3Z-7

論証支援システムのための論理式エディタ

土屋恭子　佐藤かおる　小野越夫　沢村一　南後朗
（富士通株式会社）

1. はじめに

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．そこでは，問題解決の方法として論理学的方法論の有効性が認められている．

問題を扱うためには，問題をモデル化しなければならない．またこのモデルは問題に応じて多種多様に存在する．そこで，様々なモデル上での問題解決が必要である．

本論証支援システムは，論理式によって表現される論理モデルに関する演繹的な証明を支援する．本システムでは，ユーザが対象とする論理モデルのための論理系の定義をすることによって，その論理系での論証を可能にする．

汎用の論証支援システム［１］［２］には，論理系の定義の支援，証明を行うのに必要な論理式の入力と編集の支援，入力した論理式に対する証明の支援の３つの支援が必要である．

本論文では汎用の論証支援システムの入力部である論理式エディタ［３］［４］の現状と今後の展開について述べる．

2. 論理式エディタとは

論理式は一般に複雑な構造を持つことが多い．論理式エディタは汎用の論証支援システムの入力を容易にするためのものである．そのような論理式エディタの実現には，次の３つの機能が検討されなければならない．

(1) 論理系の記号・構文の定義機能．
(2) パーザの生成機能．
(3) 論理式編集機能．

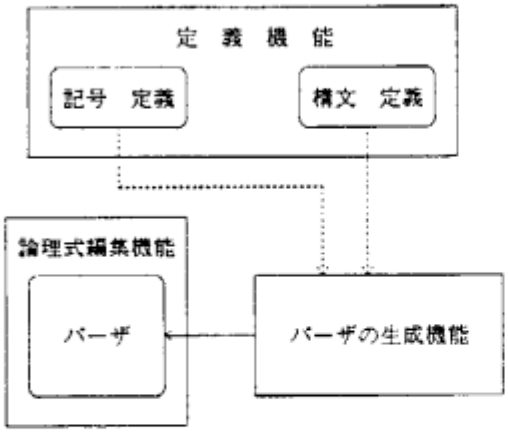

図１　定義機能・パーザの生成機能・論理式編集機能の関係

図１に定義機能，パーザの生成機能，論理式編集機能の関係を示す．

以下の節で，(1)・(3)の現状と(2)の検討結果について述べる．

3. 論理式エディタの開発の現状

3.1 記号定義

記号はすべて構文的型（変数・定数・関数子・演算子）と型（自然数・ブール等）で定義されている．記号のうちアリティ（引数の個数）が０のものを特別に定数という．それ以外の記号を関数子または演算子という．演算子については，記法や結合性に任意性がある．そこで結合性・優先順位の定義も必要である．

ここで記号定義の例を次の図２に挙げる．

定義は，

＜構文型＞ ⇒ ＜記号＞ ":" ＜型＞，

あるいは，

＜構文型＞ ⇒ ＜記号＞ ":" ＜型＞
"<" ＜結合性＞，＜優先順位＞ ">"

の形式で記述する．また，引数を持つものの型は，

＜定義域の型＞ "->" ＜値の型＞

で表す．例えば，図２の最初の行は，a で始まる文字列を整数型と解釈し，その構文型は定数であることを示している．

```
constant ⇒ a*:int
variable ⇒ x*:int
variable ⇒ y :int
functor  ⇒ f :(int,int)->int
functor  ⇒ g :bool -> int
operator ⇒ + :(int,int)->int <yfx,1000>
```

図２　記号定義の例

3.2 編集機能

論理式は一般的に木構造で表現できる．そこで，木構造を用いた構造エディタとして，論理式編集機能を作成した．例えば，図２に示された記号の情報を用いて図３のような木構造を表示する．

本エディタ特有の編集機能として，項の代入・項の書き換えなどがある．項の代入とは，変数と定義された記号に任意の項を代入する機能である．項の書き換えとは，等価の論理式への書き換えを行う機能である．図４に項の代入の例を示す．この例では，図３で示した論理式において，変数と定義された yに定数a3を代入する．他の編集機能については参考文献［４］を参照されたい．

Well-formed formulas editor for the Reasoning Assistant System
Kyoko TSUCHIYA,Kaoru SATOH,Etsuo ONO,Hajime SAWAMURA,Toshiro MINAMI
Fujitsu,Ltd.

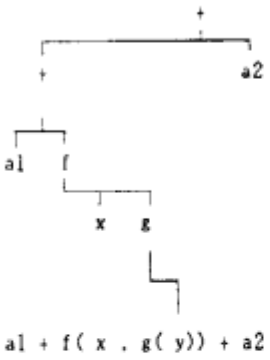

図3　論理式エディタの表示例

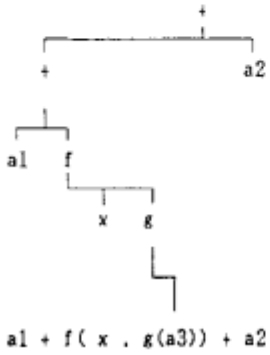

図4　項の代入の例

4.　構文の定義法とパーザ生成への展開

4.1　構文の定義法

以上で述べたように，記号定義と演算子順位文法のためのパーザを用いて，項と演算子順位文法に基づく論理式の木構造を認識できる．そこで定義できる記法は，前置記法・中置記法・後置記法等である．例えば，次のような例は定義できる．

f(x)　，　A　⇒　B

しかし，論理式がどんな構文的型の記号から構成されているかを定義することができない．例えば，

∀x f(x)

を考えたとき，∀の後ろは変数であることを記号定義だけでは表せない．これを補うために，論理式の構文の定義を記述するメタ言語が必要である．その記述言語を用いてユーザは論理式の構文を定義することができる．例えば，上記の例の構文を定義すると，次のようになる．

＜論理式＞：：＝" ∀" ＜変数＞＜項＞

定義された構文を認識するためには，ユーザの書いた構文規則からパーザを生成するパーザ・ジェネレータが必要となる．

4.2　パーザ生成

パーザ・ジェネレータを作成する前に，パーザ・ジェネレータとしてどのようなものが適当なのかを特定する必要がある．そこで，既存のパーザ・ジェネレータの能力を調べるためにDCG［5］とYacc（字句解析としてLexを使用）［6］［7］について調査を行った．文法，すなわちパーザ・ジェネレータへの入力，の記述能力を調べるために命題論理・第一階述語論理・内包論理の定義を行った．このうち内包論理は型を持つ論理である点が他の2つの論理と異なっている．

その結果，次のことがわかった．まず，既存のインプリメントによるDCGでは左再帰的な定義ができないので，左結合の演算子を定義できない．そのため，論理式の定義を行うには不適当であることがわかった．また，Yaccを使用すると命題論理・述語論理に属する論理式は定義できる．しかし，内包論理の論理式は不自然な定義になってしまうことがわかった．Yaccの構文では内包論理の型の整合性を自然な形で表現できないからである．そこで，論理式の構文を定義するにはより強力な構文記述が必要であることがわかった．

5.　まとめ

本論文では，論理式エディタの現状を述べ，構文の定義法とパーザ生成について考察を加えた．

今後，文字列の形式だけでなく付随した情報も同時に表現できる構文記述言語と，記述された文法からある程度の効率を持ったパーザが生成できるようなパーザ・ジェネレータの研究が必要である．

謝辞

日頃御指導いただく相橋部門長，三宅部長代理，國藤室長，寂東善氏に感謝いたします．

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］南俊朗他：論証支援システムの一構成，日本ソフトウェア科学会，1986．

［2］南俊朗他：論証支援システムのための証明コンストラクタ，情報処理学会第33回全国大会，1986．

［3］沢村一他：論理式エディタ，構造エディタに関するワークショップ（ＷＳＥ），日本ソフトウェア科学会，1986．

［4］佐藤かおる他：論証支援システムのための論理式エディタ，情報処理学会第33回全国大会，1986．

［5］溝口文雄編：Prologとその応用2，総研出版，1985．

［6］Stephen C.Johnson : Yacc : Yet Another Compiler-Compiler , Computing Science Technical Report No.32,1975.

［7］M.E. Lesk and E. Schmidt : Lex - A Lexical Analyzer Generator, Computing Science Technical Report No.39,1975.