自然言語処理 65－5
（1988. 3. 25）

自然語によるプログラム仕様の，抽象的データタイプを用いた構文及び意味解析

並河 英二　松村 享　関 浩之　藤井 譲　嵩 忠雄
大阪大学基礎工学部情報工学科

抽象的データタイプの概念を用いて，プログラム仕様の記述に用いる自然語の形式的意味定義を行った．本手法では，まず，HPSGのカテゴリに似た中間表現と呼ばれる形式を定め，自然語の句を中間表現に対応づける規則を与える．

中間表現は，属性およびその値の二字組であり，特に，属性として，その句の表す関数や述語の引数と値のデータタイプを表す属性を導入した．

そして，データタイプ名の集合上に，一方が他方の部分データタイプであることを表す半順序関係を導入し，この半順序関係に基づく単一化の概念を用いて，部分句のデータタイプに関する制約条件から，親の句のデータタイプに関する制約条件を定義した．データタイプに関する制約条件を利用することにより，構文解析時の曖昧さが減ること，また，等位接続や複数形名詞句の形式的意味定義も簡潔に行えることを示した．

A Formal Definition of Natural Language Specifications Based on Abstract Data Types

Eiji NABIKA　Takashi MATSUMURA　Hiroyuki SEKI
Mamoru FUJII　Tadao KASAMI

Department of Information and Computer Sciences
Faculty of Engineering Science, Osaka University
Toyonaka, Osaka, 560 JAPAN

A formal definition of natural language specifications is defined based on abstract data types. In our method, a phrase in a sentence of a natural language is transformed into a function or a predicate on an appropriate abstract data type. We introduce a partial ordering, denoted by $\leq$, on the class of abstract data types, where $t_1 \leq t_2$ denotes that t_1 is a subtype of t_2. Using the concept of unification with respect to this ordering $\leq$, the data type of the function or predicate corresponding to a phrase in a natural language sentence, is defined from those of its constituents. Syntactic ambiguities may be reduced by the data type information of each phrase, and the semantics of plural noun phrases and coordinations can be defined formally.

1. まえがき

プログラムの段階的詳細化において，プログラムが仕様を満足していることの形式的な検証が行えるためには，仕様やプログラムの意味が形式的に定義されていることが望ましい[7]．我々の研究グループではプログラム仕様の記述に有用と思われる自然語（L_{NS}と呼ぶ）の形式的意味を定義し，その定義に従って，自然語による仕様を代数的仕様に変換する方法についての研究を行ってきた[8,9]．対象として，プログラム仕様に用いる自然語を考えているので，日常用いられる自然語に比較してあいまいさや省略が少ないと考えられる反面，形式的仕様への変換を可能にするためには，厳密な意味定義を行う必要がある．また，プログラム仕様に用いる自然語文に限定したとしても，問題領域に依存して定義すべき個々の語句は量的に膨大になると思われる．これらの問題点を考慮し，本研究では，L_{NS}の構文と意味定義を以下の1)～5)の方法で行っている．

1)英語を対象とし，構文の記述法として，GPSG[3]を採用している．現在，関係代名詞節，分詞構文，能動態と受動態，形式主語，分詞，不定詞，同格などの構文を含み，GPSGの非終端規則数が約100である．

2)解釈としてエルブラン解釈のみを考える多ソートの一階述語論理（ＭＨ１とよぶ）を採用し，L_{ns}の文とＭＨ１の論理式との対応づけ（１対１と限らない）を定義し，対応する論理式を文の一つの意味とする．意味定義の対象はプログラム仕様に用いる自然語であるので，様相については，プログラムの入力に関する量化を簡潔に表現する機能があれば十分であり，高階論理を用いる必要はない．なお，多ソートの概念は，プログラムや仕様における抽象的データタイプを定義するのに有用である．

3)自然語文と論理式との対応づけに必要な情報のうち，たとえば，述語の仮引数に対応する実引数の決定，引数が満たすべき条件の決定などは，構文木をボトムアップ式に処理することにより得ることができる．もちろん，これらの局所的な構文だけで自然語文の意味が確定できるわけではない．指示代名詞の先行詞や，量化のスコープを指定する機能がその例である．自然語文と論理式の対応づけを考えるとき，先行詞の決定，あるいは量化の問題を，その他の部分と切り分けて考える方が見通しがよいと思われる．そこで，本稿では，構文木から論理式へ直接変換するのでなく，構文木上をボトムアップ式に，逐次決まる意味上の情報を表現する形式として，中間表現を用いている．構文木から中間表現を構成する規則（中間表現構成規則とよぶ）は次のように定義される．(a)まず，終端記号に対応する中間表現を定め，次に，(b)各構文規則を，それがもつ“意味上の機能”に応じて分類し，構文規則の右辺に対応する中間表現と左辺に対応する中間表現との関係を，その“機能”別に定める．

4)中間表現と論理式を対応づける規則（論理式生成規則とよぶ）を定める．その定義法としては，文献(8)(9)などで提案した方法を採用する．ただし，(8)(9)では，直接自然語文の構文木と論理式との対応を定義していたが，本稿では，段落（文の系列）に対する中間表現の系列からの論理式生成規則を定義する（詳細は(10)を参照）．

5)個々の語句の意味は，論理式生成規則とは別に与える．意味定義の対象をプログラム仕様に限定しているので，個々の語句はいわゆる抽象データタイプ上の演算や述語として定義する．

L_{NS}の文法及び語句の定義を最初から網羅的に行うことはできないので，実例に即して，随時構文規則及び語彙の追加を行う必要がある．本研究の手法では，中間表現の形式が個々の構文規則に依存しないうえ，語句の意味は論理式生成規則とは別に与えられるため，構文規則及び語彙の追加が容易に行える．

本稿では特に，中間表現に求められる機能について考察し，それをもとに，実際に中間表現及び中間表現構成規則を定め，自然語文と中間表現の対応づけを定義した．

本稿における中間表現とは，＜属性，値＞の二字組の集合であり，LFG[6]におけるf-構造，あるいは，HPSGにおけるカテゴリに対応している．LFGやHPSG等の文法理論においては，例えば，語句walksの意味をwalks'と表す，というだけで，定義されていない．これに対し，ここでは，実際に自然語のプログラム仕様を代数的仕様に変換することを目標としている．代数的仕様では，抽象データタイプ上の関数や述語の満たすべき性質を代数的公理によって記述する．本手法では，自然語による仕様の文の系列を公理に変換するが，その際，各語句を適当な抽象データタイプ上の関数や述語に対応づける．このためには，これらの語句に対応する関数や述語の引数及び値のデータタイプを決定する必要がある．本研究で導入する中間表現には，LFGやHPSGで用いられている属性の他に，その中間表現の表す関数や述語のデータタイプを表す属性（後述のdatatype）がある．関数や述語がどのようなデータタイプをもちうるかという制約も一般的に表現できるよう，データタイプの間に順序関係を導入し，単一化の概念を利用して，部分句のデータタイプに関する制約から，親の句のデータタイプの制約を定義している．これらを利用して，関係節の先行詞の決定の際などに起こる構文のあいまいさ，同型異義語の意味の決定の際のあいまいさなどを減らすことができる（3.3 参照）．

また，英語の複数名詞句にはさまざまな用法があり，それらをすべて形式化するような研究結果は知られていない．そこで，４．でプログラム仕様中における複数名詞句や等位接続された名詞句の機能について考察し，それらの機能も表現できるように中間表現を拡張する．

２．中間表現構成規則

2.1 中間表現の定義

Ｆを属性名の有限集合とする．Ｆの元としてどのようなものを考えるかは，2.2で述べる．Ｆの各属性ｆに対して，ｆのとりうる値を表す有限集合V(f)が定まっている

とする．ただし，後に述べるように，中間表現自身，あるいはそれらの集合や系列を値としてもつ属性も許す．V(f)には１つの半順序関係（特にことわらない限り≤で表す）が定義されているとする．そして，V(f)の元α，βの，その順序関係における最大下界（単一化ともいう）を$\alpha \wedge \beta$とかき，$\alpha \wedge \beta$を求めることを，α，βを単一化するという．また，属性の値に対する順序関係が，陽に定義されていないときには，$\alpha \wedge \beta$は，$\alpha = \beta$のときαと定義され，それ以外では未定義とする．属性とその値の２字組の有限集合，

$$\{(f, v) \mid f \in F,\ v \in V(f)\}$$

で，属性名が等しく値の異なる対を含まないような任意の集合を，中間表現と呼ぶ．

中間表現$\{<\alpha_1, \beta_1>, <\alpha_2, \beta_2>, \cdots, <\alpha_n, \beta_n>\}$を，

$$\begin{bmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ & \vdots \\ \alpha_n & \beta_n \end{bmatrix}$$

とかく．

V(f)上の半順序を拡張することにより，中間表現の集合の上に次のような半順序を導入する．

A，Bを任意の中間表現とする．任意の$<f, v> \in A$に対し，次の(i)または(ii)が成り立つときかつそのときのみ$A \leq B$が成り立つと定義する．

(i)　$<f, v'> (v' \in V(f))$の形の組がBに含まれない．

(ii)　$<f, v''> \in B$かつ，$v \leq v''$が成り立つ．

属性の値の集合でのように，単一化等の概念を，この順序関係≤に関して同様に定義する．

本稿では，文献(2)に従い，中間表現の属性が値を共有することを許すものとする．

$$\begin{bmatrix} f1 & \boxed{x}\ v1 \\ f2 & \begin{bmatrix} f1 & \boxed{x} \\ f3 & v3 \end{bmatrix} \end{bmatrix} \qquad (*)$$

において，各f1の値は，常に同じ値をもつ．また，このような中間表現にも半順序≤は定義される．

Aを属性の値の共有を含む中間表現，BをAに含まれるある値の共有をやめ，代わりに共有していた値と同じ値で置き換えたものとする（(*)をAとするとBは

$$\begin{bmatrix} f1 & v1 & \\ f2 & \begin{bmatrix} f1 & v1 \\ f3 & v3 \end{bmatrix} \end{bmatrix}$$

となる）．

このとき$A \leq B$とする．これを用いて単一化も定義できる．詳しい形式化は(2)を参照されたい．

自然語文は中間表現を介して論理式に変換されるが，自然語文中の各句が，変換された論理式においてその句に対応する項，あるいは式をその句の意味要素と呼ぶことにする．

2.2 属性について

ここでは，主要な属性の種類と，その役割について簡単に説明する．

1)surface

自然語の句を値としてもつ．<surface, α>を含む中間表現は，句αに対応していることを表す．

2)property

動詞，名詞，形容詞，接続詞などの語句の意味要素を記述する．具体的には，属性funcとargからなる中間表現を値としてもつ．func，argはそれぞれ値として意味要素の関数記号，および中間表現のリストをとる．また，

$$\begin{bmatrix} \text{func} & \alpha \\ \text{arg} & x_1, \cdots, x_n \end{bmatrix}$$

を$\alpha(x_1, \cdots, x_n)$と書くことがある．目的語をとらない名詞の場合，funcの値とデータタイプ（後述）が一致した場合省略する．

3)args

意味要素が関数，述語またはデータタイプの構成子であるような句に対応する中間表現に記述される属性．値はその関数等の引数に関する制約条件を表すような中間表現のリストである．

4)datatype

datatypeは，その句の意味要素のデータタイプを表す属性である．たとえば，自然語の句の意味要素がデータタイプ名β，あるいはデータタイプβのインスタンス（対象）であるとき中間表現はそれぞれ，

<datatype, (n, β)>　<datatype, (i, β)>

を含む（ただし混乱がない限り属性名datatypeは省略する）．

データタイプβ_1，β_2に対し，$\beta_1 \leq \beta_2$は直観的にはβ_1はβ_2の部分データタイプであることを表す．例えば，系列は“順序”が定義されているという意味で，集合の特別な場合ということができる．よって，系列≤集合であるとする．データタイプの詳細は３．で述べる．

5)restriction

関係節，分詞，前置詞句によって修飾された名詞の中間表現がもつ属性．値は関係節，分詞，前置詞句に対応する中間表現のリスト．

その他の属性については，必要となったとき順次説明する．

2.3 自然語の句と中間表現の対応関係の定義

2.3.1 語句に関して

語句に関しての対応づけは，一般には語句の意味を定義する人に委ねられる．ここでは，名詞，動詞の場合について例示するのみにとどめる．

(a)名詞file

$$\begin{bmatrix} \text{surface} & \text{file} \\ (n, \text{file}) & \end{bmatrix}$$

(b)動詞contains

$$\begin{bmatrix}\text{surface contains}\\ (\text{i, bool})\\ \text{property contain}(\boxed{X},\boxed{Y})\\ \text{args}\quad \begin{bmatrix}\boxed{Y}\ [(\text{i, d})\quad]\\ \boxed{X}\ [(\text{i, set(d)})]\end{bmatrix}\end{bmatrix}$$

ここで，argsの値の順は自然語の構文においてcontainsと先に結び付くものほど先になるとする．

2.3.2 複合されて得られる句に関して

まず自然語の構文をその機能に応じて以下のi)～v)に分類する．そして分類された各グループに対して，右辺の非終端記号に対応する中間表現から左辺に対応する中間表現を定義する．

(a)構文の分類

自然語の構文を，それが対応する論理式の操作を考え，次の５つに分類する．

i)あるデータタイプから元を一つ取り出す操作に対応する構文．

ii)引数に対する関数や述語の適用を表す構文．

iii)ある部分句が別の部分句の満たすべき制約条件を表しているような構文．

iv)局所的には意味上の機能をもたないような構文．

v)その他．

(b) i)～iv)の構文について，それらの左辺に対応する中間表現を以下で定義する．また，v)に属する構文には，①同格，②名詞の形容詞的用法，③分詞構文，および④complementizerを加える構文がある．説明は紙面の都合で省略する．詳しくは(10)を参照されたい．

i) i)に属する構文を生成する構文則は，

NP→DET N^3

である．N^3はデータタイプ名に対応し，NPはそのインスタンスに対応すると考える．したがってN^3の中間表現は，

$$\begin{bmatrix}\text{surface } \alpha\\ (\text{n}, \beta)\end{bmatrix}$$

を含む（ただし，αはN^3より導出される語，βはαに対応するデータタイプ）．そして，NPに対応する中間表現においては，datatypeに含まれるnを，インスタンスであることを表すiに変え，属性detに冠詞の種類を加える．冠詞をθとすると，NPの中間表現は，

$$\begin{bmatrix}\text{surface } \theta\ \ \alpha\\ (\text{i}, \beta)\\ \text{det}\quad \theta\end{bmatrix}$$

となる．

ii) ここに類別される規則は，右辺の非終端記号が関数または述語とその引数に対応しているようなものであり，それらは，対象とする自然語を英語に限定するならば，A→B_1 Hまたは，A→H B_1・・・B_mという形である（Hは関数，述語または，構成子に対応する非終端記号）．この場合の中間表現構成規則は，HPSGにおけるsubcategorization原理に対応しており，次のように定義される．

構文規則をA→B_1 H，またはA→H B・・・B_mとする．関数または述語に対応する中間表現を，

$$\begin{bmatrix}(\text{s}, \alpha)\\ \text{property}\quad \text{f}(\boxed{X_1}, \boxed{X_2}, \cdots, \boxed{X_n})\\ \vdots\\ \boxed{X_{I1}}\quad \begin{bmatrix}(s_1, \beta_1)\\ \vdots\end{bmatrix}\\ \vdots\\ \boxed{X_{Ii}}\quad \begin{bmatrix}(s_i, \beta_i)\\ \vdots\end{bmatrix}\\ \text{args}\ \begin{bmatrix}\boxed{X_{I\,i+1}}\ [(s_{i+1}, \beta_{i+1})]\\ \vdots\\ \boxed{X_{In}}\quad [(s_n, \beta_n)]\end{bmatrix}\end{bmatrix}$$

とし，B_kに対応する中間表現をR_k，Hに対応する中間表現をWとする（ただし，s，s_1，・・・，s_nはiまたはnとする．I_iはpropertyの引数の順とargsの順を対応づける整数とする）．左辺に対応する中間表現を，以下の①～③ように定義する．

①k＝１，Y_0＝Wとする．

②Y_{k-1}のargsの値の先頭をポップし，取り出した中間表現をR_kと単一化する．ポップした後のY_{k-1}をY_kとする．

③k＜mならk：＝k＋１とし，②を繰り返す．

k＝mなら得られた中間表現が左辺に対応するとする．

②で単一化できないときは，その構文規則は適用できないとする．

次に，関係代名詞の生成や主題化（Topicalization）などの際に適用される，句の移動の規則S→NP S/NPについては，GPSGと同様，次のように取り扱う．すなわちεに対応する中間表現を，

$$\begin{bmatrix}(\text{i, d})\\ \text{gap}\ +\end{bmatrix}$$

とする．VP/NP→V NP/NP，S/NP→NP VP/NPの規則の適用の際は，VP→V NP，S→NP VPの場合と全く同様にして中間表現を得る．

S→NP S/NPの左辺に対応する中間表現は，＜gap，+＞を含む中間表現をNPに対応する中間表現との単一化で置き換えたものである（ただし，単一化をとるとき＜gap，+＞は除く）．

iii) 関係節，前置詞句の修飾により加わる制約を，属性restrictionで表す．restrictionの値は関係節，前置詞句などに対応する，中間表現のリストである．制約は述語を用いて行う．例えば，who is walkingに対応する中間表現は，

$$\begin{bmatrix}\text{surface who is walking}\\ (\text{i, bool})\\ \text{property walk}(\boxed{X})\\ \text{args}\ \begin{bmatrix}\boxed{X}\begin{bmatrix}(\text{i, man})\\ \text{fparameter}\ +\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

とする．ここで，属性fparameterの値がtであることは，それを含む中間表現が，制約を表す述語の仮引数であることを表している．

また，これらの中間表現を名詞のrestrictionに加えるとき，名詞のdatatypeと，(fparameter,t)を含む中間表現のdatatypeの単一化を行う（ただし，このときはiとnの違いは無視する）．もし単一化できないときはその修飾の規則は適用できない．

iv) iv)に属する構文については左辺に対応する中間表現は右辺に対するものと同じとする．

3．データタイプについて

3.1 データタイプ

ここではまず，中間表現の属性datatypeのとりうる値の集合と，その集合の上の単一化を定義する．Dをデータタイプ名の集合とする．Dの上には，1つの半順序≤が定義されているとする．他の属性や中間表現の場合と同様，データタイプt_1，t_2に対し，$t_1 \leq t_2$は，直観的には，t_2がt_1より一般的なデータタイプであることを表す．

半順序≤をどのように定義するかは，自然語文に対応づけられる論理式の従う論理体系において，どのようなデータタイプを定義するかに依存する．例えば，データタイプがt_1であるような式がすべてt_2の式にもなるときかつそのときに限り，$t_1 \leq t_2$と定義する．

[例3.1] $D \triangleq \bigcup_{i \geq 0} D_i$，但し，con＝{set,seq}，$D_0$＝{int,real}，$D_i$＝{c(n)｜c∈con，n∈$D_{i-1}$}（i≧1）とし，con上の半順序⊑を，⊑＝{(set,set)，(seq,set)，(seq,seq)}と定義する．D上の関係≤を，次の条件を満たす最小の半順序関係と定義する．

(i) int≤real.

(ii) もし，$c_1 \sqsubseteq c_2$（c_1，c_2∈con）かつ$t_1 \leq t_2$（t_1，t_2∈D）ならば，$c_1(t_1) \leq c_2(t_2)$． □

Dをデータタイプ名の集合とする．データタイプ名の集合を表す一般に変数を含む式の集合DEXPを次のように定義する：

(1)φ，D，{t}，d∈DEXP，ただし，tはデータタイプ名，dは変数．各変数には，それがとりうる値の範囲を表す2^Dの部分集合dom(d_i)が与えられているとする．dom(d_i)が陽に指定されていないとき，dom(d_i)≜2^Dとする．

(2)α_1，α_2∈DEXPであるとき，$\alpha_1 \cap \alpha_2$，$\alpha_1 \cup \alpha_2$，$\overline{\alpha_1}$，$\alpha_1 \downarrow$∈DEXP

(3)上の(1)，(2)で定まるもののみがDEXPの元である．

φ，D，{t}，∩，∪，$\overline{\ }$ の意味は通常どおり定義し，

$\alpha \downarrow \triangleq$ {t'｜tはαの表すデータタイプ名の集合，$t' \leq t$，t'∈D}

と定義する．また，例3.1のsetやseqのようなデータタイプの構成子cとデータタイプ名の集合βに対し，

$$c(\beta) = \bigcup_{t \in \beta} c(t)$$

と定義する．混同のない限り，式α∈DEXP自身をそれが表すデータタイプの集合として取り扱う．属性datatypeのとりうる値の集合v(datatype)は，

v(datatpe)≜ {(u,α)｜u∈{n,i}，α∈DEXP}

と定義する．v(datatype)上の単一化を次のように定義する．(u_1,α_1)，(u_2,α_2)∈v(datatype)に対し，$u_1 = u_2$かつ$\alpha_1 \wedge \alpha_2$が存在するときかつそのときのみ，(u_1,α_1)と(u_2,α_2)の単一化が存在し，それは，

(u_1,α_1)∧(u_2,α_2)≜(u_1,$\alpha_1 \wedge \alpha_2$)

と定義される．ここで，α_1とα_2の単一化は次のように定義される．まず，変数d_1，…，d_nを含む式α∈DEXPにおいて，各d_i（$1 \leq i \leq n$）にそれぞれβ_i（β_i∈dom(d_i)）を代入して得られる式を$\alpha\ [d_1/\beta_1, \cdots, d_n/\beta_n]$とかく．また，$\alpha_1$，$\alpha_2$∈DEXPとし，$\alpha_1$または$\alpha_2$のいずれかに含まれる変数を$d_1$，…，$d_n$とする（$\alpha_1$，$\alpha_2$の両方に含まれる変数も許す）．任意の組$\beta_1$，…，$\beta_n$（$\beta_i$∈dom($d_i$)，$1 \leq i \leq n$）に対し，集合，

$$\alpha_1\ [d_1/\beta_1, \cdots, d_n/\beta_n] \wedge \alpha_2\ [d_1/\beta_1, \cdots, d_n/\beta_n]$$

を考え，それらの全ての集合和をとって得られる集合が空集合でないとき，それを$\alpha_1 \wedge \alpha_2$と定義する．最小上界$\alpha_1 \vee \alpha_2$も同様に定義する．dom(d_i)'∈2^D($1 \leq i \leq n$)を次の条件を満たす最大集合の組と定義する．

[条件] dom(d_i)'⊆dom(d_i)であり，かつ任意のβ_i∈dom(d_i)'($1 \leq i \leq n$)に対し，次の(1)，(2)が成り立つ．

(1)$\alpha_1 \wedge \alpha_2 \sqsubseteq \alpha_1\ [d_1/\beta_1, \cdots, d_n/\beta_n] \cap \alpha_2\ [d_1/\beta_1, \cdots, d_n/\beta_n]$．

(2)もし，組β_1'，…，β_n'（β_i'∈dom(d_i)，$1 \leq i \leq n$）に対して，$\alpha_1 \wedge \alpha_2 \sqsubseteq \alpha_1\ [d_1/\beta_1', \cdots, d_n/\beta_n'] \cap \alpha_2\ [d_1/\beta_1', \cdots, d_n/\beta_n']$が成り立つならば，ある$j$($1 \leq j \leq n$)が存在して，$\beta_j' \not\leq \beta_j$． □

dom(d_i)'($1 \leq i \leq n$)は，α_1，α_2に現れる変数の動く範囲を，単一化$\alpha_1 \wedge \alpha_2$によって得られる情報を用いて制限したものである．例えば，例3.1において，seq(d)∧set(int)＝seq(int)であり，dom(d)'＝{int}である．属性datatypeの値(u,α)に単一化操作を行った場合，αに現れる変数dの動く範囲はそれ以降dom(d)'に制限されると定める．以下，簡単のため，(u,$\alpha\downarrow$)（u∈{i,n}，α∈D）を，$\downarrow$を省略して(u,α)とかく．

3.2 例

The file contains a record.

を考える．contains，a record，the fileに対する中間表現は，それぞれ，

```
[surface contains]        [surface a record]
[(i,bool)]                [(i,record)]
                          [det a]
[property contain(X,Y)]
[args  [Y [(i,d)]     ]]
       [X [(i,set(d))]]
```

```
[surface the file]
[(i,file(d))]
[det the]
```

となる．上のcontainsの中間表現で，yとxの属性data-typeにおいて，dは，同一の指標をもつ（共有された）変数とする．ここでcontainsとa recordが結び付くと，d↓∩record↓＝record↓より次の中間表現が得られる．

```
[surface contains a record
(i,bool)
property contain(X,Y)
args    X[[ (i,set(record)) ]]]
```

ここで，

```
[Y (i,record)
  det a]
```

さらに，これと，the fileの中間表現より

```
[surface the file contains a record
(i,bool)
property contain(X,Y)]
```

ここで，

```
X [(i,file(record))
   det the]
Y [(i,record)
   det a]
```

但し，file(d)↓∩set(record)↓＝file(record)↓と仮定している．

3.3 データタイプの情報を用いた曖昧性の除去

A batch preceding the T1 card
which consists of T3 cards.

において，自然語の構文を考えただけでは，関係代名詞whichの先行詞がbatchであるかfirst T1 cardであるかには曖昧性が残る．しかし，これらの句の中間表現が，

```
[surface batch]      [surface first T1 card]
[(n,seq(card))]      [(n,card)]
[   ...    ]      [   ...    ]
```

```
[surface which consists of T3 cards
(i,bool)
property  consist(X,Y)
          ...                ]
```

ここで

```
X  [(i,set(card))
    gap +]
```

であるとし，任意のデータタイプtに対してseq(t)≦set(t)，また，cardとset(card)の間には順序関係がないと仮定すると，関係代名詞whichのデータタイプset(card)↓と，名詞batchのデータタイプseq(card)↓は，seq(card)↓に単一化できる．これに対し，set(card)↓と，句first T1 cardのデータタイプは単一化できない．従って，batchはwhichの先行詞になり得るが，first T1 cardはなり得ない．このように，単一化によって表現されるデータタイプに関する制約条件を考えることにより，構文解析時の曖昧さを減少させることができる．

4．複数名詞の取り扱いについて

英語の複数名詞の形式的な意味に関する研究は，たとえば，Carlson[4]によるものがあるが，プログラム仕様の意味定義という目的からは形式化が不十分である．本研究では，各複数名詞句は抽象データタイプ集合に対応するとして，以下のように意味の定義を行っている．

4.1 仕様記述における複数名詞の機能

以下の三つの例を考える．

The file contains customer records. (4-1)

Count the cards. (4-2)

This stream is available as a sequence of digits. (4-3)

文(4-1)のrecordsはthe fileに含まれるようなrecordの集合に対応する．この文におけるcontainsの意味は，

The file contains a record. (4-4)

におけるcontainsの意味の拡張として定義できる．すなわち，(4-1)はrecordsに含まれる各recordに対し，単数の場合のcontainsが成り立つという意味である．これを以下では（単数の場合の）拡張とよぶ．

文(4-2)においても，複数名詞句the cardsは集合に対応している．しかし，これは(4-1)の場合と異なり，単数の場合の拡張にはなっておらず，the cardsの意味要素が，直接，countに対応する述語の引数となる．

(4-3)において，句sequence of digitsは，数字からなる系列というものの性質を表しており，複数名詞句digitsは，digitのある集合というより，“digit”というデータタイプの名前に対応すると考える．

論理式への変換の際，この三つは異なる取り扱いをしなければならない．以下その理由を述べる．

(4-1)と(4-2)は一見同様な取り扱いができるように思える．すなわち，customer records，the cardsに対応するような集合型の変数，CR，Cを考え，(4-1)，(4-2)に対応する論理式をそれぞれ，

contain*(f,CR)，count(C)

とする．そして，contain*(f,CR)の意味を，単数の場合のcontainsに対応する述語contain(f,cr)を用いて，

contain*(f,CR) ⇔ ∀cr∈CR⊃contain(f,cr) (*)

と定義する．(*)は語句“contain*”の定義の問題であるし，CR，Cの量化は，(4-1)，(4-2)に共通の問題である．しかしこの方法では(4-5)のような文の意味をうまく定義できない．

All processes print their current states. (4-5)

all processes，their current statesの意味要素を，それぞれPR，STとすると，前述の方法では，(4-5)に対応する論理式は，

print*(PR,ST)

という，集合PRとSTの関係を述べるものとなる．しかしながら，(4-5)はすべてのプロセスは自分自身のcurrent stateをprintするというものであり，PR，STに属する個々の元の関係についても言及している．print*の語句の

定義によりこの関係を表すのは不自然であろう．

4.2 複数名詞句を考慮した中間表現

4.1で述べた問題点を解決するため，名詞句に対応する中間表現に，複数形であることを表す属性plu，及び単数形の拡張としての複数形を表す属性extを導入する．そして，複数形の中間表現としては，単数形の拡張としての取り扱いを受ける場合と，そうでない場合に対応して，2種の中間表現を用意する．ただし，単数形の拡張とみなす場合，属性datatypeは単数形のそれと等しくしておく．例えば，cardおよびcardsの中間表現は次のようになる：

```
[surface  the card
(i, card)
plu  -                ]
```
(4-6)

```
[surface  the cards
(i, card)
plu  +
ext  +                ]
```
(4-7)

```
[surface  the cards
(i, set(card))
plu  +
ext  -                ]
```
(4-8)

これに対し，引数に単数形をとらない句の中間表現には，対応する場所にplu +をつけておく：

```
[surface count
(i, bool)
property count(X, Y)
args  [Y [(i, set(d))
         plu  +      ]
       X [(i, program)]]]
```
(4-9)

```
[surface contains
(i, bool)
property contain(X, Y)
args  [Y [(i, d)]
       X [(i, set(d))]]]
```
(4-10)

(4-6)，(4-9)では，属性pluの値が異なるので，句count the cardは許されない．また，(4-7)，(4-9)では，属性datatypeの単一化が行えない（countの引数yはデータタイプがset(d)の形でない単数形の拡張としての複数形と認めないことを示す）．(4-8)と(4-9)なら，単一化が行え，

```
[surface  count the cards
    ...
args  [Y [(i, set(card))
          plu  +
          ext  -
           ...          ]]]
```

が得られる．一方，(4-6)～(4-8)と(4-10)はいずれも結び付くことができる．特に，(4-7)と(4-10)からは，

```
[surface  contains the cards
args  [Y [(i, card)
          plu  +
          ext  +       ]]]
```

が得られ，countの引数yが単数形（データタイプはcard）の拡張であることを表している．

また，複数名詞句がデータタイプの名前に対応する場合は，それが単数の場合のデータタイプの集合型の名前となることはないので，単数の場合とまったく同様に扱う．たとえば，

```
[surface sequnce          ]   [surface of digits]
(n, seq(d))                   (n, digit)
property sequence(X)          comp of
args  [X [(n, d)]]            ]
```

に対し，

```
surface sequence of digits
(n, seq(digit))
property sequence(X)
```

ここで，

```
X [(n, digit)
   comp of   ]
```

となる．

4.3 等位接続された名詞句について

andにより等位接続された名詞句も4.2の方法で取り扱うことができる．例えば，文

x, y and z are integers.

は，次の文と等価であると考えられる．

x is an integer and
y is an integer and
z is an integer.

このような“等位接続の展開”は，4.2で述べた単数形の場合の拡張としての複数形の意味として扱うことができる．等位接続に関する構文規則は，2.3.2の関数適用の規則に分類される．例として構文規則，

NP→NP and NP

を考える．句s1，s2がNPから生成されるとし，s1，s2の中間表現をそれぞれt1，t2とすると，s1 and s2に対応する中間表現は，複数名詞句の場合と同様，

```
[surface   s1 and s2
property   and(X, Y)
           X t1  Y t2
datatype   set(α1∨α2)
plu        +
ext        -          ]
```

と

```
[surface   s1 and s2
property   and(X, Y)
           X t1  Y t2
datatype   α1∨α2
plu        +
ext        +          ]
```

であると定義する．例えば，The cards and the batchesの中間表現は，

```
[surface  the cards and the batches
(i, set(set(card∨batch)))
plu       +
ext       -
          ...                      ]
```
(4-11)

及び，

```
[surface  the cards and the batches
(i, set(card∨batch))
plu       +
ext       +
          ...                      ]
```
(4-12)

となる．4.2の(4-9)と(4-12)より，

```
[surface  count the cards and the batches
property count(X,Y)
         Y[(i,set(card\batch))
           plu  +
           ext  +
            ...
   ...                                    ]
```

が得られ，引数yの属性extが+であることにより，上の句が，count the cardsとcount the batchesに“展開”できることを表している．

5．あとがき

本稿で述べた方針に従って，自然語による仕様から論理式への変換及び語句の意味の定義の蓄積を支援するシステムを作成した．

論理式への変換部は①構文解析部，②中間表現構成部，③論理式生成部の３つからなる．①，②はC-prologで記述され，プログラムサイズは①がDCGのレベルで約400行，②も約400行程度である．また③では，指示代名詞の先行詞，量記号の有効範囲などは，簡単なヒュウリスティック及び，利用者への質問により決定している．この部分は言語Ｃを用いて記述されており，行数は約1500行である（ただし，機能拡張を計画中）．

このシステムを用いて，実際に文献(5)の例題の解析を行っている．

【謝辞】　文献(1,2,3)等について御教示頂いた，本学言語文化部の郡司隆男助教授，および，BUPトランスレータを快く提供して下さった，（財）新世代コンピュータ技術開発機構の松本裕治氏に深謝します．

文　　献

(1) C. J. Pollard: “Lectures on HPSG”, unpublished manuscript, Stanford University (1985-02).

(2) C. J. Pollard and I. A. Sag: “Information-Based Syntax and Semantics, Vol. 1: Fundamentals”, CSLI Lecture Notes Series No. 12, Center for the Study of Language and Information, Stanford University, also published by The University of Chicago Press, Chigo(1987)

(3) G. Gazdar, E. Klein, G. Pullum and I. Sag: “Generalized Phrase Structure Grammar”, Basil Blackwell (1985).

(4) G. N. Carlson: “Reference to Kinds in English”, Garland Publishing, Inc (1980).

(5) M. A. Jackson: “Principles of Program Design”, Academic Press (1975).

(6) R. M. Kaplan and J. Bresnan “Lexical-Functional Grammmar: A Formal System for Grammatical Representation”, in J. Bresnan: ‘The Mental Representation of Gramatical Relations’ Cambridge, Mass., The Mit Press(1982).

(7) 市川，蓬莱，佐伯，米崎，榎本：“自然言語に基づく静的システムの仕様のプロトタイププログラムへの変換手法”，情処論，Vol. 27, No. 11, pp. 1112-1118(1986)

(8) 関，並河，藤井，嵩：“自然語によるプログラム仕様の形式的意味定義　－自然語による仕様から代数的仕様への変換－”，情報処理学会研報，SW-52-7 (1987-2).

(9) 並河，松村，関，藤井，嵩：“プログラム仕様に用いる自然語の形式的意味定義について”，情報処理学会研報，NL-63-7(1987-9)

(10) 並河　英二：“自然語によるプログラム仕様から代数的仕様への変換法について”，大阪大学基礎工学研究科修士学位論文(1988-02)