多値論理 EALP の知識表現による一貫性制約を用いた議論の意味論の計算

深山 竜太[1]　　龍沢 昌宏[2]　　若木 利子[3]
芝浦工業大学 システム工学部 電子情報システム学科*

1 はじめに

近年，複数の人間による議論をマルチエージェント (知的ソフトウェア) にシミュレートさせるために，*Dung* により 4 つの議論の意味論が提案された．この議論の意味論構築の為に，これまでエージェントの知識表現に拡張論理プログラム (*ELP*)[1] 等が用いられている．ところで現実の人間社会で行う議論では，しばしば「この映画は面白いともいえるし，面白くないともいえる」といった一見矛盾と思える知識や，「この映画は 90%程度面白い」といった不確実な知識を議論に用いているが，*ELP* 等による知識表現では，このような知識を扱えない．このような知識は準無矛盾論理や多値論理で研究されている．最近，新潟大学の沢村らがリテラルに注釈を付けた *EALP* 言語を用いて，このような知識の表現を試み，その上で対話的証明論 (即ち，単一 *extension* の *grounded* な意味論) に基づくエージェントの議論の方法の提案とシステム実装を行っている [2] が，複数 *extensions* を持つ *preferred* や *stable* 意味論に基づく議論の計算システム実装は行われていない．また，*EALP* の一貫性制約を扱う議論の意味論とその計算方法が提案されていない．本研究では，エージェントの知識を *EALP* で表現しても，*grounded semantics* 以外の *Dung* の意味論に基づく議論判定の必要性があることを示すと共に，一貫性制約を議論判定に用いる方法を提案し，それらを実装した *EALP* 上の議論計算エンジンの開発を行った．

2 多値論理による知識表現と議論

本研究ではエージェントの知識を，*ELP* に注釈を付けることでより高い表現能力を有する *EALP* で表現する．

定義 1 (注釈の完備束と注釈付リテラル) 注釈 (真理値) の完備束を $(\mathcal{T}, \leq)$ とする．原子式 A, 注釈 $\mu \in \mathcal{T}$ とした時，$A{:}\mu$ は注釈付原子式であり，注釈付原子式とその存在論的否定 ($\sim A{:}\mu$) を注釈付リテラルという．また認識論的否定 ($\neg A{:}\mu$) で $\neg A{:}\mu = A{:}\neg\mu$ が成り立つ．

定義 2 (拡張注釈付論理プログラム)
拡張注釈付論理プログラム *EALP* P は次の形の注釈付規則 r の集合である．

$$L_0 \leftarrow L_1, ..., L_m, not\ L_{m+1}, ..., not\ L_n \quad (1)$$

ここで *not* はデフォルト否定である．$L_i (0 \leq i \leq m)$ は注釈付リテラル，$not\ L_i (m+1 \leq i \leq n)$ は注釈付デフォルトリテラルである．L_0 を r の頭部，L_1 , ... , $not\ L_n$ を r の本体という．$body^- = \{L_{m+1}, ..., L_m\}$ とする．

定義 3 (注釈付論証) *EALP* P の注釈付論証とは (1) の形の規則の有限系列 $Ag{=}[r_1, ..., r_n]$ である．但し，全ての $i (1 \leq i \leq n)$ で r_i が P の規則もしくは極小還元 (詳細は省略) であり，r_i の本体にある全ての注釈付原子式 $A{:}\mu$ に対し，頭部が $A{:}\rho (\rho \geq \mu)$ となるような $r_k (n \geq k > i)$ が存在する．また，r_i の本体にある全ての存在論的否定 $\sim A{:}\mu$ に対し，頭部が $\sim A{:}\rho (\rho \leq \mu)$ となるような $r_k (n \geq k > i)$ が存在する．r_1 の頭部を Ag の *claim* と呼ぶ．

定義 4 (攻撃関係) 攻撃関係 def_P は論証集合 $Args_P$ 上の 2 項関係 ($def_P \subseteq Args_P^2$) として定義し，*rebut*, *undercut*, *attack* 等が存在する [2].

定義 5 (反論:*rebut*)
Ag_1 が Ag_2 を反論する ($i.e.(Ag_1, Ag_2) \in r$) $\overset{def}{\Leftrightarrow} \mu_1 \geq \mu_2$ となる $A : \mu_1 \in concl(Ag_1)$ と $\sim A : \mu_2 \in concl(Ag_2)$ が存在する．但し，$concl(Ag)$={$L | L$ は $r \in Ag$ の頭部 }.

Computing Constrained Argumentation Semantics built on EALPs handling integrity constraints
[1]Ryuta Miyama, [2]Masahiro Tatsuzawa, [3]Toshiko Wakaki
*Shibaura Institute of Technology

定義 6 (無効化:*undercut*)
Ag_1 が Ag_2 を無効化する ($i.e.(Ag_1, Ag_2) \in u$) $\overset{def}{\Leftrightarrow} \mu_1 \geq \mu_2$ となる $A : \mu_1 \in concl(Ag_1)$ と $not\ A : \mu_2 \in assm(Ag_2)$ が存在するか，$\mu_1 \leq \mu_2$ となる $\sim A : \mu_1 \in concl(Ag_1)$ と $not\ \sim A : \mu_2 \in assm(Ag_2)$ が存在する．但し，$assm(Ag) = \{L | L \in body^-(r)\ for\ r_i \in Ag\}$

定義 7 (攻撃:*attack*)
Ag_1 が Ag_2 を攻撃 (*attack*) する ($i.e.(Ag_1, Ag_2) \in a$) $\overset{def}{\Leftrightarrow} Ag_1$ が Ag_2 を反論するか，Ag_1 が Ag_2 を無効化する．

定義 8 (議論の意味論 [3]) *EALP* P に関する議論フレームワーク $AF_P{=}(Args_P, def_P)$ において，$S \subseteq Args_P$ を *conflict-free* な論証集合，単調関数 F を $F : 2^{Ar} \rightarrow 2^{Ar}$, $F(S) = \{a | a$ は S に関して *acceptable*$\}$ とする．議論の意味論 *complete/grounded/preferred/stable semantics* は次に定義する *complete/grounded/preferred/stable extension* E によって与えられる．

- E が *complete extension* $\overset{def}{\Leftrightarrow} E = F(E)$
- E が *grounded extension* $\overset{def}{\Leftrightarrow} E$ は *complete extension* の中で $\subseteq$ に関して極小な論証集合である
- E が *preferred extension* $\overset{def}{\Leftrightarrow} E$ は *complete extension* の中で $\subseteq$ に関して極大な論証集合である
- E が *stable extension* $\overset{def}{\Leftrightarrow} E$ は *preferred extension* の中で $Args_P \setminus E$ の任意の論証を攻撃する論証集合である

3 制約付き議論フレームワークの意味論

本研究で提案する一貫性制約を扱える議論の意味論と論証の正当化による議論の判定を以下に示す．

定義 9 (制約付き議論フレームワーク *EALP* $P \cup C$ に関する制約付き議論フレームワーク: $CAF(P, C)$ は以下で定義される．

$$CAF(P, C) \overset{def}{=} (Args_P, attack_P, C)$$

但し，P は (1) の形式の規則集合，C は (2) の形式，即ち一貫性制約の規則 r_{ic} の集合とする．

$$\leftarrow L_1, ..., L_m, not\ L_{m+1}, ..., not\ L_n \quad (2)$$

定義 10 (制約の充足)
A, B をリテラルとする．$E \subseteq Args_P$ なる E について，以下で定義する $claim(E)$ を用いて制約充足を定義する．

$$claims(E) = \{L | L = claim(a)\ for\ a \in E\}$$

1. $E \subseteq Args_P$ で，E は C に違反する
   $\overset{def}{\Leftrightarrow}$ 次のことが成り立つ一貫性制約 r_{ic} が存在する．
   (a) $L_i = A : \rho$ について，$claims(E) \ni A : \mu$ (但し，$1 \leq i \leq m$, $\rho \leq \mu$)
   (b) $L_i = \sim A : \rho$ について，$claims(E) \ni \sim A : \mu$ (但し，$1 \leq i \leq m$, $\mu \leq \rho$)
   (c) $L_i = B : \rho$ について，$claims(E) \ni B : \mu$ (但し，$m \leq i \leq n$, $\rho \leq \mu$)
   (d) $L_i = \sim B : \rho$ について，$claims(E) \ni \sim B : \mu$ (但し，$m \leq i \leq n$, $\mu \leq \rho$)
2. E が C に違反しないなら，E は C を満たすという．

CAF の意味論は以下の *C-extensions* で与えられる．

定義 11 (*C-extensions*)
Sname は *complete, preferred, grounded, stable* のいずれかを表す．*Sname* 意味論の下で，E は *EALP* $P \cup C$ に関する制約付き議論フレームワーク: $CAF(P, C)$ の *C-extension* であるのは「E は議論フレームワーク AF_P における *Sname* 意味論の *extension* であり，かつ C を満たす」ことであり，その時に限る．

定義 12 (*skeptically/credulously* に正当化)
論証 $Ag \in Args_P$ について，$CAF(P, C)$ における *Sname* 意味論の全ての (或る) *C-extension* E において $Ag \in E$ である時，Ag は $CAF(P, C)$ に対して *skeptically(credulously)* に正当化されるという．

4 議論の例題

例 1 *EALP* での *preferred* 意味論での議論の有用性

「プロジェクトは何日に終了するか？」を議題とし，エージェントは以下の知識を持つとする．

> A が 3,4,5 日に働き 5 日に部品が届けば 6 日にプロジェクトは終わる．
> A が 3,4,6 日に働き 6 日に部品が届けば,7 日にプロジェクトは終わる．
> 部品が 5 日に届くか 6 日に届くかのいずれかである．
> 5,6,12,13 日は休日である．A は 3,4,5,6 日に働く．
> 5 日が休日で B が 5 日に働く事実がなければ A は 3,4,5 日に働かない．
> B は 12,19,26 日以外は働く．

この例で注釈 *(*真理値*)* の完備束 $(\mathcal{P}(\{1,...,31\}),\subseteq)$ とする．注釈は時相である．*EALP* K で以下の知識ベースを表現し，K で構成される論証集合を $Args_K$ とする．

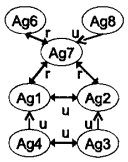
K

$finish(project)$:{6} ← $work(A)$:{3, 4, 5}, $arrive(comp)$:{5},
$finish(project)$:{7} ← $work(A)$:{3, 4, 6}, $arrive(comp)$:{6},
$arrive(comp)$:{5} ← $not\ arrive(comp)$:{6},
$arrive(comp)$:{6} ← $not\ arrive(comp)$:{5},
$holiday$:{5, 6, 12, 13} ←,
$work(A)$:{3, 4, 5, 6} ←,
∼ $work(A)$:{3, 4, 5} ← $holiday$:{5}, $not\ work(B)$:{5},
¬$work(B)$:{12, 19, 26} ←

$Args_K$

Ag_1 = [$finish(project)$:{6} ← $work(A)$:{3, 4, 5}, $arrive(comp)$:{5};
$work(A)$:{3, 4, 5, 6} ← ; $arrive(comp)$:{5} ← $not\ arrive(comp)$:{6}]
Ag_2 = [$finish(project)$:{7} ← $work(A)$:{3, 4, 6}, $arrive(comp)$:{6};
$work(A)$:{3, 4, 5, 6} ← ; $arrive(comp)$:{6} ← $not\ arrive(comp)$:{5}]
Ag_3 = [$arrive(comp)$:{5} ← $not\ arrive(comp)$:{6}]
Ag_4 = [$arrive(comp)$:{6} ← $not\ arrive(comp)$:{5}]
Ag_5 = [$holiday$:{5, 6, 12, 13} ←]
Ag_6 = [$work(A)$:{3, 4, 5, 6} ←]
Ag_7 = [∼ $work(A)$:{3, 4, 5} ← $holiday$:{5}, $not\ work(B)$:{5};
$holiday$:{5, 6, 12, 13} ←]
Ag_8 = [¬$work(B)$:{12, 19, 26} ←]

Ag_8 は結論として ¬work(B):{12,19,26} を持ち，Ag_7 は前提としてnot work(B):{5}を持つので Ag_8 は Ag_7 に無効化の攻撃をする．Ag_1 と Ag_7 はお互いの結論にwork(A):{3, 4, 5, 6}，∼$work(A)$:{3, 4, 5} を持ち，{3, 4, 5} ⊆ {3, 4, 5, 6} より反論関係が成り立つ．Ag_2 と Ag_7 でも同様に成り立つ．$Args^2_K$ 上の攻撃関係 def に *attack* を用いると議論フレームワークは右図，*complete extensions* は以下の E_1, E_2, E_3 になる．*preferred extensions* は E_1, E_2, *grounded extension* は E_3 のみ．



$E_1 = \{Ag_1, Ag_3, Ag_5, Ag_6, Ag_8\}$
$E_2 = \{Ag_2, Ag_4, Ag_5, Ag_6, Ag_8\}$
$E_3 = \{Ag_5, Ag_6, Ag_8\}$

skeptically に正当化されるのは Ag_5, Ag_6, Ag_8 であり，議題に関する論証 Ag_1, Ag_2 は含まれていないので，先行研究にあるような *grounded semantics* の計算では，プロジェクトの終了日が不明である．しかしここで *preferred semantics* による計算を行うと *credulously* に正当化される議論判定が可能となり，部品が 5 日に届けばプロジェクトは 6 日に終わる $(Ag_1 \in E_1)$，或は部品が 6 日に届けばプロジェクトは 7 日に終わる $(Ag_2 \in E_2)$ という結論が得られる．このように *EALP* の議論での *preferred semantics* の計算の必要性が示される．

例 2 一貫性制約を用いた意味論での計算例

「A さんは明日 80%の確率で車で出かけるか?」を議題とし，エージェントは以下のような知識を持つとする．

> A さんは 80%の確率で明日雨が降るということが分からなければ，
> 車で出かける．B さんは 100%車で出かける．
> 70%雲行きが怪しければ，明日は 80%雨が降る．
> 70%晴れの予報があれば，明日は 60%雨が降らない．
> 100%雲行きが怪しい．80%晴れの予報がある．

この例で注釈 (真理値) の完備束を $([0,1],\leq)$ とする．*EALP* P で以下の知識ベースを表現し，P で構成される論証集合を $Args_P$ とする．

P

$go(car, A) : 0.8 \leftarrow not\ rain(tomorrow) : 0.6$,
$go(car, B) : 1.0 \leftarrow$,
$rain(tomorrow) : 0.8 \leftarrow cloud(many) : 0.7$,
$\sim rain(tomorrow) : 0.6 \leftarrow forecast(fine) : 0.8$,
$cloud(many) : 1.0 \leftarrow$, $forecast(fine) : 1.0 \leftarrow$

$Args_P$

Ag_1 = [$go(car, A) : 0.8 \leftarrow not\ rain(tomorrow) : 0.6$],
Ag_2 = [$go(car, B) : 1.0 \leftarrow$],
Ag_3 = [$rain(tomorrow) : 0.8 \leftarrow cloud(many) : 0.7; cloud(many) : 1.0 \leftarrow$],
Ag_4 = [$\sim rain(tomorrow) : 0.6 \leftarrow forecast(fine) : 0.7; forecast(fine) : 0.8$],
Ag_5 = [$cloud(many) : 1.0 \leftarrow$], Ag_6 = [$forecast(fine) : 1.0 \leftarrow$]

議論の結果，以下の 2 個の *preferred extensions* が得られた．$AF_P = (Args_P, def_P)$ *(*但し,def_P は *attack)* の下で，議題に関する *claim* を持つ論証は Ag_1 である．

$E_1 = \{Ag_1, Ag_2, Ag_4, Ag_5, Ag_6\}$
$E_2 = \{Ag_2, Ag_3, Ag_5, Ag_6\}$

結果から，$Ag_1 \in E_1$ より，場合によっては A さんは車で出かけることが出来ると判定される．ここで，以下のような一貫性制約 r_{ic} が C に存在する．

$$C = \{\leftarrow go(car, A) : 0.8,\ go(car, B) : 0.8\ \}$$

この一貫性制約は，*(*車が 1 台しかないので*)*A と B は同時に車で出かけることが出来ないという意味を示す．ここで AF_P における議論の *Extensions* を見ると，Ag_1 と Ag_2 を同時に含む E_1 は C を満たさないので，E_1 は $CAF(P, C)$ の *C-extension* として認められないが，E_2, は *C-extension* として認められることが分かる．E_2 には Ag_1 が含まれていないので，$CAF(P, C)$ で *credulously* に正当化されず，A さんは 80%の確率で車で出かけるとはいえないと判定される．

5 実装

EALP 上で一貫性制約を用いて議論の意味論の計算を行うシステムを *JAVA* により実装した．本システムでは，はじめに *EALP* で表現された知識ベースと一貫性制約が記述されたテキストファイルを入力とし，注釈の完備束の種類，攻撃関係を設定した後，議論フレームワークを作成する．作成されたルール集合，論証集合，攻撃関係はオプションで表示することができる．その後意味論，判定方法，議題，使用する一貫性制約を設定した後に計算を開始し，議題の判定結果と *Extensions* が出力される．*Extensions* の計算には解集合プログラミングによって議論意味論の計算を行うツール [4] を一貫性制約を扱えるように拡張したものを使用する．図 1 に入力画面，図 2 に論証集合と議題の判定結果の出力画面を示す．

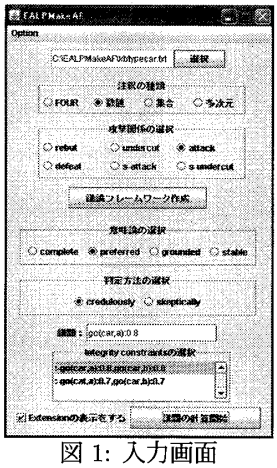


図 1: 入力画面

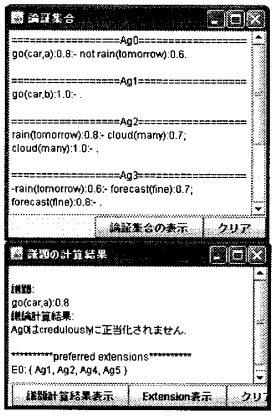


図 2: 結果出力画面

6 おわりに

EALP 上で 4 つの意味論に基づく議論判定を一貫性制約を用いて行うことが可能なプログラムを実装した．今後はツールを用いた応用システムを構築し，有用性の評価を行う．

参考文献

[1] R. Schweimeier, M. Schroeder: A Parameterised Hierarchy of Argumentation Semantics for Extended Logic Programming and its Application to the Well-founded Semantics,Theory and Practice of Logic Programming, pp.207-242, 2005.

[2] Y.Takahashi H.Sawamura:A Logic of Multiple-Valued Argumentation. Proc. ofAAMAS 2004, pp.800-807, 2004.

[3] P. M. Dung: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence, 77, pp.321-357, 1995.

[4] 伊藤，若木：解集合プログラミングによる議論の意味論の計算，情報処理学会第 70 回全国大会，2008.