人　工　知　能　107－2
（1997．1．21）

テキストからの数学知識の獲得

北村 輝夫† 鈴木 淳之‡

†静岡大学大学院工学研究科
‡静岡大学情報学部情報科学科

数学知識ベースシステムODISを対象に、数学のテキストから知識を獲得する方法について検討する。本論文では、まずODISにおける定義・定理知識の形式について説明する。次に、私たちが対象とする文章の構造について述べる。そして、ODIS言語の構成要素に対応する自然言語表現を述語形式を媒介としてODIS言語に変換する方法を説明する。最後に、定義・定理知識を獲得する方法について説明する。本研究では、初等解析学のテキストを題材に研究を行なっている。

Acquisition of Mathematical Knowledge from Text

Teruo Kitamura† Atsuyuki Suzuki‡

†Graduate School of Engineering, Shizuoka University
‡Department of Computer Science, Faculty of Information, Shizuoka University

We discuss a method to acquire a knowledge from text for the knowledge base system of Mathematics, ODIS. In this paper, first of all, we explain the ODIS language and the frame structures on both a definitional knowledge and a theorem knowledge. Next, we show the sentence instances which we dealt with for the knowledge acquisition. After semantic processing, we acquire the natural language phrases corresponding to the ODIS language constructs. Then, we translate these phrases into the corresponding ODIS constructs. After that, we give a method to acquire a definitional or a theorem knowledge. We adopted introductory mathematical analysis as a knowledge base domain.

1 はじめに

知識ベースシステムの開発において知識獲得を支援することは重要な問題である。ODIS[4] は、数学の公理・定義・定理などの知識を蓄えた知識ベースとその知識を定理証明などの問題解決に利用するための推論機構をそなえた知識ベースシステムである。数学に関する知識の多くは、教科書など数学のテキストという形で記録されており、このテキストから知識を抽出すれば、効率的に知識を獲得できると考えられる。本研究では、初等解析学のテキスト [1] を題材に、簡単な構造をした定義・定理文章から定義・定理知識を獲得する方法を考えた。

まず、2 節で ODIS における定義知識と定理知識の形式について説明する。そして 3 節で私たちが対象としている定義文章・定理文章の構造について述べ、4 節では、述語形式 [3] を用いて自然言語表現を ODIS 言語に変換する方法を説明する。5 節で定義知識・定理知識を獲得する方法を述べ、6 節でまとめを行なう。

2 ODIS について

この節では、知識ベースシステム ODIS についての簡単な説明を行なう。

2.1 ODIS 言語

ODIS の知識ベース中では公理、定義、定理知識は、ODIS 言語を使って表現される。ODIS 言語は、Ontic[2] をもとに作られた順序ソート付き一階集合論理の構文的変形体であり、また、ODIS 言語は依存型を持っているので、数学の知識をより自然に表現できるといった特徴を持っている。ODIS 言語は、(1) 項 (2) 論理式 (3) 関数 (4) 型 (5) 型生成子の 5 つの構文要素からなる。型は 1 変数の述語に相当し、型生成子は多変数の述語記号に相当する構文要素である。以下に各構文要素を列挙する。

項

- 定数記号。
- 変数 x。
- [f,t_1,...,t_n]。ただし、f は n 変数の関数で、t_i は項。
- [the_set_of_all, τ]。ただし、τ は型。
- [the_rule, f]。ただし、f は 1 変数の λ 関数。
- [the,τ]。ただし、τ は型。

論理式

- contradiction
- [is,t,τ]。ただし、t は項で、τ は型。
- [=,t_1,t_2]。ただし、t_1 と t_2 は項。
- [and,A,B], [or,A,B], [implies,A,B], [not,A]。ただし、A と B は論理式。
- [forall,[(x,τ)],A], [exist,[(x,τ)],A]。ただし、x は変数、τ は型、A は論理式。

関数

- 定関数記号。
- [lambda,[(x_1,τ_1),...,(x_n,τ_n)],T] ただし、x_i と T は項、τ_i は型。
- [the_function,t]。
  ただし、t は [the_rule,f] のような項。

型

- 定型記号
- [g,t_1,...,t_n]。ただし、g は n 変数の型生成子で、t_i は項。
- [lambda, [(x_1,τ_1),...,(x_n,τ_n)],A] ただし、x_i は項、τ_i は型、A は論理式。

型生成子

- 定型記号
- [lambda, [(x_1,τ_1),...,(x_n,τ_n)],T] ただし、x_i は項、τ_i と T は型、A は論理式。

2.2 ODIS の知識フレーム

ODIS の知識ベースでは、定義・定理知識はつぎのようなスロットを持つフレームの形で記憶されている。定義フレームは、主なスロットとして次のようなものを持っている。

name 定義されている概念の名前
definition ODIS 言語で書かれた概念の定義
rule 定義をルールコンパイラで変換したプロダクションルール

ここで definition スロットの値は次のような形をしている。

[def, [定義名,[引数の宣言リスト]]
　定義の本体]

たとえば、union では次のようになる。

```
[def, [union, [(a, set),(b, set)]],
  [the_set_of_all,
    [lambda, [(x, thing)],
      [or, [is, x, [member_of, a]],
           [is, x, [member_of, b]]]]]]
```

また、定理フレームの主なスロットとしては次のようなものがある。

name 定理名

arguments 変数の宣言リスト
assumptions 定理の仮定のリスト
conclusions 定理の結論のリスト
rules ルールコンパイラで変換されたプロダクションルール
ここで、仮定と結論の各リストの要素は連言の関係を持っている。

3 文章の構造

3.1 定義文章

ここでは、私たちが対象としている定義文章の構造について説明する。

> Given a sequence (x_n) in R and a real number x, we say that x is a **limit** of the sequence (x_n) if for every neighborhood U of x, the sequence (x_n) is eventually in U. When x is a limit of (x_n), we write $x_n \to x$ as $n \to \infty$.

上の文章は、[1] で数列の極限を定義している文章である。この文章は (1) オブジェクト (変数) の宣言文、(2) 定義文、(3) 数式表現の定義の 3 つの部分でできていると考えられる。まず、最初の文の Given 節がオブジェクトの宣言に対応し、'we say that ... if ...' が定義文に対応する。そして、第 2 文の 'When ..., we write ...' が数式の定義文に対応している。[1] に出てくる定義を調べた結果、多くの定義がこのような形をしていた。

上述の 3 種類の文の中で、ODIS の知識ベースを作るのに必要な情報は、オブジェクトの宣言文と定義文に含まれている。そこで、私たちはこの 2 種類の文から成る定義を扱うことにし、現在のところは、数式表現の定義を行なっている文は扱っていない。

また、この 2 種類の表現には、いくつかのバリエーションがあることも分かった。たとえば、オブジェクトの宣言文には、

- Suppose that A is a set.
- Let A be a set.

のように suppose や let を使った表現があり、定義文には、

- A sequence (x_n) is said to be **convergent** if ...
- the **power set** is defined to be ...

のように受動態を使った文や define, call といった動詞を使った文があった。

3.2 定理文章

つぎに、定理文について考えてみる

> Suppose that (x_n), (y_n), (z_n), are sequences of real number, that $x \in R$, that both (x_n) and (z_n) converge to x, and that for every natural number n we have $x_n \leq y_n \leq z_n$. Then $y_n \to x$.

この文は、サンドイッチの定理について述べている。この文は、(1) オブジェクト (変数) の宣言、(2) 仮定文、(3) 結論文で構成されていると考えられる。まず、Suppose 文では、最初の 2 つの that 節がオブジェクトの宣言文で、次の 2 つの that 節が仮定文になる。そして、'Then, ...' が結論文になる。

定義文章と同じように、この 3 種類の文にもいくつかのバリエーションがある。定義と同様に、

- Let A be a set.

のような文もオブジェクトの宣言文である。また、オブジェクトの宣言文と仮定文に同じ 'suppose that ...' が使われるが、ここでは、that 節が単文でその主語がまだ文中に現れていないオブジェクトを表すときにオブジェクトの宣言文と考え、それ以外のとき、たとえば that 節の主語がすでに文中に出現しているオブジェクトを指すときは仮定文であると考える。また、定理文には、

> Every cauchy sequence is convergent.

のように、単文だけの定理も多く存在する。そのような定理は、結論文だけからなる定理文と考えることにする。

4 自然言語から ODIS 言語への変換

ここでは、数学領域の概念を表す自然言語表現をどのように ODIS 言語表現に変換するかを説明する。前の節では、定義・定理文章には、'suppose that ...' や 'we say that ... if ... ' のようにその文の役割を表す部分があることを述べた。一方で、文中には、'a sequence (x_n) in R' や 'for every neighborhood U of x, the sequence (x_n) is eventually in R' のように数学の領域の概念を表す表現が存在する。このような表現は、最終的には ODIS 言語による表現に変換される。自然言語表現を ODIS 言語表現のような論理形式に変換するには、(1) 照応の問題、(2) 量化の問題を解決する必要がある。私たちは、中間形式として述語形式 [3] を使い、自然言語表現から ODIS 言語表現への変換を行なっている。

図 1は、自然言語表現から ODIS 言語表現への変

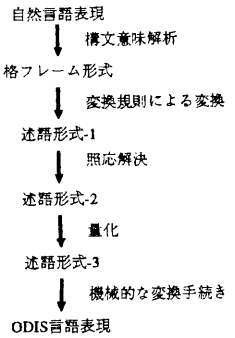

図 1: 自然言語表現から ODIS 言語表現への変換

換の過程を示している。入力された自然言語表現を構文・意味解析によって、格フレーム形式に変換し、変換規則を用いて述語形式に変換する。そして、照応解決・量化を行ない、最後に機械的な変換手続きによって ODIS 言語表現への変換を行なう。

4.1 述語形式の定義

述語形式による表現は、図 2のような木構造をしている。[3] では、述語形式は、(1) 述語、(2) 名辞、(3) 変数 (名辞の識別子) の 3 種類の要素で構成されている。ここでは、ODIS 言語に変換することを考えて、さらに、(4) 関数、(5) 集合を加えることにする。各ノードの説明を行なう。

述語

述語は、ODIS 言語における **and**, **or**, **implies** などの論理演算子や型・型生成子を表す。型は 1 引数の述語に対応し、n 引数型生成子は $n+1$ 引数述語に対応する。図 2では、述語ノードは楕円形で表されている。述語は次の情報を持っている。

述語名 文字列

引数リスト 述語名が論理演算子を表すのならば述語の並び。そうでなければ、変数、名辞、関数、集合の並び。

量化子リスト 名辞あるいは、negation からなる並び

名辞

名辞は、オブジェクト (変数) を表す。修飾子は、そのオブジェクトの型に対応する。名辞は次の情報を持っている。

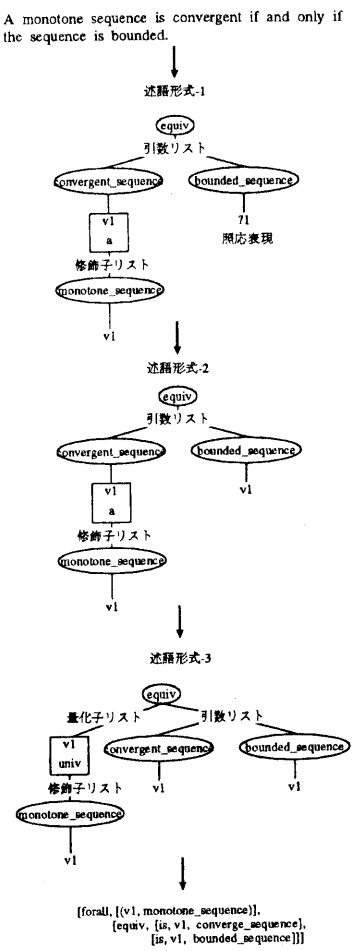

図 2: ODIS 言語への変換例

識別子 変数

量化子の種類 量化子タイプの決定前は、a, all, every, some など数量詞、量化子タイプの決定の後は、univ, exist, neg-u のいずれか。

修飾子リスト 述語の並び

名辞は図 2では正方形で表されている。

関数

関数ノードは、$[f, t_1, \ldots, t_n]$ のような項に対応する。関数は次の情報を持っている。

関数名 文字列

引数リスト 変数、名辞、関数、集合の並び

集合

集合は、[the_set_of_all, τ] に対応する。これは、修飾子リストにある述語を満たすオブジェクトのすべてからなる集合を意味する。集合は次の情報を持っている。

識別子 変数
修飾子リスト 述語の並び

4.2 照応解決

文中で代名詞や定名詞句などの照応表現が使われているとき、その表現が参照するものを決定しなければならない。本研究では、自然言語表現を述語形式に変換するとき、照応表現を未確定変数 (図 2では?1) に変換する。そして、未確定変数の確定を次のような簡単な規則で行なっている。

1. 記号をともなった照応表現の場合、同じ記号をもつオブジェクトを参照先とする。
2. 修飾子が一致する最後に現れたオブジェクトを参照先とする。

私たちは、照応表現の参照先を名辞・変数・関数・集合のオブジェクトを表すものだけに限定し、動詞句や文全体を指すような照応表現は扱っていない。

4.3 量化

量化とは、不定冠詞や数量詞によって導入された不定名詞句が表す変数の

- 量化子のタイプ
- 量化子の作用域 (スコープ)

を決定することである。本研究では、[3] に述べられている方法を使って量化の処理を行なっている。この方法では、量化は名辞を量化子として述語の量化子リストに帰属させ、量化子タイプを決定することによって量化を行なっている。

4.4 ODIS 言語への変換

ここでは、量化された述語形式を ODIS 言語に変換する方法について説明する。量化された述語形式から ODIS 言語への変換は、次のように定義された写像 F によって行なわれる。

述語ノード

述語ノードの変換は、写像 F' と F'' を使って定義する。まず、F' の定義をおこなう。述語ノード P の述語名を $name$, 引数リストを $p_1,\ldots,p_n$, 量化子リストを $Q_1,\ldots,Q_m$ とする。

- 述語 P が論理演算子 (and, or, implies など) のとき、
  $F'(P) = [name, F(p_1), \ldots, F(p_n)]$.
- 述語 P が型 (1 引数の述語) を表すとき、
  $F'(P) = [is, F(p_1), name]$
- 述語 P が型生成子 (多引数の述語) を表すとき、
  $F'(P) = [is, F(p_1), [name, F(p_2), \ldots F(p_n)]]$

そして、各量化子 Q_i の変数を v_{Q_i}, 修飾子リストを $M_{Q_i,1},\ldots,M_{Q_i,l}$、$T_i = (v_{Q_i}, [M_{Q_i,1},\ldots,M_{Q_i,l}])$ とし、F'' を次のように定義する。

- 名辞の量化子タイプが univ のとき
  $F''(Q_i, A) = [forall, [(v_{Q_i}, \tau(T_i))], A]$
- 名辞の量化子タイプが exist のとき
  $F''(Q_i, A) = [exist, [(v_{Q_i}, \tau(T_i))], A]$
- 名辞の量化子タイプが neg-u のとき
  $F''(Q_i, A) = [forall, [(v_{Q_i}, \tau(T_i))], [not, A]]$
- 量化子が negation のとき
  $F''(Q_i, A) = [not, A]$

ここで、τ は変数とその修飾子からその変数の型を返す関数である。τ の定義はあとで行なう。これらの F' と F'' を使って、F を定義する。

$F(P) = F''(Q_1, \ldots F''(Q_m, F'(P)))$

変数

変数 v は、写像 F によってそれ自身に変換される。

$F(v) = v$

関数ノード

関数ノード P の関数名を $name$、引数リストを $p_1,\ldots,p_n$ とする。

- n=0 のとき、
  $F(P) = name$
- $n > 0$ のとき、
  $F(P) = [name, F(p_1), \ldots, F(p_n)]$

集合ノード

集合ノード P の変数を v、修飾子リストを $M_1,\ldots,M_n$ とし、$T = (v, [M_1,\ldots,M_n])$ とする。

$F(P) = [the\_set\_of\_all, \tau(T)]$。

型への変換 τ の定義

$T = (v, [M_1,\ldots,M_n])$ とする。

- 修飾子リストが空 ($n = 0$)、
  $\tau(T) = thing$
- $n = 1$ で $F(M_1) = [is, [v, [P, p_2, \ldots, p_n]]]$、
  $\tau(T) = [P, p_2, \ldots, p_n]$
- $n = 1$ で $F(M_1) = [is, [p_1, [P, p_2, \ldots, p_n]]]$、
  $\tau(T) = [lambda, [(v, thing)], F(M_1)]$
- $n > 2$ で $F(M_1) = [is, [v, [P, p_2, \ldots, p_n]]]$、
  $\tau(T) = [lambda, [(v, [P, p_2, \ldots, p_n])],$
  $[and, F(M_2), ..., [and, [F(M_{n-1}), F(M_n)]]]]$
- $n > 2$ で $F(M_1) = [is, [p_1, [P, p_2, \ldots, p_n]]]$
  $\tau(T) = [lambda, [(v, thing)],$
  $[and, F(M_1), ..., [and, [F(M_{n-1}), F(M_n)]]]]$

5 知識獲得システム

5.1 定義知識の獲得

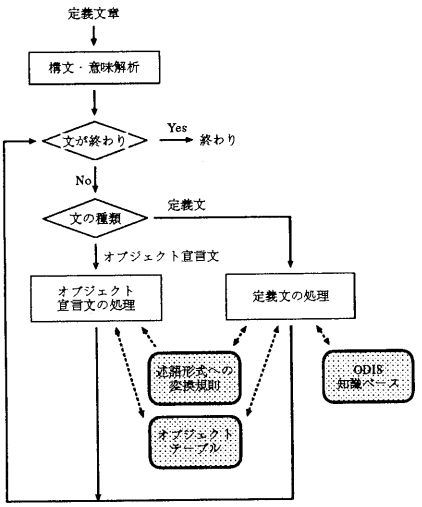

図 3: 定義知識獲得の流れ

ここでは、定義文章から定義知識を獲得する方法を説明する。図 3はその処理の流れを表している。入力文章の各文は、構文・意味解析によって格フレーム形式に変換される。そして、各文の種類に応じて処理を行なう。オブジェクトの宣言文や不定名詞句などによって新たにオブジェクトが導入されたとき、そのオブジェクトに関する情報がオブジェクトテーブルに格納される (この情報は、主として照応関係の決定に使われる)。定義文の処理では、その文で定義されている概念を ODIS 言語で表したものと、定義されている概念を表す自然言語表現 (格フレーム形式) から述語形式に変換するための規則が生成される。

定義フレームの作成

ODIS の知識ベース中の定義フレームで、知識獲得において重要なスロットは、 definition スロットである。ここでは、 definition スロットの値の決定方法について説明する。 definition スロットは次のような形をしている。

[def, [型生成子名 (または関数名), [引数のリスト]],
　　定義本体]
あるいは、
[def, 型名 (または定数名),
　　定義本体]

定義本体は、型・型生成子の場合は型になり、関数・定数の場合は項になる。したがって、定義を獲得するために、 (1) 定義本体、 (2) 引数とその引数の型を決める必要がある。テキストに現れる定義文にはいくつかの種類 (図 4) がある。タイプ 1,2,3 は、型・型生成子の定義に使われ、タイプ 4 は、項・関数の定義に使われる。図 4のほかにも受動態を用いた定義文などがある。ここでは、タイプ 1 について definition スロットの値を作成する手順を述べる。

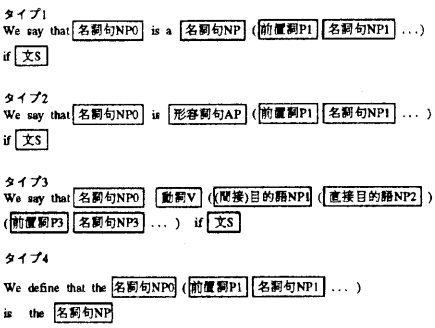

図 4: 定義文の種類の例

1. 名詞句 NP_0、名詞句 NP を修飾する前置詞句 P_i の目的語 NP_i、文 S を述語形式に変換し、照応関係を決定する。 (ここで得られた述語形式を PF_{NP_i}, PF_S とする)
2. NP_i を引数とする一時的な述語 tmp_pred を作成する。
3. 述語 tmp_pred と PF_S を引数とする述語 equiv を作成する。
4. 述語 equiv に対して量化を行なう。

5. 名詞句 NP_0 に対応する量化子 Q_0 をのぞく述語 equiv に帰属する量化子 Q_i が、ここで定義される型生成子の引数になる。(もしそのような量化子がなければ定義されているのは型)。
6. 量化子 Q_j を ODIS 言語に変換し、変数 p_j とその型 τ_j のペア (p_j, τ_j) を作成する。
7. PF_S に対応する量化された述語形式を ODIS 言語に変換する (その論理式を F_S とする)。
8. 名詞句 NP から型生成子 (型) 名 name を決定する。
9. ODIS 言語による定義式を作成し、
   [def, [name, [(p_1,τ_1),...,(p_n,τ_n)]],
   [lambda, [(p_0,τ_0)],
   F_S]]
   を生成する。

図 5は、全射の定義による definition スロットを作成する過程を表している。

述語形式への変換規則の獲得

一度定義された概念は、定理文章や別の概念の定義文章の中で使われる。その文章から知識を獲得するとき、定義された概念を表す自然言語表現を述語形式に変換する必要がある。そのために、定義知識の獲得と同時に述語形式に変換する規則を獲得しなければならない。たとえば、前節のタイプ 1 の定義文で定義された概念については、次のような情報が必要になる。

- 概念を表す名詞句 NP
- 型生成子 (型) 名 name
- 前置詞 P_i と引数 p_j の対応
- 引数 p_j とその修飾子

5.2 定理知識の獲得

図 6は、定理知識獲得の処理の流れを表している。入力文章は、定義文と同様に構文・意味解析によって格フレーム表現に変換され、その後、文の種類に対応して処理が行なわれる。定義と同様に、オブジェクトの宣言文や不定名詞句などによって新たに導入されたオブジェクトについての情報がオブジェクトテーブルに保存される。仮定を表す文は、述語形式に変換され、照応の解決が行なわれ、仮定リストに記憶される。また、結論を表す文も、述語形式への変換、照応解決が行なわれ、結論リストに記憶される。すべての文に関して処理が終わった時点で仮説リストと結論リストから定理フレームの生成を行なう。

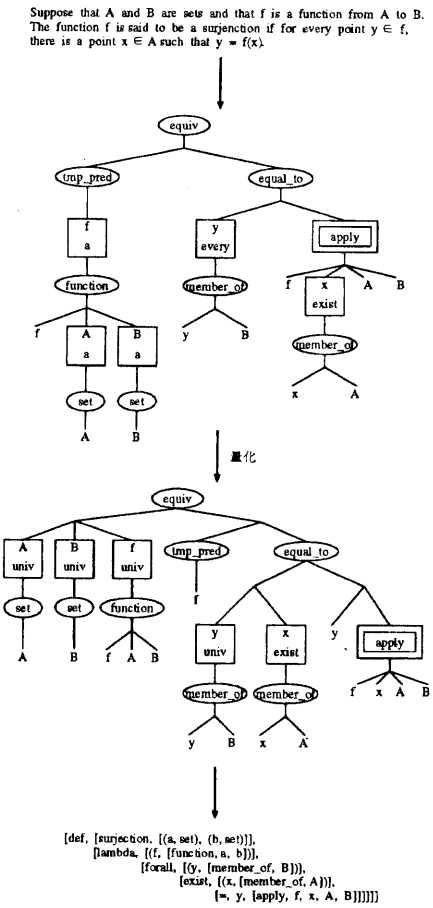

図 5: 定義獲得の例

定理フレームの作成

定理フレームは、すでに得られている仮定リストと結論リストから作成される。定理フレームの作成では、arguments スロット、conclusions スロット、conclusions スロットの値を決定する必要がある。この処理は、次のような手順で行なわれる。

1. 仮定リスト中の述語を述語 and で連結する (それを ass と呼ぶ)。
2. 結論リスト中の述語を述語 and で連結する (それを con と呼ぶ)。
3. ass と con を述語 implies で連結する。
4. 述語 implies に対して量化を行なう。

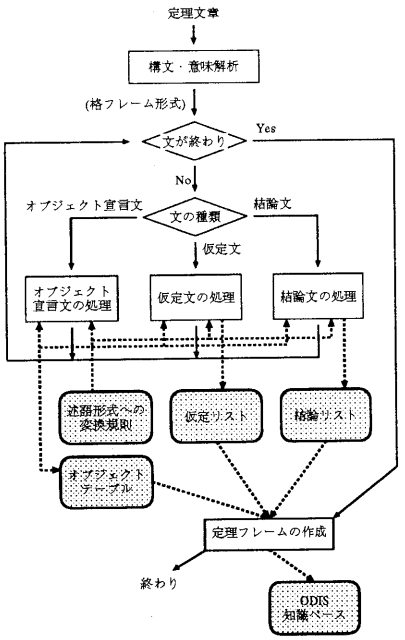

図 6: 定理知識獲得の流れ

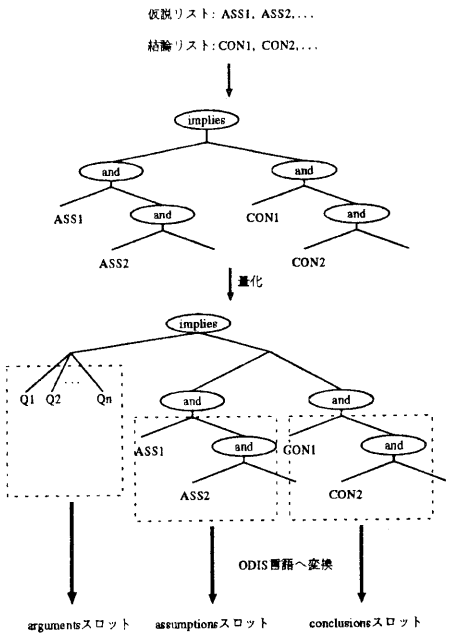

図 7: 定理フレーム作成の流れ

5. 述語 implies に帰属する量化子 Q_i から変数 v_i とその型 p_i のペアを作り、arguments スロットを作る。
6. 量化された述語形式の仮定部分を ODIS 言語に変換し、assumptions スロットを作る。
7. 量化された述語形式の結論部分を ODIS 言語に変換し、conclusions スロットを作る。

図 7は、この過程を図で表したものである。

6 まとめ

本研究では、数学知識ベースシステム ODIS における知識獲得支援の方法として、述語形式を媒介とすることによって数学のテキストから定義・定理知識を抽出し、定義フレーム・定理フレームを作成する方法を考えた。また、定義文から自然言語表現を述語形式に変換するための規則を獲得する方法も同時に考えた。しかし、(1) 定義・定理文章を簡単な構造の文章に限っている、(2) 構文解析における曖昧性を考慮していない、(3) 照応解決について簡単な規則でおこなっており深い推論が必要な照応を扱っていないなどの問題点があり、実用的な知識獲得システムを開発するためには上記の問題点を解決する必要がある。

参考文献

[1] Jonathan Lewin, Myrtle Lewin, "An Introduction to Mathematical Analysis", McGraw-Hill, 1992

[2] David A. McAllester, "ONTIC", MIT Press, 1989.

[3] 丹羽 芳樹, "述語形式を媒介とする自然言語文の意味抽出方式", 情報処理学会研究報告 90-NL-75, 1990

[4] 王 国傑, "局所的概念駆動型数学知識ベースシステムに関する研究", 静岡大学 博士論文, 1994

[5] 大脇 宏仁, "数学テキストにおける対話型証明検証システム", 静岡大学 修士論文, 1995