卒業論文

LLMマルチエージェント弁証法的推論による質問応答

指導教官　　村上 陽平 教授

立命館大学情報理工学部
先端社会デザインコース

増尾 柚希

2026年1月1日

LLM マルチエージェント弁証法的推論による質問応答

増尾 柚希

内容梗概

大規模言語モデル（LLM）を用いた質問応答システムは，近年，さまざまなアプリケーションや情報システムにおいて実用化が進んでいる．さらに，LLM に複数の価値観や立場を付与し，それらを相互に対話させるマルチエージェント型の手法が提案され，模擬的な議論や意思決定を生成する研究が活発化している．多文化共生社会においては，質問応答は単なる情報検索にとどまらず，異なる文化的・宗教的・倫理的背景をもつ人々の相互理解を促進する役割を担うことが期待されている．

しかしながら，既存の LLM ベースのマルチエージェント対話は，対立する立場の優劣を決定するディベート的構造に依存するものが多く，最終的な結論も表層的な妥協案にとどまる傾向がある．たとえば,「学校清掃を生徒が行うべきか職員が行うべきか」といったトピックにおいて，現状の LLM は「日常的な清掃は生徒が担い，専門的な清掃は職員が行う」といった折衷的回答を生成しがちである．しかし，このような回答は，価値観の根底にある対立を踏まえ，清掃行為そのものを教育活動として再定義し，生徒と職員の共同実践として位置づけ直すといった止揚的な解決を十分に表現しているとは言い難い．

そこで本研究では，多様な価値観の対立を内包するトピックを対象とし，LLM を用いたマルチエージェント対話によって，新たな価値・制度・概念の再構成を導く弁証法的対話プロトコルを提案する．本手法では，従来の形式論理に基づく弁証法的議論および統合手続きを参照しつつ，それらの形式的プロセスを自然言語上で再現可能な段階的対話モデルとして再設計する．

本手法の実現にあたり，主に以下の二つの課題に取り組む．

議論の収束

質問応答システムに適用される推論プロトコルは，その推論過程が必ず終了することを保証する必要がある．弁証法的対話においては，エージェント同士の議論を収束させることで，推論過程の終了が担保される．記号論理に基づく従来の議論では，知識ベースや推論規則が有限かつ明示的に定義されるため，議論は必ず収束する．一方，自然言語を用いる LLM エージェント同士の対話では，意味的に同一または類似した主張や反論が繰り返し生成される可能性があ

り，プロンプトによる制御のみで議論の収束を保証することは困難である．

高次な止揚論証の生成

統合（止揚）は，単なる折衷ではなく，対立する価値観の前提を再構成することによって，より高次な概念や視点を創出することを意味する．しかし，自然言語に基づく対話において，止揚が成立する条件や構築過程は明確に定義されておらず，形式的定義のみから意味的に妥当な統合であるかを判断することは難しい．

本研究では，既存の論理ベースエージェントの議論に則って，LLM エージェントの発話を構造化されたスキーマ上で行うように制御した．加えて，反論の手法も具体的に定義し，構造化された論証の中で明確な反論対象を提示させるようにした．したがって，エージェントの発話は自然言語でありつつも，何を根拠に述べているのか，何が結論で，何が前提なのかが明確であり，これらを用いて議論を行うことで，議論の論理構造を把握することが容易になる．これらにより，議論の中での表層的な反論の応酬を防ぎ，LLM エージェント自身のスタンスに反論する根拠や事実がなければ，きちんと反論不能であると判断することを可能にした．この判断を可能性は，論理ベース同様に議論の収束を保証する．止揚論証の構築においては，形式的な統合手続きを自然言語プロンプトとして再構成し，前提の拡張，論証の特化，および上位概念の導出を段階的に実行させることで，価値観の前提そのものを再構成した止揚論証を生成できることを確認した．

さらに，本研究では提案プロトコルを組み込んだ弁証法的推論に基づく質問応答システムを実装し，既存の止揚論証例との比較を行った．その結果，以下の貢献を確認した．

議論の収束

提案した対話プロトコルは，既存研究同様の弁証法議論を展開し，LLM エージェントが自身のスタンスに基づいて，論理的な判断，発話をすることを可能にした．その上で，LLM エージェントの自由発話では困難な議論の収束を可能にした．

高次な止揚論証の生成

既存の対話例を再現した実験において，生成された最終的な止揚論証が意味的に一致することを確認した．

LLM Multi-agent Dialectical Reasoning for Question Answering

Yuzuki MASUO

Abstract

Question answering systems based on large language models (LLMs) have recently been widely deployed in various applications and information systems. Furthermore, multi-agent approaches that assign multiple values or stances to LLMs and allow them to interact have been proposed, and research on generating simulated discussions and decision-making has become active. In multicultural societies, question answering is expected not only to retrieve information but also to promote mutual understanding among people with different cultural, religious, and ethical backgrounds.

However, many existing LLM-based multi-agent dialogue systems rely on debate-like structures that determine the superiority of opposing positions, and their final conclusions often remain superficial compromises. For example, on topics such as whether school cleaning should be performed by students or staff, current LLMs tend to generate conciliatory answers that fail to address and reconstruct the underlying value conflicts in a dialectical manner.

To address this issue, this study proposes a dialectical dialogue protocol that targets topics involving conflicts among diverse values and derives the reconstruction of new values, institutions, and concepts through multi-agent dialogue using LLMs. Referring to conventional dialectical argumentation and integration procedures based on formal logic, the proposed method redesigns these processes as a step-by-step dialogue model reproducible in natural language.

In implementing this approach, we mainly address the following two challenges.

Convergence of Argumentation

Reasoning protocols applied to question answering systems must guarantee termination. In dialectical dialogue, termination is achieved by converging discussions between agents. In symbolic logic-based argumentation, convergence is guaranteed because knowledge bases and inference rules are finite and explicit. In contrast, in natural language dialogue between LLM agents, semantically

similar claims and rebuttals may be repeatedly generated, making convergence difficult to ensure through prompt-based control alone.

Generation of Higher-Order Dialectical Synthesis

Dialectical synthesis is not mere compromise but the creation of higher-order concepts or perspectives through the reconstruction of conflicting premises. However, in natural language-based dialogue, the conditions and construction process of synthesis are not clearly defined, making it difficult to assess semantic validity based solely on formal definitions.

In this study, following existing logic-based argumentation frameworks, we constrain LLM agents ' utterances to structured schemas. Rebuttal methods are explicitly defined so that rebuttal targets are clearly indicated. Although utterances are expressed in natural language, their premises, conclusions, and grounds are made explicit, facilitating analysis of the logical structure. This prevents superficial rebuttal exchanges and enables agents to judge when rebuttal is no longer possible due to the absence of relevant grounds, thereby ensuring convergence. For dialectical synthesis, formal integration procedures are reformulated as natural language prompts, enabling premise expansion, argument specialization, and higher-level concept derivation to be performed step by step.

Furthermore, we implemented a question answering system based on dialectical reasoning incorporating the proposed protocol and compared its outputs with existing examples of dialectical synthesis. The results confirm the following contributions.

Convergence of Argumentation

The proposed dialogue protocol enables LLM agents to engage in dialectical discussions comparable to existing studies and ensures convergence of argumentation.

Generation of Higher-Order Dialectical Synthesis

In experiments reproducing existing dialogue examples, the generated final dialectical syntheses were found to be semantically consistent with those examples.

LLM マルチエージェント弁証法的推論による質問応答

目次

第1章　はじめに

近年，大規模言語モデル（LLM）を用いた質問応答は，さまざまなアプリケーション，システムで実用化が進んでいる．同時に世界では，多文化共生がますます進んでおり，さまざまな価値観を持つ人々が共に暮らす社会ができている．質問応答には情報検索としての役割に加えて，そのような社会に適応するために，異なる価値観を持つ人々の相互理解を促進する，合意可能な回答を生成する役割も求められている．この背景に対し，複数の役割や価値観を付与したLLMエージェント同士の対話により結論を導くマルチエージェント議論の枠組みが提案されている．しかし既存手法の多くは，対立する立場をもとに勝敗を決めるディベート的議論，あるいは双方の主張を並列した表層的な折衷案に収束しやすい．その結果，対立を保持したまま共通項を抽出し，新たな価値・制度・概念の再構成として解決を導く止揚的な回答生成には至りにくいという課題がある．

そこで本研究では，記号論理で形式化された弁証法的推論の枠組みに基づき，複数のLLMエージェントが論証構築・反論・統合を段階的に行う対話プロトコルを提案する．具体的には，Kido, Kurihara らの弁証法議論形式 [1] を参照し，記号論理で示された対話プロトコルと統合の手法を，2体のLLMエージェントを用いた弁証法的推論による質問応答システムに適用する．プロトコルは論証構築フェーズ，反論フェーズ，統合フェーズから構成され，論証構築とそれに対する反論は，2体の対話エージェントによって相互に行われる．

本提案手法を実現するにあたり，取り組むべき課題は以下の2点である．

議論の収束

LLMの自由発話による議論は，発話の役割や論点が明示的に制約されないため，話題の逸脱や同一主張の反復が生じやすく，対立が解消されないまま議論が循環するという問題がある．また，どの主張がどの前提に基づき，どの反論によって否定されたのかが不透明となり，収束の判定が困難である．本研究ではこの問題に対し，発話の役割を明示した構造化メッセージスキーマを導入し，各ターンを結論と最小限の前提からなる論証として表現する．さらに，反論が生成不可能となった時点を終了条件として定義することで，議論の進行を形式的に管理し，無限ループを抑制する．これにより，議論の可視性を保ちながら収束性を担保する．

高次な止揚論証の生成

Kido, Kurihara[1] によって提案された統合のアルゴリズム（Reasoning Method）は，記号論理に対する操作は可能であるものの，自然言語に対する操作は不可能であり，これをそのまま用いて止揚論証を構築することは困難である．本研究では，統合のための論証スキーマも提案した上で，Reasoning Method を意味的に解釈し，指示プロンプトに反映することで，LLM 上で一般化（generalization）の操作を擬似的に実装する．これにより，対立する論証の前提構造を再編成し，単なる折衷ではなく，上位概念に基づく高次な止揚論証の生成を可能とする．また，Kido, Kurihara が実際に行っていた対話例を本提案システム上でも実行し，同様の止揚論証が得られることを確認した．

以下，本論文では 2 章において関連研究を示した上で本研究の位置付けを行い，3 章では実際に提案するシステム，プロトコルについて詳細に説明する．続いて 4 章では評価として，Kido, Kurihara らの対話例を実際に行わせ，5 章でそれらを評価した．最後に，6 章で評価結果から考えられる考察と，本研究の将来的な展望を述べる．

第2章　関連研究

2.1　記号論理に基づく弁証法的議論とその限界

対立する主張を形式的に扱う枠組みとして，弁証法議論に関する研究が行われてきた．

Sawamura，Umeda ら [2][3] は，Prakken, Sartor ら [4] の拡張論理プログラミングに基づき，弁証法議論の枠組み（Computational Dialectics）を提案している．この枠組みでは，論証の構築と，相手の論証に対する攻撃（rebut, undercut）を段階的に行い，複数のエージェントが対話を通じて弁証法的に統合を行う議論フレームワークを形式定義している．ただし，Sawamura，Umeda らは文献の中で，具体的な止揚論証の導出方法は”a sort of oracle”であると述べており，形式定義される止揚論証の意味的実体や，その構築方法は未実装であった．

これに対して，Kido, Kurihara ら [1] は，Specialization および Generalization によって，形式定義された止揚論証を構築する手法を提案した．この手法により，対話を通じて双方の立場を包含する統合を行うための形式的操作が明確化された．

一方で，これらの手法はいずれも記号論理による知識表現を前提としており，議論に用いられる事実や規則，およびそれらの組合せは，あらかじめ定義された知識ベースに強く依存する．したがって，これらの手法は結論ありきの議論設計になっていることが多く，さまざまな議題や価値観の衝突に対する応用は困難である．また，形式的に導出された止揚論証が，どのような意味的解釈を持ち，どのような制度や価値の再構成を示すのかは，論理式そのものからは直接的に読み取ることができず，依然として課題として残されている．

2.2　大規模言語モデルによる弁証法的議論とその限界

そこで近年では，大規模言語モデル（LLM）を複数のエージェントに分担させ，異なる視点から推論を進めるマルチエージェント対話が注目されている．

Anghel ら [5] は，複数の LLM を用いて弁証法的推論を模倣する枠組みを提案し，(1) 主張（thesis）の生成，(2) 反論（antithesis）の生成，(3) 統合（synthesis）の生成という三段階の推論フローを設計している．同手法では，主張，反論，そして統合を行うエージェントを別々に定義し，自由発話によってそれら構築する．さらに生成された推論過程や最終的な統合結果に対して，別の評価エージェ

ントがルーブリックに基づく評価を行う点に特徴がある．

一方で，この手法は各エージェントが発話を一度だけしか行わず，議論を行うことはしていない．これは，形式定義された弁証法議論には則っておらず，あくまで擬似的に弁証法的な推論を行ったに過ぎず，対立するエージェントの主張点を明確にした上で統合を行うことはしていない．また，このシステムを多ターンに展開するとしても，自由発話によって構築された論証では，各発話の役割や攻撃関係，および前提の継承関係が構造的に管理されないため，論点の逸脱や同一主張の反復が生じやすく，議論の深化や収束が保証されない．その結果，対立の整理は行われても，前提の再編成や新たな概念形成を伴う止揚には至らず，表層的な対立調整にとどまる可能性が高い．

以上より，止揚（統合）を弁証法的に妥当な形で生成するためには，統合に先立って多ターンの議論を通じて争点を段階的に明確化し，各主張の前提構造と，それらの前提間の衝突関係を，攻撃関係および依存関係として構造的に表現・管理する枠組みが必要である．

これらの関連研究から，記号論理に基づく弁証法議論は，攻撃関係や統合操作を形式的に定義できる一方で，知識ベースへの強い依存や，止揚論証の意味的解釈の困難さといった課題を抱えていることが分かる．一方，大規模言語モデルを用いたマルチエージェント対話は，自然言語による柔軟な推論や多様な視点の導入を可能にするが，議論構造や攻撃関係が明示的に管理されないため，弁証法的に妥当な止揚へと収束させることが難しい．

そこで本研究では，記号論理に基づく弁証法議論の構造的厳密性と，大規模言語モデルの持つ自然言語理解・生成能力を統合することを目的とし，弁証法的推論の進行規則を明示的に定義した「弁証法プロトコル」に基づく質問応答システムを提案する．次章では，本研究で設計したシステム全体の構成と，各コンポーネントの役割について述べる．

第3章　弁証法的推論による質問応答システム

3.1　概要

本システムは，ユーザから与えられたトピック（Issue）と，対話を行うエージェントのスタンスを入力とし，2体のLLMエージェントによる対話的推論を通じて，最終的に止揚論証（統合された回答）を生成するマルチエージェント推論システムである．本研究では，弁証法的推論の規則や進行手順を「弁証法プロトコル」として抽象化し，それを実行するための計算基盤としてシステムアーキテクチャを設計する．本章では，提案プロトコルの内容には立ち入らず，システムを構成する各コンポーネントと，それらの間の情報の流れに焦点を当てる．

3.2　システム構成

本システムは，図1に示すように，入力，議論制御，弁証法プロトコル，LLM Agents，出力の5つの主要コンポーネントから構成される．これらのコンポーネントは，対話の進行管理と論証構造の制御を行う部分と，自然言語による主張生成を行う部分を分離する形で配置されている．そのため，議論の流れや論証の整合性を保ったまま，LLMの生成能力を柔軟に利用できる構成となっている．

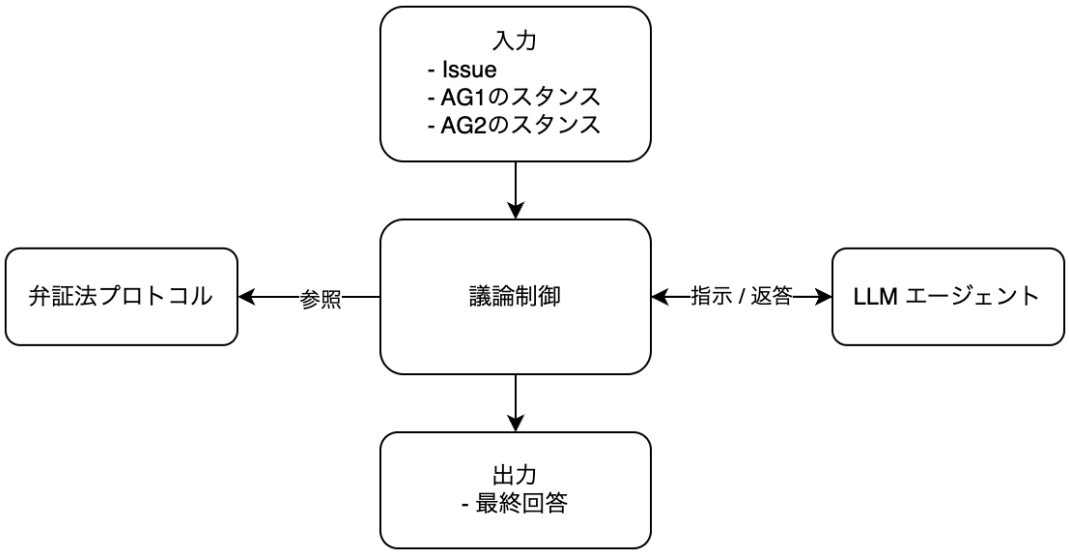

図1: 提案システムの全体アーキテクチャ

入力コンポーネントはトピックと各エージェントのスタンスをユーザーから受け取り，議論制御コンポーネントに渡す（入力例の図 2）.

今回は，Kido, Kurihara の対話例を再現することを目的として，文献中のカメラ選択問題をもとに，各エージェントのスタンスを事実および規則の集合として与えている．AG1 は「コンパクトで軽く，ユーザーフレンドリーなカメラ」を重視する立場，AG2 は「高画質かつバッテリー性能に優れたカメラ」を重視する立場をそれぞれ表しており，両者の価値基準が対立する設定となっている.

このように，各エージェントのスタンスを明示的な命題および推論規則として与えることで，議論フェーズにおいて，どの前提に基づいて主張が構成され，どの規則が衝突しているのかを構造的に追跡可能とする．以降では，本入力を用いて，多ターンの議論を通じて対立点がどのように明確化され，最終的に止揚論証がどのように生成されるかを示す.

次に，弁証法プロトコルコンポーネントは，議論の状態と，エージェントが

```
Issue: Which camera should we buy?

AG1's stance:
  a is a camera.
  c is a camera.
  a is compact.
  a is light.
  c has long battery life.
  c is user-friendly.
  b is over budget.
  If a camera is compact and light, we should buy it.
  If something is over budget, we should not buy it.
  If something is compact and light, it is user-friendly.

AG2's stance:
  b is a camera.
  a is out of stock.
  b has long battery life.
  b has high image quality.
  If a camera has high image quality and long battery life, we should buy it.
  If something is out of stock, we should not buy it.
```

図 2: 入力例

行う操作を定義したものである．議論制御コンポーネントは弁証法プロトコルを参照しながら議論を進行し，エージェントが行う操作を LLM エージェントに指示する．LLM エージェントは指示に対して返答を行い，議論制御コンポーネントに返す．使用するモデルは GPT 5-mini である．プロトコルにおいて議論の終了状態に遷移した時，議論制御コンポーネントは議論を終了し，最終回答を出力する．

次章では，本研究の主題である弁証法プロトコルの詳細設計について説明する．

3.2.1 プロトコル設計とメッセージスキーマ

本研究で提案する弁証法プロトコルは，論証構築フェーズ，反論フェーズ，統合フェーズの 3 段階から構成される．各フェーズの LLM エージェントの指示には，出力をあらかじめ定義した論証スキーマに従わせるように制御する．また，本プロトコルの終了条件は以下の 2 通りである．

- それぞれの論証に関する議論において，論証構築を行なったエージェントの意見が正当化されたとき．
- 最終回答（止揚論証）が構築されたとき

これにより，論理的に構造化されたメッセージで対話を行うことを可能にし，建設的な議論を促進する．以下の各フェーズの説明では，具体的なスキーマの構造と，その役割について詳細に述べる．

論証構築フェーズ

論証構築では，各エージェントはシステム入力として与えられた Issue および自身のスタンスに基づき，最初の主張（main argument）を 1 つ生成し，反論フェーズに遷移する．

論証の出力形式は，単なる自然文ではなく，推論規則の列（rules）から構成される論証として表現される．各規則は，結論を導くための最小限の前提（antecedent.strong）および，証拠不在を仮定する弱否定（antecedent.weak_negation）から，結論（consequent）を導く形式を取り，規則列は結論に至る推論の依存関係を明示する順序構造を持つ．また，各論証は，結論の集合（Conc）および弱否定の集合（Ass）を明示的に保持する．Conc, Ass はそれぞれ，consequent の集合リストと，弱否定の集合リストである．これにより，各エージェントの主張がどの前提に基づいて構成されているかを構造的に把握可能とする．

以下に，本フェーズにおける json スキーマを示す（図 3）．

```
{
  "Argument": {
    "rules": [
      {
        "id": "r1",
        "antecedent": {
          "strong": [
            "Minimum necessary premise 1 to derive conclusion",
            "Minimum necessary premise 2 to derive conclusion",
            ...
          ],
          "weak_negation": [
            "Assumption that there is no evidence for an argument (not negation of facts) premise 1",
            "Assumption that there is no evidence for an argument (not negation of facts) premise 2",
            ...
          ]
        },
        "consequent": "Natural language statement of the conclusion (conclusion only) derived from
strong and weak_negation",
      }
    ],
    ...
    "Conc": [
      "consequent of r1",
      "consequent of r2",
    ...],
    "Ass": [
      "weak_negation of r1",
      "weak_negation of r2",
      ...
    ]
  }
}
```


図 3: メッセージスキーマ (論証構築)

反論フェーズ

反論フェーズでは，以下の手続きを反復的に実行することで，エージェント間の対立を段階的に掘り下げる．

1. 相手の直前の論証（初期主張または再反論）に対して，反論が可能かを判定する．反論が可能な場合は，rebut（結論への反駁）または undercut（推論規則の前提への攻撃）のいずれかを選択し，新たな論証を構築する．反論が不可能な場合，相手の論証が正当化されたとみなし，議論を終了し，システムにプロトコルの終了を通知する．
2. 反論が生成された場合，今度は先に主張したエージェントが，同様にその反論に対して再反論可能かを判定し，可能であれば再反論を構築し，不可能であれば当該エージェントの論証が defeated（敗北）したものとして議論を終了し，エージェントの主張が行われていなければそちらを，そうでなければ統合フェーズに進む．

3. 上記の手続きを，どちらかのエージェントが反論不能になるまで繰り返す.

反論の手法は，Kido, Kurihara らに則り，以下のように定義し，エージェントはこれらが実行可能なとき，反論可能と判断する.

- **rebut**: 相手の論証及び反論における consequent を否定する.
- **undercut**: 相手の論証及び反論における weak_negation（ある事象が成り立たないという仮定）を否定し，それらが実際に成り立つことを証明する.

反論の出力形式は，反論可能判断（YES / NO），攻撃方法（rebut / undercut），前提（strong, weak_negation），結論（consequent），および Conc, Ass を明示する．これにより，どの前提がどのように攻撃され，どの仮定が否定されたのかを，論証間の関係として形式的に追跡可能とする.

以下に，本フェーズにおける json スキーマを示す（図 4）.

```
{
  "can_defeat": "YES or NO",
  "Argument": {
    "rules": [
      {
        "id": "r1",
        "attack": "rebut or undercut",
        "antecedent": {
          "strong": [
            "Minimum necessary premise 1 to derive conclusion",
            "Minimum necessary premise 2 to derive conclusion",
            ...
          ],
          "weak_negation": [
            "Assumption that there is no evidence for an argument (not negation of facts) premise 1",
            "Assumption that there is no evidence for an argument (not negation of facts) premise 2",
            ...
          ]
        },
        "consequent": "For rebut: statement negating opponent's conclusion; For undercut: statement
showing opponent's rule cannot be applied",
      }
    ],
    ...
    "Conc": [
      "consequent of r1",
      "consequent of r2",
      ...
    ],
    "Ass": [
      "weak_negation of r1",
      "weak_negation of r2",
      ...
    ]
  }
}
```

図 4: メッセージスキーマ (反論)

統合フェーズ　Kido, Kurihara の提案手法は，入力となる各エージェントの論証の最後の rule（warrant）のペアに対し，お互いの前提と背景知識を用いながら自身の規則列にリテラルを付加していき，最終的にそれらが同時に成り立つ新たな規則を同定することで，統合を行なっていた．これを参考に，統合フェーズでは，各エージェントの論証（main argument）における最後の rule（warrant）を入力とし，以下の段階を経て，最初に論証構築を行なったエージェント（AG1）最終回答を構築する．

1. 性質化（Characterization）：対立する 2 つの主張について，それぞれの前提（antecedent.strong）と結論（consequent）を「性質レベル」に抽象化する．これは，Kido, Kurihara の提案手法において，特定の対象を指す表現を除去し，論理式を clause（節の集合の論理式）に変換する処理と対応しており，対立する主張の性質を明らかにすることで，最終的に合意を達成する論証の性質を同定するために必要な処理である．この処理を経た論証のペアを（C1, C2）とする．
2. 一般化（Generalization）：性質化された 2 つの論証（C1, C2）に基づき，両者が受け入れ可能な合意核（E）を構築する．このとき，共通する前提が存在する場合は必ず含めるとともに，元の前提をそのまま再利用するのではなく，両者を包摂する上位概念として再構成する．

   今回は，Kido, Kurihara の提案手法における Specialization を適用することなく，対立する性質のペアの両方を包含する 1 つの性質をそのまま出力させる．なぜなら，Specialization は，本来，対立する論証の前提をそれぞれ個別に特化させ，両者の適用条件を明確に分離した上で統合処理へと導く操作であるためである．この操作は，記号論理においては，対立する clause に対して新たなリテラルを付加することで適用範囲を制限し，衝突を回避する役割を果たす．しかし，自然言語による議論においては，このような特化操作は，対立構造を解消するというよりも，単に条件を追加する形で論証を分岐させる結果となり，価値観の再構成という観点からは逆に，多様な止揚論証の可能性を狭めてしまう場合がある．したがって Specialization を適用しない方針をとり，それに伴って必要になる背景知識（background knowledge）も使用しない．
3. 最終回答の生成（Answer）：得られた合意核 E に基づき，E の前提条件を可能な限り満たす具体的な解決策を提示する．この際，元の論証に含まれ

ていた個別の結論をそのまま再利用することは避け，合意核に基づく新たな視点から，両者が納得可能な形で再構成された結論として最終回答（止揚論証）を出力する．

最後に，弁証法プロトコルの全体像をシーケンス図で表す（図 5）．これの図はこの図では，それぞれの最初の主張をそれぞれ A1，A2 で示す．

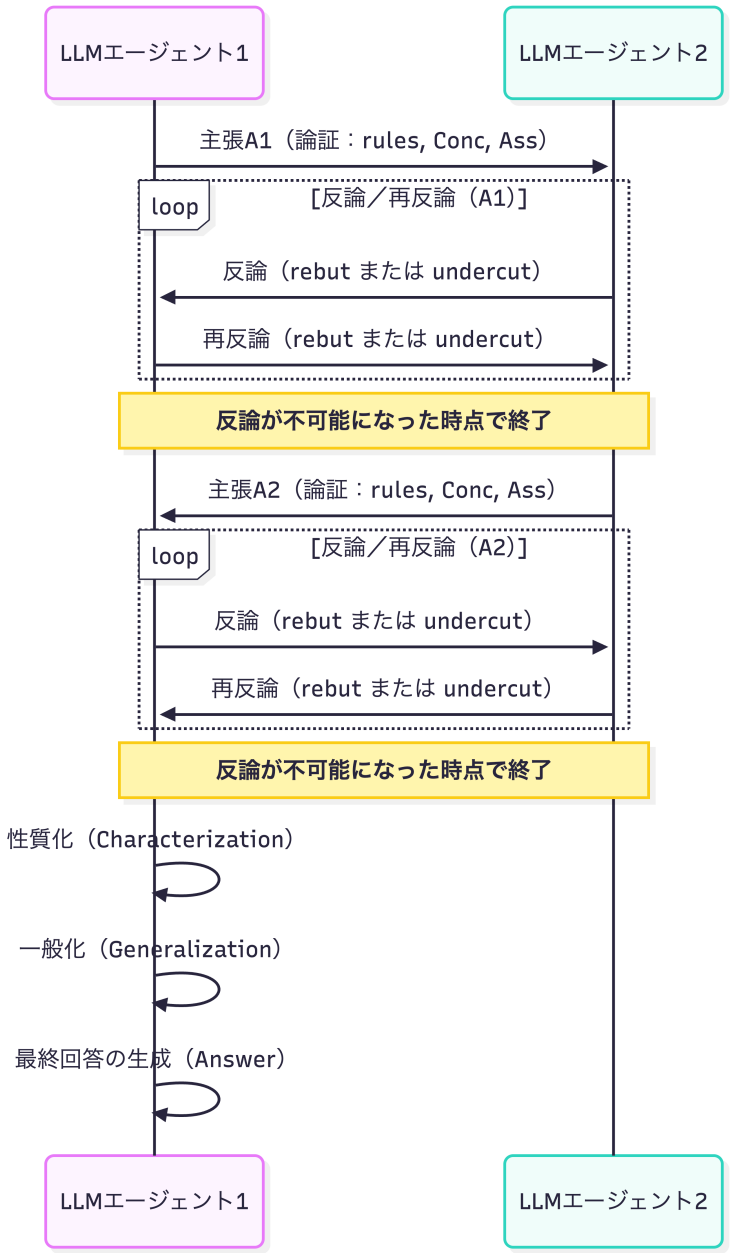

図 5: 弁証法プロトコルを用いたシーケンス図

第4章　実験

4.1　実験の目的

本章では，前章で定義した弁証法プロトコルを実際の問題設定に適用し，どのような対話が生成され，最終的な止揚論証がどのような過程を経て導出されるのかを，具体的な対話例を通して示すことを目的とする．

また，本章で得られた対話ログおよび最終回答は，次章において，既存研究との整合性および統合結果の妥当性を評価するための基礎データとして用いる．

4.2　対話の実行と結果

4.2.1　問題設定・実行

題材には，Kido，Kurihara[1] 第6章で用いられているカメラ購買問題を採用した．議題（Issue）は”Which camera should we buy”とし，2体のエージェントには，それぞれ異なる事実および推論規則からなるスタンスを入力として与えた．入力の具体例については，3章で示した図2の通りである．本問題設定では，一方のエージェントが「携帯性・操作性」を重視する立場，他方のエージェントが「画質・バッテリー性能」を重視する立場をとることで，評価基準の異なる2つの観点が意図的に対立するよう設計されている．

これらの入力を，前章で定義した弁証法プロトコルを適用した提案システムに与え，実行する．各ターンにおけるエージェントの出力はすべて構造化された論証として生成・記録される．

これにより，どの前提がどの結論を導いているのか，反論においてどの結論が否定され，あるいはどの仮定が攻撃されているのか，さらに統合フェーズにおいて，対立する前提構造がどのように再編成されているのかを，対話の進行とともに追跡可能とした．

4.2.2　対話結果

以下では，主張，反論，再反論の各段階で生成された論証を，対話ログの画像として提示する．

実際は前章で説明したjsonスキーマの通り出力されるが，余白の関係で簡略化したものを示す（図6）．それぞれの変数名は，前章で定義したものと同じである．

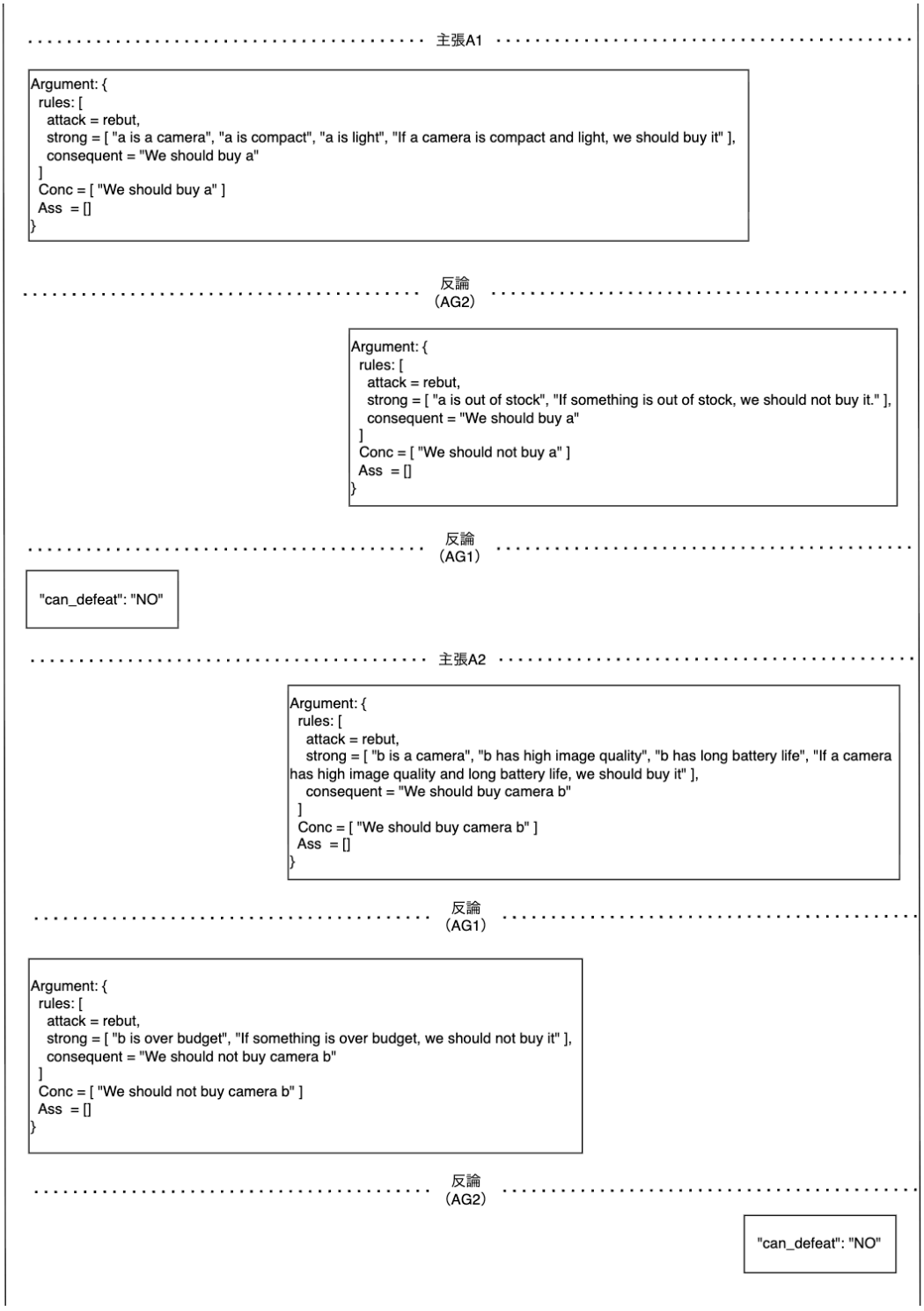
主張A1
Argument: {
rules: [
attack = rebut,
strong = ["a is a camera", "a is compact", "a is light", "If a camera is compact and light, we should buy it"],
consequent = "We should buy a"
]
Conc = ["We should buy a"]
Ass = []
}
反論
(AG2)
Argument: {
rules: [
attack = rebut,
strong = ["a is out of stock", "If something is out of stock, we should not buy it."],
consequent = "We should buy a"
]
Conc = ["We should not buy a"]
Ass = []
}
反論
(AG1)
"can_defeat": "NO"
主張A2
Argument: {
rules: [
attack = rebut,
strong = ["b is a camera", "b has high image quality", "b has long battery life", "If a camera has high image quality and long battery life, we should buy it"],
consequent = "We should buy camera b"
]
Conc = ["We should buy camera b"]
Ass = []
}
反論
(AG1)
Argument: {
rules: [
attack = rebut,
strong = ["b is over budget", "If something is over budget, we should not buy it"],
consequent = "We should not buy camera b"
]
Conc = ["We should not buy camera b"]
Ass = []
}
反論
(AG2)
"can_defeat": "NO"

図 6: 対話実行のログ

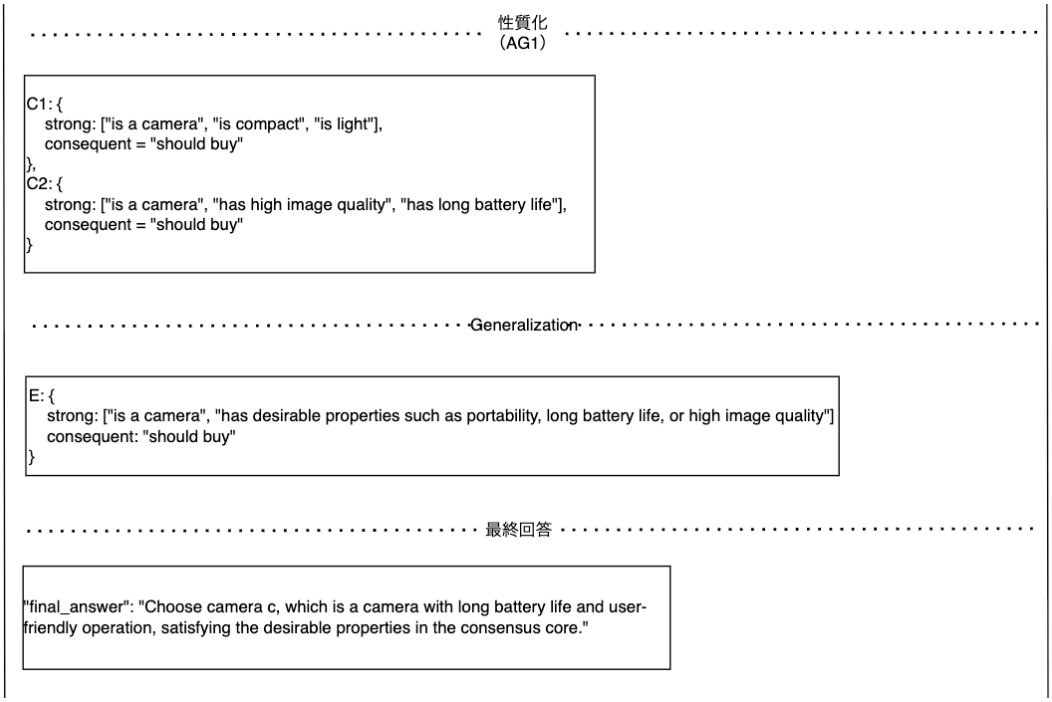

図 7: 統合フェーズによる止揚論証の生成

4.2.3 止揚論証の生成

対話の結果，いずれの主張も単独では正当化されない状態となり，反論フェーズで得られた議論履歴に対して統合フェーズが適用された．図 7 は，性質化および一般化を経て得られた合意核に基づき，最終的な止揚論証が構築される過程を示したものである．

図 6，図 7 に示すように，本研究のプロトコルでは，多ターンの議論を通じて対立点が段階的に明確化され，最終的に性質化および一般化を経て，単なる折衷ではない止揚論証が構築されていることが確認できる．次章では，これらの対話結果および最終回答を，既存の形式例と比較することで，提案手法の妥当性について評価を行う．

第5章　評価

5.1　評価手法

本章では，提案する弁証法プロトコルの妥当性を検証するための評価手法について述べる．本研究における評価の主目的は，提案プロトコルによって生成される議論過程および止揚論証が，既存の計算的弁証法モデルに基づく形式例と，構造および意味の両面において対応しているかを確認することである．

従来の質問応答システムの評価では，単一の正解を持つタスクに対する正解率が主な指標として用いられることが多い．しかし，弁証法的議論に基づく推論では，議題に対する唯一の正解を定義することが困難であり，議論の進行構造，反論関係の形成，および最終的な統合の構成そのものが評価対象となる．本研究では，提案プロトコルの動作が既存研究で示されている形式例をどの程度再現できているかに着目し，生成過程の対応関係をトレースすることによって妥当性を検証する．

本研究における評価方針は，以下の二点に基づく．

- 構造的一致:提案プロトコルによって生成される議論構造が，Kido，Kurihara によって定義された計算的弁証法の形式例と対応しているかを検証する．具体的には，論証の構築順序，反論関係（defeat 関係）の形成，および議論終了後に統合が生成される過程について，形式例と照合しながら確認する．
- 統合の再現性：生成される最終回答が，既存の形式例における統合結果と意味的に対応しているかを検証する．ここでは，新規性や一般性の評価は行わず，対立構造の整理や判断根拠の構成が，既存の統合例をどの程度再現できているかに着目する．

なお，本章の評価は，提案プロトコルが既存の計算的弁証法モデルの振る舞いをどの程度正確にトレースできているかを確認することを目的としており，多様なトピックに対する汎用的性能や定量的な比較評価は対象外とする．

5.2　評価に用いる比較データ

本研究では，提案する弁証法プロトコルの妥当性を検証するための比較対象として，Kido，Kurihara によって提案されたカメラ購買問題に関する対話例（文献 [1] 第 6 章）を用いる．本節では，評価の基準となる形式例を明確にするため，文献中で用いられている問題設定および対話過程を示す．

まず，文献では次の議題と 2 つの知識ベースが与えられる．

$$Issue = buy(x) \wedge camera(x)$$

$$\begin{aligned} S_1 = \{ & camera(a),\ camera(c),\ compact(a),\ light(a), battery(c, \text{long}), \\ & userFriendly(c),\ overTheBudget(b), \\ & compact(x) \wedge light(x) \wedge camera(x) \rightarrow buy(x), \\ & overTheBudget(x) \rightarrow \neg buy(x), \\ & compact(x) \wedge light(x) \rightarrow userFriendly(x) \} \end{aligned}$$

$$\begin{aligned} S_2 = \{ & camera(b),\ outOfStock(a), \\ & battery(b, \text{long}),\ resolution(b, \text{high}), \\ & resolution(x, \text{high}) \wedge battery(x, \text{long}) \wedge camera(x) \rightarrow buy(x), \\ & outOfStock(x) \rightarrow \neg buy(x) \} \end{aligned}$$

ここで，S_1 および S_2 は，それぞれエージェント 1 およびエージェント 2 に与えられる知識ベースを表す．S_1 は携帯性や使いやすさを重視する立場に対応しており，S_2 は高画質およびバッテリー性能を重視する立場に対応している．両者は評価基準が異なるため，購入すべきカメラについて対立した判断を導く．

次に，文献中で示されている対話の進行を原表記に基づいて示す．ここで Arg_i^j は,「エージェント i が構築した j 番目の論証」を表す.

まず，エージェント 1 が主張を構築し，エージェント 2 がこれを反論する．

$$\begin{aligned} \mathrm{Arg}_1^1 = [& compact(a),\ light(a),\ camera(a),\quad compact(a) \wedge light(a) \wedge camera(a) \\ & \rightarrow buy(a)] \quad \text{(I want to buy 'a' since it is a compact and light camera.)} \end{aligned}$$

$$\mathrm{Arg}_2^2 = [\, outOfStock(a),\ outOfStock(a) \rightarrow \neg buy(a) \,] \quad \text{(It is out of stock.)}$$

この反論に対して，エージェント 1 はこれを打ち負かす論証を構築できない．したがって，Arg_1^1 は defeated となり，エージェント 2 の主張へ移る．

続いて，エージェント 2 が新たな主張を行い，エージェント 1 がこれに反論する．

$$\begin{aligned} \mathrm{Arg}_2^3 = [& resolution(b, \text{high}),\ battery(b, \text{long}),\ camera(b), camera(b) \\ & \wedge resolution(b, \text{high}) \wedge battery(b, \text{long}) \rightarrow buy(b)] \end{aligned}$$

（Would you like 'b'? Because it is a high-resolution camera with a long battery life.）

$\mathrm{Arg}_1^4 = [\, overTheBudget(b),\ overTheBudget(b) \rightarrow \neg buy(b)\,]$ （It exceeds the budget.）

この結果，Arg_1^1 および Arg_2^3 のいずれも正当化されず，さらに両エージェントはこれ以上の主張を構築できない状態となる．そこで，エージェント 1 は，対立する主張の warrant（Arg_1^1,Arg_2^3）に対して統合のアルゴリズム（Reasoning Method）を適用し，両者の根拠（warrant）を統合した新たな主張を構築する．

$\mathrm{Arg}_1^5 = [\, userFriendly(c),\ battery(c, \mathrm{long}),\ camera(c), userFriendly(c)$
$\wedge\, battery(c, \mathrm{long}) \wedge camera(c) \rightarrow buy(c)]$ （I will then buy ‘c.’
Since this is a user-friendly camera with a long battery life.）

この Arg_1^5 は，携帯性・使いやすさとバッテリー性能という異なる評価軸を同時に満たす対象として c を選択するものであり，Arg_1^1 と Arg_2^3 の根拠を包摂した統合的な論証である．

5.3 評価結果

本節では，前節で示した形式例（文献 [1] 第 6 章）と，図 6 および図 7 に示した本研究の実行結果を比較し，提案プロトコルによって生成される議論過程および統合過程が，構造的に既存の計算的弁証法モデルと一致しているかを検証する．ここでいう構造的一致とは，各段階において，(1) 用いられる前提，(2) 適用される推論規則（評価基準），(3) 導出される結論，および (4) 反論不能性に基づく主張権の遷移が，形式例と同型であることを指す．

まず，形式例では，エージェント 1 が知識ベース S_1 に基づき,「コンパクトかつ軽量であり，かつカメラである対象は購入すべきである」という規則を適用し，対象 a がコンパクトで軽量なカメラであるという前提から,「カメラ a を購入すべきである」という主張を導出する．これは，次の論証として表現されている．

$\mathrm{Arg}_1^1 = [\, compact(a),\ light(a),\ camera(a),\quad compact(a) \wedge light(a) \wedge camera(a)$
$\rightarrow buy(a)\,]$ (I want to buy ‘a‘ since it is a compact and light camera.)

本研究の実行結果においても，対話の初期段階でエージェント 1 は，”a is a camera”, ”a is compact”, ”a is light” という性質を前提とし，”If a camera is compact and light, we should buy it”という前提を用い，特定の対象について購買結論”We should buy a”を導出している．この推論は，携帯性に関する性質集合から購買結論を導くという点で，形式例の Arg_1^1 と構造的に一致している．

これに対して形式例では，エージェント 2 が,「在庫切れであれば購入すべきでない」という規則を適用し，対象 a が在庫切れであるという前提から，先の購買結論を否定する反論を構築する．これは，次の論証として与えられている．

$$\mathrm{Arg}_2^2 = [\, outOfStock(a),\ outOfStock(a) \rightarrow \neg buy(a) \,] \quad \text{(It is out of stock.)}$$

本研究の実行結果においても，初期主張に対してエージェント 2 は，　”a is out of stock” という事実と，”If a camera is out of stock, we should not buy it” という前提を用いて当該購買結論を否定する反論”We should not buy a”を構築している．この反論に対して，初期主張側のエージェントは，反論の可能性を検討するが，rebut の論証は Ass が空であり，これに対する undercut は構築できない．

その結果，形式例と同様に議論は終了する．これに伴い，次の主張権は反論を行ったエージェント 2 へと移行する．この「反論不能性を条件とした主張権の遷移」は，計算的弁証法における議論進行規則そのものであり，本研究のプロトコルが形式例と同型の遷移条件を実装していることを示している．

続いて形式例では，エージェント 2 が,「高画質であり，かつバッテリーの持続時間が長く，カメラである対象は購入すべきである」という規則を適用し，対象 b が高画質かつ長時間利用可能なカメラであるという前提から，新たな購買主張を構築する．これは，次の論証として示されている．

$$\mathrm{Arg}_2^3 = [\, resolution(b, \mathrm{high}),\ battery(b, \mathrm{long}),\ camera(b), camera(b)$$
$$\wedge\, resolution(b, \mathrm{high}) \wedge battery(b, \mathrm{long}) \rightarrow buy(b) \,] \quad \text{（Would you like ‘b‘?}$$
$$\text{Because it is a high-resolution camera with a long battery life.）}$$

本研究の実行結果においても，主張権の移行後，エージェント 2 は，”b is a camera”, ”b has high image quality”, ”b has long battery life” という事実と，”If a camera has high resolution and long battery life, we should buy it” という前提を用いて購買主張”We should buy camera b”を構築している．

これに対して形式例では，エージェント 1 が,「予算超過であれば購入すべきでない」という規則を適用し，対象 b が予算を超過しているという前提から，再び購買結論を否定する反論を構築する．これは次の論証として表される．

$\mathrm{Arg}_1^4 = [\, overTheBudget(b),\ overTheBudget(b) \rightarrow \neg buy(b) \,]$ （It exceeds the budget.）

本研究の実行結果においても，第二の購買主張に対してエージェント 1 は，”b is over budget”という前提と，”If a camera is over the budget, we should not buy it”という前提を用いて当該主張を否定する反論”We should not buy camera b”を構築している．この結果，エージェント 1 および 2 のいずれの主張も正当化されず，両エージェントはこれ以上の主張を構築できない状態に至る．

以上の議論収束を受けて，形式例では，対立していた二つの購買主張（Arg_1^1 と Arg_2^3）の warrant を入力として，統合アルゴリズムが適用され，携帯性・使いやすさとバッテリー性能という異なる評価軸を同時に満たす対象 c を選択する統合論証 Arg_1^5 が構築される．

統合の具体的な流れについては既存研究の対話例の中に示されていなかったため，対応関係を示すことはできないが，warrant の入力から双方の購買条件となる性質をそれぞれ抽出し，それらを包含する条件を新たに構築することで，同様の結論”Choose camera c,...”が導かれている．したがってこの点においても形式例の Arg_1^5 と構造的に一致していると言える．

以上より，本研究の実行結果は，形式例における主張構築，反論，の流れと，反論不能による主張エージェントの交代，および統合という一連の推論過程を，論理構造を保ちながら再現できており，提案プロトコルが構造的に既存の計算的弁証法モデルと一致していることが確認された．

第6章　考察

本章では，提案システム及びプロトコルの応用性について検討する．本研究では提案手法に対し，既存研究との整合性を検証したのみで，実際にさまざまなトピックについて有用な回答が得られたかどうかの検証は行っていない．したがって，本研究の結果からは，提案手法が既存研究の形式例と整合的に動作することは確認できたものの，入力として与えるスタンスの表現方法を変更した場合に，どのような性質の回答が得られるかについては，さらなる検討の余地がある．

そこで考察として，従来のように「カメラa」「カメラb」といった架空の対象および明示的な事実・規則を与える形式ではなく，エージェントに対して評価観点のみを自然言語で与える入力形式を用いた．具体的には，一方のエージェントには携帯性や使いやすさを重視する立場を，他方のエージェントには画質やバッテリー性能を重視する立場を与え，個別の製品名や事実命題は明示的には指定していない．これらの入力例については，図8に示す．

Issue: Which camera should we buy?

AG1's stance:
You are discussing which camera to buy.
You prioritize ease of use for the user, such as compactness and lightness.

AG2's stance:
You are discussing which camera to buy.
You prioritize camera performance, such as good image quality and battery life.

図8: 入力

このような抽象的なスタンスのみを与えた条件下において，本システムを適用したところ，最終的な回答として，実在するカメラ製品名を含む具体的な提案が生成された．例えば，以下のような回答が得られた．

”final_answer”: ”Sony RX100 VII — a compact pocket camera that provides easy portability while delivering strong image quality.”

この結果は，合意核として満たすべき性質— すなわち「携帯性が高く使いやすいこと」と「十分な性能を備えていること」 —を，LLMエージェントが実

在する製品知識と対応付け，具体的な選択肢として提示した例であると解釈できる．

従来の Kido, Kurihara らの対話例では，架空の対象に対して形式的に定義された事実と規則の組合せのみが議論の対象となっていた．そのため，LLM を用いた場合においても，生成される結論は入力として与えられた対象の範囲内に限定されると考えられていた．実際，架空の「カメラ c」が最終回答として選択された場合，それは一般知識に基づく選択ではなく，あくまで与えられたスタンス内での形式的な帰結に過ぎない．

しかしながら，本考察で示したように，評価観点のみを与える抽象的な入力形式においては，LLM が有する一般知識を活用し，合意核を満たす具体的な実在製品を自律的に対応付ける挙動が確認された．このことは，本システムが入力として与えられたスタンスのみに閉じた推論を行うのではなく，議論の構造を保ったまま，現実世界の知識を用いた回答生成へ拡張しうる可能性を示している．

以上より，本結果は，文献中の形式例を忠実に再現するという観点では想定外の挙動を含むものの，提案システムを将来的に一般的な実問題や実在対象を含む意思決定問題へ適用する上で，有効な拡張性を有していることを示唆するものである．特に，評価軸のみを与えることで，具体的な選択肢の生成を LLM に委ねる設計は，多様なトピックへの応用において有望なアプローチであると考えられる．

第7章　おわりに

本研究では，大規模言語モデル（LLM）を用いたマルチエージェント対話に，計算的弁証法の理論的枠組みを導入することで，対立する主張を形式的に扱い，その上で止揚論証を生成する質問応答システムを提案した．論証を構造化スキーマとして表現し，rebut および undercut に基づく反論フェーズと，性質化および一般化に基づく統合フェーズからなる弁証法プロトコルを設計・実装した点に本研究の特徴がある．さらに，Kido，Kurihara の形式例を評価基準として用いることで，提案手法が既存の計算的弁証法モデルと意味的に対応する議論構造および統合過程を再現できることを示した．

本研究の意義は，既存の計算的弁証法モデルに基づく形式的な議論過程を，大規模言語モデルを用いた自然言語対話として構造的に再現可能であることを示した点にある．論証構造と反論の手法を明確に定義することで，LLM が持つ幅広い一般的な知識を引き出しつつも，表層的な反論の応酬を防ぎ，建設的な議論を促進することができた．またそれにより，自由発話では困難な議論の停止が可能になり，推論プロトコルとしての終了を保証する．

一方で，本研究にはいくつかの課題も残されている．評価は既存文献の形式例をトレースすることに主眼を置いており，多様なトピックに対する汎用的な性能やスケーラビリティの検証には至っていない．

今後の発展としては，主に三つの方向性が考えられる．第一に，本研究では評価対象を特定の形式例に限定したが，本プロトコルをさまざまなトピックに適用し，社会的・倫理的問題や価値観の対立を含む多様な議題において，議論構造および統合過程がどの程度一貫して再現されるかを検証することが重要である．これにより，本手法の適用範囲と限界を体系的に明らかにできると考えられる．

第二に，本研究では特定の LLM を用いてエージェントを構成したが，今後はモデル規模や学習方針の異なる複数の LLM を用いたエージェント間対話を通じて，議論の進行特性や統合結果の差異を比較することが課題である．また，プロンプト設計や内部表現の調整（チューニング）を通じて，反論の生成傾向や統合の抽象度がどのように変化するかを分析することで，弁証法的推論に適したエージェント設計指針の確立が期待される．

第三に，議論過程におけるエージェントの発言の一貫性に対する評価枠組み

の構築が挙げられる．具体的には，各エージェントが初期の立場や評価基準を議論の進行中にどの程度維持しているか，また統合段階においてそれらがどのように再構成されているかを定量的に測定する指標の設計が求められる．これにより，最終的な出力結果のみならず，推論過程そのものの安定性や妥当性を評価することが可能になると考えられる．

本研究が，弁証法的推論と大規模言語モデルの融合に向けた基盤的な試みとして，今後の計算的対話システムの発展に寄与することを期待する．

謝辞

本研究を行うにあたり，熱心なご指導，ご助言を賜りました村上陽平教授に深く感謝申し上げます．また，普段からお世話になっている社会知能研究室の皆様にも心より感謝申し上げます．

参考文献

[1] Kido, H. and Kurihara, M.: Computational Dialectics Based on Specialization and Generalization, *Proc. JSAI 2008*, pp. 228–241 (2008).

[2] Sawamura, H., Umeda, Y. and Meyer, R. K.: Computational Dialectics for Argument-based Agent Systems, *Proc. 4th International Conference on MultiAgent Systems (ICMAS 2000)*, pp. 271–278 (2000).

[3] Umeda, Y. and Sawamura, H.: Towards an Argument-Based Agent System, *Proc. 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES 1999)*, pp. 30–33 (1999).

[4] Prakken, H. and Sartor, G.: Argument-based extended logic programming with defeasible priorities, *Journal of Applied Non-Classical Logics*, Vol. 7, No. 1-2, pp. 25–75 (1997).

[5] Anghel, C. et al.: Multi-Model Dialectical Evaluation of LLM Reasoning Chains, *Informatics*, Vol. 12, No. 3, p. Art. 76 (2025).

付録

A.1 プロンプト

本付録では，提案システムにおいて使用される 5 つのプロンプトテンプレートの詳細を示す.

論証構築

```
Task:
Based on the given Issue and your stance, construct one main Argument
    representing your position.

Constraint:
The main Argument must follow the structure of argumentation below.

Argumentation structure:
  - rules: An array of inference rules. Each rule represents an
    inference from premises (antecedent) to a conclusion (consequent)
     Important: rules have an order. The consequent of an earlier rule
    can be used in the antecedent.strong of a later rule.
     Include only the minimum necessary rules to derive the final
    conclusion (consequent of the last rule).
     - id: Rule identifier (r1, r2, ...)
     - antecedent.strong: A list of natural language statements
    representing the minimum necessary premises to derive the
    conclusion (may include consequents from previous rules)
     - antecedent.weak_negation: A list of natural language statements
    representing assumptions that there is no evidence for certain
    arguments (not negation of facts) necessary to derive the
    conclusion
     - consequent: A natural language statement of the conclusion
    (conclusion only) derived from strong and weak_negation
  - Conc: A list collecting the consequent of each rule
  - Ass: A list collecting the weak_negation of each rule

Output only the following JSON:
{
  "Argument": {
```

```
    "rules": [
      {
        "id": "r1",
        "antecedent": {
          "strong": [
            "Minimum necessary premise 1 to derive conclusion",
            "Minimum necessary premise 2 to derive conclusion",
            ...
          ],
          "weak_negation": [
            "Assumption that there is no evidence for an argument
    (not negation of facts) premise 1",
            "Assumption that there is no evidence for an argument
    (not negation of facts) premise 2",
            ...
          ]
        },
        "consequent": "Natural language statement of the conclusion
    (conclusion only) derived from strong and weak_negation",
      }
    ],
    ...
    "Conc": [
      "consequent of r1",
      "consequent of r2",
    ...],
    "Ass": [
      "weak_negation of r1",
      "weak_negation of r2",
      ...
    ]
  }
}
```

反論

The above is the opponent's argument.
Each has the following meaning.

The argument has the following structure:
- rules: An array of inference rules. Each rule represents an inference from premises (antecedent) to a conclusion (consequent)
  Important: rules have an order. The consequent of an earlier rule can be used in the antecedent.strong of a later rule.
  Include only the minimum necessary rules to derive the final conclusion (consequent of the last rule).
  - id: Rule identifier (r1, r2, ...)
  - (attack): Attack method (rebut or undercut)
  - antecedent.strong: A list of minimum necessary premises to derive the conclusion (may include consequents from previous rules)
  - antecedent.weak_negation: A list of assumptions that there is no evidence for certain arguments (not negation of facts) necessary to derive the conclusion
  - consequent: Conclusion (conclusion only) derived from strong and weak_negation
- Conc: A list collecting the consequent of each rule
- Ass: A list collecting the weak_negation of each rule

Task:
Determine whether you can defeat the opponent's Argument, and if possible, generate one counterargument.
Answer NO only if you truly cannot counterargue.

Constraint:
For counterarguments, choose one of the following attack methods.
Counterarguments must follow the argumentation structure below.
Counterarguments must ""always"" be stated from your own stance.
Counterarguments must ""never"" reuse your past antecedent.strong.

Attack methods:・
rebut: When your stance semantically contradicts or conflicts with a proposition in the opponent's argument's conclusion (Conc)
  → Construct an inference rule that negates the opponent's conclusion or derives a conclusion incompatible with the opponent's conclusion
  → If the opponent's argument's strong is empty, rebut is not

```
    possible. •
undercut: When your stance can provide evidence that a proposition in
    the opponent's argument's assumption (Ass) is correct
  → Construct an inference rule that negates the weak negation
    assumption used in the opponent's inference rule, or shows that
    the assumption does not hold
  → If the opponent's argument's weak_negation is empty, undercut is
    not possible.

Argumentation structure:
- rules: An array of inference rules. Each rule represents an
    inference from premises (antecedent) to a conclusion (consequent)
  Important: rules have an order. The consequent of an earlier rule
    can be used in the antecedent.strong of a later rule.
  Include only the minimum necessary rules to derive the final
    conclusion (consequent of the last rule).
  - id: Rule identifier (r1, r2, ...)
  - attack: Attack method (rebut or undercut)
  - antecedent.strong: A list of minimum necessary premises to derive
    the conclusion (may include consequents from previous rules)
  - antecedent.weak_negation: A list of assumptions that there is no
    evidence for certain arguments (not negation of facts) necessary
    to derive the conclusion
  - consequent: Conclusion (conclusion only) derived from strong and
    weak_negation
- Conc: A list collecting the consequent of each rule
- Ass: A list collecting the weak_negation of each rule

    Output only the following JSON:
{
  "can_defeat": "YES or NO",
  "Argument": {
    "attack": "rebut or undercut",
    "rules": [
      {
        "id": "r1",
        "antecedent": {
```

```
        "strong": [
          "Minimum necessary premise 1 to derive conclusion",
          "Minimum necessary premise 2 to derive conclusion",
          ...
        ],
        "weak_negation": [
          "Assumption that there is no evidence for an argument
   (not negation of facts) premise 1",
          "Assumption that there is no evidence for an argument
   (not negation of facts) premise 2",
          ...
        ]
      },
      "consequent": "For rebut: statement negating opponent's
   conclusion; For undercut: statement showing opponent's rule
   cannot be applied",
    }
  ],
  ...
  "Conc": [
    "consequent of r1",
    "consequent of r2",
    ...
  ],
  "Ass": [
    "weak_negation of r1",
    "weak_negation of r2",
    ...
  ]
 }
}
```

性質化

```
The above are two conflicting arguments, Argument1 and Argument2.
Each has the following meaning.
The argument has the following structure:
 - antecedent.strong: A list representing the properties of the
```

```
    minimum necessary premises to derive the conclusion
  - consequent: The conclusion (conclusion only) derived from strong

Task:
Characterize the given Argument1 and Argument2.
This operation clarifies the characteristics of each of the two
    conflicting arguments.

Constraint:
Output must follow the argumentation structure below.

Argumentation structure:
- antecedent.strong: A list representing the properties of the
    minimum necessary premises to derive the conclusion
- consequent: The property of the conclusion (conclusion only)
    derived from strong

Output only the following JSON:
Output:
{
  "Argument": {
    "C1": {
      "strong": [
        "Property 1 of minimum necessary premises to derive
    conclusion (do not include expressions uniquely identifying
    specific objects)",
        "Property 2 of minimum necessary premises to derive
    conclusion (do not include expressions uniquely identifying
    specific objects)",
        ...
      ],
      "consequent": "Natural language statement representing the
    property of the conclusion (conclusion only) derived from strong",
    },
    "C2": {
      "strong": [
        "Property 1 of minimum necessary premises to derive
```

```
    conclusion (do not include expressions uniquely identifying
    specific objects)",
        "Property 2 of minimum necessary premises to derive
    conclusion (do not include expressions uniquely identifying
    specific objects)",
        ...
      ],
      "consequent": "Natural language statement representing the
    property of the conclusion (conclusion only) derived from strong
    (do not include expressions uniquely identifying specific
    objects)",
    }
  }
}
```

一般化

```
The above are C1 and C2, representing the properties of a pair of
    conflicting arguments.

Each argument (C1, C2) has the following structure:
Argumentation structure:
- antecedent.strong: A list representing the properties of the
    minimum necessary premises to derive the conclusion
- consequent: The property of the conclusion (conclusion only)
    derived from strong

Task:
Based on the given C1 and C2, construct one consensus core E
    (Generalization result) that both parties can easily accept.

Constraint:
If there are facts (strong) common to both, they must be included.
Do not use the strong of C1 and C2 as they are.
Each premise of the consensus core must be able to encompass both
    positions.

Output only the following JSON:
```

```
Output:
{
  "Argument": {
    "E": {
      "strong": [
        "Consensus core premise (natural language) 1",
        "Consensus core premise (natural language) 2",
        "..."
      ],
      "consequent": "Conclusion (conclusion only) derived from the
    consensus core"
    }
  }
}
```

最終回答

```
The above are the warrants of two conflicting arguments Argument1 and
    Argument2, and the consensus core E of both.

Structure of premise information:
- Warrant of Argument1, Argument2: Premises and conclusions of
    individual cases claimed by each agent
- Consensus core E: Property-level rules that both parties can agree
    on

Task:
Based on the given consensus core E, present a solution that
    satisfies the conditions of E.strong as much as possible, and
    make it the final answer.

Constraint:
- Individual conclusions appearing in the warrants of Argument1 and
    Argument2 ""must"" not be used as they are in the final answer
- The final answer must be a concrete solution that satisfies E.strong

Output only the following JSON:
{
```

```
  "final_answer": "Final answer (simple)"
}
```