

Dado que configurar o preâmbulo é um tanto complexo e que mesmo no corpo do texto às vezes há comandos especiais (para a geração da bibliografia ou tabelas, por exemplo), usar algum documento existente como base para criar seu texto em geral é uma boa ideia. O IME/USP oferece um conjunto de modelos adequados para teses/dissertações, artigos, apresentações e pôsteres (gitlab.com/ccsl-usp/modelo-latex) que pode ser adaptado para outros usos e outras instituições. Há também uma família de modelos (www.abntex.net.br) que procura seguir as normas da ABNT para diversos tipos de documentos científicos, e algumas publicações científicas fornecem modelos de acordo com suas diretrizes.

4.3 Estrutura de um documento LaTeX

O preâmbulo LaTeX começa com a definição da *classe* a ser utilizada, que determina boa parte da configuração do documento. As principais classes são book, article e beamer (para apresentações); você pode saber mais sobre elas (e outras) em qualquer texto introdutório sobre LaTeX na Internet (veja a Seção 3.1)[2]. A seguir, são carregadas várias *packages* ("*plugins*") que acrescentam funcionalidades ou modificam as classes padrão; qualquer documento LaTeX utiliza várias delas. A classe é definida com o comando \documentclass{nome-da-classe}; packages são carregadas com o comando \usepackage{nome-da-package}. Classes e packages podem receber opções adicionais entre colchetes (\usepackage[opção1,opção2...]{nome-da-package}); a documentação de cada package e classe (veja a Seção 3.1) detalha as opções disponíveis.

LaTeX ignora quebras de linha e trata sequências de vários espaços como se fossem apenas um. Isso significa que você pode usar quebras de linha e espaços no texto que está digitando como "dicas visuais" da estrutura do texto durante a edição. É muito comum fazer isso com listas de itens, por exemplo (veja a Seção 4.5). Uma ou mais linhas em branco sinalizam o fim de um parágrafo e o início de outro. O caractere "%" indica que o restante da linha é um comentário, ou seja, um trecho de texto que não tem nenhum efeito sobre o resultado final do documento. Comentários podem ser usados como lembretes sobre alguma decisão, para indicar um parágrafo que ainda precisa de revisão etc. Por conta desse significado especial, para inserir um caractere % "normal" no texto é preciso digitar "\%".

Como mencionado anteriormente, LaTeX divide o trabalho de produção de um texto entre a preparação do conteúdo e a definição da forma de apresentação. Assim, os comandos usados durante a produção do conteúdo procuram expressar o *significado* de cada elemento, e não sua aparência. Por exemplo, para realçar uma palavra é comum usar texto *em itálico*; embora exista um comando especificamente para gerar textos em itálico em LaTeX, o recomendado é que se utilize o comando \emph ("enfatizado"), pois em alguns casos pode ser melhor utilizar **negrito**, VERSALETE ou outro mecanismo para dar ênfase a uma palavra. Essa é uma orientação geral para a escrita de textos com LaTeX: procure definir a estrutura, não a aparência.

[2] Algumas revistas acadêmicas têm suas próprias classes; por exemplo, a AMS (American Mathematical Society) disponibiliza as classes amsart, amsbook e amsproc; você pode usá-las para seus trabalhos mesmo que não pretenda publicar com a AMS, veja texdoc Author_Handbook_Journals.

Um exemplo de documento LaTeX simples (lembre-se, "%" indica um comentário):

```
% O documento começa com o preâmbulo
% Vamos usar a classe "book" com fonte no tamanho 11pt
\documentclass[11pt]{book}
% Vamos escrever em português do Brasil
\usepackage[brazilian]{babel}
% Finaliza o preâmbulo e inicia o conteúdo:
\begin{document}
% Estas linhas não imprimem nada, apenas definem as
% informações que serão usadas por "\maketitle" a seguir
\author{Fulano de Tal}
\title{Começando a usar o \LaTeX{}}
% Cria um bloco ou página de título com os dados acima
\maketitle
% Capítulos, seções etc. são numerados automaticamente
\chapter{Cheguei!}
Oi, Galera!
% É preciso sinalizar o final do documento
\end{document}
```

Esse exemplo mostra como definir o nome de um capítulo. Existem também os comandos \section, \subsection, \subsubsection e \paragraph (a classe book inclui também \part, um nível acima de \chapter). Usar o nome do comando seguido de um asterisco (\chapter* etc.) faz o capítulo/seção não ser numerado (nem considerado na contagem de capítulos, seções etc.) nem incluído no sumário. Este modelo ainda define \unnumberedchapter, \unnumberedsection e \unnumberedsubsection, que eliminam a numeração mas incluem o capítulo ou seção no sumário (úteis para a introdução, por exemplo).

4.4 Executando LaTeX e comandos auxiliares

Depois de escrever o arquivo .tex, é preciso *compilá-lo*, ou seja, processá-lo para gerar o PDF desejado. Isso envolve executar, além do próprio LaTeX (veja a Seção 4.11), alguns programas auxiliares (em geral, biber ou bibtex e makeindex). Nesse processo, LaTeX quase sempre precisa ser executado três ou mais vezes antes de gerar o PDF final[3]. Por conta dessa complexidade, é comum utilizar alguma ferramenta para automatizar o processamento. Existem diversas opções, mas a mais comum é o latexmk, que é capaz de identificar automaticamente os passos necessários para a geração do documento, executando os programas na ordem correta quantas vezes forem necessárias[4]. Assim, embora seja possível gerar o PDF executando apenas pdflatex nome-do-arquivo.tex, acostume-se a compilar o documento sempre com latexmk -pdf nome-do-arquivo.tex. Note que editores especializados em LaTeX costumam ter uma opção de menu para a compilação do documento; dentre as configurações possíveis do editor, prefira sempre a que simplesmente aciona latexmk.

[3] A cada vez, ele gera uma nova versão intermediária do arquivo PDF, mas essas versões têm defeitos, como citações e referências cruzadas incorretas ou sumário inexistente.

[4] É possível personalizar o comportamento de latexmk com o arquivo de configuração latexmkrc.

4.5 Mais sobre estrutura

Para criar listas de itens, você pode fazer[5]:

```
\begin{itemize}
    \item Pedra
    \item Papel
    \item Tesoura
\end{itemize}
```

Além de "itemize", há também "enumerate" (auto-explicativo) e "description":

```
\begin{description}
    \item[Pedra:] perde para papel;
    \item[Papel:] perde para tesoura;
    \item[Tesoura:] perde para pedra.
\end{description}
```

Citações curtas normalmente são incluídas no fluxo normal do texto e colocadas entre aspas; para citações mais longas, use \begin{quote} ou \begin{quotation} (este último é mais adequado para citações com vários parágrafos). A package csquotes acrescenta recursos sofisticados para citações.

Para poesia, use \begin{verse} (a package verse acrescenta vários recursos ao comando verse). Estrofes são separadas por uma linha em branco e versos são separados por *. O asterisco é opcional; ele instrui LaTeX a manter as linhas na mesma página.

Para inserir uma nota de rodapé, use o comando \footnote{texto da nota}.

4.6 Figuras e tabelas (*floats*)

É possível utilizar \includegraphics para acrescentar figuras ao texto (nos formatos PDF, PNG e JPEG), mas normalmente elas não são inseridas diretamente. A razão é que, se você simplesmente inserir uma figura em qualquer lugar, ela pode ser grande demais para o espaço disponível na página, o que forçará LaTeX a deixar um espaço em branco e colocá-la na página seguinte. O mesmo vale para tabelas (criadas com \begin{tabular}). Para contornar esse problema, LaTeX possui *floats*, que são blocos com algum conteúdo cuja localização é flexível: LaTeX procura colocar um *float* "perto" de onde ele foi definido, mas não necessariamente no lugar exato.

Ao invés de um único comando como "\begin{float}" a ser usado tanto para figuras quanto para tabelas, LaTeX define \begin{figure} e \begin{table}. Ele faz isso porque, assim como com capítulos e seções, LaTeX também numera figuras e tabelas — mas, para isso, ele precisa saber qual é o tipo de cada *float*[6]. À parte isso, o conteúdo de um *float* pode ser qualquer coisa mas, em geral, é \includegraphics ou \begin{tabular} respectivamente.

---

[5] Observe o uso de espaços no início das linhas com \item para deixar a estrutura visualmente mais clara durante a edição.

[6] É possível criar outros tipos de *float* também: como pode ser visto no Captítulo 5, este modelo define o tipo program.

Uma consequência importante (e proposital) dos tipos diferentes de *floats* é que LaTeX garante que a sequência das figuras e a sequência das tabelas sejam respeitadas (a Figura 6 nunca aparece depois da Figura 7). No entanto, isso *não* se aplica a *floats* de tipos diferentes, ou seja, se você definiu a Figura 5, a Tabela 3 e a Figura 6, elas podem aparecer no documento na ordem "Figura 5, Tabela 3, Figura 6", "Figura 5, Figura 6, Tabela 3" ou "Tabela 3, Figura 5, Figura 6".

4.7 Referências cruzadas

É comum que um trecho do texto faça referência a outro trecho ("como discutimos no Capítulo X…"). Isso pode ser feito diretamente, mas se você reorganizar o documento ou acrescentar seções, a numeração pode mudar. Para evitar esse problema, você pode gerar essas referências automaticamente com o par de comandos \label{nome-sugestivo} e \ref{nome-sugestivo} (para o número da seção/capítulo) ou \pageref{nome-sugestivo} (para o número da página).

Esse mecanismo também é muito útil para figuras e tabelas. Dentro do *float*, além da figura em si, em geral é uma boa ideia acrescentar uma legenda com \caption. Além disso, é possível inserir um \label dentro da legenda para que se possa fazer referência à figura/tabela no texto (com os comandos \ref e \pageref).

4.8 Referências bibliográficas e bibliografia

A geração de bibliografias no LaTeX é feita através da package biblatex e do programa auxiliar biber[7] e envolve três passos:

1. A criação de um banco de dados, no formato ".bib", das obras de interesse. Esse banco de dados pode incluir obras que não vão ser de fato referenciadas no documento final. Isso significa que você pode criar um único banco de dados e utilizá-lo em todos seus documentos[8].
2. A inserção de referências às obras ao longo do texto, usando diferentes comandos dependendo do caso: \cite, \citet, \citep etc. Como já mencionado, esses comandos estão descritos na documentação da package natbib (Daly, 2010).
3. A escolha do estilo bibliográfico (usando as opções da package biblatex) que formata as citações ao longo do texto e gera a bibliografia automaticamente através do comando \printbibliography. Normalmente, apenas as obras efetivamente citadas são incluídas na lista de referências, mas é possível forçar a inclusão de uma obra sem citá-la explicitamente com o comando \nocite.

O banco de dados é um arquivo de texto contendo uma *entrada* para cada item da bibliografia e, em cada entrada, uma série de *campos* com os dados (título, autor etc.). A entrada inclui também uma *chave*, que é usada para inserir as citações no texto. Há vários

[7] Antigamente, usava-se a package natbib e o comando auxiliar bibtex. O funcionamento geral dos dois mecanismos é similar e o formato do banco de dados de ambos é o mesmo.

[8] É comum criar bancos de dados desse tipo separados por assunto, mas isso não é necessário.

tipos de entrada (para artigos, livros, sítios web etc.) e, para cada tipo, uma lista de campos possíveis (considere que periódicos normalmente incluem o número do volume, mas teses não). O exemplo abaixo é um livro cuja chave é "dissertjourney"; ele pode ser citado com o comando \cite{dissertjourney}:

```
@book{dissertjourney,
    author    = {Carol M. Roberts},
    title     = {The Dissertation Journey},
    publisher = {Corwin},
    year      = 2010,
    edition   = 2,
    location  = {Thousand Oaks, CA},
}
```

Em alguns casos, LaTeX troca as letras maiúsculas definidas em \title para minúsculas. Para evitar que isso afete siglas ou nomes próprios, basta colocá-los entre chaves ("Automated Application-Level Checkpointing of {MPI} Programs").

Os campos author e publisher podem incluir uma lista de nomes separados por and; biblatex reconhece que cada nome é composto por nome e sobrenome, às vezes com partículas como "de", "dos" ou "von" e, dependendo do estilo bibliográfico, pode abreviar nomes, mudar sobrenomes para caixa alta etc. Isso evidentemente não funciona quando o autor é, na verdade, uma instituição; nesses casos, basta colocar o nome inteiro da instituição entre chaves ("{Universidade de São Paulo — Sistema Integrado de Bibliotecas}") para que biblatex não faça alterações desse tipo. Se o nome é longo, pode ser interessante definir o campo shortauthor.

A fonte mais detalhada de informações sobre o banco de dados é a documentação da package biblatex (Lehman *et al.*, 2018, em especial as seções 2.1.1 e 2.2.2), mas o material ali é um tanto denso. Há muito material introdutório ao formato ".bib" e ao bibtex disponível *online*, e você pode se inspirar em exemplos para criar seu banco de dados bibliográfico. Além disso, ferramentas como Zotero ou Mendeley (o uso de uma delas é altamente recomendado!) podem exportar para o formato .bib. Observe que biblatex oferece recursos bastante sofisticados para o tratamento de referências e bibliografias. Se você precisar de alguma funcionalidade especial, consulte a documentação do pacote ou a Internet; é quase certeza que biblatex oferece uma solução.

4.9 Fórmulas matemáticas

A diagramação de fórmulas matemáticas tem regras específicas: letras são interpretadas como variáveis e espaços em branco são ignorados (LaTeX usa o contexto da fórmula para definir o espaçamento). Assim, para criar fórmulas em LaTeX, é preciso usar um comando para iniciar o modo matemático. Isso pode ser feito de duas formas:

- Pequenas fórmulas no meio do texto ($e^{i\pi} + 1 = 0$) são inseridas com $*fórmula*$ (e, portanto, para inserir um caractere $ normal no texto, é preciso usar \$).

- Fórmulas mais longas ou que devem aparecer em um parágrafo separado são inseridas com \[*fórmula*\] (ou \begin{displaymath}).

LaTeX é capaz de oferecer uma boa solução para praticamente qualquer problema de diagramação para matemática; basta ler a documentação.

4.10 Formatação manual

Às vezes é preciso inserir formatação de forma manual; os comandos mais importantes são: \emph (texto *enfatizado*, em geral itálico), \texttt (texto `teletype`, imitando um terminal de texto ou uma impressora), \textit (*itálico*), \textbf (**negrito**), \textsf (fonte sem serifa), \textsc (texto Versalete — nem todas as fontes oferecem essa possibilidade), \normalsize (tamanho normal), \small (tamanho reduzido), \footnotesize (ainda menor), \scriptsize (ainda menor), \tiny (ainda menor), \large (tamanho aumentado), \Large (ainda maior), \LARGE (ainda maior), \Huge (ainda maior), \vspace{\baselineskip} (deixa uma linha em branco), \begin{center} (centraliza parágrafos), \begin{flushleft} (alinha parágrafos à esquerda), \begin{flushright} (alinha parágrafos à direita)[9], \- (sugere um possível local para hifenização localizada), \linebreak[0–4] (sugere um possível local para mudar de linha; o número indica quão forte é a sugestão, ou seja, 4 faz a mudança obrigatória. Por ser uma mudança de linha "normal", se o parágrafo é justificado, a linha é justificada normalmente), \newline ou \\ (força uma quebra de linha; por ser uma quebra forçada, a linha *não* é justificada nesse caso), \pagebreak[0–4] (sugere um possível local para mudar de página; como \linebreak, o número indica quão forte é a sugestão. Por ser uma mudança de página "normal", o texto da página é espalhado verticalmente de maneira a fazer a última linha alinhada com o final das demais páginas) e \newpage (força uma quebra de página; por ser uma quebra forçada, o final da página *não* é alinhado com o final das demais páginas nesse caso).

Mas, como discutido na Seção 4.3, não é recomendável usar esses comandos ao longo do texto: o ideal em LaTeX é expressar o significado de cada elemento, não a sua forma de apresentação, pois isso permite que você faça alterações na formatação com mais facilidade. Assim, quando os recursos pré-definidos do LaTeX (\itemize, \chapter etc.) não forem suficientes, o mais adequado é definir comandos novos, em geral usando os comandos de formatação mencionados acima. Esse é um tópico avançado, mas você pode consultar o início do arquivo LaTeX deste capítulo para alguns exemplos simples.

4.11 Versões do LaTeX

Assim como há packages para o LaTeX, o próprio LaTeX é, na verdade, um conjunto de extensões para o programa TeX. Assim, se você encontrar referências a "TeX" ou a "plain TeX", basta saber que esse é o sistema que funciona "por baixo" do LaTeX.

LaTeX é um sistema em evolução (desde os anos 80!). Uma das consequências disso é que há, na verdade, quatro versões diferentes dele:

[9] É altamente recomendável carregar a package ragged2e (já incluída neste modelo) e utilizar Center, FlushLeft e FlushRight ao invés de center, flushleft e flushright.

1. LaTeX "tradicional", que gera arquivos em formato DVI que, por sua vez, precisam ser convertidos para o formato PDF. Essa versão não é capaz de usar as fontes instaladas no sistema; ela só pode usar fontes adaptadas para uso com o LaTeX. Hoje em dia não há boas razões para usar essa versão.

2. pdfLaTeX, que gera arquivos PDF e dá suporte a alguns recursos avançados de tipografia adicionais. É a versão mais usada hoje em dia, embora também só possa usar as fontes adaptadas para uso com o LaTeX.

3. XeLaTeX que, além dos recursos do pdfLaTeX, opera internamente em UTF-8 (ou seja, funciona melhor com múltiplas línguas) e pode funcionar não só com as fontes adaptadas para o LaTeX como também com as fontes instaladas no sistema. XeLaTeX foi muito importante ao ser lançado, mas atualmente a comunidade está mais empenhada em evoluir o sistema com LuaLaTeX.

4. LuaLaTeX, que oferece os mesmos recursos que o XeLaTeX e também pode ser estendido internamente com mais facilidade (através da linguagem de programação Lua).

Todas essas versões são instaladas quando você instala LaTeX na sua máquina. Em geral, se você pretende escrever apenas com línguas no alfabeto latino e não pretende usar fontes diferentes das disponíveis por padrão, qualquer das três versões modernas (pdfLaTeX, XeLaTeX e LuaLaTeX) é adequada; pdfLaTeX é um pouco mais rápido, mas LuaLaTeX gera arquivos PDF um pouco menores. Se você pretende usar outros alfabetos, gostaria de escolher fontes diferentes ou precisa de recursos tipográficos específicos (texdoc fontspec, texdoc unicode-math), use LuaLaTeX.

4.12 Limitações do LaTeX

Como qualquer ferramenta, LaTeX tem limitações e características indesejáveis:

- A linguagem é muito prolixa: é bastante tedioso escrever coisas como "\begin{itemize}" etc. Linguagens como asciidoc (asciidoctor.org), markdown (commonmark.org), bookdown (bookdown.org) e reStructuredText (sphinx-doc.org) operam de maneira similar a LaTeX, mas sua sintaxe é bem mais enxuta. Elas funcionam muito bem para a geração de páginas web, mas LaTeX oferece mais recursos e geralmente produz resultados impressos melhores.

- LaTeX gera muitas mensagens pouco importantes durante o processamento do documento, o que dificulta a identificação de problemas (o programa auxiliar texlogsieve, incluído com versões recentes de LaTeX, pode minimizar esse incômodo). Além disso, quando ocorrem erros durante esse processamento, as mensagens explicativas muitas vezes são confusas ou, pior, não indicam o problema real que causou a falha.

- LaTeX procura ser uma linguagem *declarativa*, ou seja, os comandos buscam expressar o que se deseja e não como fazer algo ("este texto é um título" e não "pule duas linhas, selecione uma fonte maior, escreva este texto, pule mais duas linhas e selecione a fonte de tamanho padrão"). No entanto, ela é insuficiente em algumas situações, obrigando o usuário a utilizar vários comandos, às vezes obscuros, para obter resultados relativamente simples.

- Há diversas packages para personalizar os aspectos básicos da formatação final do documento, como o tipo de fonte, tamanho dos títulos das seções, espaçamento etc. No entanto, quando se quer fazer modificações maiores, é preciso lidar com partes complexas da linguagem e diversos comportamentos surpreendentes.
- Às vezes há incompatibilidades entre packages; em alguns casos, isso pode ser contornado mudando a ordem em que elas são carregadas, mas em outros pode simplesmente não ser possível combiná-las.
- A colocação automática dos *floats* e o algoritmo que encontra as quebras de página em geral funcionam bem, mas às vezes é possível obter resultados melhores manualmente (veja as Seções 5.5 e 5.4).
- As classes padrão (book, article etc.) não foram criadas para serem facilmente modificadas, o que deu origem a inúmeras packages voltadas para possibilitar a personalização de diversos aspectos da apresentação final do documento. Esse mecanismo não é ideal, por diversas razões. Por conta disso, existe um conjunto de versões alternativas dessas classes (scrbook no lugar de book, scrartcl no lugar de article etc.) chamado KOMA-Script, com mais recursos e mais possibilidades de customização. A classe memoir tem o mesmo objetivo, mas procura dar suporte a livros e artigos com uma única classe. Ambas abordagens são muito boas, mas a maioria dos modelos usados por revistas e outras publicações é baseada nas classes padrão. A versão 3 de LaTeX está em desenvolvimento com vistas a resolver boa parte dos problemas atuais do sistema, mas ainda deve demorar muitos anos para ficar pronta. ConTeXt é um "irmão mais novo" de LaTeX com diversas vantagens, mas com sintaxe diferente e que ainda não é tão popular.

Chapter 5

Exemplos e dicas de LaTeX

Neste capítulo, apresentamos exemplos comuns com alguma complexidade e, principalmente, pequenas dicas para evitar surpresas indesejáveis. Mesmo que você já conheça LaTeX, vale a pena analisar este material, incluindo o código-fonte do capítulo. Se você ainda não conhece nada sobre LaTeX, o Capítulo 4 e outros materiais citados na Seção 3.1 apresentam os conceitos básicos.

5.1 Bibliografia e referências

A documentação do pacote biblatex (Lehman *et al.*, 2018) é bastante extensa e explica (nas Seções 2.1.1 e 2.2.2) os diversos tipos de documento suportados, bem como o significado de cada campo. Na prática, às vezes é preciso fazer escolhas sobre o que incluir na descrição de um item bibliográfico e muitas vezes é mais fácil aprender copiando exemplos já existentes, como estes (consulte o arquivo `bibliografia.bib` para ver como foi criado o banco de dados e a bibliografia na página 39 para ver o resultado impresso):

- @Book: Knuth *et al.*, 1996.
- @Article (em periódico): Mittelbach, 2014.
- @InProceedings (ou @Conference): Alves *et al.*, 2003.
- @InCollection (capítulo de livro ou coletânea): Babaoglu and Marzullo, 1993.
- @PhdThesis: Garcia, 2001.
- @MastersThesis: Schmidt, 2003.
- @Techreport: Alvisi *et al.*, 1999.
- @Manual: Lehman *et al.*, 2018.
- @Misc: Allcock, 2003.
- @Online (para referência a artigo *online*): Fowler, 2004.
- @Online (para referência a página web): FSF, 2007.

A maioria das revistas científicas ainda utiliza bibtex e não biblatex, mas isso não faz muita diferença na prática: latexmk identifica automaticamente qual sistema usar durante a geração do documento e os modelos normalmente já incluem os comandos \bibliography, \printbibliography, \bibliographystyle etc. conforme o caso. O único detalhe importante

se refere a datas: biblatex prefere o uso do campo "date" para definir ano, mês etc. No entanto, se você quiser garantir compatibilidade tanto com biblatex quanto com bibtex, use os campos "year" e "month". Ambos reconhecem diversos formatos para o campo "month", mas apenas um funciona corretamente com os dois: o nome do mês em inglês, abreviado com três letras minúsculas e sem chaves, ou seja:

```
...
author = {Fulano de Tal},
year = {2011},
month = oct,
title = {Um título grandioso},
...
```

Para citar material *online*, há três casos:

- Para citar publicações *online* que se enquadram em formatos tradicionais (como um ebook ou a versão online de um artigo científico, independentemente de a revista existir ou não no formato impresso), use o tipo correspondente (@article, @book, @inproceedings etc.) e acrescente o campo url no arquivo .bib, aceito por todos os tipos de documento do bibtex/biblatex.
- Para citar materiais essencialmente *online* que possuem título e autor definidos (como uma postagem ou comentário em blog ou uma mensagem de email para uma lista de discussão), use o tipo @online de biblatex. Bibtex, por padrão, não tem um tipo específico para isso; com ele, normalmente usa-se o tipo "misc" e seu campo "howpublished" para especificar que se trata de um recurso *online*.
- Se o que você quer citar não é propriamente uma ideia (que normalmente possui um autor e faz parte de um texto com título etc.) mas sim um sítio (como uma empresa, produto ou repositório de software), pode ser mais adequado colocar a referência apenas como nota de rodapé e não na lista de referências. Outra opção é criar uma segunda lista de referências especificamente para recursos *online* desse tipo (biblatex permite criar múltiplas bibliografias).

5.2 Identificando problemas

Como mencionado na Seção 4.12, LaTeX gera um grande volume de mensagens informativas durante o processamento, o que torna mais difícil encontrar problemas. O arquivo de configuração latexmkrc deste modelo utiliza o programa texlogsieve para filtrar essas mensagens, apresentando apenas as mais importantes para o usuário; considere usá-lo em outros trabalhos com LaTeX também.

5.3 Modo matemático

O modo matemático do LaTeX tem sintaxe própria, mas ela não é complicada e há bastante documentação *online* a respeito. Por exemplo, "massa e energia são grandezas relacionadas pela Equação $E = mc^2$, definida inicialmente por Einstein", ou ainda "equações de segundo grau (Equação 5.1) são estudadas no ensino médio. As raízes de uma equação

de segundo grau podem ser encontradas por (5.2) — a fórmula de Bháskara. O valor do discriminante Δ (Equação 5.3) determina se a equação tem zero, uma ou duas raízes reais distintas".

$$ax^2 + bx + c = y \quad \forall x \in \mathbb{R} \tag{5.1}$$

$$y = 0 \Leftrightarrow x = \frac{-b \pm \sqrt{\Delta}}{2a} \Leftrightarrow x \text{ é raiz da equação} \tag{5.2}$$

$$\Delta \ (delta) = b^2 - 4ac \tag{5.3}$$

Para inserir um espaço explicitamente no modo matemático, use \quad ou \enspace. Para inserir texto "normal" em uma fórmula matemática, use \text{texto} (para texto de fato) ou \mathit{texto} (para nomes de variáveis ou funções com mais de uma letra). Pode ser necessário deixar um espaço no início do texto para evitar que ele fique colado com o caractere matemático que o antecede.

Para recursos mais sofisticados, incluindo frações com múltiplas linhas, matrizes, sistemas de equações alinhadas, setas, acentos etc., procure a documentação das packages amsmath e mathtools. Para teoremas, lemas, conjecturas etc., leia a documentação da package amsthm e decida de quais tipos de estrutura você vai precisar no seu documento. Aqui criamos três: "Pegadinha", "Teorema" e "Conjectura" (observe as numerações):

Pegadinha 1 :-) $1 = 0$

Proof. Tomemos dois números, a e b, tais que $a = b + 1$.

$$\begin{aligned}
a &= b + 1 \\
(a - b)a &= (a - b)(b + 1) \\
a^2 - ab &= ab + a - b^2 - b \\
a^2 - ab - a &= ab - b^2 - b \\
a(a - b - 1) &= b(a - b - 1) \\
\cancel{a(a - b - 1)} &= \cancel{b(a - b - 1)} \\
a &= b \\
b + 1 &= b \\
1 &= b - b \\
1 &= 0
\end{aligned}$$

□

Teorema 1. *É sempre possível colorir os vértices de um grafo sem que dois vértices adjacentes tenham a mesma cor usando no máximo quatro cores diferentes.*

Proof. A demonstração do Teorema 1 é um exercício a cargo do leitor.

Conjectura (?) 1: *Dado qualquer inteiro $n > 2$, não existem inteiros positivos a, b e c tais que $a^n + b^n = c^n$.*

Proof. Este espaço é muito pequeno para apresentá-la.

Teorema 2. P≠NP

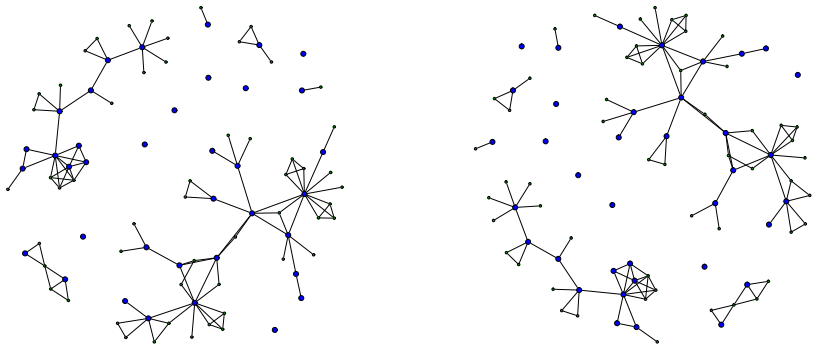
Proof. **P** tem apenas uma letra, enquanto **NP** tem duas letras. □

5.4 Quebras de página

O algoritmo que LaTeX usa para quebrar páginas funciona bem, minimizando linhas órfãs ou viúvas e garantindo uma distribuição homogênea do texto na página, mas não é excelente. Assim, se houver quebras de página ruins no seu texto final, pode ser útil modificá-las manualmente. Uma técnica usada por editores profissionais é mudar ligeiramente a altura do texto impresso em algumas páginas, melhorando a distribuição geral do texto. Para isso, ao invés de comandos como \pagebreak ou \newpage, o mais adequado é usar \enlargethispage{\baselineskip} (ou -1\baselineskip). Esse comando instrui LaTeX a fazer a página ligeiramente maior (ou menor), tornando possível acomodar mais uma linha de texto (ou uma linha a menos). Em documentos frente e verso, lembre-se de sempre garantir que a página adjacente também tenha seu tamanho modificado para que a alteração não seja tão perceptível. Um outro truque às vezes útil é aplicar o comando \looseness=1 (ou -1) a um parágrafo, que faz LaTeX tentar reorganizar as quebras de linha de maneira a fazer o parágrafo ter uma linha a mais (ou a menos), se isso for possível.

5.5 Figuras, gráficos e outros *floats*

Evidentemente, LaTeX permite inserir figuras no texto; além disso, ele também permite girá-las e criar subfiguras (com sublegendas), como no exemplo da Figura 5.1, que inclui as subfiguras 5.1a e 5.1b.



(a) *Uma figura simples.* **(b)** *O mesmo exemplo, girado.*

Figure 5.1: *Exemplo de subfiguras.*

Floats em geral incluem uma legenda e um *label*. Prefira sempre colocar o comando \label de uma figura ou tabela dentro do comando \caption; não fazê-lo muitas vezes funciona, mas às vezes causa problemas.

Você pode carregar arquivos de imagem de um subdiretório usando `dir/img.pdf`, mas é mais fácil modificar o comando \graphicspath próximo ao início de cada arquivo `.tex` de exemplo.

Para centralizar uma imagem mais larga que o texto da página, use a *package* adjustbox (incluída neste modelo):

```
\begin{figure}
\adjustbox{center}{\includegraphics[width=1.2\textwidth]{img.pdf}}
\caption{...}
\end{figure}
```

Uma "figura", na verdade, pode ser qualquer tipo de conteúdo ilustrativo (um exemplo interessante é o cronograma mostrado na Figura 5.2) mas, com a *package* float, também é possível definir ambientes específicos para cada tipo de conteúdo adicional (cada um com numeração independente), como é o caso do Programa 5.1. Há mais informações e dicas sobre recursos específicos para inclusão de código-fonte e pseudocódigo no Apêndice A[1].

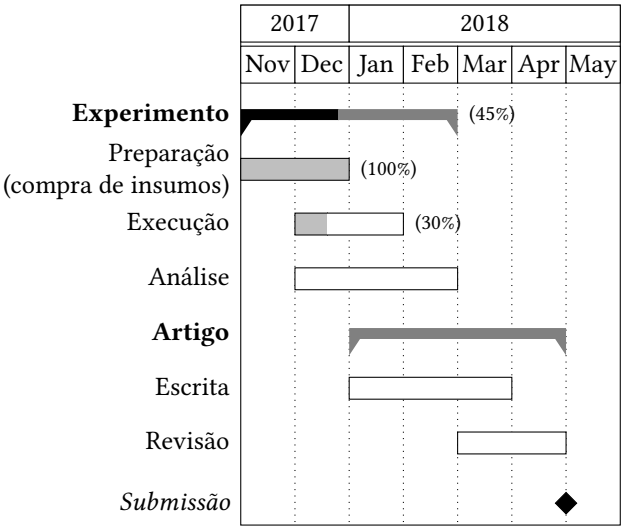


Figure 5.2: *Exemplo de cronograma.*

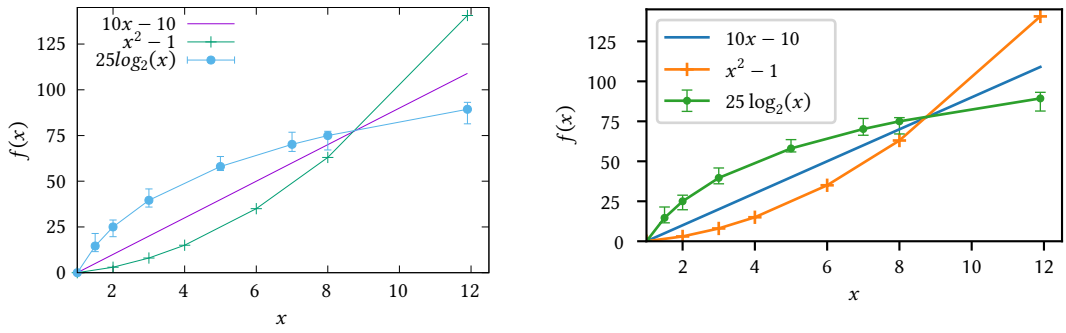
LaTeX também é capaz de gerar ilustrações e diagramas diretamente, mas usar esses recursos em geral não é trivial. Em particular, a package tikz oferece bons mecanismos para a criação de figuras (incluindo funções pré-prontas para formas geométricas, grafos, matrizes etc.) e é fácil usá-la para traçar linhas ou curvas simples.

[1] Observe que o nome do Apêndice ("A") foi impresso em uma linha separada, o que não é muito bom visualmente. Para evitar que isso aconteça (não só no final do parágrafo, mas em qualquer quebra de linha), utilize um espaço não-separável para fazer referências a figuras, tabelas, seções etc. ou antes de símbolos: "…no Apêndice~\ref{ap:pseudocode}", "O discriminante é denotado por~Δ".

Program 5.1 Exemplo de laço em Java.

```
1   for (i = 0; i < 20; i++)
2   {
3      // Comentário
4      System.out.println("Mensagem...");
5   }
```

Gráficos de dados ou funções matemáticas de excelente qualidade podem ser gerados com a *package* pgfplots (há um exemplo comentado neste arquivo; experimente des-comentar para ver o resultado). Também é possível importar gráficos gerados por matplotlib, gnuplot e R como qualquer outra imagem, mas nesse caso a fonte usada nesses gráficos provavelmente será diferente do corpo do texto. Felizmente, isso pode ser solucionado: Gnuplot (com o *driver* lua tikz[2]), matplotlib (com o *backend* PGF[3]) e R (com tikzDevice[4]) são capazes de exportar gráficos de dados na forma de comandos para tikz[5]: o resultado pode ser visto na Figura 5.3.



(a) *gnuplot*.

(b) *matplotlib*.

Figure 5.3: *Exemplos de gráficos gerados externamente*

Note que a colocação automática dos *floats* em geral funciona bem, mas às vezes pode ser melhorada. Isso acontece porque LaTeX decide o posicionamento de cada *float* individualmente, sem levar em conta os próximos *floats*, e nunca reavalia essa decisão. No exemplo da Seção 4.6, se a ordem "Figura 5, Tabela 3, Figura 6" for aceitável, esse vai ser o resultado, mesmo que a ordem "Tabela 3, Figura 5, Figura 6" seja melhor. Apenas se não for possível encontrar um lugar aceitável para a Figura 5 imediatamente (ou seja, na página atual) é que LaTeX processa os *floats* seguintes e, depois, procura novamente um lugar para ela. Por isso, depois que seu trabalho estiver finalizado, vale a pena avaliar se a colocação dos *floats* pode ser melhorada; se sim, mudar o lugar em que eles são definidos no documento (veja algumas dicas em Mittelbach, 2014) pode fazer LaTeX gerar um resultado melhor (mas lembre-se que isso só faz sentido depois que o documento estiver pronto, pois qualquer mudança no texto pode mudar totalmente a posição final dos *floats*).

[2] www.gnuplot.info/docs_5.2/Gnuplot_5.2.pdf#section*.516

[3] matplotlib.org/users/pgf.html

[4] cran.r-project.org/package=tikzDevice

[5] Você pode se interessar também pela *package* `gnuplottex`.

5.6 Tabelas

Talvez você precise organizar a apresentação da informação na forma de tabelas[6]; um exemplo simples é a Tabela 5.1. Para um resultado visual excelente, não deixe de ler a documentação da *package* booktabs.

Código	Abreviatura	Nome completo
A	Ala	Alanina
C	Cys	Cisteína
...	...	...
W	Trp	Triptofano
Y	Tyr	Tirosina

(a) *Com linhas de cores alternadas.*

Código	Abreviatura	Nome completo
A	Ala	Alanina
C	Cys	Cisteína
...	...	...
W	Trp	Triptofano
Y	Tyr	Tirosina

(b) *Com cabeçalhos girados.*

Table 5.1: *Exemplos de tabelas (códigos, abreviaturas e nomes dos aminoácidos).*

Normalmente, o fim de cada linha de uma tabela é indicado por \\. No entanto, se sua tabela causar erros misteriosos, experimente usar \tabularnewline ao invés de \\.

Se a tabela tem muitas linhas e, portanto, não cabe em uma única página, é possível fazê-la continuar ao longo de várias páginas com a *package* longtable, como é o caso da Tabela 5.2. Nesse caso, a tabela não é um *float* e, portanto, ela aparece de acordo com a sequência normal do texto. Se, além de muito longa, a tabela for também muito larga, você pode usar o comando landscape (da *package* pdflscape) em conjunto com longtable para imprimi-la em modo paisagem ao longo de várias páginas. A Tabela 5.2 tem essa configuração comentada; experimente des-comentar as linhas correspondentes[7]. Ela também demonstra o uso da *package* siunitx para alinhar as colunas numéricas pelo separador decimal.

Ângulo		Função					
graus	rads	sen	cos	tan	cotan	sec	cosec
0	0.0000	0.0000	1.0000	0.0000	-	1.0000	-
2	0.0349	0.0349	0.9994	0.0349	28.6363	1.0006	28.6537
4	0.0698	0.0698	0.9976	0.0699	14.3007	1.0024	14.3356
6	0.1047	0.1045	0.9945	0.1051	9.5144	1.0055	9.5668
8	0.1396	0.1392	0.9903	0.1405	7.1154	1.0098	7.1853

continua ⟶

Table 5.2: *Exemplo de tabela com valores numéricos.*

[6] Para defini-las com LaTeX, pode valer a pena usar o sítio www.tablesgenerator.com.

[7] Observe que, nesse caso, vai sempre haver uma quebra de página no texto para fazer a tabela começar em uma página em modo paisagem.

Ângulo		Função					
graus	rads	sen	cos	tan	cotan	sec	cosec
10	0.1745	0.1736	0.9848	0.1763	5.6713	1.0154	5.7588
12	0.2094	0.2079	0.9781	0.2126	4.7046	1.0223	4.8097
14	0.2443	0.2419	0.9703	0.2493	4.0108	1.0306	4.1336
16	0.2793	0.2756	0.9613	0.2867	3.4874	1.0403	3.6280
18	0.3142	0.3090	0.9511	0.3249	3.0777	1.0515	3.2361
20	0.3491	0.3420	0.9397	0.3640	2.7475	1.0642	2.9238
22	0.3840	0.3746	0.9272	0.4040	2.4751	1.0785	2.6695
24	0.4189	0.4067	0.9135	0.4452	2.2460	1.0946	2.4586
26	0.4538	0.4384	0.8988	0.4877	2.0503	1.1126	2.2812
28	0.4887	0.4695	0.8829	0.5317	1.8807	1.1326	2.1301
30	0.5236	0.5000	0.8660	0.5774	1.7321	1.1547	2.0000
32	0.5585	0.5299	0.8480	0.6249	1.6003	1.1792	1.8871
34	0.5934	0.5592	0.8290	0.6745	1.4826	1.2062	1.7883
36	0.6283	0.5878	0.8090	0.7265	1.3764	1.2361	1.7013
38	0.6632	0.6157	0.7880	0.7813	1.2799	1.2690	1.6243
40	0.6981	0.6428	0.7660	0.8391	1.1918	1.3054	1.5557
42	0.7330	0.6691	0.7431	0.9004	1.1106	1.3456	1.4945
44	0.7679	0.6947	0.7193	0.9657	1.0355	1.3902	1.4396
46	0.8029	0.7193	0.6947	1.0355	0.9657	1.4396	1.3902
48	0.8378	0.7431	0.6691	1.1106	0.9004	1.4945	1.3456
50	0.8727	0.7660	0.6428	1.1918	0.8391	1.5557	1.3054
52	0.9076	0.7880	0.6157	1.2799	0.7813	1.6243	1.2690
54	0.9425	0.8090	0.5878	1.3764	0.7265	1.7013	1.2361
56	0.9774	0.8290	0.5592	1.4826	0.6745	1.7883	1.2062
58	1.0123	0.8480	0.5299	1.6003	0.6249	1.8871	1.1792
60	1.0472	0.8660	0.5000	1.7321	0.5774	2.0000	1.1547
62	1.0821	0.8829	0.4695	1.8807	0.5317	2.1301	1.1326
64	1.1170	0.8988	0.4384	2.0503	0.4877	2.2812	1.1126
66	1.1519	0.9135	0.4067	2.2460	0.4452	2.4586	1.0946
68	1.1868	0.9272	0.3746	2.4751	0.4040	2.6695	1.0785
70	1.2217	0.9397	0.3420	2.7475	0.3640	2.9238	1.0642
72	1.2566	0.9511	0.3090	3.0777	0.3249	3.2361	1.0515
74	1.2915	0.9613	0.2756	3.4874	0.2867	3.6280	1.0403
76	1.3265	0.9703	0.2419	4.0108	0.2493	4.1336	1.0306
78	1.3614	0.9781	0.2079	4.7046	0.2126	4.8097	1.0223
80	1.3963	0.9848	0.1736	5.6713	0.1763	5.7588	1.0154
82	1.4312	0.9903	0.1392	7.1154	0.1405	7.1853	1.0098
84	1.4661	0.9945	0.1045	9.5144	0.1051	9.5668	1.0055
86	1.5010	0.9976	0.0698	14.3007	0.0699	14.3356	1.0024
88	1.5359	0.9994	0.0349	28.6363	0.0349	28.6537	1.0006
90	1.5708	1.0000	0.0000	-	0.0000	-	1.0000

Table 5.2: *Exemplo de tabela com valores numéricos.*

Tabelas mais complexas são um tanto trabalhosas em LaTeX; a Tabela 5.3 mostra como construir uma tabela em forma de ficha. Além de complexa, ela é larga e, portanto, deve ser impressa em modo paisagem. No entanto, usamos um outro mecanismo para girar a tabela: o comando sidewaystable (da *package* rotating). Com esse mecanismo, ela continua sendo um *float* (e, portanto, não força quebras de página no meio do texto), mas sempre é impressa em uma página separada.

Resumindo:

- Se uma tabela cabe em uma página, defina-a como um *float* (`\begin{table}`);
- Se cabe em uma página mas é muito larga e precisa ser impressa em modo paisagem, use sidewaystable (que também é um *float*);
- Se não cabe em uma página por ser muito longa, use longtable;
- Se não cabe em uma página por ser muito longa e precisa ser impressa em modo paisagem por ser muito larga, use longtable em conjunto com landscape. Nesse caso, vai haver uma quebra de página no texto para que a tabela inicie em uma nova página em modo paisagem.

5.7 Caracteres especiais

Um espaço não-separável é indicado pelo caractere til (“~”) e é possível forçar uma quebra de linha com “\\”. Aspas tipográficas (“ ” e ‘ ’) são inseridas com ` ` '' e ` '. Os principais símbolos matemáticos estão listados em texdoc undergradmath e você pode consultar a lista completa de símbolos disponíveis com texdoc symbols-a4 ou em www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf. Uma outra maneira de encontrar símbolos é usar este sítio: detexify.kirelabs.org/classify.html.

5.8 Línguas e hifenização

Detalhes sobre o suporte a diferentes línguas estão descritos na documentação da *package* babel. Para mudar temporariamente a língua do documento (por conta de uma citação, por exemplo), use \foreignlanguage{língua}{texto} (isso altera a hifenização das palavras e algumas convenções tipográficas, como espaços antes de pontuação em francês). Para trocar completamente de língua (o que inclui “captions”, ou seja, palavras geradas automaticamente como “Capítulo”, “Sumário” etc.), use \selectlanguage{língua}. Se você quiser apenas desativar temporariamente a hifenização, faça \begin{hyphenrules}{nohyphenation} (para uma única palavra, \mbox{palavra} é mais simples). LaTeX em geral é capaz de hifenizar palavras de maneira excelente, mas você pode “ensiná-lo” a hifenizar uma palavra de maneira diferente do padrão acrescentando ao preâmbulo \babelhyphenation{al-gu-mas pa-la-vras} (todas as línguas) ou \babelhyphenation[língua, língua…]{al-gu-mas pa-la-vras} (uma ou mais línguas específicas).

Experimento número:	1		**Data:**			jan 2017
Título:	Medições iniciais					
Tipo de experimento:	Levantamento quantitativo					
Locais	São Paulo	Rio de Janeiro	Porto Alegre	Recife	Manaus	Brasília	Rio Branco
Valores obtidos	0.2	0.3	0.2	0.7	0.5	0.1	0.4

Table 5.3: *Exemplo de tabela similar a uma ficha.*

Appendix A

Código-fonte e pseudocódigo

Com a *package* listings, programas podem ser inseridos diretamente no arquivo, como feito no caso do Programa 5.1, ou importados de um arquivo externo com o comando \lstinputlisting, como no caso do Programa A.1.

Program A.1 Máximo divisor comum (arquivo importado).

```
1  FUNCTION euclid(a, b)  ▷ The g.c.d. of a and b
2      r ← a mod b
3      while r ≠ 0  ▷ We have the answer if r is 0
4          a ← b
5          b ← r
6          r ← a mod b
7      end
8      return b  ▷ The g.c.d. is b
9  end
```

Trechos de código curtos (menores que uma página) podem ou não ser incluídos como *floats*; trechos longos necessariamente incluem quebras de página e, portanto, não podem ser *floats*. Com *floats*, a legenda e as linhas separadoras são colocadas pelo comando \begin{program}; sem eles, utilize o ambiente programruledcaption (atenção para a colocação do comando \label{}, dentro da legenda), como no Programa A.2[1]:

Program A.2 Máximo divisor comum (em português).

```
1  FUNCAO euclides(a, b)  ▷ O máximo divisor comum de a e b
2      r ← a mod b
3  enquanto r ≠ 0  ▷ Atingimos a resposta se r é zero
4          a ← b
5          b ← r
```

cont ⟶

[1] listings oferece alguns recursos próprios para a definição de *floats* e legendas, mas neste modelo não os utilizamos.

```
    ⟶ cont
6               r ← a mod b
7           fim
8       devolva b  ▷ O máximo divisor comum é b
9       fim
```

Além do suporte às várias linguagens incluídas em listings, este modelo traz uma extensão para permitir o uso de pseudocódigo, útil para a descrição de algoritmos em alto nível. Ela oferece diversos recursos:

- Comentários seguem o padrão de C++ (`//` e `/* ... */`), mas o delimitador é impresso como "▷".
- ":=", "<>", "<=", ">=" e "!=" são substituídos pelo símbolo matemático adequado.
- É possível acrescentar palavras-chave além de "if", "and" etc. com a opção "morekeywords={pchave1,pchave2}" (para um trecho de código específico) ou com o comando \lstset{morekeywords={pchave1,pchave2}} (como comando de configuração geral).
- É possível usar pequenos trechos de código, como nomes de variáveis, dentro de um parágrafo normal com \lstinline{blah}.
- "$...$" ativa o modo matemático em qualquer lugar.
- Outros comandos LaTeX funcionam apenas em comentários; fora, a linguagem simula alguns pré-definidos (\textit{}, \texttt{} etc.).
- O comando \label também funciona em comentários; a referência correspondente (\ref) indica o número da linha de código. Se quiser usá-lo numa linha sem comentários, use `///` \label{blah}; "`///`" funciona como `//`, permitindo a inserção de comandos LaTeX, mas não imprime o delimitador (▷).
- Para suspender a formatação automática, use \noparse{blah}.
- Para forçar a formatação de um texto como função, identificador, palavra-chave ou comentário, use \func{blah}, \id{blah}, \kw{blah} ou \comment{blah}.
- Palavras-chave dentro de comentários não são formatadas automaticamente; se necessário, use \func{}, \id{} etc. ou comandos LaTeX padrão.
- As palavras "Program", "Procedure" e "Function" têm formatação especial e fazem a palavra seguinte ser formatada como função. Funções em outros lugares *não* são detectadas automaticamente; use \func{}, a opção "functions={func1,func2}" ou o comando "\lstset{functions={func1,func2}}" para que elas sejam detectadas.
- Além de funções, palavras-chave, strings, comentários e identificadores, há "specialidentifiers". Você pode usá-los com \specialid{blah}, com a opção "specialidentifiers={id1,id2}" ou com o comando "\lstset{specialidentifiers={id1,id2}}".

Annex A

Table of Analysis Passes

Table A.1: *LLVM Analysis Passes*

Name	Description
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778

Continued on next page

Table A.1 – continued from previous page

Name	Description
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778

Annex B

Table of Transformation Passes

Table B.1: *LLVM Transformation Passes*

Name	Description
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778

Continued on next page

Table B.1 – continued from previous page

Name	Description
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778
abcdef ghjijklmn	123.456778

References

[Allcock 2003] William Allcock. *GridFTP Protocol Specification. Global Grid Forum Recommendation (GFD.20)*. 2003 (cit. on p. 23).

[Alves *et al.* 2003] Carlos E. R. Alves, Edson N. Cáceres, Frank Dehne, and Siang W. Song. "A parallel wavefront algorithm for efficient biological sequence comparison". In: *ICCSA'03: The 2003 International Conference on Computational Science and its Applications*. Springer-Verlag, May 2003, pp. 249–258 (cit. on p. 23).

[Alvisi *et al.* 1999] Lorenzo Alvisi, Elmootazbellah Elnozahy, Sriram S. Rao, Syed A. Husain, and Asanka Del Mel. *An Analysis of Comunication-Induced Checkpointing*. Tech. rep. TR-99-01. Austin, USA: Department of Computer Science, University of Texas at Austin, 1999 (cit. on p. 23).

[Babaoglu and Marzullo 1993] Ozalp Babaoglu and Keith Marzullo. "Consistent global states of distributed systems: fundamental concepts and mechanisms". In: *Distributed Systems*. Ed. by Sape Mullender. 2nd ed. 1993, pp. 55–96 (cit. on p. 23).

[Blanco-Cuaresma and Bolmont 2016] Sergi Blanco-Cuaresma and Emeline Bolmont. "What can the programming language Rust do for astrophysics?" *Proceedings of the International Astronomical Union* 12.S325 (2016), pp. 341–344 (cit. on p. 1).

[Cooper *et al.* 2005] Keith D Cooper *et al.* "Acme: adaptive compilation made efficient". *ACM SIGPLAN Notices* 40.7 (2005), pp. 69–77 (cit. on p. 2).

[Daly 2010] Patrick W. Daly. *Reference sheet for `natbib` usage*. Sept. 13, 2010. url: mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf (visited on 12/20/2018) (cit. on p. 18).

[Fowler 2004] Martin Fowler. *Is Design Dead?* May 2004. url: martinfowler.com/articles/designDead.html (visited on 01/30/2010) (cit. on p. 23).

[FSF 2007] Free Software Foundation. *GNU General Public License*. 2007. url: www.gnu.org/copyleft/gpl.html (visited on 01/30/2010) (cit. on p. 23).

[Garcia 2001] Islene C. Garcia. “Visões Progressivas de Computações Distribuídas”. PhD thesis. Campinas, Brasil: Instituto de Computação, Universidade de Campinas, Dec. 2001 (cit. on p. 23).

[Knuth *et al.* 1996] Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts. *Mathematical Writing*. The Mathematical Association of America, Sept. 1996 (cit. on p. 23).

[Lattner and Adve 2004] Chris Lattner and Vikram Adve. “Llvm: a compilation framework for lifelong program analysis & transformation”. In: *International symposium on code generation and optimization, 2004. CGO 2004.* IEEE. 2004, pp. 75–86 (cit. on p. 1).

[Lehman *et al.* 2018] Philipp Lehman, Philip Kime, Moritz Wemheuer, Audrey Boruvka, and Joseph Wright. *The `biblatex` Package*. Oct. 30, 2018. url: mirrors.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf (visited on 12/20/2018) (cit. on pp. 19, 23).

[Levy *et al.* 2017] Amit Levy *et al.* “The case for writing a kernel in Rust”. In: *Proceedings of the 8th Asia-Pacific Workshop on Systems*. 2017, pp. 1–7 (cit. on p. 1).

[Matsakis and Klock 2014] Nicholas D Matsakis and Felix S Klock. “The Rust language”. *ACM SIGAda Ada Letters* 34.3 (2014), pp. 103–104 (cit. on p. 1).

[Mittelbach 2014] Frank Mittelbach. “How to influence the position of float environments like figure and table in LaTeX?” *TUGboat. Communications of the TeX Users Group* 35.3 (2014). url: tug.org/TUGboat/tb35-3/tb111mitt-float.pdf (visited on 01/09/2020) (cit. on pp. 23, 28).

[Schmidt 2003] Rodrigo M. Schmidt. “Coleta de Lixo para Protocolos de *Checkpointing*”. MA thesis. Campinas, Brasil: Instituto de Computação, Universidade de Campinas, Oct. 2003 (cit. on p. 23).

[The Rust Project Developers 2018] The Rust Project Developers. *Rust Compiler Development Guide*. Jan. 2018. url: https://github.com/rust-lang/rustc-dev-guide (cit. on p. 1).

Index