初等幾何 AI AlphaGeometry と Newclid の教育利用

福岡教育大学　藤本 光史
Mitsushi Fujimoto, University of Teacher Education Fukuoka

1　はじめに

これまでの初等幾何定理の自動証明には，代数的手法（algebraic proof）と合成的手法（synthetic proof）の二つのアプローチがある．代数的手法は幾何的対象を代数方程式系などで表現し，その数式を機械的に変形することで命題の真偽を判定するものであり，Wu's method，Groebner bases method，Quantifier elimination method などがある．一方，合成的手法は前提で与えられた図形から補助点の追加・直線の延長・円の描画などの図形操作を用いて証明を組み立てるものであり，Area method，Full-angle method，Deductive database method などがある．代数的手法と比較して，合成的手法は適用範囲が限定的であるが，通常の人間による証明に近いものを出力するという特徴がある．

本稿では，合成的手法の一つである Deductive database method に着目し，それを用いた初等幾何 AI の AlphaGeometry と Newclid の利用方法について紹介する．また，これらを用いた福岡県高校入試問題の証明生成実験および GeoGebra との連携についても報告する．

2　Deductive database method と JGEX

AlphaGeometry と Newclid で重要な役割を果たす Deductive database method と JGEX について述べる．

2.1　Deductive database method について

初等幾何定理の自動証明における Deductive database method は [1] で提案された手法である．

図 1 は Deductive database method の処理の流れを表したものである[1]．Deductive database method は命題の仮定から初期事実のリストを生成し，データベースを構築する．そこから既知の定義やルールを用いて新事実を生成してデータベースの更新を繰り返す．新事実が生成できなくなった状態（不動点）に結論が含まれていれば「証明完了」．そうでなければ何らかの方法で補助点を追加し，データベースの更新に戻る．事前に設定された個数まで補助点を追加しても証明が完了しなかった場合は「証明不能」で終了する．

[1] この図は [2] に掲載されたものを日本語化したものである．

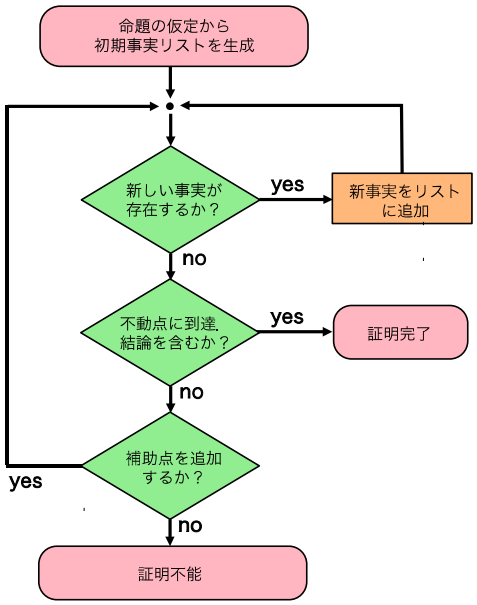


図 1: Deductive database method のフロー

2.2 JGEX について

Java Geometry Expert (JGEX [3]，図 2) は GeoGebra のような動的幾何のユーザインタフェースを有する自動幾何定理証明アプリである．JGEX は 2017 年まで Zheng Ye により GitHub 上で管理されていたが，その後 Zoltán Kovács によりフォークされ，現在も開発が継続されている [4]．既存の代数的手法や合成的手法のアルゴリズムが実装されており，容易に初等幾何の自動証明を体験できるだけでなく，アニメーションを用いた証明のプレゼンテーションツールとしての機能も有している．

この JGEX で利用されている図形描画コマンドが AlphaGeometry や Newclid の図形記述言語のベースになっている．

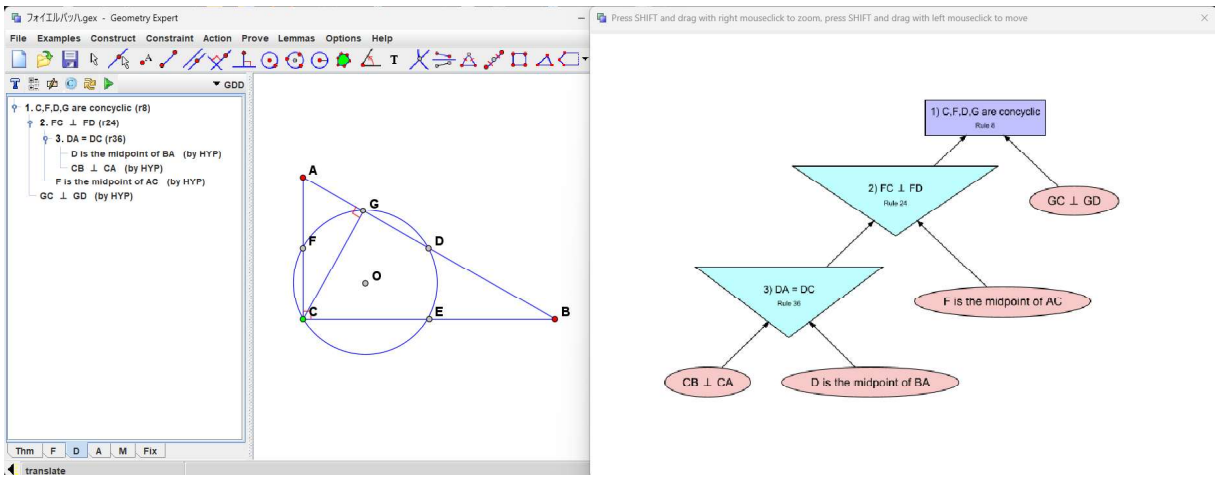


図 2: JGEX

3 AlphaGeometry

3.1 AlphaGeometry について

AlphaGeometry は Google DeepMind によって開発された初等幾何 AI であり，2024 年 1 月に Nature の論文 [5] として発表された．ベンチマークの国際数学オリンピックの幾何問題（30 問）の 8 割（25 問）を解いたことから，金メダリスト（上位 1/12 以内）に相当する能力を有するとされ，多くのネットニュースで話題となった．演繹データベース（DD）と代数的推論（AR）を組み合わせた記号演繹ソルバー DDAR と大規模言語モデル（LLM）を利用していることが大きな特徴である．

DDAR は 2.1 で述べた Deductive database method をベースに以下のような代数的な記号処理を行うものであり，LLM[2]は補助点の追加の際に利用される．

辺の延長	$\|AB\| + \|BC\| = \|AC\|$
三平方の定理	$\|AB\|^2 + \|BC\|^2 = \|AC\|^2$
三角形の相似	$\|AB\| : \|BC\| = \|DE\| : \|EF\|$

以下は AlphaGeometry 内部の処理を擬似コードで表したものである．

処理手順 1 AlphaGeometry の基本的な処理の流れ

$D \leftarrow$ 前提からデータベースを構築
$beams \leftarrow \{D\}$ （初期状態）
for $i = 0$ **to** $DEPTH$ **do**
  $new\_beams \leftarrow \emptyset$
  for each $D \in beams$ **do**
    while D が DDAR で更新可能 **do**
      $D \leftarrow$ DDAR で D を更新
    end while
    if 結論 $\in D$ **then**
      traceback で証明ステップを抽出
      return 証明を出力
    else
      $candidates \leftarrow$ LLM により補助構成の候補を $BATCH\_SIZE$ 個生成
      $new\_beams \leftarrow new\_beams \cup \{D + cand \mid cand \in candidates\}$
    end if
  end for
  $beams \leftarrow top\_k(new\_beams, BEAM\_SIZE)$
end for
return 証明失敗

[2]AlphaGeometry の LLM は，ランダム生成した図形の前提条件（premises）集合に対し，記号ソルバーが構築した〈仮定・結論・証明〉の大規模データ（1 億例，うち約 900 万は補助構成を含む）を用いてトレーニングされている．

LLM は *BATCH_SIZE* 個の補助点候補を生成し，各候補に付与されたスコアに基づいてビーム探索を行う．ここで，*DEPTH* は探索の最大深さ，*BEAM_SIZE* は各探索段階で保持するビーム幅を表す．

3.2 AlphaGeometry のインストール

ここでは，Linux 環境[3]に AlphaGeometry をインストールする方法について解説する．

Python 仮想環境

AlphaGeometry は Python 3.10 環境が必要なので，次のように Anaconda を用いて仮想環境[4]を作成する．

```
$ conda config --set auto_activate_base false
$ conda create -n alphageometry python=3.10
$ conda activate alphageometry
```

ここでは仮想環境名を alphageometry として作成した．この環境から抜ける場合は以下を実行すればよい．

```
$ conda deactivate
```

必要ファイルのダウンロード

次のように GitHub から AlphaGeometry のソースコードを取得する．

```
$ conda activate alphageometry
$ git clone git@github.com:google-deepmind/alphageometry.git
$ cd alphageometry
$ pip install --require-hashes -r requirements.txt
$ conda install -c conda-forge gdown
$ gdown --folder https://bit.ly/alphageometry
```

`gdown` は，Google Drive 上のファイルをダウンロードするための Python 製コマンドラインツールである．これによって，AlphaGeometry のソースコードと共に以下のパラメータファイルが入手できる．

`checkpoint_10999999` (1.2GB): 学習済みパラメータの状態を記録したファイル．トレーニングの 10999999 ステップ後の状態を記録しており，推論を行う際にこの状態をロードすることでモデルが既に学習済みの知識を活用できる．

`geometry.757.model` (14KB): モデルのアーキテクチャ（層の構成や接続，ハイパーパラメータなど）の情報が記述されたファイル．チェックポイントで保存されたパラメータを正しく反映させるために必要となる．

[3]筆者は Windows 11 の WSL2 にインストールした Debian 上で動作確認を行っている．

[4]ホスト OS にインストールされている Python とは独立した環境を作成し，ライブラリの依存関係による問題を回避するためのもので，Anaconda の他に venv や uv などのツールがある．

geometry.757.vocab (10KB): モデルが扱う入力や出力を数値化するための語彙情報を保持したファイル．データの前処理（テキストや記号のトークン化）や，出力を解釈可能な形に戻す後処理で利用される．

AlphaGeometry の LLM は Meliad（LLM を効率的に学習・推論するためのライブラリ）を用いているため，それもインストールする必要がある．しかし，2025 年 6 月上旬に Meliad 側のメインブランチが更新されて AlphaGeometry が動作しなくなったため，次のように 2025 年 5 月 31 日の日付を指定してチェックアウトする．

```
$ mkdir meliad_lib; cd meliad_lib
$ git clone https://github.com/google-research/meliad
$ cd meliad
$ git fetch origin
$ REV=$(git rev-list --before="2025-05-31" -n 1 origin/main)
$ git checkout -b ag-compatible $REV
```

起動前準備

AlphaGeometry を起動する前に，以下のように環境変数を設定する．

コード 1: preset.sh

```
DATA=ag_ckpt_vocab
MELIAD_PATH=meliad_lib/meliad
export PYTHONPATH=$PYTHONPATH:$MELIAD_PATH
DDAR_ARGS=(
  --defs_file=$(pwd)/defs.txt \
  --rules_file=$(pwd)/rules.txt \
);
BATCH_SIZE=2
BEAM_SIZE=2
DEPTH=2
SEARCH_ARGS=(
  --beam_size=$BEAM_SIZE
  --search_depth=$DEPTH
)
LM_ARGS=(
  --ckpt_path=$DATA \
  --vocab_path=$DATA/geometry.757.model \
  --gin_search_paths=$MELIAD_PATH/transformer/configs \
);
```

以上で AlphaGeometry のインストールは完了である．

3.3 AlphaGeometry の利用

AlphaGeometry で「三角形の外心・重心・垂心は同一直線上にある[5]」という命題を証明させてみる．

命題ファイルの準備

以下の内容のファイル `examples.txt` を `alphageometry` ディレクトリに作成する[6]．

コード 2: examples.txt

```
euler
a b c = triangle a b c; h = orthocenter a b c; g1 g2 g3 g = centroid g1 g2
↪ g3 g a b c; o = circle a b c ? coll h g o
```

1 行目は問題名であり，2 行目がこの命題に対応した記述である．?記号の左側が命題の「仮定」で，右側が「結論」になっている．この記述に対応する図を以下に示す．

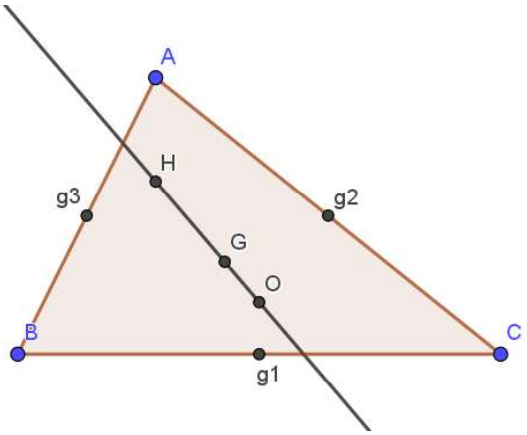


図 3: オイラー線

これは JGEX で用いられた記法 [6] を拡張して AlphaGeometry 用に新たに策定された図形構築言語を用いて記述されており，**JGEX 記法**と呼ばれている．

AlphaGeometry の実行

以上の準備の後，AlphaGeometry は以下によって実行することができる．

```
$ python -m alphageometry \
  --alsologtostderr \
  --problems_file=$(pwd)/examples.txt \
  --problem_name=euler \
  --mode=alphageometry \
  "${DDAR_ARGS[@]}" \
  "${SEARCH_ARGS[@]}" \
  "${LM_ARGS[@]}"
```

[5] この直線はオイラー線と呼ばれる．

[6] 3.2 で述べたインストール作業に従えば，`examples.txt` は既に存在しているはずである．

主なオプションの意味は次の通りである．

--problems_file: 命題が記述されたファイルを指定するためのオプション．このファイルには複数の命題を記述することができる．

--problem_name: --problems_file で指定されたファイルから解かせたい命題を指定するためのオプション．

--mode: 利用する証明エンジンを指定するためのオプション．alphageometry の場合は DDAR+LLM で推論し，ddar の場合は DDAR のみで推論する．

AlphaGeometry による証明

以下は AlphaGeometry が出力したこの命題の証明である．

```
* Proof steps:
001. C,B,E are collinear [02] & EB = EC [03] ⇒ E is midpoint of CB [12]
002. C,F,A are collinear [04] & FC = FA [05] ⇒ F is midpoint of CA [13]
003. E is midpoint of CB [12] & F is midpoint of CA [13] ⇒ EF ∥ BA [14]
004. EF ∥ AB [14] & F,B,H are collinear [06] & A,H,E are collinear [07] ⇒ FE:FH = AB:BH [15]
005. G,A,B are collinear [10] & GA = GB [11] ⇒ G is midpoint of BA [16]
006. F is midpoint of CA [13] & G is midpoint of BA [16] ⇒ FG ∥ CB [17]
007. E is midpoint of CB [12] & G is midpoint of BA [16] ⇒ EG ∥ CA [18]
008. C,B,E are collinear [02] & BC ∥ FG [17] & AC ∥ EG [18] ⇒ ∠ FGE = ∠ BEG [19]
009. G,A,B are collinear [10] & AB ∥ EF [14] & AC ∥ EG [18] ⇒ ∠ FEG = ∠ BGE [20]
010. ∠ FGE = ∠ BEG [19] & ∠ FEG = ∠ BGE [20] (Similar Triangles) ⇒ EF = GB [21]
011. FE:FH = AB:BH [15] & EF = GB [21] & GA = GB [11] ⇒ GA:FH = AB:BH [22]
012. EB = EC [03] & IB = IC [08] ⇒ CB ⊥ EI [23]
013. IA = IB [09] & IB = IC [08] ⇒ IC = IA [24]
014. FC = FA [05] & IC = IA [24] ⇒ CA ⊥ FI [25]
015. CB ⊥ EI [23] & AD ⊥ BC [00] & CA ⊥ FI [25] & BD ⊥ AC [01] ⇒ ∠ ADB = ∠ EIF [26]
016. CA ⊥ FI [25] & BD ⊥ AC [01] & AB ∥ EF [14] ⇒ ∠ ABD = ∠ EFI [27]
017. ∠ ADB = ∠ EIF [26] & ∠ ABD = ∠ EFI [27] (Similar Triangles) ⇒ AB:BD = FE:FI [28]
018. FE:FI = AB:BD [28] & EF = GB [21] & GA = GB [11] ⇒ GA:FI = AB:BD [29]
019. GA:FH = AB:BH [22] & GA:FI = AB:BD [29] ⇒ FH:FI = BH:BD [30]
020. F,B,H are collinear [06] & CA ⊥ FI [25] & BD ⊥ AC [01] ⇒ ∠ HFI = ∠ HBD [31]
021. FH:FI = BH:BD [30] & ∠ HFI = ∠ HBD [31] (Similar Triangles) ⇒ ∠ FHI = ∠ BHD [32]
022. ∠ FHI = ∠ BHD [32] & F,B,H are collinear [06] ⇒ HI ∥ DH [33]
023. HI ∥ DH [33] ⇒ I,H,D are collinear
```

AlphaGeometry による証明を精読した結果，以下のような傾向が観察できた．

- 命題入力時に与えたことを再び証明の中で記述する．
- 回りくどい証明である．そのため，証明ステップが長くなっている．
- 人間であれば通常しない変形を行う．
- 証明のキーポイントがわかりにくく，人間が再現するのは困難である．
- 実行する度に異なる証明を出力する．

3.4 AlphaGeometry を利用する際の問題点

AlphaGeometry を利用する際の最大の問題点は，JGEX 記法の詳細が不明確であることである．筆者は GitHub の Issues に投稿された “ALPHAGEOMETRY SYNTAX USED IN REPRESENTING GEOMETRIC ELEMENTS” という文書と，ベンチマーク用の IMO 問題ファイル imo_ag_30.txt を参考にしたが，記述方法に誤りがあるのでは，という不安がいつも付き纏った．

この記法問題は次節の Newclid の登場により，大幅に改善された．

4 Newclid

4.1 Newclid について

Newclid は Lagrange Mathematics and Computing Research Center（LMCRC[7]）によって開発された AlphaGeometry ベースの初等幾何 AI であり，2024 年 11 月に arXiv プレプリント [7] として発表された．AlphaGeometry の実験結果から LLM より DDAR の貢献が大きいと判断し，DDAR の性能を高めるアプローチを採用している．2025 年 8 月に公開された Version 3 では，記号演繹ソルバー DDAR の主要部を C++で実装し直すことで推論の高速化を実現している．AlphaGeometry と同様にソースコードは GitHub 上に公開されている．また，利用する際に必要な情報が整理されており，命題を記述するための JGEX 記法の詳細 [8] も明確にされている[8]．

4.2 Newclid のインストール

Newclid のインストールは AlphaGeometry と比較して非常に簡単である．

```
$ git clone https://github.com/Newclid/Newclid.git
$ cd Newclid
$ pip install uv
$ uv sync
$ source .venv/bin/activate
```

このように Python 仮想環境 uv を利用して，`uv sync` を実行するだけでソースコードがビルドされ，バイナリファイルが `.venv/` 以下に生成される[9]．

4.3 Newclid の利用

`problems_datasets/examples.txt` に解かせたい命題を JGEX 記法で記述し，以下を実行する．

```
$ newclid -o outputs/euler jgex --problem-id euler --file
↪ ./problems_datasets/examples.txt
```

主なオプションの意味は次の通りである．

`--file`: 命題が記述されたファイルを指定するためのオプション．このファイルには複数の命題を記述することができる．

[7] 2020 年に HUAWEI がパリに開設した数学と計算のため研究所．
[8] この記法のほとんどが AlphaGeometry でも利用可能である．
[9] ただし，boost-1.83 ライブラリを利用するので，事前にインストールしておく必要がある．

--problem-id: `--file` で指定されたファイルから解かせたい命題を指定するためのオプション．

-o: 結果の証明と図を出力するディレクトリを指定するためのオプション．

このコマンドによって，`outputs/euler` ディレクトリに，実行結果の証明ファイル `proof_steps.txt` と図ファイル `proof_figure.svg` が生成される．

5 高校入試問題を用いた実験

福岡の県立高校入試問題（数学）は，平面図形と空間図形の問題が毎年出題されており，このうち平面図形では記述式の証明問題が含まれている．2013 年から 2024 年の 12 年間に出題された 13 問の証明問題を，AlphaGeometry と Newclid に解かせる実験を行ったので，その概要を紹介する[10]．

以下の図 4 は，2024 年の証明問題（抜粋）である．

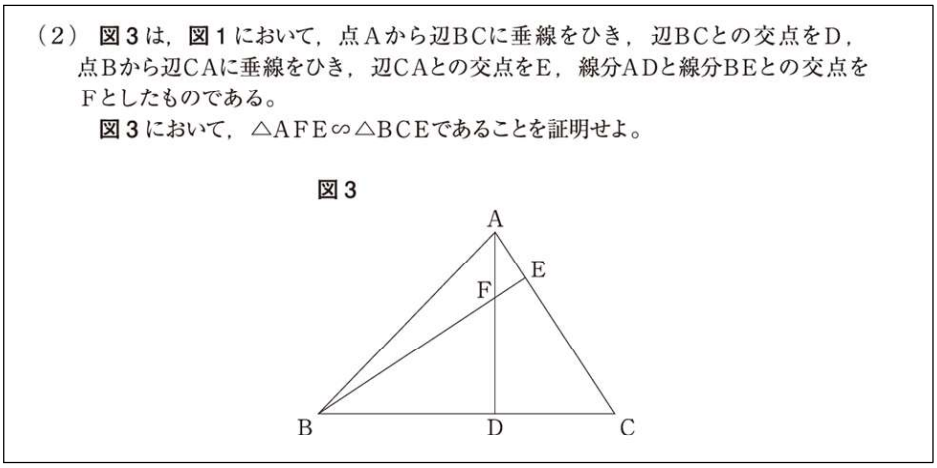


図 4: 2024 年度 福岡県立高校入試問題

この問題を JGEX 記法で記述したものが次である．

コード 3: examples.txt

```
2024
a b c = triangle a b c; d = foot d a b c; e = foot e b c a; f = on_line f a
↪ d, on_line f b e ? simtri a f e b c e
```

このように 13 問の証明問題を JGEX 記法で記述し，AlphaGeometry と Newclid で解かせた結果を表 1 に示す．実行時間は表 2 の通りであった．

この実験で用いた問題は，いずれも補助点を追加することなく証明可能なものである．AlphaGeometry は LLM の使用有無に関わらず，起動時に LLM を読み込む仕様となっている．そのため，実際の計算処理とは直接関係しない初期化処理に時間を要し，全体

[10] この実験は本研究室での卒業研究 [9] をベースにして，問題の追加，JGEX 記述の修正，実行環境の変更，Newclid のアップデートなどを行った上で実施したものである．

表 1: 実験結果

年度	問題	AlphaGeometry	Newclid
2024	三角形内の三角形の相似	○	○
2023-1	正方形内の三角形の合同	○	○
2023-2	正方形内の三角形の相似	△	○
2022	円に内接する二等辺三角形内の二辺の相等	○	○
2021	平行四辺形内の三角形の合同	○	○
2020	円内の三角形の相似	△	○
2019	円内の二辺の相等	○	○
2018	半円内の三角形の相似	△	○
2017	円に内接する三角形の相似	○	○
2016	円内の三角形の相似	○	○
2015	円に内接する五角形内の三角形の相似	×	○
2014	円に内接する四角形内の三角形の相似	△	○
2013	半円内の三角形の相似	△	○

○は「証明成功」，△は「1200 秒で計算打ち切り」，×は「証明に誤りあり」を表す．

表 2: 実行時間（○と×のみ）

	AlphaGeometry	Newclid
平均時間 [秒]	19.0	0.034
最小〜最大 [秒]	18 〜 21	0.012 〜 0.083

実行環境
CPU: AMD Ryzen AI 9 HX PRO370（2.00–5.10GHz）
Memory: 64GB
OS: Debian 13（Windows 11 上の WSL）

の実行速度が遅くなる傾向がある．一方，Newclid は LLM を搭載しておらず，記号演繹ソルバーを C++ により実装しているため，初期化に伴うオーバーヘッドが小さく，高速に動作する．

この実験結果から，AlphaGeometry のバグや実装速度の問題が Newclid で改善されていることがわかる．

6　JGEX による命題記述の課題とその対応策

前節の実験により，高校入試における（補助点の追加が不要な）証明問題は Newclid で解けることが確認できた．しかし，この実験から，幾何 AI の教育利用を実現するには「JGEX 記法による命題の記述が難しい」という課題が見えてきた．ここでは，この課題について，二つのアプローチを試みる．

6.1 生成 AI の利用

生成 AI（Generative AI）は，入力されたデータや指示に基づいて，新しいコンテンツ（文章・画像・音声・コード）を自動的に生成する新しい AI 技術の一つである．膨大なデータから統計的なパターンを学習し，その学習結果を使って人間に匹敵するものを出力する．

ここでは，コード生成が可能な生成 AI（ChatGPT，Claude，Gemini，Grok）を使用して，日本語で記述された問題文から JGEX への変換を試みる．以下が生成 AI に指示したプロンプトである．

```
## 初等幾何 AI の Newclid に高等学校入試問題を解かせたい。Newclid を利用するために必要な命題記述言語 JGEX の仕様は以下に記載されている。
- 命題の前提 (?記号の前) の記述
https://newclid.github.io/Newclid/manual/building_a_problem_setup/jgex/definitions.html
- 命題の結論 (?記号の後) の記述
https://newclid.github.io/Newclid/manual/concepts/predicates.html#newclid-predicates

## この言語仕様に従って、以下の問題を JGEX で記述したところ、Newclid で受理され、証明も出力できた。
---
- 2015（円に内接する五角形内の三角形の相似）
円 O の周上に 4 点 A,B,C,D を弧 AB=弧 BC=弧 CD となるようにとる。弧 BC を除く円周上に点 E を弧 AE=弧 ED となるようにとる。対角線 BD と CE の交点を F とする。三角形 FCD と三角形 ABE が相似であることを証明せよ。
JGEX:
e b c = iso_triangle e b c; o = circle o e b c; a = on_circle a o b, eqdistance a b b c; d = on_circle d
↪ o c, eqdistance d c c b; f = on_line f b d, on_line f c e ? simtri f c d a b e
- 2014（円に内接する四角形内の三角形の相似）
線分 BD を直径する円 O をとる。円 O 上に点 A をとり、点 A を含まない弧 BD 上に点 C をとる。点 A を通り線分 BC に平行な直線と線分 CD との交点を E、線分 BD との交点を F とする。三角形 ACD と三角形 FBA が相似であることを証明せよ。
JGEX:
b d = segment b d; o = midpoint o b d; a = on_circle a o b; c = on_circle c o d; e = on_line e c d,
↪ on_pline e a b c; f = on_line f a e, on_line f b d ? simtrir a c d f b a
---
## 以上の情報を参考にして、次の問題を Newclid で受理されるように JGEX で記述せよ。
- 2013（半円内の三角形の相似）
線分 AB を直径とし、点 O を中心とする半円がある。半円周上に点 D,E を弧 DE=弧 EB となるようにとる。線分 AE と BD の交点を F とする。三角形 AEB と三角形 BEF が相似であることを証明せよ。
```

このプロンプトに対する各生成 AI の回答は表 3 の通りである．残念ながら，すべての回答が不完全なものであった[11]．

各回答の評価値の根拠は次による．

- `eqdistance d e e b` とするべきところを `eqdistance e d d b` と間違った式を用いている．(GPT-4.1)
- `simtrir` とすべきところを `simtri` を使用している．(o3，GPT-4.1，Gemini)
- `eqdistance` の使用法[12]に誤りがある．(GPT-5 Thinking，Claude，Grok)

この結果から，生成 AI を利用した JGEX への変換は現時点で不完全といえる．

[11] この問題の正しい JGEX は，`"a b = segment a b; o = midpoint o a b; e = on_circle e o a; d = on_circle d o a, eqdistance d e e b; f = on_line f a e, on_line f b d ? simtrir a e b b e f"` である．

[12] `eqdistance x a b c` は，点 a,b,c が与えられたとき，xa=bc となるような点 x を生成する．

表 3: 生成 AI による変換結果

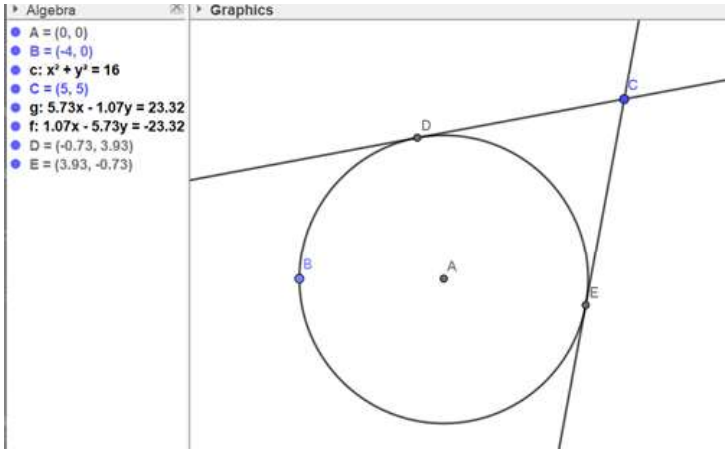
生成 AI	JGEX	評価
ChatGPT (o3)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o b, eqdistance d e e b; f = on_line f a e, on_line f b d ?  simtri a e b b e f`	50
ChatGPT (GPT-4.1)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o a, eqdistance e d d b; f = on_line f a e, on_line f b d; ?  simtri a e b b e f`	30
ChatGPT (GPT-5 Thinking)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o a, eqdistance d e e b; f = on_line f a e, on_line f b d ?  simtrir a e b b e f`	60
Claude (Opus 4.1)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o a, eqdistance e b d e; f = on_line f a e, on_line f b d ?  simtrir a e b b e f`	70
Grok (Grok 4)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o a, eqdistance d e e b; f = on_line f a e, on_line f b d ?  simtrir a e b b e f`	60
Gemini (2.5 Pro)	`a b = segment a b; o = midpoint o a b; d = on_circle d o a; e = on_circle e o a, eqdistance d e e b; f = on_line f a e, on_line f b d ?  simtri a e b b e f`	50

6.2 GeoGebra の利用

もう一つのアプローチとして，命題の「前提」の入力に GeoGebra ファイルを利用する方法を紹介する．

Step 1 問題名を tangents とし，`tests-exp/tangents` ディレクトリを作成する．

Step 2 GeoGebra で証明したい問題の図を作成し，`tangents.ggb` というファイル名で保存する．ここでは，ファイルの保存場所を `problems` ディレクトリとする．



Step 3 「結論」を JGEX 記法で記述し，`--goals` オプションで渡して，以下のように実行する．

```
$ newclid -o outputs/tangents ggb --file ./problems/tangents.ggb
↪ --goals "cong C D C E"
```

実行結果の証明ファイル `proof_steps.txt` と図ファイル `proof_figure.svg` は Step 1 で作成した `tests-exp/tangents` ディレクトリに出力される．

このアプローチは，JGEX 記法による命題の「前提」の記述を避けることができ，教育利用に適したものと言える．ただし，この機能は実験的な実装であり，すべての GeoGebra ファイルが Newclid に受理されるわけではない．今後のバージョンアップが期待される．

7　まとめと今後の課題

本稿では，Deductive database method を基盤とする初等幾何 AI である AlphaGeometry および Newclid の概要と，これらを高校入試問題の証明生成に適用した実験結果について述べた．特に，Newclid が JGEX 記法の整備や DDAR の高速化により，教育現場での利用に耐えうる実行の安定性と処理速度を備えていることを確認した．また，GeoGebra との連携機能により，命題記述の負担を軽減できる可能性も示した．

一方で，幾何 AI を学校教育へ実際に導入するためには，多くの課題が残されている．今後の研究課題として，以下を挙げる．

1. **Newclid と GeoGebra との連携の強化** GeoGebra ファイルのインポート機能の性能向上，ならびに生成した証明結果をステップ毎にアニメーション表示するための GeoGebra ファイルのエクスポート機能の追加が求められる．
2. **補助点の追加が必要な問題に対する検証** 本稿での実験で使用した問題はすべて補助点の追加なしで解けたが，入試問題では補助点の追加が不可欠な難問も多い．そのような問題に対応するために，AlphaGeometry の LLM と Newclid のヒューリスティック機能による補助点生成の検証が必要である．
3. **Newclid 用 LLM の開発** 現在の Newclid は補助点の追加を行う LLM を有していない．上の項目で述べた検証を行い，ヒューリスティック機能では不十分となれば LLM の開発が必要になる．その場合，AlphaGeometry が公開しているパラメータファイルの活用や，日本の中学校数学の問題体系に合わせた自前モデルの構築といった方向性が考えられる．また，日本語の問題文から，それに対応する JGEX へ変換を行う LLM も求められる．
4. **教育利用のための GUI の開発** JGEX 記法は強力である一方，一般の教師・生徒には難解である．命題入力を補助したり，証明結果を図を用いながら段階的に提示するわかりやすいインタフェースが不可欠である．
5. **日本の中学校での利用に特化したカスタマイズ** 学習指導要領に準拠し，日本の教科書の記述に沿った読みやすい証明を生成するようにしなくては授業で利用できない．また，学年毎に段階的に進行する教育課程に即した調整[13]が求められる．

[13] AlphaGeometry では，使用する定義と既知命題を `defs.txt` と `rules.txt` によって制御でき，教育課程に合わせたチューニングも可能である．

これらの課題の解決を目指すことで，幾何 AI は教育現場での証明学習支援ツールになり得ると考える．

謝辞

本研究は JSPS 科研費 21K12157 及び 25K15373 の助成を受けている．

参考文献

[1] Shang-Ching Chou, Xiao-Shan Gao, Jing-Zhong Zhang: A Deductive Database Approach to Automated Geometry Theorem Proving and Discovering, Journal of Automated Reasoning, vol. 25 (2000) pp.219–246.

[2] N. Baeta and P. Quaresma: Towards a Geometry Deductive Database Prover, Annals of Mathematics and Artificial Intelligence, vol.91 (2023) pp.851–863.

[3] Zheng Ye, Shang-Ching Chou, Xiao-Shan Gao: An Introduction to Java Geometry Expert, ADG2008, Lecture Notes in Artificial Intelligence, vol. 6301 (2011) pp.189–195.

[4] JGEX, `https://github.com/kovzol/Java-Geometry-Expert`

[5] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, Thang Luong: Solving olympiad geometry without human demonstrations, Nature, vol. 625 (2024) pp.476–482.

[6] Zheng Ye, Shang-Ching Chou, Xiao-Shan Gao: Visually Dynamic Presentation of Proofs in Plane Geometry – Part 1. Basic Features and the Manual Input Method, Journal of Automated Reasoning, vol. 45 (2010) pp.213–241.

[7] Vladmir Sicca, Tianxiang Xia, Mathïs Fédérico, Philip John Gorinski, Simon Frieder, Shangling Jui: Newclid A User-Friendly Replacement for AlphaGeometry with Agentic Support, arXiv preprint, `https://arxiv.org/abs/2411.11938`, 2024.

[8] JGEX definitions, `https://newclid.github.io/Newclid/manual/building_a_problem_setup/jgex/definitions.html`

[9] 柳川真里奈: AlphaGeometry による福岡県高校入試問題の解法について, 福岡教育大学卒業論文, 2025.