矛盾を扱える論理プログラミングシステム ALPS-HI

7M－2

合志和晃 程京徳 牛島和夫
九州大学 大学院 システム情報科学研究科

1. はじめに

複数の知識源から得た知識には矛盾が存在しうる。したがって、知識処理システムは、矛盾を扱える必要がある。矛盾を扱うには２つの方法がある。一つは、知識中に矛盾の存在を許さず矛盾のない状態で知識を利用する方法である。もう一つは、知識中の矛盾の存在を許し矛盾が存在しても知識を利用しつつ、可能ならば矛盾を取り除いていこうとする方法である。前者は知識利用の観点からみれば理想的である。しかし現実の知識は矛盾を含み得るものであるため常に矛盾のない状態に保てるとは限らない。このため、より実際的な知識を扱おうとするならば後者が適している。従って我々は後者の方法を採用する。

知識処理システム構築の方法として論理プログラミングによるアプローチがある。矛盾を扱える論理プログラミングシステムへの基本要求は以下のものである。(1) 矛盾の扱いに適したプログラミング言語。(2) 矛盾に関する情報を提供できる矛盾解析ツール。

矛盾を扱える論理プログラミングシステムとして、注釈付き論理プログラミング(Annotated Logic Programming)が提案され研究されている[1]。しかし、上記の要求を満たすものではない。我々は、**ALPS-HI**(Annotated Logic Programming System with Hypothetical Implication)を試作した。

2. 概要

ALPS-HI は以下の機能を持つ。

(1) 矛盾の扱いに適したプログラミング言語

注釈により矛盾を記述することができる。矛盾を扱うメタ知識を記述することができる。仮説的含意により状況の変化が記述できる。

(2) 矛盾に関する情報提供ができる矛盾解析ツール

複数の矛盾を起こす節が矛盾の原因の可能性が高いという考えに基づき原因探索を行う[2]。

(3) 矛盾に関する情報の視覚化ツール

ユーザは、矛盾の原因の可能性が高い節を、グラフ化した証明木の上で検討ができる。

3. 文法と計算

$\mathfrak{I}$ = {none, true, false, both}を四値論理の真理値の集合とする。本稿では、それぞれの真理値を表す注釈の記号として n, t, f, b を用いる。

ALPS-HIプログラムの文法は以下の通りである。ここでA = { n, t, f, b } A' = { t, f }である。

Definition 1. もし l がリテラル(literal)ならば $l:\mu$ は注釈付きリテラル(annotated literal)である。ここで$\mu \in$ Aでありμはlの注釈である。

Definition 2.

(1) もしLが注釈付きリテラルならば、Lは**ALPS-HI** 式(**ALPS-HI** formula)である。

(2) もし L_0 が注釈付きリテラルで $F_1, F_2, \ldots, F_n$ が **ALPS-HI** 式ならば、$(L_0 :- F_1, F_2, \ldots, F_n):\mu'$は **ALPS-HI** 式である。ここで、$\mu' \in$ A'である。

(3) 以上のもののみが**ALPS-HI**式である。

Definition 3. もしFが**ALPS-HI**式ならば、$F.$は**ALPS-HI**節(**ALPS-HI** clause)である。

Definition 4. もしLが注釈付きリテラルで$F_1, \ldots, F_n$が**ALPS-HI**式ならば、$L:-.$(or $L.$)は事実(fact)であり、$L:-F_1, F_2, \ldots, F_n.$は規則(rule)であり、$:-F_1, F_2, \ldots, F_n.$ (or $?-F_1, F_2, \ldots, F_n.$)は質問(query)である。

Definition 5. ALPS-HI節の有限集合が**ALPS-HI**プログラムである。

ALPS-HI プログラムの計算は、基本的には Prolog プログラムと同様である。注釈を扱うために、クローニング(cloning)と失敗としての未知(unknown as failure)を含む[2]。また、相関論理プログラミング[3]の手法に基づいて仮説的含意を扱う。

ALPS-HI: An Annotated Logic Programming System with Hypothetical Implications
Kazuaki Goshi, Jingde Cheng, and Kazuo Ushijima
Department of Computer Science and Communication Engineering
Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-81 JAPAN

4. 実装

我々は、MS-Windows95/NT 上で SWI-Prolog を用いて **ALPS-HI** を実装した。基本的な原理は注釈を引数として扱うことである。ALPS-HI はシェル、節コンバータ、ソルバー、ディテクタ、ビューアの５つの部分からなる。シェルはユーザとのインターフェースである。節コンバータは **ALPS-HI** のプログラムを Prolog のプログラムに変換する。ソルバーは導出を行う。ディテクタは矛盾の原因個所の探索を行う。ビューアは矛盾についての情報を表示する。ビューアのみ Delphi(Object Pascal)を用いて実装し、TOutline のクラスにより証明木の表示を行う。**ALPS-HI** の主な部分は Prolog で記述してあるので **ALPS-HI** は移植性が高い。

5. 例

仮説的含意を含む質問を使った知識修正の例である。鳥は飛ぶが、鳥であるペンギンは飛ばないという矛盾を扱う。システムは知識の自動修正は行わない。知識の修正には人間の判断が必要である。

(1) (fly(X):t :- bird(X):t):t.　(2) bird(penguin):t.
(3) fly(penguin):f.

もし、ユーザが fly(penguin):t について質問すると当然矛盾である。

```
??-fly(penguin):t.
success
fly(penguin):t []
-Contradiction Check?(y/n)>y
*** Contradiction *** both fly(penguin):t
fly(penguin):f exist.
-(;,[Enter],i,t)->tInformation (File)
```

ユーザは矛盾についての情報をビューアで見ることができる。(図 1)

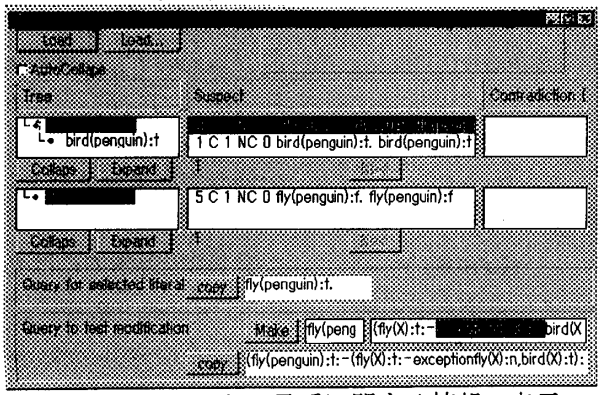

図 1 ビューアによる矛盾に関する情報の表示

システムは矛盾の原因である可能性が高い順に節のリストを表示する。ここで、もし、以前に他の飛べない鳥(例えばダチョウ)について質問があればシステムは(1)が矛盾の原因の可能性が高いと示す。ユーザはその情報を知識修正の際に参考にすることができる。さらに、ビューアは、修正の対象となる節を条件として含んだ質問を表示するので、ユーザはその質問を再度質問する際に利用できる。ここでは、exception_fly というリテラルの導入によって例外を記述し矛盾を取り除くことを試みる。

```
??-paste.
  (fly(penguin):t:-(fly(X):t:-
exception_fly(X):n,bird(X):t):t):t.
success
```

もちろん、この質問は依然として成功する。なぜならペンギンが例外であると付け加えていないからである。さらに追加し質問を行う。

```
??-paste.
 (fly(penguin):t:-(fly(X):t:-
exception_fly(X):n,bird(X):t):t,exception_fly(peng
uin):t):t.
fail
```

質問は失敗し、ユーザは修正が正しいと知る。このように **ALPS-HI** は矛盾が起こった際にユーザの矛盾への対処を支援することができる。

6. おわりに

本稿では、新しい注釈付き論理プログラミング **ALPS-HI** について説明した。**ALPS-HI** は矛盾を含みうる知識処理システムのプログラミングに有効である。

参考文献

[1] H. A. Blair and V. S. Subrahmanian, "Paraconsistent Logic Programming," Theoretical Computer Science 68, pp. 135-154, 1989.

[2] K.Goshi, N.Sakamoto, J.Cheng, and K.Ushijima, "Improving the Inconsistency Processing in Annotated Logic Programming," Proc. of SPICIS'94, pp. 243-248, 1994.

[3] A.W.Bollen, "Relevant Logic Programming," Journal of Automated Reasoning 7, pp. 563-585, 1991.