# *Automated Theorem Proving*

**Claudio Busatto, Henrique Weber, Jefferson Stoffel, João Gross**

# *Prova Automática de Teoremas*

- Trata do desenvolvimento de programas capazes de demonstrar que uma determinada proposição é conseqüência lógica de outras proposições.

- Campo mais desenvolvido da área de raciocínio matemático.

- Utilizados de maneira interativa por um especialista no domínio da aplicação que guia o processo de inferência, ou determina lemas intermediários que devem ser provados no decorrer da demonstração.

# Histórico

- Início em 1957 com a publicação do artigo "The Logic Theory Machine", por Newell, Shaw e Simon.

- Foi capaz de provar 38 teoremas do Principia Mathematica, e introduziu diversos conceitos básicos da ATP e técnicas de IA.

- Em 1960 o matemático Hao Wang desenvolveu programas capazes de provar todos os teoremas de lógica proposicional do Principia Mathematica em poucos minutos. Por esse trabalho ele recebeu a primeira Milestone Award for Automated Theorem Proving, em 1983.

# *Histórico*

- Paul Gilmore desenvolveu provavelmente o primeiro procedimento de prova mecanizado para o cálculo de predicados, e era capaz de provar teoremas de dificuldade moderada.

- Nos anos 60 surgiu um procedimento chamado “eliminação de modelos”. Levou ao desenvolvimento de um formato de resolução linear, chamado SL-resolution.

- SL-Resolution (Selective Linear Resolution) é regra de inferência básica para linguagens de programação lógicas, tais como Prolog.

# *Histórico*

- Em 1976, Kenneth Appel e Wolfgang Haken provaram o teorema das quatro cores. Esse foi o primeiro teorema importante provado por um sistema de ATP, ainda que boa parte do problema tenha sido resolvida manualmente.

- Em 1997, uma prova mais simples foi publicada por Robertson, Sanders, Seymour e Thomas, utilizando outro sistema de ATP.

- Em 2005, o teorema foi novamente provado por Gonthier usando um sistema de ATP de propósito geral.

# *Histórico*

- A partir dos anos 80, os provadores automáticos de teoremas começaram a ser usados comercialmente.

- Hoje, todas as grandes empresas de hardware utilizam provadores automáticos de teoremas para verificação e desenho de seus circuitos integrados.

# *Técnicas Usadas*

**Lean Theorem Prover**

- Implementado com o objetivo de alcançar a máxima eficiência utilizando a mínima quantidade de código e recursos.

- Geralmente são implementados em Prolog e podem ser muito pequenos, com algumas centenas de bytes de código fonte.

- Um dos principais exemplos do Lean Theorem Prover é o ileanCoP.

# *Técnicas Usadas*

**Method of Analytic Tableaux**

- Ligada à lógica clássica de primeira ordem.

- Seu objetivo é “quebrar” fórmulas complexas em outras de menor tamanho, para que possa ser analisada de uma forma mais simples.

- Método de prova por refutação (prova-se um teorema pelo insucesso na tentativa de construção sistemática de um modelo para sua negação).

# *Técnicas Usadas*

**Indução Matemática**

- Método de prova fundamental na matemática.

- Define-se o método da seguinte forma:

(i) P(1) é verdadeira; e

(ii) qualquer que seja n ε IN, sempre que P(n) é verdadeira, segue que P(n + 1) é verdadeira. Então, P(n) é verdadeira para todo n ε IN.

# *Técnicas Usadas*

**Método Geométrico de Wu**

- Introduzido em 1977 por Wu Went-Sün.

- O método de Wu apenas pode provar certos tipos de teoremas geométricos envolvendo incidência, congruência e paralelismo.

- Sendo mais específico, o método de Wu pode ser usado para resolução de um problema cuja hipótese e tese podem ser convertidas em equações polinomiais com coeficientes racionais.

# *Técnicas Usadas*

**Teorema de Prova Interativo**

- Campo da lógica que se preocupa com o desenvolvimento de ferramentas para obter provas formais com a interação máquina-homem.

- Usuário pode remover caminhos de pesquisa infrutíferos, reduzindo a quantidade de processamento para realizar uma prova.

# *Técnicas Usadas*

**Model Checking**

- Verifica-se automaticamente a validade de propriedades acerca do comportamento de sistemas reativos.

- Enumeração exaustiva de todos os estados alcançáveis.

- Muito empregada na verificação formal de hardwares e softwares.

# *Principais Implementações*

**LCF (Logic for Computable Functions)**

- Teorema de prova automático interativo desenvolvido nas universidades de Edinburgh e Stanford por Robin Milner em 1970.

- Introduziu o uso da linguagem de programação ML, uma linguagem funcional imperativa de propósito genérico. (Linguagem imperativa: define “como“ computar; Linguagem puramente lógica: define “o que” computar, e.g. Prolog)

- O sistema automático de prova LCF também permite que um prova seja construída para trás (backwards), através de uma conjectura a ser provada. Uma tática do LCF é uma função que reduz a meta a zero ou mais submetas.

# *Principais Implementações*

**Isabelle**

- Última Versão: Isabelle2009-1, de 2009.

- É um teorema de prova baseado em LCF (Logic for Computable Functions), ou seja, é baseado em pequenos núcles lógicos que garantem resultados lógicos corretos.

- O método principal de prova do ATP Isabelle é a versão de resolução de lógica de alta ordem, baseada em unificação de lógica de alta ordem.

# *Principais Implementações*

**HOL4 (Higher Order Logic 4)**

- Última Versão: kananaskis-5, de 2009.

- Evolução do HOL88.

- Variáveis podem ser funções ou predicados (por isso, "alta ordem").

- Não há categoria de separação sintática (variáveis booleanas satisfazem essa regra).

- Escrito em Standard ML.

# Principais Implementações

**Gandalf**

- Versão comercial chamada G, desenvolvida e distribuída pela Safelogic A.

- Contém um grande número de diferentes estratégias de busca e é capaz de automaticamente selecionar estratégias adequadas.

- No modo normal, o Gandalf tenta apenas encontrar insatisfatibilidade. Para encontrar satisfatibilidade tem que ser chamado com um "-sat flag.

- Implementado em Scheme e compilado para o C usando o compilador Hobbit.

# *Principais Implementações*

**Otter**

- Última versão: 3.3, de 2004.

- Primeiro ATP de alta performance para lógica deprimeira ordem amplamente distribuído, sendo assim pioneiro em várias importantes técnicas de implementação.

- Foi muito estável durante toda sua época, porém não é mais hoje desenvolvido. Hoje encontra-se em domínio público.

# Principais Implementações

**Snark**

- Última versão: snark-20080805r018b, de 2008.

- É um ATP desenvolvido em Common LISP. É utilizado como núcleo dedutivo do Amphion (sistema da NASA), para, por exemplo, realizar cálculos de astronomia planetária.

- Suas principais regras de inferência são resolução e paramodulação.

- Esse estilo de prova de teoremas é similar ao Otter.

# *Aplicações Práticas*

- Verificação de Software

- Engenharia de Software

- Verificação de Hardware

- Empresas: Intel, AMD, outras

# Conclusão

- Importância do uso e constante desenvolvimento de sistemas automáticos de prova para a indústria e área acadêmica.

- Proporcionou o desenvolvimento da computação.

- Impulsionou a criação de máquinas capazes de por em prática a teoria desenvolvida em torno da prova de teoremas devido ao alto poder de processamento exigido.

- O estudo e uso de provadores automáticos de teoremas apresenta-se, portanto, como uma forma bastante eficiente de garantir a confiabilidade de software e hadrware.