論証支援システム EUODHILOSのための構文記述法の改良とパーザの実現

大橋恭子　南俊朗　沢村一　大谷武
(富士通株式会社)

1. はじめに

今日，計算機科学や人工知能の分野で，計算機による問題解決や支援の研究がなされている．論理を用いた問題解決は，形式的な手続きによって行うことができるため，有効であることが認められている．

問題解決を論理の下で行うには，解決する問題に適した論理系の下で行うことが望ましいが，適切な論理系は問題ごとに異なる．論理系ごとに支援システムを構築するのは，その度ごとに類似の努力を要し，無駄が多い．そのため，論理系を定義するためのメタな枠組みを提供し，ユーザの定義した論理系の下での支援を行うようなシステムが有効である．我々は，このような要求を満たす汎用論証支援システム EUODHILOS[1]を開発した．

EUODHILOSは，この目的を満たすため，ユーザが言語を定義する機能と，その定義から内部構造変換機能付きパーザとアンパーザを自動生成する機能を有している[2]．

言語の定義に用いる構文記述法は，確定節文法（ＤＣＧ）[3]に構成子定義を加えたものである．構成子定義とは，項や式において演算子や述語記号，関数記号として用いる記号の定義，および，演算子の優先順位を定義を行うものである．図1にMartin-Lofの型理論の構文規則を示す．ＤＣＧには，書きやすい，文脈に依存した情報も書ける，Prologで記述したパーザに変換する手法[4]が知られている，といった特長がある．

```
judgement --> term,"∈",type;
term --> bind_op,variable,".",term1 | term1;
term1 --> term1,"・",term2 | term2;
term2 --> "(",term,")" | or_intro,"(",term,")";
......
bind_op --> "λ";
or_intro --> "inl" | "inr";
......
type --> "(",type,"⊃",type1,")" | type1;
type1 --> "(",type1,"∨",type2,")" | type2;
type2 --> "～",type2;
......
operator
  "～","∨","⊃",bind_op,"∈";
predicate
  or_intro, …
```

図1　Martin-Lofの型理論の構文規則

2. 構文記述法の改良

1節で示した構文規則と構成子定義を実際に記述する時，ユーザは次のことに注意しなければならない．

・構文規則を記述する時に，適当な非終端記号を用いて，演算子の優先順位や結合性を表す．

・演算子の優先順位を構成子定義でも（構文規則と矛盾しないように）記述する．

適当な非終端記号を導入しなければならないことにより，

①構文規則の記述量（構文規則の数）が増える．

②ユーザのイメージする構文を素直に表したものと言えない．

という記述上の問題が生じた．また，演算子の優先順位を構文規則と構成子定義で記述しなければならないので，

③両者の間に不整合の起きる可能性がある．

という問題が生じた．[5]

これらの問題を改善，あるいは，解決するために構文記述法を改良した．図1に示した構文規則を，改良した記述法を用いて書き換えると，図2のようになる．この構文規則では，構成子定義で優先順位と結合性の定義を行い，構文規則でのそれらの記述を不要にした．図2の構成子定義では，演算子の後ろに，左結合ならばleft，右結合ならばright と記述することより結合性を表している．優先順位についてはこれまでと同様である．構文規則には，曖昧さが生じるが，構成子定義を参照することによって取り除くことができる．

```
judgement --> term,"∈",type;
term --> bind_op,variable,".",term1
       | term,"・",term
       | "(",term,")"
       | or_intro,"(",term,")";
......
bind_op --> "λ";
or_intro --> "inl" | "inr";
......
type --> "(",type,"⊃",type1,")"
     | "(",type1,"∨",type2,")"
     | "～",type2;
......
operator
  "～","∨":left,"⊃":left,
        "・":left,bind_op,"∈";
predicate
  or_intro,
```

図2　改良したMartin-Lofの型理論の構文規則

Improved syntax rules and implementation of the parser for reasoning assistant system EUODHILOS

Kyoko OHASHI, Toshiro MINAMI, Hajime SAWAMURA, Takeshi OHTANI
Fujitsu, Ltd.

3. パーザの実現

本節では，2節で示した構文規則と構成子定義からパーザを作成する方法について述べる．本システムで用いているパーザは，ボトム・アップ・パーザのＢＵＰ[4]である．

図2の構文規則では演算子の優先順位や結合性を記述していない．それらは，構成子定義で定義されている．構成子定義から適当な数を得て，それと非終端記号を組み合わせることによって，曖昧さのないＢＵＰパーザを作成する．その数は，非終端記号の引数に付加する．

以下，ＢＵＰパーザを作成するのに必要な，構文規則をＢＵＰ形式のProlog節へ変換する時に非終端記号に付加する数の決め方を(1)，停止条件節，および，link節については(2)，(3)で述べる．また，図2に示した構文規則を用いることにより，新たに作成しなければならなくなった節（以後，優先順位下降節と呼ぶ）がある．この節については，(4)で述べる．

(1)ＢＵＰ形式のProlog節への変換

まず，以下の説明に必要な用語を定義しておく．「非終端記号NTの最高（最低）の優先順位」とは，NTが左辺の構文規則の右辺に含まれる構成子の優先順位の中で，最も高い（低い）優先順位である．

ここでは，述語記号や関数記号の優先順位を０としている．また，右辺に構成子の含まれる構文規則がない場合には，両方とも０とする．

次に，非終端記号に付加する数の決定方法を示す．

①左辺の非終端記号

・右辺に構成子があるならば，構成子の優先順位．
・そうでなければ，左辺の非終端記号の最高の優先順位．

②右辺の非終端記号

・左辺の非終端記号と等しいならば，右辺に含まれる構成子の定義に基づいて，以下のように決める．
  (a)構成子が優先順位Ｐで，左（右）結合の場合，
    ・構成子よりも前（後）にあるならば，Ｐ．
    ・構成子よりも後（前）にあるならば，Ｐよりも１高い優先順位，すなわち，Ｐ－１．
  (b)構成子が優先順位Ｐで，結合性が未定義ならば，Ｐ．
・そうでなければ，非終端記号の最低の優先順位．

```
例1) term --> bind_op,variable,".",term;
  ⇒  bind_op(P,G,Gp,[A1],X,S1,Sn):-
         link(term,G),
         goal(variable,0,A2,S1,["."|S2]),
         goal(term,5,A3,S2,S3),
         term(5,G,Gp,[A],X,S3,Sn);
例2) term --> term,"・",term;
  ⇒  term(P,G,Gp,[A1],X,["・"|S1],Sn):-
         link(term,G),      P ≦ 4,
         goal(term,3,A2,S1,S2),
         term(4,G,Gp,[A],X,S2,Sn);
例3) term --> variable;
  ⇒  variable(0,G,Gp,[A],X,S1,Sn):-
         link(term,G),   term(0,G,Gp,[A],X,S1,Sn);
  例4) bind_op --> "λ";
  ⇒dict(bind_op,0,["λ"|S1],Sn,G,Gp):-
         link(bind_op,G);
```

(2) 停止条件節の生成法

通常のＢＵＰと同様に作成する．還元された非終端記号とゴールの非終端記号に付加する数も等しくする．

(3) link節の生成法

通常のＢＵＰと全く同様に作成する．link節の非終端記号には優先順位を示す数字は，以下の理由により不要である．

・異なる非終端記号とのlink関係を調べるときには，優先順位は不要である．
・等しい非終端記号とのlink関係は，必ず作成するlink(A,A)のパターンで調べられる．演算子の優先順位と結合性の関係によっては，link関係の無いことがあるが，それは，構文規則を変換した節でチェックしているのでlink節で調べる必要はない．

(4) 優先順位下降節

これは図1に含まれていた以下の構文規則に相当する節である．

```
term1 --> term2;
```

他の節にマッチしないときにのみ使われ，引数に付加する優先順位を下げるのと同様の効果を持つ．この節がないと，目的とする非終端記号と還元された非終端記号が等しいが優先順位が異なる場合に，構文解析が失敗してしまう．

この節は，還元された非終端記号に付加した数に１を加えて，再びProlog節を呼び出すように作成する．ただし，還元された非終端記号の最低の優先順位より大きな数になったら，失敗させる．

4. おわりに

論証支援システム EUODHILOSにおける構文記述法の改良と，その記述法による構文規則からＢＵＰパーザを生成する方法について報告した．改良した記述法を用いると，構文規則の数が減り（図2の例では，５個（18%）減），また，記述する際に注意すべきことが減るので，ユーザの負担を軽減することができた．

なお，本研究は，新世代コンピュータの開発の一環としてＩＣＯＴの委託によって行ったものである．

《参考文献》

[1] Sawamura,H.,Minami,T.,Yokota,K. and Ohashi,K.: Potential of General-Purpose Reasoning Assistant System EUODHILOS, SOFTWARE SCIENCE AND ENGINEERING-SELECTED PAPERS FROM THE KYOTO SYMPOSIA, pp.164-188, 1991.

[2] 大橋，横田，南，沢村，大谷：確定節文法のための内部構造変換機能付きパーザとアンパーザの自動生成方式，情報処理学会論文誌，第31巻第11号，pp.1616-1626,1990.

[3] Pereira, F.C.N., et al.: Definite Clause Grammars for Language Analysis -A Survey of the Formalism and a Comparison with Augmented Transition Networks, AI Journal 13, pp.231-278, 1980.

[4] 松本，田中，平川，三吉，安川，向井，横井：Prologに埋め込まれたボトム・アップ・パーザ：ＢＵＰ,Proc. of the Logic Programming Conf, '83, pp.1-9,ICOT, 1983.

[5] Minami,T.,Sawamura,H.,Ohashi,K.,Yokota,K.: On Designing the Method of Syntax Descriptions for Logics, 情報処理学会39回（平成元年度後期）全国大会講演論文集，pp.277-278, 1989.