K_092

非単調論理の論証状態分析による議論支援
Argumentation State Analysis of Non-monotonic Logic for Supporting Argument

柴田 裕介 † 山口 和紀 ‡
Yusuke Shibata Kazunori Yamaguchi

1. 導入

近年論証構造に基づく議論支援システムが研究されている．例えば法的な論証の構造化を念頭に置いたArguMed [5] や，政策立案での利用に焦点を当てたZeno [3] などがある．著者もまた独自に議論を記述する枠組みとしてP++を提案した [6].

これらのシステムはいずれも推論機構を備えており，賛成/反対といった論証構造に基づいて主張の是非を導出できるようになっている．しかし，ではどうして肯定(否定)に導出されるのかは推論機構のアルゴリズムに立ち返る必要があり，直感的に導出理由を理解するのは困難である．

また，これらのシステムはいずれも推論機構の定義で終わっており，積極的に議論のプロセスに介入し議論支援を行う手法などは提案されていない．しかし，議論を構造化して記述しているため，構造を計算することで議論支援へと踏み込む余地は少なくない．

本研究では，非単調論理の分野における議論フレームワーク理論 ([2] など) を利用することで，議論の導出理由を明示的に表現する論証状態グラフを提案し，どうして肯定 (否定) になるのかを直感的に理解できる手法を提案する．

また，論証状態グラフを元に議論支援へと踏み込み，結論が得られたときに結論にクリティカルに効く部分だけを集める要約と，肯定側/否定側それぞれの視点から“どこを補強/反論すべきか”を表すビューを定義する．

2. 言語

議論記述のための言語としてSartor [4] を元に独自に変更を加えたものを用いる．Sartor [4] はDung [1] の議論フレームワーク (AF) を元に，強弱二種類の否定やルールの優先度を組み込んだ論理を構築している．本研究では，論理ではなく議論を記述するという観点から，ルールの集合として論理を記述するのではなく，それと等価な有向グラフによる記述を考える．また簡略化のため現時点ではルールの優先度は考慮していない．

定義: ルール ルールは次の形式で記述される．

$$\sim \overline{L_0} \wedge \sim \overline{L_1} \wedge, ..., \sim \overline{L_m} \Rightarrow S$$

L_i はリテラル，S は言明を表し リテラル L_j または $\overline{L_j}$(L_j の否定を表す) である．$\sim \overline{L_i}$ はデフォルト前提であり“$\overline{L_i}$ である根拠がない”ことを表す．

前提がすべてデフォルト前提であるが，本研究では議論の記述を考えるため，すべての主張は前提にすぎない (反論があれば崩れる) という立場をとる．したがってファクトも考えず，前提が空集合になることはない．

定義: 全体議論グラフ 議論はすべてルールにより記述される．ルールの集合 R に対して，全体議論グラフ $TAG = (N, A)$ を次のように定義する．TAG のノード N はルールおよびルールに含まれるすべての言明からなる．アーク A はルールの各々の前提からルール自体へのアークと，ルール自体から結論へのアークからなる．このとき，結論が否定形つまり $\overline{A_i}$ の形をしている場合は，ルールのノードに $-$，そうでない場合はルールのノードに $+$ を記述し区別する．

全体議論グラフが，議論の構造化に適するようにいくつかの制約を与える．

- 結論と前提が循環するルールの順序列，すなわちループを認めない．
- グラフの唯一の終点 (議題) が存在する．

この制約を満たす全体議論グラフを議論記述Πとする．

例: 全体議論グラフ 例えば，図1の全体議論グラフ (囲み枠の説明は後述) をルールで表現すると次のようになる．

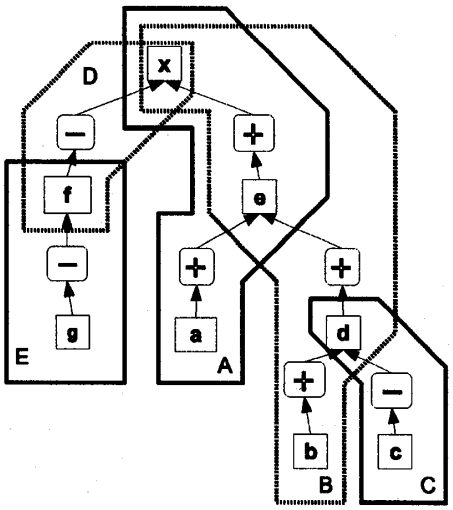

図 1: 全体議論グラフと議論列変換の例

†東京大学大学院総合文化研究科．Graduate School of Arts and Sciences, The University of Tokyo. shiva@graco.c.u-tokyo.ac.jp
‡東京大学情報基盤センター．Information Technology Center, The University of Tokyo. yamaguch@mail.itc.u-tokyo.ac.jp

$$\sim\overline{a} \Rightarrow e, \sim\overline{b} \Rightarrow d, \sim\overline{c} \Rightarrow \overline{d},$$
$$\sim\overline{d} \Rightarrow e, \sim\overline{e} \Rightarrow x, \sim\overline{g} \Rightarrow \overline{f}, \sim\overline{f} \Rightarrow \overline{x}$$

– が記述されたルールのノードは，対応するルールの結論において否定 ($\overline{a}$ の形) に変換されている点に注意せよ．

次に，議論フレームワークとして議論を捉えるために，ルールよりも大きな単位である議論列と，議論記述から議論列を生成する議論列変換を導入する．

定義: 議論列 議論列 AA とは，次を満たすルールの順序列 $AA = [r_1, r_2, ..., r_n]$ である．r_i の結論 c_i が $r_j (j > i)$ の前提に $\sim\overline{r_j}$ として現れる．ただし r_n の結論 c_n は他のルールに一切現れない．また r_i の結論が $\overline{c_i}$ であるときそれ以降続くルールはない $(i = n)$．循環や自己矛盾してはならない．

定義: 議論列の結論 議論列 $AA = [r_1, r_2, ..., r_n]$ の結論 $CONC(AA)$ とは，末端のルール r_n の結論である．

定義: 議論列変換 議論列変換とは，全体議論グラフ TAG に対し極大となる議論列をすべて集めることである．

例: 議論列変換 図1の議論列変換は図中の A,B,C,D,E となる．

議論列を元に議論フレームワークを考える．まず，議論列の間の関係として defeat を導入する．

定義: defeat A_1，A_2 を議論列とするとき，A_1 が A_2 を defeat するとは，

- **rebut** A_1 に属するルールの結論のいずれか (L_i とする) が，A_2 に属するルールの結論と矛盾する (つまり $\overline{L_i}$ となる)，または
- **undercut** A_1 に属するルールの結論のいずれか (L_i とする) が，A_2 に属する前提のいずれかと矛盾する (つまり $\sim\overline{L_i}$) である

特に A_1 が A_2 を defeat するが，A_2 は A_1 を defeat していない場合，A_1 は A_2 を**厳密に defeat** しているという．

議論列に対しすべての defeat 関係を表現した論証状態グラフを考える．

定義: 論証状態グラフ 議論記述 Π の論証状態グラフ ASG とは，次のように構成されるラベル付き有向グラフ $ASG = (Args, defeat, label)$ である．ここで $Args$ は議論記述 Π を議論列変換により得られる議論列の集合でノード，$defeat$ は defeat 関係でアーク，$label$ はアークに付与したラベルで，その値は rebut または undercut しているリテラル L である．

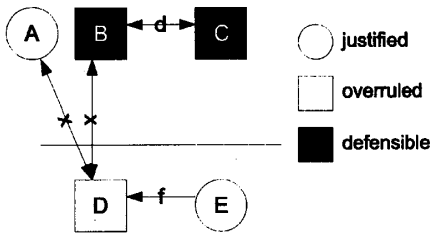

図 2: 図1の議論に対する論証状態グラフ

例: 論証状態グラフ 図1の議論列グラフは図2となる (justified/overruled/defensible については後述)．

論証状態グラフは議論列の対立関係を表している．議論フレームワーク理論 (??など) に倣い議論列に対し justified / overruled / defensible の三値を与える．

定義: acceptable 議論列 A_0 を defeat するすべての議論列が，ある議論列 $A_s \in Args$($Args$ は議論列の集合) により厳密に defeat(defeat の定義を参照) されているとき，議論列 A_0 は $Args$ に対して **acceptable**(受容可能) であるという．

定義: 特性関数 議論列変換されたすべての議論列からなる集合を $Args$ とする．$S \in Args$ とするとき，特性関数 Γ を次のように定義する．

- $\Gamma : Pow(Args) \rightarrow Pow(Args)$
- $\Gamma(S) = \{A \in Args | A \text{ is acceptable w.r.t. S}\}$

$\Gamma(\{\})$ を反復的に適用して，それ以上要素が増えないときに得られる集合 (least fixpoint of Γ) を $JustArgs$ という．

定義: justified/overruled/defensible 議論列 A に対して，justified / overruled / defensible の三値を以下のように割り当てる．

- **justified** $A \in JustArgs$ の場合．
- **overruled** A がある $A_i \in JustArgs$ により defeat されている場合．
- **defensible** A が justified でも overruled でもない場合．

また言明 L に対して，justified / overruled / defensible の三値を以下のように割り当てる．

- **justified** $\exists A \in JustArgs$ s.t. $CONC(A) = L$ の場合．
- **overruled** $\exists A \in JustArgs$ s.t. $CONC(A) = \overline{L}$ の場合．
- **defensible** A が justified でも overruled でもない場合．

例: justified/overruled/defensible 図2の枠の形が justified / overruled / defensible を表している．

3. 論証状態分析

論証状態グラフを用いることで，ある言明の状態 jusified / overruled / defensible (J/O/D) に対し “なぜ J/O/D なのか” を解析することができる．

従来議論フレームワーク理論は，矛盾したルールからなる集合からリテラルの値 (J/O/D) を決定するための方法として用いられてきた．論証状態分析では逆向きに，つまりリテラルの値からその推論理由を明示するために用いる．

具体的な説明として，図 3 の BSE 問題に関する議論を例にとる．

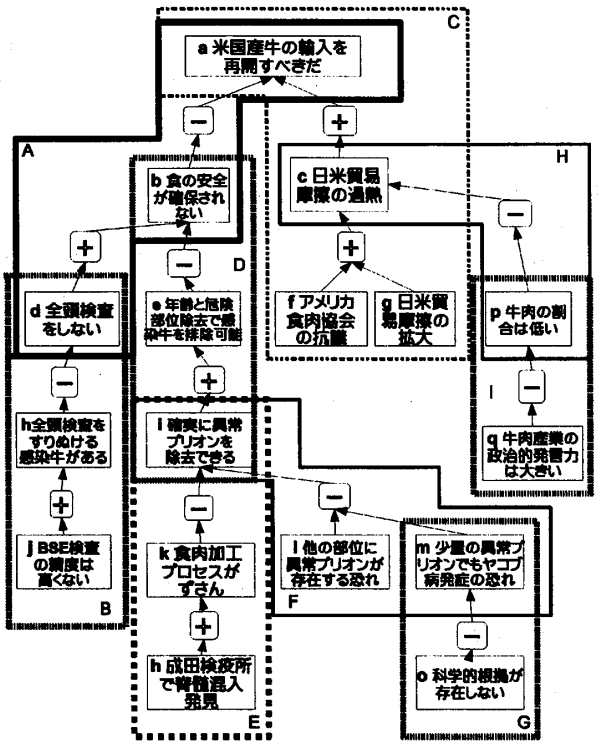

図 3: BSE 問題をめぐる議論記述および議論列変換

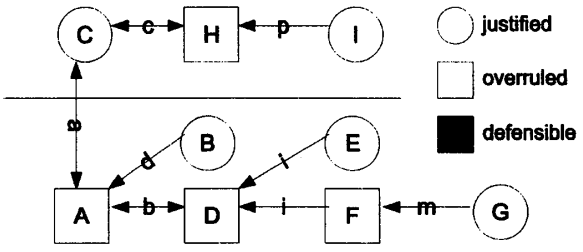

図 4: BSE 問題をめぐる議論の論証状態グラフ

図 4 は図 3 の論証状態グラフである．ここで重要なことは，J/O/D は論証状態グラフのアーク (defeat 関係) のみにより決定される点である．図 3 のあるノードに着目したとき，どうしてそのノードが J/O/D なのかは論証状態グラフより読み取ることができる．

例えば，議論列 A が overruled なのは justified された議論列 B により defeat されているためであり，C が justified なのは対立する H, A が両方とも overruled なためである．また，例えば C を論破 (defeat) しようとするならば，C を defeat する新しい議論列を考えるか，H, A が justified になるよう H, A を defeat している I または B(D は overruled なので関係ない) を defeat すればよいことが分かる．

4. 議論支援機能

ここでは，BSE 問題に関する議論 (図 3,4) を例にとり，論証状態分析を用いた議論支援機能として，要約と，肯定側/否定側のビューの 2 つを紹介する．

4.1 要約

議論には，議題に直接影響しているような重要な箇所と，そうではない枝葉末節的な非重要な箇所が存在する．最も重要な箇所は justified された議題を導く議論である．

要約機能では，このことを論証状態グラフの上で考え，議題の値 (是非) が決定している場合に，なぜその値になったのかを簡潔に説明する．

今議論記述 Π と，Π に対応した論証状態グラフ ASG が与えられたとき，議題を (肯定または否定に) 導く議論が justified されている場合，Π の要約は次の議論からなる集合として定める．

なお，ここでは議題を肯定側に導く議論が justified されているとして記述する．否定側についても同様に定義される．

- justified された肯定議論 (1)
- (1) を defeat する議論 (すべて overruled)(2)
- (2) を defeat する justified された議論 (3)
- (3) を defeat する議論 (すべて overruled)(以下繰り返し)
- 否定議論 (すべての overruled)(4)
- (4) を defeat する justified された議論 (5)
- (5) を defeat する議論 (すべて overruled)(以下繰り返し)

ここで，論点 I の肯定/否定議論とは，$I/\overline{I}$ を結論とする部分議論列 (議論列 $[r_1, r_2, ..., r_n]$ の部分議論列とは議論列 $[r_1, r_2, ..., r_j](j < n)$) を持つ議論のことをいう．

これにより，例えば否定議論を defeat しているが justified されていない議論など，結論に効果を及ぼしていない枝葉末節的な議論が除かれる．

BSE 議論の例では，要約は議論 C, H, I および A, B からなる集合となる．議論 D, E, F, G の箇所はいくら議論を深めても，B が justified されているので結論に対して影響を及ぼさないという意味で，枝葉末節的である．

4.2 肯定側/否定側のビュー

議論は肯定側と否定側に分かれて行われる．肯定側と否定側で補強/反論すべき箇所は当然異なってくる．議論記述は議論の全体像を表しているが，肯定側/否定側にとって補強/反論すべき箇所を表すのがビューである．

ビューは次のように再帰的に定義される．

補強すべき議論列 (S) について 補強すべき議論列 A を defeat している justified または defensible な議論列を「攻撃すべき議論」とする

攻撃すべき議論列 (A) について 攻撃すべき議論列 A を defeat している overruled または defensible な議論列を「補強すべき議論」とする

ただし，再帰的に広げる際の初期条件を次のように定める．

肯定側 補強すべき議論列 (S) は議題を肯定している overruled または defensible な議論列．攻撃すべき議論列 (A) は議題を否定している justified または defensible な議論列．

否定側 補強すべき議論列 (S) は議題を否定している overruled または defensible な議論列．攻撃すべき議論列 (A) は議題を肯定している justified または defensible な議論列．

初期条件で指定される議論を 0 次 (の S/A) とし，以下 1 ステップの再帰ごとに次数を 1 増加させる．

BSE 議論の例では，肯定側は 0 次で (S) なし，(A) なしとなる．これはすでに議題が肯定されているためである．否定側は次のようになる．

- (0 次)(S)A，(A)C
- (1 次)A について (S) なし，(A)B ※1
- (1 次)C について (S)H，(A) なし
- (2 次)H について (S) なし，(A)I
- (2/3 次)B/I について (S) なし，(A) なし

なお※1で，C は overruled されているので (A) に C を含まない点に注意．

実際に表示する場合は，一度に最大次まで表示すると分かりにくいので，ユーザーとのインタラクションで順次階層的に展開するインターフェイスが望ましい．

5. まとめ

議論フレームワークの理論を利用して，非単調論理で記述された議論の分析を試みる手法を提案した．

具体的には，議論列変換や論証状態グラフといった手法により，議論の対立軸や，なぜある議論列が justified / overruled / defensible かといった推論理由が一目で分かるようになった．

さらに，論証状態グラフを分析することにより，要約や肯定側/否定側のビューといった推論以外の議論支援の可能性を示した．

今後の課題として，(1) 議論支援機能をより多様化・精緻化する，(2) 論理の枠組みを拡張しルールの優先度を導入する，(3) 本支援機能を用いて議論する場合議論の時間的・内容的効率化が図られることを示す実験を行う，などを考えている．

参考文献

[1] Dung, P. M.: An Argumentation Semantics for Logic Programming with Explicit Negation., *International Conference on Logic Programming*, pp. 616–630 (1993).

[2] Dung, P. M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming., *International Joint Conferences on Artificial Intelligence*, pp. 852–859 (1993).

[3] Gordon, T. F. and Karacapilidis, N. I.: The Zeno Argumentation Framework, *International Conference on Artificial Intelligence and Law*, pp. 10–18 (1997).

[4] Sartor, G.: A simple computational model for nonmonotonic and adversarial legal reasoning, *ICAIL '93: Proceedings of the 4th international conference on Artificial intelligence and law*, New York, NY, USA, ACM Press, pp. 192–201 (1993).

[5] Verheij, B.: Artificial argument assistants for defeasible argumentation, *Artificial Intelligence*, Vol. 150, No. 1-2, pp. 291–324 (2003).

[6] 柴田裕介，山口和紀：優先度付き LA 推論に基づいた議論支援システム P++，第 45 回情報処理学会全国大会 (2006).