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

Modal Logics

This chapter introduces K_n, a modal logic language, determined semantically by an account of necessity and possibility.

On classical logic, formulae are either evaluated to true or false, in any model. Propositional logic and predicate logic, for instance, do not allow for any further possibilities. From many points of view, however, this lacks expressivity. In natural language we often distinguish between various modalities of truth, such as *necessarily* true, *known to be* true, *believed to be* true or yet true *in some future*, for example.

Modal logic adds connectives to express one, or more, of these different modes of truth.

2.1 Syntax

The language of K_n is equivalent to its set of *well-formed formulae*, denoted WFF_{K_n}, which is constructed from a denumerable set of *propositional symbols* $\mathcal{P} = \{p, q, r, \ldots\}$, the negation symbol $\neg$, the disjunction symbol $\vee$ and the modal connectives $\boxed{a}$, that express the notion of necessity, for each index a in a finite, fixed set $\mathcal{A} = \{1, \ldots, n\}, n \in \mathbb{N}$.

Definição 1 The set of well-formed formulae, WFF_{K_n}, is the least set such that:

1. $p \in WFF_{\mathsf{K}_n}$, for all $p \in \mathcal{P}$
2. if $\varphi, \psi \in WFF_{\mathsf{K}_n}$, then so are $\neg\varphi, (\varphi \vee \psi)$ and $\boxed{a}\varphi$, for each $a \in \mathcal{A}$

When $n = 1$, we often omit the index in the modal operators, i.e., we just write $\Box\varphi$ and $\Diamond\varphi$, for a formula φ. Other logic operators may be introduced as abbreviations, as usual: $\varphi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$ (conjuction), $\varphi \Rightarrow \psi \stackrel{\text{def}}{=} \neg\varphi \vee \psi$ (implication), $\varphi \Leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$ (equivalence), $\diamondsuit\varphi \stackrel{\text{def}}{=} \neg\boxed{a}\neg\varphi$ (possibility), $\mathbf{false} \stackrel{\text{def}}{=} \varphi \wedge \neg\varphi$ (*falsum*), $\mathbf{true} \stackrel{\text{def}}{=} \neg\mathbf{false}$ (*verum*). Parentheses may be omitted if the reading is not ambiguous.

A *literal* is a propositional symbol $p \in \mathcal{P}$ or its negation $\neg p$. We denote by $\mathcal{L}$ the set of all literals. A *modal literal* is a formula of the form $\boxed{a} l$ o $\diamondsuit \neg l$, with $l \in \mathcal{L}$ and $a \in \mathcal{A}$.

The maximal number of nesting modal operators in a formula is defined as its *modal depth* and denoted *mdepth*. The maximal number of modal operators in which scope the formula occurs is defined as the *modal level* of that formula, and it is denoted *ml*. For instance, in $\boxed{a} \diamondsuit p$, $mdepth(p) = 0$ and $ml(p) = 2$.

2.2 Semantics

The semantics of K_n is presented in terms of Kripke structures.

Definição 2 A Kripke model for $\mathcal{P}$ and $\mathcal{A}$ is given by the tuple $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$, where W is a non-empty set of possible worlds with a distinguinshed world w_0, the root of $\mathcal{M}$; each R_a, $a \in \mathcal{A}$, is a binary relation on W, and $\pi : W \times \mathcal{P} \longrightarrow \{false, true\}$ is the valuation function that associates to each world $w \in W$ a truth-assignment to propositional symbols.

Satisfiability and validity of a formula in a given world is defined as follows.

Definição 3 Let $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ be a Kripke model, $w \in W$ a world, and $\varphi, \psi \in WFF_{\mathsf{K}_n}$. The *satisfiability relation*, denoted by $\langle \mathcal{M}, w \rangle \models \varphi$, between a world w and a formula φ, is inductively defined by:

1. $\langle \mathcal{M}, w \rangle \models p$ if, and only if, $\pi(w, p) = \mathbf{true}$, for all $p \in \mathcal{P}$;
2. $\langle \mathcal{M}, w \rangle \models \neg\varphi$ if, and only if, $\langle \mathcal{M}, w \rangle \not\models \varphi$;
3. $\langle \mathcal{M}, w \rangle \models \varphi \vee \psi$ if, and only if, $\langle \mathcal{M}, w \rangle \models \varphi$ or $\langle \mathcal{M}, w \rangle \models \psi$
4. $\langle \mathcal{M}, w \rangle \models \boxed{a}\varphi$ if, and only if, for all $t \in W$, $(w, t) \in R_a$ implies $\langle \mathcal{M}, t \rangle \models \varphi$

Satisfiability is defined in terms of the root of a model. A formula $\varphi \in WFF_{\mathsf{K}_n}$ is said to be *satisfiable* if there exists a Kripke model $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ such that $\langle \mathcal{M}, w_0 \rangle \models \varphi$. A formula is said to be *valid* if it is satisfiable in all models.

The local satisfiability problem in K_n corresponds to determining the existence of a model at which a formula is satisfied. The local satisfiability problem for K_n is PSPACE-complete [1].

Chapter 3

Resolution

Nonclausal methods, in general, require a large number of resolution rules, making implementation difficult.

3.1 Clausal Resolution

Clausal resolution was suggested as a proof method for classical logic by Robinson in 1965 [2], and was claimed to be suitable to be performed by computer, as it has only one rule of inference that may be applied many times.

Chapter 4

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 [3]. Great theoretical and practical efforts have been 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 [4]. In essence, SAT solvers provide a generic combinatorial reasoning and search platform.

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** [4].

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: DPLL-recursive(F, ρ)

Input : A CNF formula F and an initially empty partial assignment ρ
Output: UNSAT, or an assignment satisfying F

$(F, \rho) \leftarrow$ UnitPropagate(F, ρ)
if *F contains the empty clause* **then**
  return UNSAT
end
if *F has no clauses left* **then**
  Output ρ
  return *SAT*
end
$l \leftarrow$ a literal not assigned by ρ
if *DPLL-recursive*$(F|_l, \rho \cup \{l\}) = SAT$ **then**
  return *SAT*
end
return *DPLL-recursive*$(F|_{\neg l}, \rho \cup \{\neg l\})$

4.1 The DPLL Procedure

4.2 Conflict Driven Clause Learning

Algorithm 2: CDCL(F, ν)

Input :
Output:

if *UnitPropagate*$(F, \nu) ==$ *CONFLICT* **then**
  return *UNSAT*
end
$dl \leftarrow 0$
while *not* *AllVariablesAssigned*(F, ν) **do**
  $(x, \nu) \leftarrow$ PickBranchingVariable(F, ν)
end
return *SAT*

4.3 MiniSat

Bibliography

[1] Spaan, E.: *Complexity of Modal Logics*. PhD thesis, University of Amsterdam, 1993. 3

[2] Robinson, J. A.: *A machine-oriented logic based on the resolution principle*. Journal of the ACM, 12(1):23–41, January 1965. 4

[3] Cook, S. A.: *The complexity of theorem proving procedures*. In *stoc71*, pages 151–158, 1971. 5

[4] Gomes, Carla P, Henry Kautz, Ashish Sabharwal, and Bart Selman: *Satisfiability solvers*. Foundations of Artificial Intelligence, 3:89–134, 2008. 5