Um Provador de Teoremas Multi-Estratégia

A Multi-Strategy Theorem Prover

Adolfo Gustavo Serra Seca Neto
DAINF - UTFPR

http://www.dainf.ct.utfpr.edu.br/~adolfo

Data desta versão: 14 de novembro de 2008
Date of this version: November 14th, 2008
Sobre


About

This is a chapter of my Ph.D. thesis entitled “A Multi-Strategy Theorem Prover”. This Computer Science thesis was defended on January 30th, 2007 at the Institute of Mathematics and Statistics (IME) of the University of São Paulo (USP). My advisor was Prof. Dr. Marcelo Finger. Thesis full text is available at http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943/. Only the first chapter was written in Portuguese. All the following appendices were written in English.
Apêndice A

Introduction

Automated deduction has been an active area of research since the 1950s [69]. Early developments within the automated deduction field have had a profound effect on the Artificial Intelligence (AI) domain, and, indeed, all of Computer Science [70]. Automated Theorem Proving (ATP) deals with the development of computer programs that show that some statement (the conjecture) is a logical consequence of a set of statements (the axioms and hypotheses). ATP systems are used in a wide variety of domains [110], such as mathematics, AI, software generation, software verification, security protocol verification, and hardware verification.

Most automated theorem provers nowadays are based either on the resolution principle [100] or on the Davis-Logemann-Loveland (DLL) procedure[32], but other methods can also be used. Tableau methods are particularly interesting for theorem proving since they exist in many varieties and for several logics [52]. Besides that, they do not require conversion of input problems to clausal form. Tableaux can be used for developing proof procedures in classical logic as well as several kinds of non-classical logics, such as Fuzzy Logic [68], Residuated Logic [71], Modal and description logics [46], Substructural logics [30], Many Valued Logics [13], and Logics of Formal Inconsistency [18].

The inference rules of automated deduction systems in general and tableau provers in particular are typically non-deterministic in nature. They say what can be done, not what must be done [52]. Thus, in order to obtain a mechanical procedure, inference rules

\footnote{Which is a restricted form of resolution also known as Davis-Putnam procedure.}
need to be complemented by another component, usually called strategy or search plan, which is responsible for the control of the inference rules [6]. That is, the inference rules of a proof method (or of an automated deduction system based on this proof method) define a nondeterministic algorithm for finding a proof; a strategy is a (deterministic) algorithm for finding proofs in this method. For each proof method, many strategies can be defined. The size of proofs as well as the time spent by the proof procedure can vary greatly as different strategies are used.

Nondeterministic algorithms are used in several areas in computer science such as automated theorem proving [94], term-rewriting systems [114], protocol specification [59], formal specification [81], optimization [10], pattern recognition [61], and decision making [84]. An algorithm is a sequence of computational steps that takes a value (or set of values) as input and produces a value (or set of values) as output [26]. A nondeterministic algorithm is an algorithm with one or more choice points where multiple different continuations are possible, without any specification of which one will be taken. A particular execution of such an algorithm picks a choice whenever such a point is reached. Thus, different execution paths of the algorithm arise when it is applied to the same input, and these paths, when they terminate, generally produce different output [115].

Nondeterministic algorithms compute the same class of functions as deterministic algorithms, but the complexity may be lower. Every nondeterministic algorithm can be turned into a deterministic algorithm, possibly with exponential slow down. That is, a deterministic algorithm that traces all possible execution paths of a polynomial time nondeterministic algorithm may have exponential time complexity. One of the most important open research problems in computer science nowadays is the “P=NP?” question [20, 21]. Informally speaking, the answer to this question corresponds to knowing if decision problems that can be solved by a polynomial-time nondeterministic algorithm can also be solved by polynomial-time deterministic algorithm.

The satisfiability problem (SAT) for classical propositional logic was the first known NP-complete problem. A decision problem is NP-complete if (1) it is in NP, and (2) any problem in NP can be reduced in polynomial time to it. SAT can be described as “given
a propositional formula, decide whether or not it is satisfiable”. Many other decision problems, such as graph coloring problems, planning problems, and scheduling problems can be encoded into SAT.

One of many logical methods that can be used to solve the satisfiability problem is the KE system. It is a tableau method originally developed for classical logic by Marco Mondadori and Marcello D’Agostino [31], but that has been extended for other logical systems. The KE system was presented as an improvement, in the computational sense, over traditional Analytic Tableaux [106]. It is a refutation system, that though close to the analytic tableau method, is not affected by the anomalies of cut-free systems [29].

We have designed and implemented KEMS, a multi-strategy theorem prover based on the KE method for propositional logics. A multi-strategy theorem prover is a theorem prover where we can vary the strategy without modifying the core of the implementation. A multi-strategy theorem prover can be used for three purposes: educational, exploratory and adaptive. For educational purposes, it can be used to illustrate how the choice of a strategy can affect performance of the prover. As an exploratory tool, a multi-strategy theorem prover can be used to test new strategies and compare them with others. And we can also think of an adaptive multi-strategy theorem prover that changes the strategy used according to features of the problem presented to it.

KEMS current version implements strategies for three logics: classical propositional logic and two paraconsistent logics: mbC and mCi. Paraconsistent logics are tools for reasoning under conditions which do not presuppose consistency [18]. We have chosen to implement KEMS for mbC and mCi because these two logics are the simplest logics from the family of Logics of Formal Inconsistency [18], a family of paraconsistent logics that internalize the notions of consistency and inconsistency at the object-language level. This family of logics has some nice proof-theoretic features and have been used in some computer science applications such as the integration of inconsistent information in multiple databases [34].
A.1 Overview

Appendix B discusses the logical systems implemented in KEMS, their tableau methods, and their complexity. It also provides the necessary background and the basic notions which will be used in the rest of the document. Appendix C presents KEMS design and implementation, discussing in more detail the implemented strategies. In Appendix D we exhibit the problems used to evaluate KEMS as well as the results obtained in the evaluation. Concluding remarks, a review of this thesis contributions and some suggestions for future works are presented in Appendix E. Next, a brief KEMS user manual is shown in Appendix F.
Referências Bibliográficas


[103] N. Scharli, S. Ducasse, O. Nierstrasz, and A.P. Black. Traits: Composable units of


[107] Sérgio Soares and Paulo Borba. AspectJ - Programação orientada a aspectos em
Java. *Tutorial no SBLP 2002, 6o. Simpósio Brasileiro de Linguagens de Pro-


[109] Friedrich Steimann. The paradoxical success of aspect-oriented programming. *SIG-


