Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

UnB-CIC: Uma classe em LaTeX para textos do Departamento de Ciência da Computação

Guilherme N. Ramos

Dissertação apresentada como requisito parcial para
qualificação do Mestrado em Informática

Orientador
Prof. Dr. Guilherme Novaes Ramos

Brasília
2014

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

UnB-CIC: Uma classe em LaTeX para textos do Departamento de Ciência da Computação

Guilherme N. Ramos

Dissertação apresentada como requisito parcial para
qualificação do Mestrado em Informática

Prof. Dr. Guilherme Novaes Ramos (Orientador)
CIC/UnB

Prof. Dr. Donald Knuth
Stanford University

Dr. Leslie Lamport
Microsoft Research

Prof.a Dr.a Ada Lovelace
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 24 de dezembro de 2014

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`). Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The* ***`UnB-CIC`*** *class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: LaTeX, scientific method, thesis

Contents

Chapter 1

Introduction

Chapter 2

Satisfiability Solvers

The problem of determining whether a boolean formula is satisfiable has the historical honor of being the first problem ever shown to be NP-Complete [1]. Great theoretical and practical efforts are directed in improving the efficiency of solvers for this problem, known as *Boolean Satisfiability Solvers*, or just *SAT solvers*. Despite the worst-case exponential run time of all the algorithms known, satisfiability solvers are increasingly leaving their mark as a general purpose tool in the most diverse areas [2]. In essence, SAT solvers provide a generic combinatorial reasoning and search tool.

The underlying represenational formalism is propositional logic. A propositional or Boolean formula is a logic expression defined over variables (or atomic expressions), that take value in any binary set whose values oppose each other, usually $\{\mathbf{true}, \mathbf{false}\}$ or just $\{0, 1\}$, and boolean connectives, any unary or binary function with one output, such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\Rightarrow$ (implication), $\Leftrightarrow$ (equivalence). Parentheses can be used to avoid ambiguous. A *truth assignment* to a set V of Boolean variables is a map $\sigma : V \to \{\mathbf{true}, \mathbf{false}\}$. A *satisfying assignment* for a formula F is a truth assignment σ such that F evaluates to **true** [2].

In the context of SAT solvers, we are interested in propositional formulae in *Conjuctive Normal Form* (CNF): F is in CNF if it is a conjunction of *clauses*, where each clause is a disjunction of *literals*, and each literal is either a variable or its negation. For example, $F = (p \vee \neg q) \wedge (\neg p \vee r \vee s) \wedge (q \vee r)$ is a CNF formula with four variables and three clauses.

Therefore, the Boolean Satisfiability Problem (SAT) can be expressed as: Given a CNF formula F, does F have a satisfying assignment? One can be interested not only in this decision problem, but also in finding an actual satisfying assignment when it exists. All practical SAT solvers do produce such an assignment if there exists one.

Algorithm 1: Euclid's algorithm for finding the greatest common divisor of two nonnegative integers

```
Input  : Two nonnegative integers $a$ and $b$
Output: gcd($a, b$)
function Euclid ($a, b$)
if $b = 0$ then
    return $a$;
else
    return Euclid($b, a \bmod b$);
end
```

2.1 The DPLL Procedure

2.2 Conflict Driven Clause Learning

Chapter 3

Modal Logics

Bibliography

[1] Cook, S. A.: *The complexity of theorem proving procedures.* In *stoc71*, pages 151–158, 1971. 2

[2] Gomes, Carla P, Henry Kautz, Ashish Sabharwal, and Bart Selman: *Satisfiability solvers.* Foundations of Artificial Intelligence, 3:89–134, 2008. 2