UNIVERSIDADE D
COIMBRA

Pedro Alexandre Pereira Teixeira

O MÉTODO DAS BASES DE DADOS DEDUTIVAS PARA A GEOMETRIA

DEDUÇÃO AUTOMÁTICA NA GEOMETRIA

Dissertação no âmbito do Mestrado em Matemática, Ramo Matemática Aplicada e Computação orientada pelo Professor Doutor Pedro Quaresma e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia.

Julho de 2022

O Método das Bases de Dados Dedutivas para a Geometria

Pedro Alexandre Pereira Teixeira

UNIVERSIDADE D
COIMBRA

Master in Mathematics
Mestrado em Matemática

June 2022 / Junho 2022

Agradecimentos

Agradeço ao meu orientador Professor Pedro Quaresma pela proposta deste tema e pelo consistente apoio e suporte ao longo do ano letivo.

Agradeço também aos meus pais e irmão pelo apoio ao longo de todo o meu percurso académico assim como aos meus familiares e amigos.

Resumo

Os provadores automáticos de geometria são programas que permitem desenvolver demonstrações formais para conjeturas geométricas de forma automática sem intervenção humana. Têm como objetivo principal a decisão, em tempo útil e relativamente a um conjunto de condições iniciais, do valor lógico de uma conclusão estabelecida. Originalmente concebidos durante a década de 50, permitem o desenvolvimento de demonstrações formais, em alguns casos produzindo uma versão legível por um matemático.

Este relatório tem como objetivo providenciar uma introdução ao conceito de provadores automáticos de geometria, uma introdução básica dos principais métodos e com um interesse principal no método das bases de dados dedutivas.

Introduz a história dos provadores automáticos, as diferentes áreas que auxiliaram o desenvolvimento e alguns dos diferentes sistemas utilizados atualmente. A secção conclui com uma breve introdução aos diferentes métodos disponíveis e uma primeira visão do método das bases de dados dedutivas.

Finalmente, é introduzido um programa, criado com a esperança de resolver problemas geométricos, utilizando uma lista de regras definida com base em um artigo publicado anteriormente. Devido à complexidade do projeto e ao tempo limitado para o executar, o programa desenvolvido, embora completo, ainda necessita de algum trabalho para efeitos de eficiência, tanto em termos de tempo, como taxa de sucesso. São documentadas as dificuldades ultrapassadas durante o seu desenvolvimento, os limites presentes no sistema e conjunto de regras utilizadas como base para o programa, bem como os problemas identificados durante o desenvolvimento do programa.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O trabalho de Dissertação de Mestrado proposto é a implementação, utilizando *C++* e *SQLite*, do *Método das bases de dados dedutivas*. Este método é um método automatizado de demonstração de teoremas para geometria, implementado primeiramente por Chou et al. [4]. É um método sintético, capaz de produzir demonstrações e também capaz da descoberta automatizada de teoremas. Espera-se que um sistema sólido e eficiente provador de teoremas automatizado de geometria seja produzido como resultado deste trabalho.

Começamos com uma secção detalhando a história geral do uso de computadores e programas para demonstrar conjeturas geométricas (ver Secção 1.1). Em seguida, na secção 2, subsecção 2.1 e subsecção 2.3 serão apresentadas as duas principais alternativas na busca da demonstração automática de teoremas, os métodos algébricos e os métodos geométricos (sintéticos).

Uma vez estabelecidos os dois métodos, nas subsecções 2.3.1–2.2.1 serão exploradas as diferentes tentativas de combinar as duas abordagens para estabelecer provadores mais poderosos, principalmente o método da área e o método das bases de dados.

Na secção 3, são exploradas as diferentes plataformas de computador com algum grau de automação da demonstração, principalmente programas de geometria dinâmica e provadores automatizados de teoremas de geometria.

Na secção 4 será apresentado o planeamento por trás de uma nova implementação do método dedutivo - O método da base de dados.

Finalmente, na secção 5, será apresentada uma visão geral do método desenvolvido, vantagens, desvantagens e problemas encontrados ao longo do seu desenvolvimento.

1.1 Dedução Automática em Geometria

Quando se trata de demonstrar teoremas geométricos usando computadores, há duas abordagens principais: sintética e algébrica (esta secção é principalmente baseada em [1]).

A abordagem sintética usa os axiomas e regras de inferência de uma dada teoria axiomática para geometria, por exemplo a Teoria Axiomática de Euclides, para desenvolver as demonstrações das conjeturas aos axiomas, aplicando regras de inferência. Dado que este tipo de demonstração leva a

uma explosão de casos, métodos de inteligência artificial são frequentemente usados para ajudar a conter o tamanho da demonstração.

A abordagem algébrica é caracterizada por uma tradução da conjetura geométrica para uma forma algébrica, ou seja, um conjunto de equações, seguido pela aplicação de manipulações algébricas puras, sem qualquer ligação com o raciocínio geométrico.

Uma maneira de atingir o objetivo de usar computadores para ajudar a demonstrar teorias é pelo uso de métodos sintéticos, em muitos casos auxiliados por heurística. Existem várias vantagens em usar métodos sintéticos como o método usado para demonstrar teoremas geométricos.

- As demonstrações são mais legíveis e mais curtas do que com métodos algébricos.
- Permite a existência e estudo de pontos fixos.
- Certos teoremas podem ser resolvidos e provados de forma muito mais elegante por este método, onde uma demonstração algébrica exigiria uma maior quantidade de recursos computacionais.

Os métodos que se baseiam em métodos sintéticos tentam utilizar os meios de inferência para demonstrações geométricas que vinham sendo utilizados desde tempos imemoriais, mas com a ajuda de poder computacional.

O uso de computadores para ajudar a demonstrar teoremas geométricos começou na década de 50 do século passado, com Gelernter e seus colaboradores. Isto foi considerado um marco na área de IA para a época, tendo como base a abordagem de simulação humana. Foi usado para demonstrar teoremas em sistemas de axiomas de Tarski para geometria elementar. Apesar de sucessos e melhorias nesses métodos, não levou ao desenvolvimento de um poderoso provador de teoremas de geometria.

No que diz respeito à abordagem da teoria da demonstração, os primeiros trabalhos datam de Hilbert. Hilbert apresentou um método de decisão em geometria afim para uma classe de conjeturas de geometria construtiva.

Em 1951, Tarski publicou um método de decisão para a teoria de corpos fechados, dando assim um método de decisão para o que ele chamou geometria elementar. Apesar das melhorias de Seindenberg e outros nos anos seguintes, a implementação de variantes do método de Tarski permaneceram impraticáveis para demonstrar teoremas que não sejam triviais em geometria. Em 1974, Collins deu uma importante contribuição em linha com Tarski, de tal forma que o seu algoritmo de decomposição algébrica cilíndrica é atualmente o melhor algoritmo geral do tipo Tarski. Este método foi implementado por Arnon e vários problemas difíceis de álgebra-geometria foram resolvidos pelo seu programa. Outra grande melhoria prática do método de Collins foi feito por Collins e Hong em 1991.

Os métodos algébricos transformam o problema geométrico em um conjunto de equações algébricas, permitindo um sistema de regras muito mais poderoso e condições para ajudar a demonstrar os teoremas em estudo. No entanto, raramente uma prova legível é produzida, apenas uma resposta binária (sim/não) é obtida.

Um avanço na prova automatizada de teoremas de geometria (GATP) foi feito por Wu. Restringindo-se a uma classe de declarações de geometria de tipo de igualdade, Wu introduziu um método em 1977 que pode ser usado para demonstrar teoremas de geometria bastante difíceis de forma eficiente. O trabalho de Wu tornou-se conhecido fora da China principalmente através de revistas de investigação, e o facto de mais de 130 teoremas terem sido provados pelo método foi bastante animador. Ko, Hussain,

Wang, Hu, Gao, Kapur e Wan também conseguiram implementar provadores de teoremas baseados em várias versões modificadas do método de Wu. Mais tarde, foi esclarecido que as ferramentas algébricas necessárias na abordagem podem ser desenvolvidas a partir do trabalho de Ritt em 1950. Devido a tal facto, esta abordagem é agora conhecida como o método do conjunto característico de Wu-Ritt (CS). É agora o caso que centenas de teoremas em Geometrias euclidianas e não euclidianas podem ser demonstradas automaticamente por programas de computador, implementações do método de Wu.

O sucesso do método de Wu reavivou o interesse em demonstrações geométricas por computadores. Em particular, foi desenvolvida investigação na aplicação do método das bases de Gröbner (GB) para a mesma classe de teoremas de geometria antes demonstrados por implementações do método de Wu. Em 1985-1986, três grupos relataram sucessos práticos.

Todos os métodos acima mencionados têm as mesmas características, isto é, primeiro há a transformação de propriedades geométricas em equações algébricas no plano Cartesiano, para, de seguida, a demonstração ser somente de índole algébrica.

A busca por um método baseado em cálculo vetorial para GATP começou em meados dos anos oitenta, porque acreditava-se que tal método produziria provas mais elegantes. No início dos anos noventa, vários métodos deste tipo foram propostos: o método da área e o método de ângulo completo foram os mais bem sucedidos.

Estudos comparativos mostram que as demonstrações produzidas por esses métodos são geralmente mais curtas do que as resultantes da aplicação dos métodos algébricos. O método de área usa invariantes geométricas, como área, razão de segmentos e diferença pitagórica. A principal vantagem deste método é que cada passo da demonstração gerada tem um significado geométrico. Implementações deste método foram capazes de produzir demonstrações de mais de 500 teoremas de geometria, algumas das quais são mais curtas do que as fornecidas por especialistas em geometria.

Os métodos de raciocínio automatizado em geometria têm uma ampla gama de aplicações, incluindo análise cinética de robótica, design de ligação, visão computacional, CAD inteligente, CAI inteligente, modelagem sólida, etc.

Capítulo 2

Métodos de Demonstração Automática em Geometria

2.1 Métodos Algébricos

Os métodos algébricos têm como passo inicial a tradução da conjetura geométrica num sistema de equações algébricas que o descrevem, e como passos subsequentes o desenvolvimento da demonstração pela aplicação de manipulações algébricas. O método do conjunto característico, também conhecido como método de Wu, o método da eliminação, o método das bases de Gröbner e a abordagem da álgebra de Clifford são exemplos de métodos práticos baseados na aproximação algébrica [12]. Dado que este tipo de métodos não é o foco desta dissertação de mestrado, não se vão aqui desenvolver.

2.2 Métodos Sintéticos

Demonstrações de geometria sintética, são demonstrações baseadas em uma teoria axiomática, desenvolvidas usando axiomas e regras de inferência, não havendo lugar à utilização de coordenadas Cartesianas e cálculos algébricos.

Ao adaptar abordagens de raciocínio «tradicionais» desenvolvidas no campo de inteligência artificial (nos anos 60 do século XX), métodos, como as abordagens de Gelernter, Nevis, Elcock, Greeno et al., Coelho e Pereira, Zhang et al. foram dedicados a automatizar processos de demonstração tradicionais. Fazendo uso de sistemas axiomáticos próximos aos usados nas escolas secundárias, esses sistemas tentaram fornecer demonstrações legíveis (por alunos e professores). Infelizmente, a explosão combinatória resultante da aplicação sucessiva das regras de inferência, levou ao uso de heurísticas adequadas, restringindo o âmbito dos GATP e impedindo o desenvolvimento de um GATP eficiente de uso geral [12].

2.2.1 Método das Bases de Dados Dedutivas

Durante 1994-1995, um programa de bases de dados dedutivas genéricas, em *Prolog*[1] foi usado para experimentar as regras de ângulo completo, tendo-se verificado que as bases de dados dedutivas genéricas tinham dois grandes problemas para o efeito:

- sequências infinitas, e.g.,$coll(A,B,C)$:- $coll(B,A,C)$:- $coll(A,B,C)$:- . . . ;
- ineficiente.

Os autores, então, implementaram uma base de dados estruturada e um conjunto de regras R consistindo em 75 regras para ângulos completos, algumas das quais são incorporadas na base de dados estruturada. Dada uma conjetura geométrica, do tipo de igualdade, cada hipótese é considerada como um facto (uma propriedade). Todas as hipóteses da afirmação formam a base de dados inicial D_0 de fatos. Em seguida, aplicamos o conjunto de regras R a D_0 para obter novos factos. A base de dados D_0 é expandida adicionando os novos factos a D_0, passando-se a ter a base de dados ampliada D_1.

O processo é repetido para D_1, originando D_2. Como o número de pontos é finito, o número de segmentos, de ângulos, de triângulos e de círculos também é finito. Então, o processo será finito. O D_k é chamado de ponto fixo da configuração geométrica. Como o conjunto de regras R e a estrutura das bases de dados são embutidos no código-fonte, o provador é muito eficiente e o ponto fixo geralmente pode ser alcançado dentro de uma fração de segundo, mesmo para um computador pessoal comum.

Se a conclusão (um facto também) está no ponto fixo, então demonstrámos que a conjetura é verdadeira e a demonstração pode ser gerada. A visualização da aplicação de regras para o método das bases de dados dedutivas é semelhante à aplicação das regras para o método do ângulo completo, exceto que eles têm conjuntos de regras diferentes. Assim como para o método do ângulo completo, o método das bases de dados dedutivas também é capaz de adicionar, automaticamente, segmentos, ângulos e círculos auxiliares caso o processo de demonstração o exiga. No conjunto de regras do método, 18 são referentes ao adicionar pontos auxiliares. A aplicação destas regras é feita de modo especial, para evitar uma explosão de novos pontos, ou seja, não se irá adicionar um novo ponto auxiliar se este ponto depender de um já adicionado anteriormente.

O métodos das bases de dados dedutivas processa as regras de inferência por encadeamento direto. No entanto, a compreensão de uma demonstração, feita por este método, não é fácil. Após a demonstração feita, a visualização da mesma é feita por encadeamento inverso, da conclusão para as hipóteses. Cada passo da demonstração pode ser «clicado» para ver os efeitos visuais correspondentes. Os efeitos para o método das bases de dados dedutivas são semelhantes aos gerados pelo método do ângulo completo. Geralmente, o método das bases de dados dedutivas requer mais etapas do que o método de ângulo completo. Existem três razões para este fenómeno:

- O método do ângulo completo usa a técnica de eliminação de pontos e linhas semelhantes ao método da área, sendo a demonstração mais direta;
- o método do ângulo completo usa regras combinadas e cada uma dessas regras consiste em várias regras do método das bases de dados dedutivas;

[1] https://pt.wikipedia.org/wiki/Prolog

- o método do ângulo-completo usa factos encontrados pelo encadeamento direto e se as demonstrações desses factos levarem a novos factos o processo entra em ciclo até que não haja novos factos.

No entanto, o método das base de dados dedutivas tem as seguintes vantagens:

- As demonstrações geradas são semelhantes às demonstrações tradicionais e, portanto, mais fáceis de compreender.
- Em comparação com o método do ângulo completo, o ponto fixo no método das bases de dados dedutivas é organizado estruturalmente e há mais regras no conjunto de regras R. Então, o encadeamento direto para o método das bases de dados dedutivas é muito mais poderoso e pode ser usado para descobrir novos factos.

2.3 Métodos Semi-Sintécticos

Os métodos semi-sintéticos (métodos livres de coordenadas) tentam combinar as aproximações sintéticas e as algébricas, conseguindo a legibilidade das demonstrações características dos provadores sintéticos e a eficiência dos provadores algébricos.

A fim de combinar a legibilidade dos métodos sintéticos e a eficiência dos métodos algébricos, algumas abordagens, como o método da área [2, 7] e o método do ângulo completo [3], representam o conhecimento geométrico na forma de expressões em relação a invariantes geométricas.

Para formular e provar conjeturas, esses métodos usam um conjunto de *quantidades geométricas* específicas que permitem o tratamento de relações e definir um sistema de axiomas.

Nas próximas secções, o *Método de Área* (§ 2.3.1), o *Método de ângulo completo* (§ 2.3.2) e o *Método de Bases de Dados Dedutivas* (§ 2.2.1) são descritos.

2.3.1 Método da Área

O método da área foi proposto em 1992 [2, 7]. Zhang encontrou muitos métodos ad-hoc elegantes baseados em áreas de triângulos para resolver problemas geométricos quando era professor do ensino médio e treinador da equipa olímpica de matemática chinesa. Esses métodos ad-hoc foram usados para desenvolver um método completo de GATP, cuja implementação é surpreendentemente poderosa na medida em que foi possível demonstrar centenas de teoremas de geometria do tipo construtivo, sendo as demonstrações, em geral, curtas e elegantes. Um programa de computador chamado *Geometry Expert*, baseado neste método, foi capaz de produzir demonstrações de 500 teoremas não triviais, de forma inteiramente automática. Este método parece ser o primeiro a produzir demonstrações legíveis para teoremas de geometria difíceis, de forma eficiente.

Em vez de coordenadas, três grandezas geométricas básicas: a razão de segmentos de retas paralelas orientados, a área orientada e a diferença pitagórica são usadas[2]. As proposições básicas, que descrevem formalmente as propriedades dessas quantidades, são a base dedutiva do método da área.

[2] A orientação dos segmentos e das áreas tem somente a ver com questões de eficiência, não interfere na generalidade do método.

As construções geométricas (conjeturas) são feitas de forma que, de um conjunto de pontos livres (do ponto de vista lógico, pontos quantificados universalmente), outros pontos são construídos de tal forma que as propriedades sobre as quais a conjetura se vai basear serão verdadeiras e a conclusão é expressa em termos de uma igualdade sobre esses pontos. A demonstração envolve a eliminação dos pontos construídos, ficando-se com uma igualdade trivial nos pontos livres.

As grandezas geométricas são:

- *proporção de segmentos orientados paralelos*,(veja Fig. 2.1) denotado, $\overline{AB}/\overline{CD}$. Se os pontos A, B, C e D são colineares, $\overline{AB}/\overline{CD}$ é a razão entre os comprimentos de segmentos orientados AB e CD. Se os pontos A, B, C e D não são colineares e contém $AB\|CD$, existe um paralelogramo $ABPQ$ tal que P, Q, C e D são colineares e então $\overline{AB}/\overline{CD} = \overline{PQ}/\overline{CD}$.
- *área orientada* para um triângulo ABC, denotado, $\mathscr{S}_{ABC}$, é a área do triângulo ABC, multiplicada por -1, se ABC tiver a orientação negativa.
- *Diferença Pitagórica*,[3] denotado $\mathscr{P}_{ABC}$, para os pontos A, B, C, definidos como $\mathscr{P}_{ABC} = \overline{AB}^2 + \overline{CB}^2 - \overline{AC}^2$.

O método da área foi estendido para demonstrar teoremas em geometria sólida (o método do volume), geometria Minkowskiana, Bolyai-Lobachevsky geometria e geometria Riemanniana [4].

Em 1996, uma nova grandeza geométrica, o ângulo completo, foi introduzida para demonstrar teoremas de geometria, com ângulos. Este método, embora não tão poderoso como o método de área, pode produzir demonstrações muito elegantes para alguns teoremas de geometria difíceis, para os quais o método da área não é capaz de providenciar uma demonstração curta.

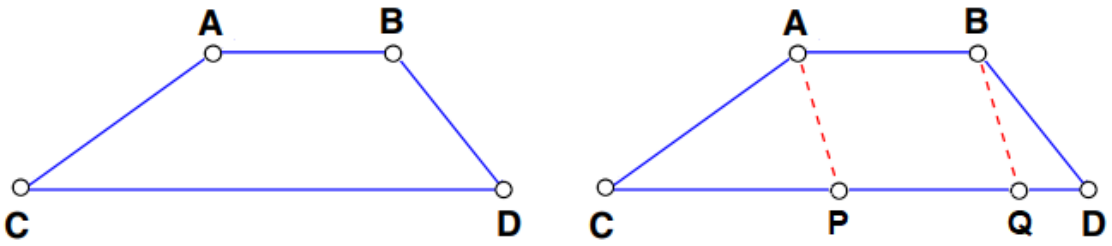


Fig. 2.1 Razão de segmentos direcionados paralelos

2.3.2 Método do Ângulo Completo

Um ângulo completo, também chamado de ângulo redondo ou perigon, é um ângulo igual a 2π radianos, correspondente ao ângulo central de um círculo.

Um ângulo completo é definido como um par ordenado de duas linhas u e v. Para denotação, consideramos $\angle[u,v]$.

Algumas propriedades são:

[3] A *Diferença Pitagórica* é uma generalização da igualdade de Pitágoras em relação aos três lados de um triângulo retângulo, para uma expressão aplicável a qualquer triângulo. Para um triângulo retângulo ABC tem-se que $\mathscr{P}_{ABC} = 0$.

- $\angle[u,v]+\angle[v,u]=\angle[0]$ significado $\angle[u,v]=-\angle[v,u]$
- $\angle[u,u]$ é equivalente a $\angle[0]$ que é um ângulo completo constante. Se u e v são duas linhas paralelas, então, $\angle[u,v]=\angle[0]$.
- Se u e v são perpendiculares, então, $\angle[u,v]=\angle[1]$. A soma de dois desses valores é igual ao ângulo completo constante, o que significa que $\angle[u,v]+\angle[u,v]=\angle[1]+\angle[1]=\angle[0]$.
- Para os ângulos completos $\angle[u,v]$ e $\angle[u',v']$ podemos denotar 3 linhas como l_1, l_2 e l_3 tal que $\angle[u,v]=\angle[l_1,l_2]$ e $\angle[u',v']=\angle[l_2,l_3]$. Podemos, então, definir a soma dos dois ângulos como $\angle[u,v]+\angle[u',v']=\angle[l_1,l_3]$.

O método de ângulo completo usa uma combinação de encadeamento direto e encadeamento inverso. O processo começa com as hipóteses do teorema, como o conjunto inicial de propriedades, e aplica regras de inferência para obter um dado conjunto de factos.

Se a conclusão estiver presente nesta lista de factos, então, o teorema está demonstrado. Caso contrário, adotamos o mesmo raciocínio aplicando um encadeamento inverso. O processo usa a técnica de eliminação de pontos e/ou linhas dos ângulos completos, semelhante ao mencionado anteriormente no método da área.

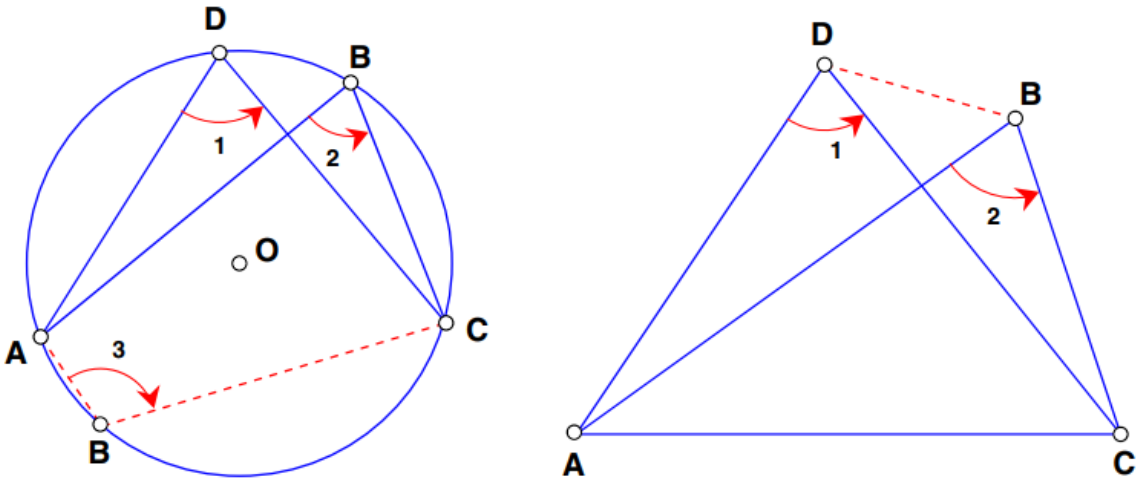


Fig. 2.2 Teorema do ângulo inscrito

Exemplo: O método de ângulo completo funciona particularmente bem para teoremas envolvendo círculos e ângulos. Sejam A, B, C e D quatro pontos não colineares. $\angle[ABC]$ denota o ângulo completo $\angle[AB,BC]$ e $\angle ABC$ denotam o ângulo tradicional. Dentro da geometria ordenada, $\angle[ABC]$ = $\angle[DEF]$ se e somente se $\angle ABC=\angle DEF$ e os dois ângulos têm a mesma orientação, ou $\angle ABC=180^\circ-\angle DEF$ e os dois ângulos têm orientações opostas.

Uma regra de inferência é:

- (Teorema do Ângulo Inscrito) Pontos A, B, C e D são cíclicos se $\angle[ABC]=\angle[ADC]$ (o lado esquerdo da Fig. 2.2)

Aplicando isso duas vezes, podemos ter uma regra combinada, digamos, ∠[BAC] =∠[BDC] iff∠[ABC] =∠[ADC] (o lado direito de Fig. 2.2). Aqui, os quatro pontos podem estar em qualquer ordem. Essas propriedades independentes do diagrama de ângulos completos tornam as demonstrações não só muito mais simples e independentes de diagramas, mas também rigorosas.

Capítulo 3

Sistemas Computacionais

3.1 Sistemas de Geometria Dinâmica

Sistemas de geometria dinâmica (Dynamic Geometry Systems, DGS), também designados de Ambientes de Geometria Dinâmica, (Dynamic Geometry Environement, DGE) são programas que permitem criar e manipular construções geométricas, principalmente em geometria plana. A manipulação dos elementos livres das construções, podem levar a conjeturar propriedades sobre as construções geométricas. Sem a natureza rigorosa de uma demonstração, estas manipulações podem ser indicadores poderosos de como uma hipótese se comporta sob muitas circunstâncias diferentes e até mesmo permitir a descoberta rápida de um contra-exemplo que essencialmente possa levar a uma demonstração por contradição. Começa-se a construção colocando alguns pontos e usando-os para definir novos objetos como linhas, círculos ou outros pontos. Depois da construção estar completa, os pontos iniciais podem-se mover. As restrições sobre todos os outros pontos, levam a que a construção retanha as suas propriedades fundamentais e, como tal, à possível descoberta de novas relações entre os objetos constituintes da construção geométrica.

Os programas de geometria dinâmica podem não conter a capacidade de provar teoremas. No entanto, o seu desenvolvimento forneceu muitas ferramentas e avanços na área de demonstração automática de teoremas geométricos, levando à criação de ambientes mais complexos, alguns já com demonstradores de teoremas incorporados.

3.2 Demonstradores Automáticos de Teoremas Geométricos

Demonstradores automáticos de teoremas geométricos (GATP)[1] são usados para analisar e concluir se um teorema geométrico é verdadeiro ou falso. Ao lado de uma conclusão, o objetivo de tais programas é elaborar uma demonstração do teorema, seja um contra-exemplo, em caso de uma conjetura falsa, ou uma série de passos lógicos para justificar uma afirmação positiva.

[1] Do Inglês Geometry Automated Theorem Provers

3.3 Sistemas Combinados

Nesta secção, serão apresentados alguns programas com recursos de dedução automatizada. Eles variam de DGS com alguns recursos GATP a programas genéricos de ATP. Esta não é uma cobertura completa da área. As escolhas feitas foram ditadas pela conveniência e/ou pela proximidade com o objetivo deste trabalho.

GeoGebra Programa de geometria dinâmica contendo provadores. [5, 9, 10]. *GeoGebra* é gratuito[2] e no *Linux* existe um «pacote» `geogebra5`, pronto para instalar (see Fig. 3.1).

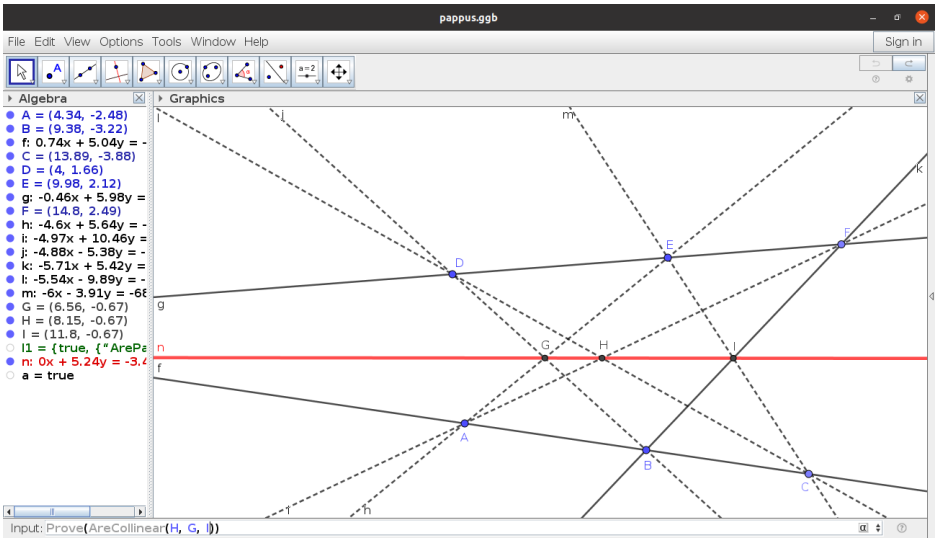


Fig. 3.1 Geogebra 5

Este programa estabelece uma conexão com provadores automáticos pelo uso dos comandos `Prove` e `ProveDetails` [10].

JGEx Programa de Geometria Dinâmica e Demonstração Automática de Teoremas [14]. Foi recentemente adicionado em código aberto ao repositório GitHub[3] (Arquivo contendo o programa e exemplos). Requer um *ambiente de desenvolvimento integrado em Java* (JD_k) (veja Fig. 3.2, lado esquerdo).

Dado o triângulo ΔABC, defina o ponto D como a interseção entre a linha perpendicular a AB que contém C. Assim sendo, conclua que as retas DE e FG são perpendiculares.

Podemos usar JGEx como DGS, movendo os pontos livres A, B e C. Com isso, podemos ter uma visão geral de como a conjetura sobre o elemento perpendicular das retas DE e FG vale para vários casos. Podemos então usar GATP, estabelecendo a conjetura, e então usando um dos provadores para criar uma verificação formal (veja Fig. 3.2, lado direito, para o método das bases de dados dedutivas) [4, 14].

[2] https://www.geogebra.org/
[3] github.com/yezheng1981/Java-Geometry-Expert

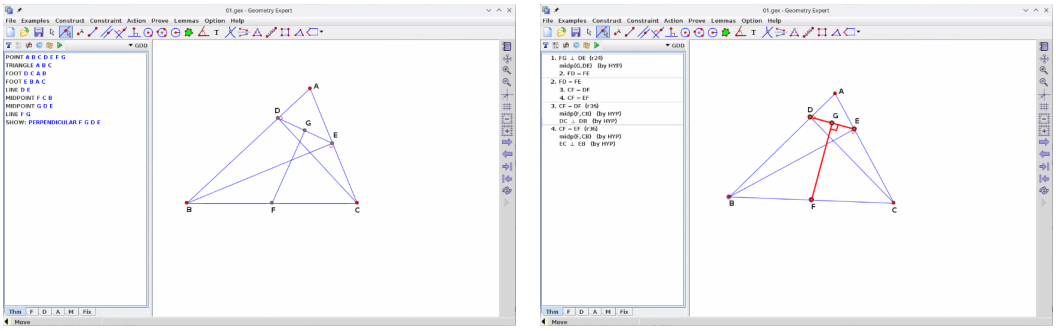


Fig. 3.2 JGEx

GCLC Provador de teoremas automático com visualização de construção [6]. Atualizado e disponível no repositório GitHub.[4]

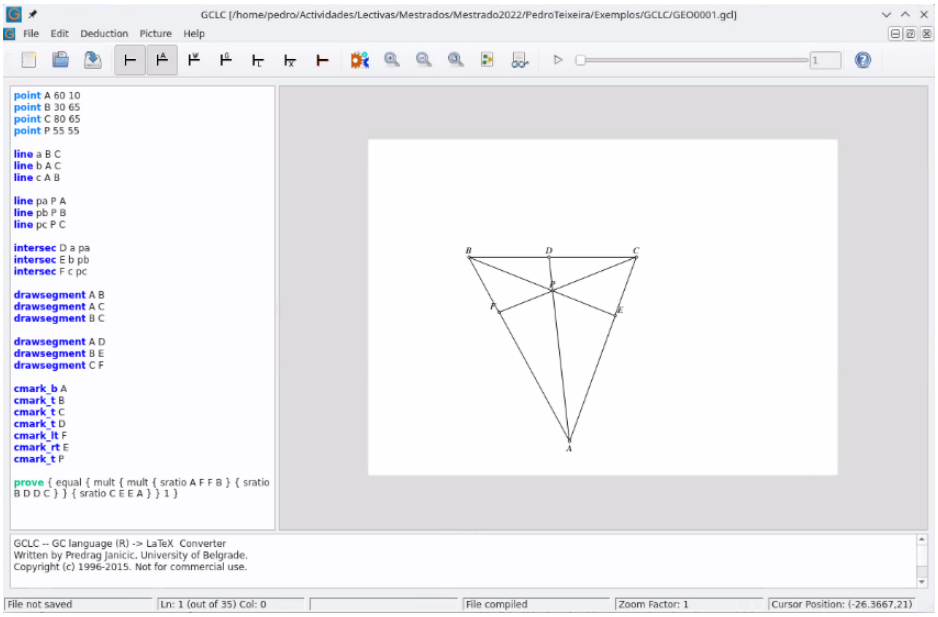


Fig. 3.3 Teorema de Ceva em GCLC

Vejamos um exemplo de aplicação do sistema GCLC, para a demonstração do Teorema de Ceva [7].

Teorema (Teorema de Ceva)**.** *Seja $\triangle ABC$ um triângulo e P um ponto arbitrário no plano. Seja D a interseção de AP e BC, E a interseção de BP e AC e F a interseção de CP e AB. Então:*

$$\frac{\overline{AF}}{\overline{FB}}\frac{\overline{BD}}{\overline{DC}}\frac{\overline{CE}}{\overline{EA}} = 1$$

Usando o método da área, GCLC é capaz de demonstrar este teorema geométrico. Apresenta-se, de seguida, um fragmento da demonstração:

[4] http://poincare.matf.bg.ac.rs/~janicic/gclc/

Proof. Pode ser demonstrado que $\frac{\overline{AF}}{\overline{FB}} = \frac{\mathscr{S}_{APC}}{\mathscr{S}_{BCP}}$. Por analogia $\frac{\overline{BD}}{\overline{DC}} = \frac{\mathscr{S}_{BPA}}{\mathscr{S}_{CAP}}$ e $\frac{\overline{CE}}{\overline{EA}} = \frac{\mathscr{S}_{CPB}}{\mathscr{S}_{ABP}}$.
Tem-se então que (traduzindo do Inglês original):

$$\begin{aligned}
\frac{\overline{AF}}{\overline{FB}}\,\frac{\overline{BD}}{\overline{DC}}\,\frac{\overline{CE}}{\overline{EA}} &= \frac{\mathscr{S}_{APC}}{\mathscr{S}_{BCP}}\,\frac{\overline{BD}}{\overline{DC}}\,\frac{\overline{CE}}{\overline{EA}} && \text{o ponto } F \text{ é eliminado}\\
&= \frac{\mathscr{S}_{APC}}{\mathscr{S}_{BCP}}\,\frac{\mathscr{S}_{BPA}}{\mathscr{S}_{CAP}}\,\frac{\overline{CE}}{\overline{EA}} && \text{o ponto } D \text{ é eliminado}\\
&= \frac{\mathscr{S}_{APC}}{\mathscr{S}_{BCP}}\,\frac{\mathscr{S}_{BPA}}{\mathscr{S}_{CAP}}\,\frac{\mathscr{S}_{CPB}}{\mathscr{S}_{ABP}} && \text{o ponto } E \text{ é eliminado}\\
&= 1
\end{aligned}$$

□

O GCLC foi capaz de demonstrar esta conjetura em menos de um segundo, produzindo um arquivo LaTeX com uma demonstração geométrica.

Vampire Um Provador Automático de Teoremas genérico. É possível usá-lo para provar conjeturas geométricas se um determinado sistema axiomático para geometria é dado [8].[5]

As regras para o GDDM foram convertidas para First-Order-Format (FOF) (formato para conjeturas em lógica de primeira ordem) do repositório *Thousands of Problems for Theorem Provers* (TPTP), uma biblioteca de problemas de diferentes áreas para serem usados como teste para os demonstradores automáticos (Automated Theorem Provers, ATP) [13]. Convertendo para este formato o problema dado acima para o JGEx.

```
%----Include Geometry Deductive Database Method axioms
include('geometryDeductiveDatabaseMethod.ax').

fof(exemplo6GDDFULL012001,conjecture,( ! [ A,B,C,D,E,F,G ] :
((  perp(D,C,A,B) & coll(D,A,B) &
    perp(E,B,A,C) & coll(E,A,C) &
    midp(F,C,B) & midp(G,D,E))
  =>
 (perp(F,G,D,E))))).
```

Listagem 3.1 *JGEx*, problem 01 (em FOF)

Vampire usa o método de resolução, fazendo a demonstração por refutação. É capaz de fazer a demonstração de forma muito eficiente, mas sem produzir uma demonstração geométrica.

```
% Version: Vampire 4.5.1 (commit 57a6f78c on 2020-07-15 11:59:04 +0200)
% Termination reason: Refutation
% Memory used [KB]: 18421
% Time elapsed: 0.866 s
```

[5] https://vprover.github.io/

Capítulo 4

A Implementação do Método das Bases de Dados Dedutivas para a Geometria

Tendo já feito a apresentação do, já longo, historial dos métodos automáticos de dedução, estamos agora em posição para começar a explorar em profundidade o método das bases de dados dedutivas. O projeto computacional que está na base desta tese de mestrado é a implementação de um demonstrador automático de teoremas para a geometria (GATP),[1] baseado nesse método. O conjunto de regras de inferência usado, como base para o mecanismo de inferência, foram as regras descritas em [4], a base para este provador. Este é um projeto de código aberto, incorporado no projeto *Open Geometry Prover Community Project*, disponível no repositório GitHub.[2]

Como parte do processo de desenvolvimento, a eficácia do provador para demonstrar conjeturas geométricas será testada usando conjeturas contidas no arquivo *Thousands of Geometric problems for geometric Theorem Provers (TGTP)* [11].

4.1 Como as Regras foram Implementadas

Cada regra é representada por uma função tal que essa função aceita como parâmetros de entrada a base de dados e o conjunto de pontos que definem o facto em estudo.

Muitas das regras requerem apenas um facto e devolvem ou um facto simples, uma permutação das posições dos pontos, ou uma conclusão direta, como, por exemplo, um conjunto que contém dois pontos e o ponto médio entre eles, correspondente a um conjunto colinear.

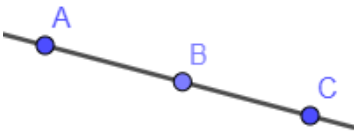
As restantes requerem factos que poderão, ou não, estar presentes na base de dados, de forma a obter novos factos para adicionar à base de dados. Nestes casos, uma pesquisa, utilizando a linguagem *data manipulation language* (DML), é necessária para procurar factos na base de dados que sejam relevantes para a aplicação da regra em estudo.

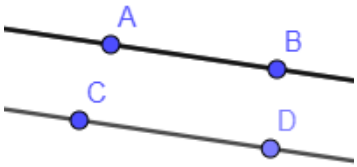
[1] Sigla Inglesa, *Geometry Automated Theorem Prover.*
[2] https://github.com/opengeometryprover

4.2 As Figuras Geométricas Representadas por Factos

Cada facto tem como objetivo criar uma estrutura abstrata que representa uma figura geométrica ou uma relação entre diferentes elementos geométricos. Certas restrições foram implementadas para reduzir o número de factos triviais obtidos pelo programa.

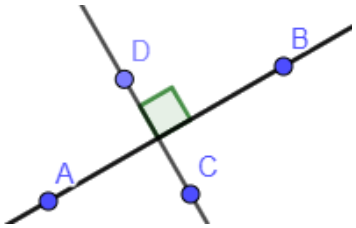
Coll(A,B,C)
Representa um conjunto de pontos *colineares* tal que existe pelo menos uma linha que os inclui a todos.
Restrições: Todos os pontos são diferentes.

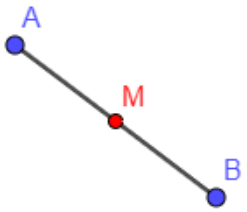




Para (A,B,C,D)
Representa um conjunto de retas $\overline{AB}$ e $\overline{CD}$ tal que estas retas são *paralelas*.
Restrições: $A \neq B$ e $C \neq D$ uma vez que não é possível concluir paralelismo entre pontos.

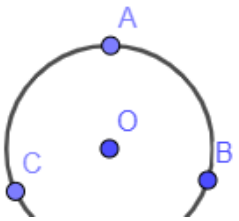
Perp (A,B,C,D)
Representa um conjunto de retas $\overline{AB}$ e $\overline{CD}$ tal que estas retas são *perpendiculares*.
Restrições: A≠B e C≠D uma vez que não é possível concluir perpendicularidade entre pontos.

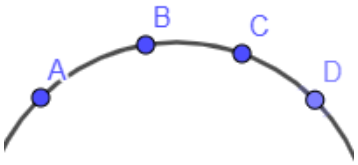




Midp(M,A,B)
Representa um ponto que é o ponto médio do segmento de reta formada pelos pontos A e B.
Restrições: Todos os pontos são diferentes.

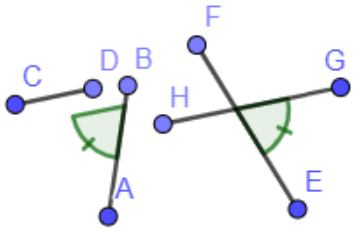
Circle (O,A,B,C)
Representa um círculo de centro O e com os pontos A,B,C contidos na fronteira.
Restrições: O centro não está contido no conjunto [A,B,C], caso contrário o raio do círculo seria zero. Os pontos da fronteira são também diferentes entre si.

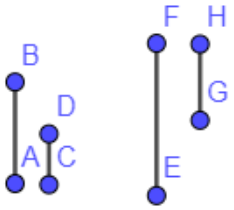




Cyclic (A,B,C,D)
Representa pontos tal que existe uma circunferência que contém todos os pontos na fronteira.
Restrições: Todos os pontos são diferentes.

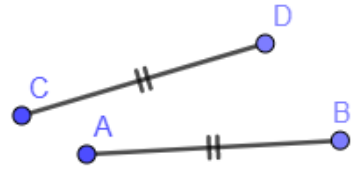
Eqangle(A,B,C,D,E,F,G,H) Representa retas tal que o ângulo entre AB e CD é equivalente ao ângulo entre EF e GH.
Restrições: A≠B, C≠D, E≠F e G≠H uma vez que não é possível concluir o ângulo entre retas e pontos.

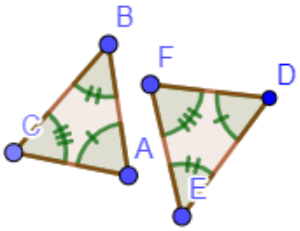




Eqratio(A,B,C,D,E,F,G,H)
Representa retas tal que a razão entre $\frac{\overline{AB}}{\overline{CD}}$ é equivalente a $\frac{\overline{EF}}{\overline{GH}}$.
Restrições: A≠B, $C \neq D$, $E \neq F$ e $G \neq H$ uma vez que não é possível concluir razão entre retas e pontos. O programa também evita casos em que os pontos presentes nas duas razões são equivalentes.

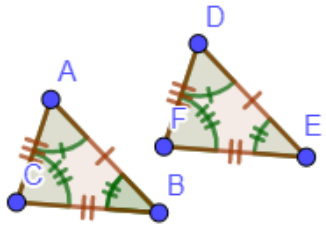
Cong(A,B,C,D)
Representa duas retas, AB e CD, que são congruentes. Isto é, as duas retas possuem a mesma distância.
Restrições: $A \neq B$ e $C \neq D$. $AB \neq CD$, caso trivial.





Simtri(A,B,C,D,E,F)
Representa dois triângulos ABC e DEF tal que estes triângulos apresentam ângulos internos equivalentes, logo a razão entre os diferentes lados dos triângulos são equivalentes.
Restrições: Os dois triângulos são distintos em pelo menos um dos pontos. Todos os pontos do triângulo são diferentes.

Contri(A,B,C,D,E,F)
Representa dois triângulos ABC e DEF tal que estes triângulos apresentam ângulos internos equivalentes e lados congruentes, logo são equivalentes.
Restrições: Os dois triângulos são diferentes em pelo menos um ponto.Todos os pontos do triângulo são diferentes.



4.3 Estrutura de cada Função

Cada regra é representada por tantas funções quantos os factos individuais relevantes. Logo, uma regra que inclua duas funções ou mais do mesmo tipo de facto requer apenas uma função. No entanto, uma regra que requer dois factos distintos terá múltiplas funções. Por exemplo:

- Regra D13: `cyclic(O,A,B,C) :- cong(O,A,O,B),cong(O,A,O,C),cong(O,A,O,D).`
  Só requer uma função uma vez que só aceita um tipo de facto como entrada. Neste caso, um facto `cong`.

- Regra D45: `midp(F,A,C) :- midp(E,A,B),para(E,F,B,C),coll(F,A,C).`
  Esta regra necessita de 3 funções diferentes, neste caso uma função que aceita factos *midp*, *para* e *coll*.

As diferentes funções estão divididas ao longo de cada tipo de facto, tal que no estudo de um facto referente ao paralelismo apenas regras que envolvem factos referente ao paralelismo serão testadas. Ver a listagem 4.1 para um exemplo de uma das funções implementadas.

Várias instruções são utilizadas para criar uma ligação entre o *SQLite* e o código em *C++*. Em particular, o mecanismo de transação das bases de dados relacionais é usado para assegurar a consistência da informação contida na base de dados. No caso da regra falhar a qualquer instante devido ao facto já estar presente na base de dados ou à falha de encontrar um ou mais factos complementares, o programa usa o comando DML "rollback" para reverter quaisquer alterações à base de dados. Caso a aplicação da regra seja bem sucedida, então o comando DML "commit" é utilizado, e todas as alterações são transcritas para a base de dados.

```
//Exemplo. Regra6: Para(A,B,C,D) & Para(C,D,E,F) :- Para(A,B,E,F)

DBinMemory Prover::ruleD6(dbim,p1,p2,p3,p4){
  std::string insertNewFact,lastInsertedRowId,lstInsRwId,insertionColl,nP,nP2
  ,querySecondGeoCmdA,querySecondGeoCmdB;

  insertNewFact = "INSERT INTO NewFact(typeGeoCmd) VALUES ('para')";
  lastInsertedRowId = "SELECT last_insert_rowid()";
  sqlite3_exec(dbim.db, "begin;", 0, 0, &(dbim.zErrMsg));
  dbim.rc = sqlite3_prepare_v2(dbim.db, insertNewFact.c_str() ,
  insertNewFact.size(), &(dbim.stmt), NULL);
  if (sqlite3_step(dbim.stmt) != SQLITE_DONE)
  {sqlite3_exec(dbim.db, "rollback;", 0, 0, 0); return(dbim);}
  //Inicia a função e cria um facto vazio do tipo esperado para esta regra.
```

```
  dbim.rc = sqlite3_prepare_v2(dbim.db, lastInsertedRowId.c_str() ,
  lastInsertedRowId.size(), &(dbim.stmt), NULL);
  sqlite3_step( dbim.stmt );
  lstInsRwId = (char*) sqlite3_column_text(dbim.stmt,0);

  nP="";
  nP2="";
  //Define novos pontos para a pesquisa.

  querySecondGeoCmdA = "SELECT p3,p4
                          FROM NewFact
                    INNER JOIN Parallel ON (newFact=id)
                         WHERE p1 = '"+p3+"' AND p2 = '"+p4+"' LIMIT 1";
  dbim.rc = sqlite3_prepare_v2(dbim.db,querySecondGeoCmdA.c_str(),
  querySecondGeoCmdA.size(), &(dbim.stmt1), NULL);
  sqlite3_step(dbim.stmt1);
  //Procura um exemplo aceitável ao longo dos novos factos.

  querySecondGeoCmdB = "SELECT p3,p4
                          FROM Facts
                    INNER JOIN Parallel ON (oldFact=id)
                         WHERE p1 = '"+p3+"' AND p2 = '"+p4+"' LIMIT 1";
  dbim.rc = sqlite3_prepare_v2(dbim.db,querySecondGeoCmdB.c_str(),
  querySecondGeoCmdB.size(), &(dbim.stmt2), NULL);
  sqlite3_step(dbim.stmt2);
  //Procura um exemplo aceitável ao longo dos factos já estabelecidos.

  if (sqlite3_data_count(dbim.stmt1) == 0 &&
      sqlite3_data_count(dbim.stmt2) == 0)

  {sqlite3_exec(dbim.db, "rollback;", 0, 0, 0); return(dbim);}
  else if (sqlite3_data_count(dbim.stmt1) != 0)
  {newPoint = (char*) sqlite3_column_text(dbim.stmt1, 0);
  newPoint2 = (char*) sqlite3_column_text(dbim.stmt1, 1);}
  else {newPoint = (char*) sqlite3_column_text(dbim.stmt2, 0);
  newPoint2 = (char*) sqlite3_column_text(dbim.stmt2, 1);}
  //Caso as duas procuras estejam vazias, a função acaba
  sem alterar a base de dados.

  insertionColl = "INSERT INTO Parallel(typeGeoCmd,p1,p2,p3,p4,newFact)
                       VALUES ('para','"+p1+"','"+p2+"'
                       ,'"+nP+"','"+nP2+"','"+lstInsRwId+"')";
  //Cria o novo facto para inserir na base de dados. Um facto paralelo com
  os pontos p1 e p2 do facto original e nP e nP2 da pesquisa.

  dbim.rc = sqlite3_prepare_v2(dbim.db, insertionColl.c_str() ,
  insertionColl.size(), &(dbim.stmt), NULL);
  if (sqlite3_step(dbim.stmt) != SQLITE_DONE)
  {sqlite3_exec(dbim.db, "rollback;", 0, 0, 0); return(dbim);}
  //Caso o novo facto já está na base de dados,
  é rejeitado e a função termina.

  sqlite3_exec(dbim.db, "commit;", 0, 0, 0);
  return(dbim);
  //O facto é adicionado com sucesso e a função termina.
}
```

Listagem 4.1 Regra 6

4.4 Regras Problemáticas

Certas regras presentes no texto original utilizavam elementos que não tinham sido introduzidos até esse momento, o que complicou a implementação dessas regras no programa e limitou a possibilidade de transformar os elementos presentes no texto em um programa capaz de resolver problemas geométricos automaticamente sem qualquer intervenção humana.

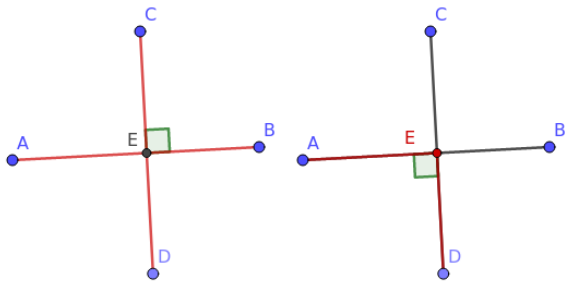
Em particular as regras designadas por D42, D43 e D61 apresentam elementos exclusivos a essas passagens.

- Regra 42 introduz um facto *colinear* de quatro pontos, diferente do caso implementado de três pontos. Neste caso basta considerar duas pesquisas uma vez que `coll(A,B,C) & coll(B,C,D) :- coll(A,B,C,D).`
- Regra 43 apresenta um problema semelhante com o caso de requisitar um facto *cíclico* com 6 pontos, enquanto que o caso implementado usa 4 pontos. Uma vez que um caso cíclico só é único quando contém pelo menos 3 pontos comuns, então é necessário um total de 3 pesquisas independentes para confirmar a existência deste facto. `cyclic(A,B,C,D) & cyclic(B,C,D,E) & cyclic(C,D,E,F):- cyclic(A,B,C,D,E,F).`
- Finalmente a regra 61 apresenta a expressão $AB = PQ$. Esta condição pode, em teoria, ser expressa por um facto de congruência, neste caso `cong(A,B,P,Q)`. Portanto, a estranha representação única nesta lista é notada como confusa.

4.5 Falhas na Lista de Regras Original

Um problema encontrado com as regras analisadas foi que, alguns casos triviais apresentaram-se como impossíveis de resolver. Um exemplo simples é o seguinte problema.

```
  perp(A,B,C,D)
& coll(A,E,B)
& coll(C,E,D)
  =>
 perp(A,E,E,D)
```



Devido a uma falta de regras que utilizem factos *perpendiculares* e/ou *colineares*, não existe nenhuma sequência que permita concluir este facto. Mesmo deixando de ter em consideração algumas restrições, permitindo ao programa encontrar factos triviais `para(A,B,A,B)` ou `para(C,D,C,D)`, o programa não conclui este facto.

Apesar de ser um caso trivial, pode ser considerado um passo importante noutros problemas e a dificuldade em provar este simples caso levou à decisão de criar regras adicionais para além da lista original em estudo permitindo, assim, encontrar uma maior quantidade de factos que possam levar a uma resolução deste e de outros problemas.

4.6 Falhas na Implementação do Programa

Certas fraquezas da estratégia planeada originalmente foram detetadas ao longo da criação, teste e desenvolvimento do programa. Isto aconteceu devido à necessidade de ter todas as regras implementadas e o programa a correr sem erros antes de ser possível analisar e testar a interação entre as regras, os problemas geométricos e a implementação das regras.

A pesquisa utilizada pelas funções não tem em consideração o potencial facto criado e a sua possível existência na base de dados. Existe, portanto, a possibilidade de o resultado da pesquisa retornar um elemento tal que o facto final já está presente na base de dados. Apesar de ser um caso válido, o programa falha no âmbito de não encontrar um caso mais relevante.

Cada função permite apenas a introdução de um facto por chamada dessa função. Enquanto que o programa tem como capacidade diferentes regras que permitem obter factos semelhantes, à medida que a quantidade de factos aumenta, existe a possibilidade de a quantidade de factos possíveis exceder o que o programa pode descobrir devido a esta limitação.

4.7 Regras Novas

Duas novas regras foram adicionadas para ajudar a resolver os problemas em estudo. Como objetivo principal, as regras seguem as propriedades da geometria. Apesar de não ter sido suficiente para um aumento significativo da quantidade de problemas que o programa era capaz de demonstrar, a simples adição de duas regras permitiu demonstrar as capacidades do programa e potencial, uma vez que permite a resolução de um problema modelo ao auxiliar as regras já estabelecidas.

- `rule P1: perp(A,B,C,D), coll(A,E,B) => perp(A,E,C,D)`

  A primeira regra tem como objetivo encontrar mais factos *perpendiculares* considerando o caso em que existe pelo menos um ponto colinear entre as retas em estudo.

- ```
  rule P2: circle(O,A,B,C), cyclic(A,B,C,D) => cong(O,A,O,D)
                                               cong(O,B,O,D)
                                               cong(O,C,O,D)
  ```

  A segunda regra permite uma mais fácil introdução de casos *cong* à lista de factos. Uma vez que existem várias conclusões e duas categorias de factos como ponto de entrada, um total de 6 funções foram incluídas para representar esta regra, permitindo a inclusão de vários valores `cong` que poderão não ter sido encontrados por outras regras.

4.8 Exemplo de Utilização do Programa

Considere-se, de novo, o exemplo `01`, do conjunto de conjeturas geométricas contido no sistema *JGEx*. Para o efeito de ser testada pelo programa desenvolvido, foi necessário a sua conversão para o formato First-Order-Format, ver a listagem 4.2.
  ```

```
fof(exemplo6GDDFULL012001,conjecture,( ! [ A,B,C,D,E,F,G ] :
((  perp(D,C,A,B) & coll(D,A,B) &
    perp(E,B,A,C) & coll(E,A,C) &
    midp(F,C,B) & midp(G,D,E))
  =>
 (perp(F,G,D,E))))).
```

Listagem 4.2 *GDDM* problem 01 (em FOF)

Podemos representar este problema pela seguinte figura.(ver 4.1)

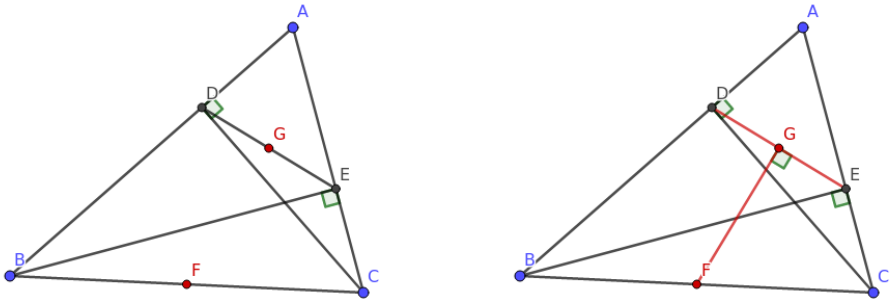


Fig. 4.1 Hipótese => Conclusão

Utilizando a linha de comandos, podemos introduzir o problema anterior, formatado de forma apropriada ao programa. Quando o programa atingir um ponto fixo, segundo o método das bases de dados dedutivas, ele testa se o facto da conclusão se encontra na lista de factos construída pelo programa. Finalmente, o programa retorna a sequência de factos que encontrou por ordem em que foram gerados, se foi demonstrado o problema e o tempo que demorou a atingir o ponto fixo.

4.9 Resultados

Apesar de uma implementação das regras, vários problemas apareceram ao longo do desenvolvimento que limitaram as capacidades do programa de resolver problemas geométricos apropriadamente. A introdução de duas regras independentes da lista original permitiu a demonstração do problema geométrico já referido em 4.8.

Na demonstração desta conjetura foi imposto um limite de tempo de 300s, tendo sido usado um sistema Debian/Linux num computador: Intel(R) Core(TM) i5-4430 CPU @ 3.00GHz, 16GiB RAM.

Foram obtidos 6076 factos, o tempo de CPU usado foi de 33,294s, tendo sido possível demonstrar a conjetura.

A introdução das regras `P1` e `P2` foi suficiente para superar os desafios encontrados em pelo menos um caso, ambos da lista de regras originais e dos problemas encontrados ao longo do desenvolvimento do programa. Permitiu, assim, demonstrar a capacidade do programa para resolver problemas geométricos automaticamente.

Utilizando como base a lista de exemplos para o método das bases de dados dedutivas, contida no programa *JGEx*, testou-se o *GDDMlite*.

Dos 100 problemas testados, apenas dois foram demonstrados pelo *GDDMlite*. Dos restantes, em um total de 75 problemas, o programa foi capaz de obter um ponto fixo dentro do tempo limite imposto de 300 segundos. Uma baixa taxa de sucesso, quando comparada com o programa JGEx, deve-se em nossa opinião a: as regras implementadas no programa JGEx diferem das apresentadas em [4], sendo que estas últimas não estão completas (ver secção 4.5); além disso, algumas melhorias no código produzido serão necessárias, nomeadamente em questões de procura de uma maior eficiência aquando da demonstração, uma vez que alguns dos casos atingiram o limite de tempo.

Capítulo 5

Conclusões

5.1 Objetivos Atingidos

O objetivo do presente trabalho era o estudo e posterior implementação do método das bases de dados dedutivas, tal como foi descrito em [4], e esse objetivo foi atingido quase na sua totalidade. Baseando-se num trabalho inicial desenvolvido pelo orientador, o Professor Pedro Quaresma, a implementação de todas as regras, com exceção daquelas que introduzem novos pontos à conjetura foi efetuada. Para uma dada conjetura, o programa atingiu um ponto-fixo e um conjunto de factos que foram possíveis de obter tendo como ponto de partida os factos contidos na conjetura. Em algumas das conjeturas testadas, foi possível obter a conclusão, isto é, conseguiu-se demonstrar a conjetura. Foi também possível chegar à conclusão que em muitas situações o conjunto de regras apresentado em [4] é insuficiente, isto é, embora um método deste tipo não seja em geral completo, e os autores do artigo original referem isso mesmo, a comparação com o programa *JGEx*, desenvolvido pelos autores do artigo, mas posterior ao mesmo [14], foi desfavorável para a nossa implementação. O conjunto de regras usado no sistema *JGEx* difere do apresentado originalmente. Foi feito algum trabalho inicial no sentido de expandir o conjunto de regras assim como tornar mais eficiente o algoritmo.

5.2 Trabalho Futuro

Ao longo do trabalho, várias vulnerabilidades relativas à implementação do programa foram encontradas. Apesar de todas as regras serem implementadas com sucesso e verificadas através de conjeturas específicas para esse efeito, existe, no entanto, quando considerados os exemplos contidos no programa *JGEx*, um conjunto significativo de conjeturas que não foram demonstradas.

Isto indicou falhas do ponto de vista das regras originais, onde uma tentativa de suplementar regras adicionais permitiu um maior sucesso de demonstrações e revelou certas limitações relativamente à estrutura considerada para o programa inicialmente, que uma vez implementada, não permitia uma correção fácil. Apesar disso, uma elevada descoberta de factos consistentes com as propriedades esperadas da geometria permite concluir que, com uma futura correção destes problemas observados e analisados ao longo deste projeto, poderá ser criada uma versão deste programa capaz de resolver problemas geométricos com elevado sucesso.

Bibliografia

[1] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z. (1994). *Machine Proofs in Geometry*. World Scientific.

[2] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z. (1996a). Automated generation of readable proofs with geometric invariants, I. multiple and shortest proof generation. *Journal of Automated Reasoning*, 17(13):325–347.

[3] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z. (1996b). Automated generation of readable proofs with geometric invariants, II. theorem proving with full-angles. *Journal of Automated Reasoning*, 17(13):349–370.

[4] Chou, S.-C., Gao, X.-S., and Zhang, J.-Z. (2000). A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3):219–246.

[5] Hohenwarter, M. (2002). Geogebra - a software system for dynamic geometry and algebra in the plane. Master's thesis, University of Salzburg, Austria.

[6] Janičić, P. (2006). GCLC — A tool for constructive euclidean geometry and more than that. In Iglesias, A. and Takayama, N., editors, *Mathematical Software - ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer.

[7] Janičić, P., Narboux, J., and Quaresma, P. (2012). The Area Method: a recapitulation. *Journal of Automated Reasoning*, 48(4):489–532.

[8] Kovács, L. and Voronkov, A. (2013). First-Order theorem proving and Vampire. In Sharygina, N. and Veith, H., editors, *Computer Aided Verification — CAV 2013*, pages 1–35, Berlin, Heidelberg. Springer Berlin Heidelberg.

[9] Kovács, Z., Recio, T., and Vélez, M. (2018). Using automated reasoning tools in geogebra in the teaching and learning of proving in geometry. *International Journal for Technology in Mathematics Education*, 25(2):33–50.

[10] Kovács, Z., Recio, T., and Vélez, M. (2022). *Mathematics Education in the Age of Artificial Intelligence*, chapter Automated Reasoning Tools with GeoGebra: What are they? What are they good for? Springer Nature.

[11] Quaresma, P. (2011). Thousands of Geometric problems for geometric Theorem Provers (TGTP). In Schreck, P., Narboux, J., and Richter-Gebert, J., editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 169–181. Springer.

[12] Quaresma, P. (2022). *Evolution of Automated Deduction and Dynamic Constructions in Geometry*, volume 17 of *Mathematics Education in the Digital Era*, chapter 1, pages 3–22. Springer.

[13] Sutcliffe, G. (2017). The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 59(4):483–502.

[14] Ye, Z., Chou, S.-C., and Gao, X.-S. (2011). An introduction to Java Geometry Expert. In Sturm, T. and Zengler, C., editors, *Automated Deduction in Geometry*, volume 6301 of *Lecture Notes in Computer Science*, pages 189–195. Springer Berlin Heidelberg.