形式論理学に基づく演繹コーパスによる言語モデルに対する演繹推論能力の付与

森下皓文　森尾学　山口篤季　十河泰弘
日立製作所　研究開発グループ
terufumi.morishita.wp@hitachi.com

概要

言語モデルに演繹推論を学習させるための事例からなる演繹コーパスを提案する．先行研究では演繹規則が恣意的に選ばれていた為，演繹推論の核心を捉えられなかった．我々はこれを再考し，形式論理学に基く根拠ある演繹規則群を採用する．実験により，提案コーパスで学習した言語モデルが既存コーパスで学習した言語モデルより頑健な演繹推論能力を持つことを示す．更に後続研究のため，コーパス・学習済みモデル・ソースコードを公開する．

1　はじめに

論理的に物事を考える機械の実現は，人工知能黎明期からの夢である [1]．このゴールに対して近年，様々な論理推論ベンチマークが提案されている [2, 3, 4, 5]．これらは通常，優れた言語理解能力を持つ最先端の言語モデルを用いて取り組まれている．しかしながら，言語モデルをもってしても論理推論は難しいことが認識されつつある [6, 7, 8]．

言語モデルは，人間が書いた文章中の多くの良質な事例から，言語理解能力を帰納的に獲得した [9]．逆に考えると，言語モデルの論理推論能力の低さは，人間が書いた文章中に良質な論理推論の事例が十分数含まれていないことを示唆している．そうであるならば，良質な論理推論の事例を多く含むコーパスを生成し，言語モデルに学習させることで，論理推論能力を獲得することができるはずである．

このようなコーパスの 1 つとして，近年提案された RuleTaker [10] が挙げられる．RuleTaker は，自動生成された多段演繹証明を含む．各演繹証明は，与えられた事実の集合に対して演繹規則を複数回適用することにより，仮説を証明 (あるいは反証) する (図 1 中の “Deduction Instance” と同様)．演繹規則としては，$\forall x F(x) \rightarrow G(x), F(a) \vdash G(a)$ ($\vdash$ は “導出する” の意) 等の，含意に類する規則が用いられた．Artifical Argument Corpus [11] は RuleTaker 同様，自動生成された一段演繹証明からなるコーパスで，演繹規則として人手で選ばれたクリティカル・シンキングに役立つ規則，例えば対偶 $\mathcal{F} \rightarrow \mathcal{G} \vdash \neg\mathcal{G} \rightarrow \neg\mathcal{F}$ ($\neg$ は否定) が用いられた．これらの演繹コーパスはいずれも，最も普遍的な論理推論能力の一つである，演繹推論能力を獲得する機会を提供し得る．

しかしながら，これら演繹コーパスで採用された演繹規則は少数に限られ，また，恣意的であった．現実世界の複雑な演繹推論では多様な演繹規則が求められるため，再考する必要がある．

本研究はこの問題に答えることを目指す．まず，形式論理学 (Formal Logic) を訪れる (2 節)．形式論理学によれば，既存コーパスで使用された演繹規則も含め，妥当な演繹規則は無限に存在する．しかし，その中でも基本的な演繹規則群 (**公理系**) が存在し，公理系による多段演繹推論によって，その他の任意の演繹規則を導出できる（**完全性**）．よって，公理系による多段演繹推論は任意の演繹規則による多段演繹推論を模擬することができる．既存コーパスの演繹規則群はこの性質を持たないので，たとえそれらを獲得したとしても，他の多様な演繹規則を模擬できない．我々はこの点を改め，演繹規則群として公理系を採用したコーパス **FLD** (**F**ormal **L**ogic **D**eduction, 3 節) を構築し，言語モデルに公理系を用いた演繹推論能力を獲得させることを目指す．

次に **FLD** の有効性を実験的に確かめる (4 節-6 節)．複数種のベンチマークにて，**FLD** で学習された言語モデルが既存コーパスで学習された言語モデルより頑健な演繹推論能力を持つことを示す．

最後に，**FLD** コーパス・ソースコード・学習済みモデルを公開する[1)]．**FLD** コーパスは既存コーパスより挑戦的なのでベンチマークとして有用である．

1) https://github.com/hitachi-nlp/FLD

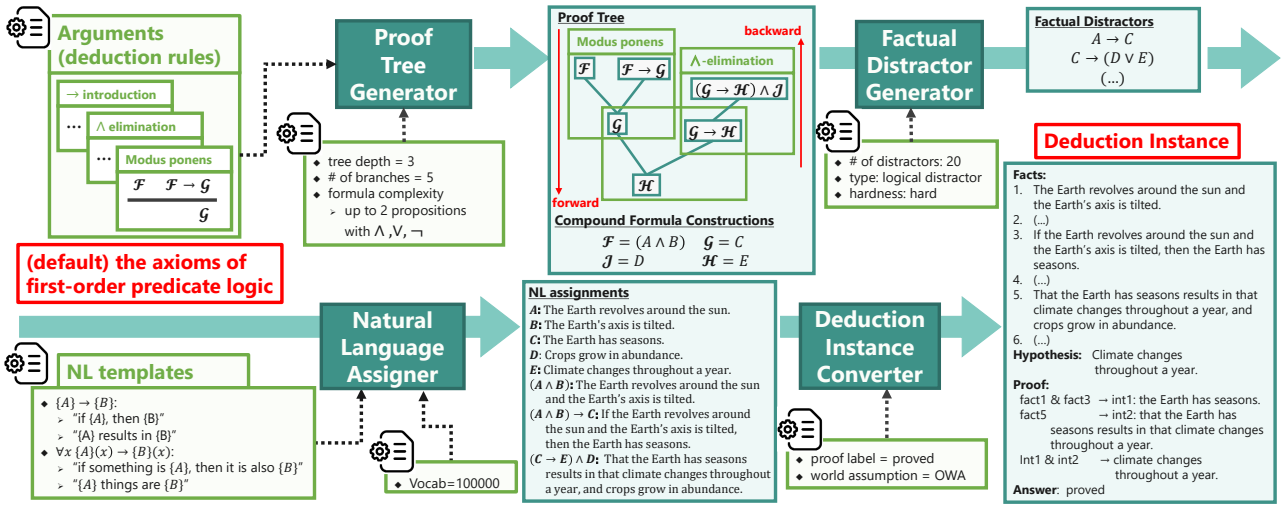


図 1: **FLD** は一階述語論理の公理系に基づく演繹推論事例を生成することを目的とする．また，将来の解析研究において様々なパターンのコーパスを生成することを見据え，多様なオプションが用意されている．

2 事前知識: 形式論理学

まず，以下の一段演繹ステップを考える:

$$\frac{\text{地球が太陽を周回している} \qquad \text{地球が太陽を周回していれば，地球には四季がある．}}{\text{地球には四季がある．}} \tag{1}$$

このステップは 2 つの前提 (棒の上) から結論 (棒の下) を導いており，記号を用いて抽象化できる:

$$\frac{\mathcal{F} \qquad \mathcal{F} \to \mathcal{G}}{\mathcal{G}}\ \text{modus ponens} \tag{2}$$

この形式の演繹ステップを modus ponens と呼ぶ．

Modus ponens 以外にも，演繹ステップは無数に存在する．例えば三段論法 (syllogism) は有名である:

$$\frac{(\mathcal{F} \to \mathcal{G}) \wedge (\mathcal{G} \to \mathcal{H})}{\mathcal{F} \to \mathcal{H}}\ \text{syllogism} \tag{3}$$

無論，妥当でないステップ[2)] を考えることもできる:

$$\frac{\mathcal{F} \qquad (\mathcal{F} \vee \mathcal{G})}{\mathcal{G}} \tag{4}$$

上記の例から分かるように，演繹推論とは，特定の規則に従って前提から結論を導き出す思考形態と定義できる．形式論理学では，このような演繹規則のことを**論証**と呼ぶ．従って，(2)–(4) は全て形式論理学における論証である．なお，$\mathcal{F}$ や $\mathcal{G}$ などは，$\mathcal{F} = \neg(A \vee \neg B)$ など (A, B は原子論理式)，任意の複合論理式でよい．更に，前提・結論に現れる論理式は無限パターン考えることができるため，論証は (妥当でないものも含めて) 無限に存在する．

2) 前提が全て真 (=1) だが結論が偽 (=0) となるような真理値割り当てが存在するステップ (論証) のこと．表 5b 参照．

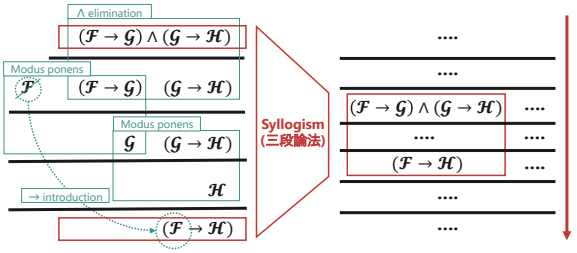


図 2: 公理系を用いた多段演繹推論．**(左):** 三段論法の導出 **(右):** ステップ数を大く取ることで，三段論法を用いた多段推論を模擬できる．

次に多段演繹推論を考える．図 2(左) から分かるように，三段論法はより「原子的な」論証による多段演繹推論によって導出できる (他の例として，先行研究で使用された論証の導出を図 4 に示す)．実は，形式論理学では**公理系**と呼ばれる原子的な論証群（図 3a）が存在し，以下のことが知られている

定理 1 (一階述語論理[3)] の完全性 Gödel, 1930)**.** 全ての妥当な論証[4)] は，公理系による多段演繹推論によって導出できる．また，公理系による多段演繹推論によって導出された論証は全て，妥当である．

ここに我々は，形式論理の核心，即ち，公理系を用いた多段演繹推論に至った．完全性により，妥当な論証は全てこの方法に導出でき，かつ，この方法によって導出される (無限種類の) 論証は全て妥当なのである．また，図 2(右) に例示するように，**公理系を用いた多段演繹推論は，原理的に，任意の論証を用いた多段演繹推論を模擬できる**ことになる．

3) 本論文は一階述語論理に議論を限る．
4) 前提が全て真 (=1) となるような任意の真理値割り当てにおいて，結論も必ず真 (=1) となる論証のこと．表 5a に例示．

3 形式論理演繹コーパスの生成

公理系による多段演繹推論事例を生成するためのフレームワーク **FLD** (**F**ormal **L**ogic **D**eduction) を提案する．直感的な理解のために図 1 を参照のこと．参考として，実際に生成された事例を図 5 に示す．

ランダム演繹による証明木生成　RuleTaker は，論理式群をランダムに生成し，それらにソルバーを走らせて偶発的に生じた演繹関係を特定し，証明木として用いた．しかしこの方法は外部のソルバーに依存するため，証明に用いる論証群を指定できない．

そこで我々は，ユーザが指定した論証群を用いてランダムな演繹推論を行うことにより証明木を生成するモジュール (図 1 中の “Proof Tree Generator’) を開発した．このモジュールは，前向き・逆向きの各ステップ（図中 “forward” と “backward”）において，論証をランダムに 1 つ選び，現在の証明木に結合する．証明木のノードの複合論理式（$\mathcal{F}$ や $\mathcal{G}$）は任意である（2 節）ため，原子論理式 (A や B 等) と論理演算子 $\wedge, \vee, \neg$ を用いてランダムに生成する[5]．

負例事実群の生成　現実の論理推論では，事実群は与えられるのではなく，検索システム等によって取得しなければならないので，ノイズとなる事実 (負例事実) が含まれる．これを模擬した負例事実群を生成する (図 1 中の “Factual Distractor Generator”)．負例事実は，正例事実と類似する論理式 (例えば正例 $(A \wedge B) \rightarrow C$ に対して負例 $A \rightarrow C$) 等を用いる．

自然言語の割り当て　各論理式に，自然言語を割り当てる（図 1 の “Natural Language Assigner”）．まず，各論理式ごとに，事前に用意された以下のようなテンプレートからランダムに 1 つ選択する:

$A \rightarrow B$: “If A, then B.”, “A leads to B.”

$F(a) \rightarrow G(b)$: “If a F, then b G.”, “When a F, b G.”

A, B, F, G, a, b のような要素には一定の文法制約の下，語彙からランダムに構築した言明を割り当てる:

A : “an Earthquake occurs”　B : “the year ends”

F : “run” G : “answer” a : “the hamburger” b : “Peter”

ランダムである理由は，演繹推論の妥当性は論理式の記号形式にのみ依存し，その内容は依存しない[6]ので，論理式への内容割り当てはいかなるものも許容されなければならないからである．

演繹推論事例の生成　事例を生成する (図 1 中の “Deduction Instance Converter”)．事例は，事実群・仮説・証明・回答ラベル，から構成される．回答ラベル「証明された」の場合: (i) 証明木の根ノードを仮説として (ii) 証明木の葉ノードと負例事実を事実群として (iii) 証明木の中間ノード群を証明として，用いる．回答ラベル「反証された」の場合，根ノードの否定文を仮説とすることにより，証明木によって仮説が反証されるようにする．回答ラベル「不明」の場合，葉ノードの一部をランダムに削除することで，仮説を証明も反証もできないようにする．

5) ただし先行研究と同様，原子論理式の数は 3 までとする．

6) 「A と $A \rightarrow B$ が成り立っている時には，B も成り立つ」という形の推論は，A の**実際の内容にかかわらず**妥当である．

表 1: 本研究で用いたコーパス (詳細は付録 A.1)．比較対象となるコーパス間の条件を揃えるため，複数のコーパスを構築した．“RT” は RuleTaker[12]．RT.BE はテストセットのみなので評価のみに使う．

コーパス名	論証群	木の深さ	木の深さ分布
RT (“D0-D3”)	含意	1 ~ 3	skewed (浅い方が多い)
RT.PR (“ParaRules”)	含意	1 ~ 5	
RT.BE (“Birds-Electricity”)	含意	1 ~ 3	
sFLD-impl	含意	1 ~ 3	
sFLD-crit	クリティカルシンキング	1	
sFLD-axiom	公理系	1 ~ 3	
RT.D5 (“D0-D5”)	含意	1 ~ 5	flat (一様)
FLD.D5	公理系	1 ~ 5	
FLD-impl	含意	1 ~ 3	
FLD	公理系	1 ~ 3	
FLD★	公理系	1 ~ 8	

4 実験

FLD で学習された言語モデルの論理推論能力を，他の演繹コーパスで学習された言語モデル，またこのような学習をしない言語モデルと比較する．

4.1 モデル

演繹タスクでは，与えられた事実の集合から与えられた仮説を証明・反証する証明系列を生成することが求められる．このために，[8] の証明器を用いる．これは T5[13] に基づく生成モデルで，“fact1 & fact3 -> The Earth has seasons.’ のように，選ばれた前提群と導出される結論からなる証明ステップを 1 つづつ生成していく．学習の詳細は付録 A.2 に示す．

4.2 ベンチマーク

演繹コーパス　表 1 中のあるコーパスで証明器を学習し，別のコーパスで性能を計測する．証明器が頑健な演繹推論能力を獲得していれば少ない事例のみで転移するはずなので，ターゲットコーパスでの

表 2: 各コーパスにおいて，全データを用いて fine-tuning された証明器の証明正解率．

RuleTaker		FLD	
RT	RT.PR	**FLD**	**FLD⋆**
92.4	93.9	66.4	37.4

表 3: コーパス間で few-shot 転移した証明器の証明正解率．これらコーパスは用いた論証群が異なる．

		Source corpus					
			RuleTaker		sFLD		
		T5	RT	RT.PR	-impl	-crit	**-axiom**
Target corpus	RT	70.1	92.4	91.3	76.2	74.4	76.7
	RT.PR	64.3	91.3	93.9	73.4	67.5	72.9
	RT.BE	56.1	88.3	88.2	75.2	79.4	85.0
	sFLD-impl	58.4	66.7	65.9	82.2	67.3	80.7
	sFLD-crit	71.9	77.7	77.2	87.8	94.0	93.6
	sFLD-axiom	54.7	54.5	54.5	67.9	63.7	79.1
	avg.	62.6	78.5	78.5	77.1	74.4	**81.3**

学習は few-shot (1%=300 事例) で行う．性能指標として証明正解率 (proof accuracy) [12] を用いる．

EntailmentBank (EB) [14] 証明木は人手で作成されており，また，各証明ステップは厳密な論理ステップだけでなく含意的なステップも許される．よって，より現実的なシーンでの演繹推論能力を測定できる．上記のように，証明ステップの性質が演繹コーパスとは異なるため，少数事例での転移は望めない．よって，演繹コーパスで学習した証明器を EB 全データで fine-tuning して性能を計測する．性能指標として証明正解率 (“AllCorrect” [14]) を用いる．

5 言語モデルは論理を解けるか?

まず，言語モデルが各演繹コーパスをどれくらい解けるか示しておく (表 2)．証明器は RuleTaker では良好な性能を発揮しているが，FLD での性能は悪い．FLD が難しい理由の一つは，以下のように考えられる (詳細は付録 A.3)．RuleTaker は含意に類する少数の論証 (図 3b) のみを使用するのに対し，FLD は公理系 (図 3a) に含まれる様々な論証を使用する．よって，FLD は証明木生成の際に，木の各レベルにおいて選択可能な論証の種類が多い．証明木は，各レベルにおいて選択された論証の組み合わせによって構成されるので，ありうる木のパターンは組み合わせ的に大きくなる (大雑把に $O(|\mathcal{A}|^d)$ パターン．ここで $|\mathcal{A}|$ は各レベルにおける論証の選択肢数，d は木の深さ)．結果，FLD は RuleTaker よりも多様な パターンの証明木を含んでおり，難易度が高いと考えられる．実際，最も木が深い **FLD⋆** は極めて難易度が高かった．

表 4: EntailmentBank における証明器の証明正解率．

		EntailmentBank		
		Task1	Task2	Task3
Source corpus	T5	36.8	31.2	6.2
	RT.D5	40.6	33.2	7.4
	FLD.D5	**40.9**	**33.8**	**8.2**

6 FLD は有効か?

6.1 論証への汎化

演繹推論能力の獲得において最も重要なのは演繹規則（論証）の獲得であるので，演繹コーパスがうまく言語モデルに論証を教えられるかを調べる．

表 3 から分かるように，論証群が異なるコーパス間の転移に関して，sFLD-axiom で学習された証明器が最も平均性能が高かった．これは，コーパス毎の結果から分かるように，論証群が異なる他のコーパスに対して最も頑健に転移したことによる．

この頑健性こそがまさしく，形式論理に基づく演繹の美徳を示している．sFLD-axiom は論証群として公理系を用いているので，完全性により，他コーパスの論証による多段演繹推論を全て模擬することができる（参考に図 4 に例を記す）．即ち，**公理系から得られる演繹推論能力は，様々な論証に汎化する**のである．一方，他コーパスの論証群は完全性のような性質を持たず，論証への汎化は得られない．

6.2 より複雑なタスクへの汎化

表 4 に，EB での性能を記す．EB は深い木 (最大で 10 以上) を含むので，深い木を含む演繹コーパスから転移させた．**FLD による学習は，より実世界に近い複雑な論理推論にも汎化する**ことが分かる．参考に表 8 に証明器の出力例を示す．「否定 (¬) の意味」「次の結論を導くために必要十分な前提のみを選ぶ」「前提から論理的に導ける事項のみ結論に含める」など，論理の基本が学習されている．

7 おわりに

形式論理学に基づく演繹コーパスを提案し，有効性を実験的に示した．今後は，事実群の獲得も求められる演繹推論や仮説推論など，より高度な推論に対して，本アプローチを昇華させていく．

謝辞

本研究は，計算機リソースとして，産総研の AI 橋渡しクラウド (ABCI) を用いた．また，日立製作所の清水正明氏には，社内大規模計算機環境の維持管理をして頂いた．両者ともに，お礼申し上げる．

参考文献

[1] John W. McCarthy. Programs with common sense. In *Proc. Tedding Conf. on the Mechanization of Thought Processes*, pp. 75–91, 1959.

[2] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[3] Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1930–1940, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[4] Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4658–4664, Florence, Italy, July 2019. Association for Computational Linguistics.

[5] Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 8713–8721, 2020.

[6] Amanda Askell. Gpt-3: Towards renaissance models. *Daily Nous Blog: Philosophers On GPT-3*, 2020.

[7] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[8] Kaiyu Yang, Jia Deng, and Danqi Chen. Generating natural language proofs with verifier-guided search. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[10] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

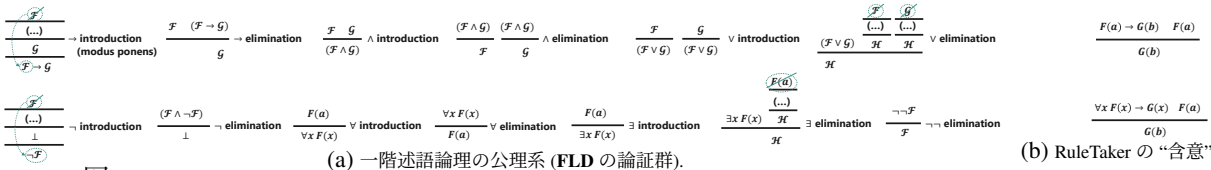
[11] Gregor Betz, Christian Voigt, and Kyle Richardson. Critical thinking for language models. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pp. 63–75, Groningen, The Netherlands (online), June 2021. Association for Computational Linguistics.

[12] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, Online, August 2021. Association for Computational Linguistics.

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, Vol. 21, No. 140, pp. 1–67, 2020.

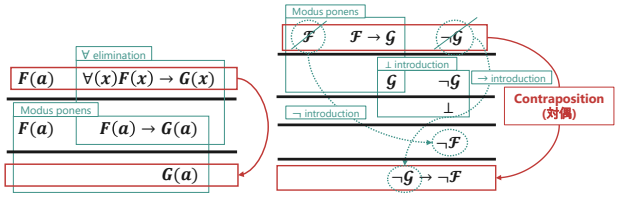
[14] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7358–7370, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

A　参考情報



(a) 一階述語論理の公理系 (**FLD** の論証群).　(b) RuleTaker の “含意” 論証群.

図 3: 本研究や先行研究で用いられた論証を示す．ただし，Artificial Argument Corpus で用いられた論証は [11] の Figure.1 参照のこと．



(a) RuleTaker の論証.　(b) Artificial Argument Corpus の論証.

図 4: 公理系を用いた多段演繹推論による，先行研究の論証の導出例.

表 5: 論証の前提 ($\mathcal{P}_i$)・結論 (C) に関わる真理値表．青色は，前提が真 (=1) の場合には結論も真．赤色は，前提が真なのに結論が偽 (=0).

(a) 妥当な論証 (3) (syllogism).

$\mathcal{P}_1 = (\mathcal{F} \to \mathcal{G}) \wedge (\mathcal{G} \to \mathcal{H})$,
$C = \mathcal{F} \to \mathcal{H}$.

$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{P}_1$	C
1	1	1	1	1
1	1	0	0	0
1	0	1	0	1
1	0	0	0	0
0	1	1	1	1
0	1	0	0	1
0	0	1	1	1
0	0	0	1	1

(b) 妥当でない論証 (4).

$\mathcal{P}_1 = \mathcal{F}, \mathcal{P}_2 = \mathcal{F} \vee \mathcal{G}$,
$C = \mathcal{G}$.

$\mathcal{F}$	$\mathcal{G}$	$\mathcal{P}_1$	$\mathcal{P}_2$	C
1	1	1	1	1
1	0	1	1	0
0	1	0	1	1
0	0	0	0	0

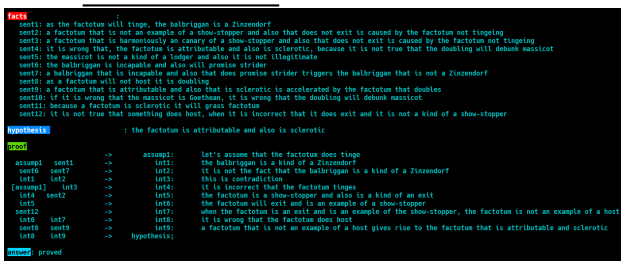


図 5: **FLD** の事例 (本例では証明中に背理法が用いられている).

A.1　コーパス詳細

“skewed” な深さ分布のコーパスは，深い木よりも浅い木をより多く含む．これは，公平な比較のために RT(D0-D3) に合わせた分布である．一方，“flat” は一様分布を持つ．全てのコーパスの事例は，負例事実群を最大 20 程度まで，ランダムに含んでいる．

RuleTaker: 他のコーパスとの条件を合わせるため，回答ラベルの分布が一様に・学習セットサイズが 30k となるように，一部をサンプリングした．“含意” 論証群は図 3b．その他詳細は [12].

FLD: オプション機能を用いて，様々な条件のコーパスを作成した．“公理系” 論証群は図 3a を，“含意” 論証群は図 3b を，“クリティカル・シンキング” 論証群は，[11] の Figure.1 記載のものを，用いた．クリティカル・シンキング論証は粒度が大きく多段に組み合わせるのが難しいので，木の深さは 1 とした．回答ラベルの分布は一様である．学習セットサイズは 30k，バリデーション・テストセットは 1k である．その他詳細はウェブ[a]参照のこと．

A.2　証明器学習の詳細

表 6: 証明器学習のハイパーパラメータ.

	演繹コーパス実験		EntailmentBank 実験	
	Source	Target	Source	Target(EB)
transformer model	T5-base	T5-base	T5-large	T5-large
# dataset instances	30000	300	30000	1313
steps	20000	2000	10000	10000
learning rate	1e-4	1e-4	1e-4	1e-4
learning rate scheduler	AdamW	AdamW	AdamW	AdamW
warmup steps	1000	500	1000	1000
batch size	64	64	64	64
gradient clipping	0.5	0.5	0.5	0.5

5 節, 付録 A.3 での “全データでの fine-tuning” では，20000 ステップ学習している．EntailmentBank は難易度が高いので，[8] の verifier も用いている．その他，学習の詳細は [8] を参考のこと．

A.3　各コーパスの難易度の詳細分析

表 7: 各コーパスで，全データで fine-tuning された証明器の証明正解率.

木の深さ分布=“skewed”			木の深さ分布=“flat”		
RT	RT.PR	sFLD-impl	FLD-impl	**FLD**	**FLD⋆**
92.4	93.9	82.2	74.6	66.4	37.7

“skewed” コーパス群において，sFLD-impl は RuleTaker より難易度が高い．論証群や木の深さ等の諸条件は同じなので，その他の実装の詳細に起因すると考えられる．例えば，FLD の負例事実は正例事実と混同しやすいように設計されており，自然言語割り当てはランダム言明によって極めて多様である (3 節). FLD-impl は sFLD-impl と同様の “含意” 論証を用いているが，より難易度が高い．これは，“flat” コーパス群は skewed に比べて，深い証明木をより多く含む (付録 A.1) からであろう．**FLD** の難易度が FLD-impl より高いのは，公理系からなる多様な論証をを用いるので，多様なパターンの証明木を含むからであろう (5 節). **FLD⋆** の極めて難しい理由は，証明木の深さが最も深く，指数関数的に多様なパターン ($|\mathcal{A}|^d$) の証明木を含むからであろ．

[a] https://github.com/hitachi-nlp/FLD

表 8: FLD.D5 での学習によって，ベースライン (T5) の誤った証明ステップが治った例.

	選ばれた前提	導出された結論	T5 の誤り
FLD.D5	1. fossils are formed by the remains of living things 2. rock is a kind of nonliving thing	rocks cannot form fossils	前提が誤っている．おそらく，否定 (¬) の意味を捉えられていないことに起因する．
T5	1. fossils are formed by the remains of living things 2. cannot means not be able to	rocks cannot form fossils	
FLD.D5	1. force causes the speed of an object to increase / to decrease 2. an increase is a kind of change 3. a decrease is a kind of change	force causes the speed of an object to change	前提が足りない．おそらく，論理積 (∧) の意味を捉えられていないことに起因する．
T5	1. force causes the speed of an object to increase / to decrease 2. an increase is a kind of change	force causes the speed of an object to change	
FLD.D5	1. the milky way is made of stars 2. light year can be used to measure the distance between stars	light year can be used to measure the distance between the stars in milky way	前提が足りず，生成された結論が前提から論理的に導かれる以上の情報を含む．
T5	1. the milky way is made of stars	the distance between the stars in milky way is light years	