ソフトウェア工学　52 — 7
（１９８７・２・１３）

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 —

関　浩之　　並河　英二
藤井　護　　嵩　忠雄
大阪大学基礎工学部情報工学科

プログラム仕様に用いる自然語の部分言語の構文と意味の形式的定義を，代数的言語ＡＳＬ／＊を用いて行った．ＡＳＬ／＊では，文脈自由文法で式(文)の集合とその構文を，また，公理の集合で文の集合の上の合同関係を定義できる．本稿では，まず，プログラム仕様の記述に有用と思われる英語の部分言語L_{NS}を設定し，L_{NS}の構文を，文脈自由文法を用いて定義する．文法の記述法としてはＧＰＳＧを採用した．次に，(1)L_{NS}から，解釈をエルブラン解釈に限定した多ソートの一階述語論理に対応する論理体系ＭＨ１の論理式の集合への変換則，(2)個々の語句の定義，及び，(3)論理体系ＭＨ１の定義を与える公理を記述することにより，L_{NS}の各文の意味を定義する．また，この意味定義に即して実際の例題を解析した結果を示す．

A Formal Definition of the Semantics of Program Specifications Written in a Natural Language

Hiroyuki SEKI, Eiji NABIKA, Mamoru FUJII and Tadao KASAMI
Department of Information and Computer Sciences
Faculty of Engineering Science, Osaka University
Toyonaka, Osaka, 560 JAPAN

A formal definition of a "fragment" L_{NS} of English which is used for writing program specifications is described using algebraic language ASL/*. The syntax of L_{NS} is defined by a generalized phrase structure grammar. The semantics of L_{NS} is defined by a set of axioms which gives (1) the definition of the transformation rules from L_{NS} to the set of formulas in a many sorted first order logic called MH1, (2) the definition of 'non-logical' words such as nouns and verbs, and (3) the definition of MH1. An example specification is analysed using the definition here.

１．まえがき

プログラムの段階的詳細化において，具体化レベルの記述が抽象レベルの記述を満たしているかどうかの形式的議論が行えるためには，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．ところで，書き手の意図をできるだけ正確に記述に反映し，かつ，記述の意味するところを直観的にわかり易いものにするためには，書き手が日常用いる自然語L_Nによって記述を行うことが望ましい．そこで，仕様の記述能力は保ちつつ，できるだけ限定したL_Nの“断片”（部分言語）L_{NS}を設定する．そして，記述のための構文の自由度が十分大きいような，形式的記述言語L_Sを一つ採用し，L_Sを用いてL_{NS}の構文及び意味定義を行う．

筆者らは，既に，文献(7)(8)で述べた理由により，L_Sとして代数的言語ＡＳＬ／＊を選択し，L_{NS}として英語の部分言語を例にとり，L_{NS}の意味定義法についての検討を行ってきた．本稿では，そこでの基本方針に従いL_{NS}の意味定義を与え，その意味定義に即して例題を解析した結果を示す．

２．代数的手法による意味定義の基本方針

2.1 意味定義の基本方針

自然語L_Nの部分言語L_{NS}の意味を定めるため，L_{NS}に属する各文ｓを，適当な論理体系のもとでの論理式に変換する規則を定める．論理式への変換による意味定義の方法として，Montagueらの意味定義法[1]を参考にした．ただし，Montagueらは，論理体系として内包論理を採用しているが，我々は，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に対応する論理体系（ＭＨ１とよぶ）を採用する．

論理における語いは，a)変数の集合，b)論理結合子及び量記号“∀”，“∃”からなる論理記号の集合と，c)対象定数，関数記号，述語記号からなる非論理記号の集合に分割される．自然語においても，語いは，対象領域には依存せずにその論理的意味が定まっていると考えられるもの，例えば冠詞，接続詞や代名詞と，対象領域ごとに意味が定義される一般の語句とに分割されると考えられる．前者を論理的要素，後者を非論理的要素とよぶ．本手法では，次の(1)～(3)の方法で意味定義を行う．

(a) 論理体系ＭＨ１の定義を行う．

(b) 論理的要素の意味を定義するため，L_{NS}からＭＨ１の論理式の集合L_{MH1}への変換則を定める．

(c) 非論理的要素の意味を，各対象領域毎に個別に定義する．

以下では，変換則の定義法の概略を述べる．自然語文，または自然語文における全部または一部の名詞句を変数で置き換えて得られる系列を，以下“擬似自然語文”とよぶ．特に，すべての名詞句を変数で置き換えて得られる系列を，“素擬似自然語文”とよぶ．同様に，名詞句において，部分句としての名詞句をすべて変数で置き換えて得られる系列を，“素擬似名詞句”とよぶ．擬似自然語文，素擬似自然語文，素擬似名詞句の集合を，それぞれ，PsL_{NS}，KnL_{NS}，KnNPとかく．ｓを擬似自然語文，ｓ’を論理式とする．“ｓとｓ’が対応する”という述語を，以下のように再帰的に定義する．

(1) s_0を素擬似自然語文とする．s_0の変数ｘに素擬似名詞句αを代入した式をs_1とし，s_0に対して，αで定まる変形をほどこして得られる式をs_1'とする（例えば，α＝a recordならば，$s_1' = \exists x[x \in \text{record} \wedge s_0]$）．このとき，$s_1$と$s_1'$は対応する．

(2) s_{i-1}とs_{i-1}'が対応していると仮定する．(1)と同様の操作によりs_{i-1}及びs_{i-1}'から得られる式をそれぞれs_i，s_i'とすると，s_iとs_i'は対応する．

(3) 上の(1)(2)で対応するといえるもののみが対応する．□

ｓとｓ’が対応し，かつ，ｓが自然語文，すなわち変数を含まない擬似自然語文ならば，ｓはｓ’に変換されるといい，ｓの意味を，論理体系ＭＨ１におけるｓ’の意味と等しいと定義する．

2.2 ＡＳＬ／＊による意味定義法

2.1の方針に従い，L_{NS}の構文と意味を代数的言語ＡＳＬ／＊[6]を用いて定義する．ＡＳＬ／＊では，文脈自由文法によって，議論の対象となる文（式，表現式などともいう）の集合とその構文を指定し，（一般に条件付きの）公理の集合によって，文の集合の上の合同関係（演算で閉じた同値関係）を指定する．これらの機能を利用し，L_{NS}，PsL_{NS}，KnL_{NS}（⊇KnNP），L_{MH1}の構文は文脈自由文法を用いて定義する．詳細は３．で述べる．また，L_{NS}の意味はＡＳＬ／＊の公理を用いて定義する．その公理は，2.1の(a)，(b)，(c)に対応して３つに分類される．(b)については４．で詳述する．また，(c)の例を５．で示す．(a)に関しては文献(7)(8)を参照されたい．これらの公理によって指定される合同関係を≡とするとき，自然語文ｓと論理式ｓ’に対し，$s \approx s' \equiv \text{true}$であるとき，かつそのときのみ，ｓはｓ’に変換されるという．

３．構文則の定義

3.1 構文則の記述方針

前述のように，我々は英語の断片L_{NS}の意味定義をＡＳＬ／＊の枠組み内で行おうとしている．したがって，構文は文脈自由文法を用いて定義する．我々が考えている英語の断片を，通常の文脈自由文法の記法で記述しようとすると，規則の数がかなり多くなり，文法全体の見通しが悪くなる．そこで，GPSG[4]を参考にして，非終端記号が引数を持つ形で構文則を記述した．ただし，引数がとりうる値の数を有限としたので，この文法は文脈自由文法に展開することができる．

また，GPSGは自然言語に対しより一般的な説明を与えようとして，複雑なマクロを用いているが，我々は対象を仕様記述に用いる英語の断片にしぼり，比較的容易に実現できるマクロを用いて構文則を記述している．

3.2 構文則の記述

3.2.1 非終端記号の記述法

文法の非終端記号は，$A\,[F_1\ fv_1, \cdots, F_{n_A}\ fv_{n_A}]$（$n_A$はＡにより決まる整数）の形で記述する．Ａをカテゴリ名と呼ぶ．例えば，S, VP, AP, NP, N, PP, DETは，それぞれ，文（または従属節），動詞句，形容詞句，名詞句，一般名詞句，前置詞句，冠詞を導出する非終端記号のカテゴリ名である．また$F_1, \cdots, F_{n_A}$を引数，$fv_1, \cdots, fv_{n_A}$を（それぞれ，引数$F_1, \cdots, F_{n_A}$の）値と呼ぶ．各カテゴリ名がどのような引数をとるかは定まっており（表１参照），また，各引数がとりうる値も有限個定まっている．例えばVPは，AGR, VFORM, CONJ, SLASH等を引数に持つが，その取りうる値と，働きは次のとおりである．

AGR: 動詞の形は主語の人称，数により変化する．また文や，形式主語がとれる動詞は限られている．この引数があれば，動詞句は，主語を生成する非終端記号が何であるかを知り，これらの性質の記述ができる．値は，NP[ATT f]，S．(NP，Sはそれぞれ，名詞句，文を導出するカテゴリ名である．また，ATTは名詞が形式主語かどうか，及び，人称，数，格が何であるかをを表わす４字組の引数で，fはATTの任意の値である．)

VFORM: 動詞の活用を示す引数．取りうる値は，BSE, FIN, PRP, PASでそれぞれ，原形，現在形，現在分詞，過去分詞に対応する．

CONJ: 先頭に接続詞を持つかどうか，及び，持つ場合それが何かを示す引数．その値は NIL, and, both, but, neither, nor, or

SLASH: 関係代名詞節を生成する際，関係代名詞を含む句に変化して節の頭部に移動した句は，もとの場所から消去

表1　カテゴリ名とその引数

STATEMENT
S [NEG, VFORM, AUX, CONJ, COMP, WH, SLASH]
VP [NEG, AGR, VFORM, AUX, COMJ, SLASH]
VL [NEG, AGR, VFORM, AUX, SUBCAT, CONJ]
AP [AGR, CONJ, SLASH]
A [AGR, CONJ, SLASH]
AL [AGR, SUBCAT, CONJ]
NP [ATT, CONJ, SLASH, WH, APP, ORD]
N [ATT, CONJ, SLASH, WH, TYPE]
NL [ATT, CONJ, WH, SUBCAT, TYPE]
PP [PFORM, LOC, CONJ, SLASH, WH]
P [PFORM, LOC, SLASH, WH]
PL [PFORM, LOC, SUBCAT]
PP_G [PFORM]
DET [AGR, WH]
SUB [SUBCAT]

されなければならない．その際消えるものの非終端記号を，この引数により記憶する．

これらの引数に具体的な値を入れることにより，導出される動詞句の形が指定できる．例えば，VFORMの値がFIN，AGRの値が三人称単数の名詞であるVPからは，" updates a file" が，VFORMの値がBSEであるVPからは，" update a file" が導出される．

3.2.2　マクロの定義と構文則

生成規則において，引数の値をすべて明記して非終端記号を記述するのは繁雑なので，以下のマクロを用いて記述する．

1）引数の省略

生成規則Pにおいて，非終端記号の引数の値が明記されていない場合には，以下に述べる条件 a)～ c) を満たす範囲で，それらの引数に許される値を任意に補って得られる可能な生成規則すべてが記述されたとみなす．

a) HEAD={NEG, VFORM, AGR, ATT, PFORM, LOC} に属する引数の値が生成規則に明記されていない時は，それらの引数の値は，左辺の非終端記号と，右辺において特に指定された非終端記号では等しくなければならない．非終端記号の指定はカテゴリ名にインデックスHをつけて行う．

b) 左辺の非終端記号におけるSLASHの値は右辺のsのついた非終端記号におけるSLASHの値の少なくとも1つに等しくなければならない．

c) 引数CONJ及びWHについては，引数の値が明記されておらず，かつ，上記a)，b)によっても値が決らないとき，値NILが指定されているとみなす．

2）メタルール

メタルールとは生成規則から新たな生成規則を作り出す規則である．受動態を作るメタルールと，形容詞，名詞の補語を省略するメタルールがある．

3）その他

a) V^2はSまたはVPを表わす．

b) Cをカテゴリ名とすると，C[SLASH s] (sはSLASHの任意の値) は，C/sと記述する．また，C[SUBCAT n]は，C[n]と記述する．(SUBCATは文型に即して単語分類するための引数で，値は整数値（有限）を取る．)

c) Aを非終端記号とする．ここで，A^+は，Aの一回以上の繰り返しを表わす．

d) Aを非終端記号，α_1，$\cdots\cdots$，α_nをそれぞれ，非終端記号の系列とする．このとき，

$A \rightarrow \alpha_1 \mid \cdots\cdots \mid \alpha_n$

は，

$A \rightarrow \alpha_1$
：
$A \rightarrow \alpha_n$

とみなす．ただし，この記法は，各α_i（$1 \leq i \leq n$）において，非終端記号の順序だけが違うときに用いる．

3.2.3　構文則の記述

3.2.2で述べたマクロを用いて，L_{NS}の構文を定義する文法を記述した．紙面の都合上，5.の例で用いる規則を含めた一部の規則のみを表2に示す．各々の詳細な説明は省略する．現在，終端記号を導出する規則を除き，87の規則がある．

なお，自然語文の集合L_{NS}だけでなく，擬似自然語文の集合PsL_{NS}，論理式の集合L_{MH1}の構文も定義しなければならない．これらは，ここで述べた文法を若干変更することにより得られる．また，MH1のアトムの構文は，L_{NS}の構文と同一とする．PsL_{NS}を生成する文法における非終端記号Aに対応するKnL_{NS}，KnNPでの非終端記号をA_{kn}で表し，変数を生成する非終端記号をVARとする．また，L_{MH1}の論理式全体を生成する非終端記号をBOOLとする．詳細は省略する．

4．変換則の定義

4.1　変換則の定義

2.で述べた方針に従い，変換則，すなわち，述語 "≈" を定義する公理を与える．本稿での定義法では，自然語文sを変換した結果得られる論理式s'において異なる変数が存在する場合，それらの変数の有効範囲は，sの構文によって定まるとする．例えば，for each record it has a flag の構文は，for each record (it has a flag) と考えられ，対応する論理式は

$\forall x[x \in \text{record} \supset \exists y[y \in \text{flag} \wedge \text{for x (x has y)}]]$

であると考えられる（ただし，実際は，for x (x has y) とx has y とが等価となるよう，語句 for の意味を別途定義する．また，擬似名詞句αに対し，$x \in \alpha$ でxがαの表すデータタイプの式であることを表す（"∈" の定義は5.参照））．そこで，変換則を定義する際，sに現れる変数xへの素擬似名詞句の代入は，sに，xよりも "有効範囲の狭い" 変数が存在しないときのみ許すように定義する．上の例では，for x (x has y) において，変数xよりも変数yの方が有効範囲が狭いことが構文より定まっているので，変数yへの代入が行われない限り，変数xへの代入は行えない，と定める．一般に，式αの変数xに代入を行うことが許されることを表す述語を，ok_sbst_for x in αで表す．

以下，ASL／＊の公理を用いて変換則を記述する．

$\ell_1 == r_1,\ \ell_2 == r_2,\ \cdots,\ \ell_m == r_m \gg L == R$

で条件付き公理を表す．特に条件部がない場合は $L == R$ とかく．ただし，公理に現れる各変数xには，非終端記号Aが一つ対応づけられているとし，xにはAから生成される任意の表現式が代入できることを表す．Aをxのタイプ指定とよぶ．簡単のため，条件部が等しいn個の公理を一つにまとめて，$\ell_1 == r_1,\ \cdots,\ \ell_m == r_m \gg L_1 == R_1,\ \cdots,\ L_n == R_n$ とかく．また，$\ell == \text{true}$ の形の公理の条件部または本体を，ℓと略記することがある．さて，2.で述べたように，擬似自然語文に代入される句は，素擬似名詞句である．従って変換則を定義する公理において，素擬似名詞句を表す変数のタイプ指定は，NP_{kn}のように，3.で導入した非終端記号に添字 'kn' をつけた非終端記号となるが，読み易さのため，あいまいさのない限り 'kn' を省略する．また，変数のタイプ指定は，3.での構文則の記述の際のマクロ記法を用いる．例えば，非終端記号の引数を省略した場合，任意の引数値の組合せに対して公理を記述したと解釈する．E(A)で非終端記号Aから生成される式全体の集合を表す．ただし非終端記号の引数を省略した場合は，任意の引数値に対してそれらの和集合をとったものを表す．公理の変数及びそのタイプ指定

表2　L_{NS}の構文則

[S1] STATEMENT→S[COMP NIL, VFORM FIN] SUB[.]
/* 平叙文 */
[S2] STATEMENT→VP[VFORM BSE] SUB[.]
/* 命令文 */
[S3] S[WH w, COMP NIL]→
NP[ATT f*NOM, WH w]$_S$ VP[AGR NP[ATT f*NOM]]$_S$
/* 文→名詞句 動詞句 */
[S4] S[WH w, COMP NIL]→NP[ATT f, WH w] S/NP[ATT f]
/* 名詞句の移動 */
[S5] S[WH w, COMP NIL]→PP[PFORM x, WH w] S/PP[PFORM x]
/* 前置詞句の移動 */
[S6] S/NIL→V^2[PAS] $S_{H:s}$ [S7] S/NIL→V^2[PRP] $S_{H:s}$
/* 分詞構文 */
[S8] VP→$VP_{H:S}$ ADVP| ADVP $VP_{H:s}$
/* 副詞による修飾 */
[S9] NP[WH w]→DET[AGR NP[ATT f], WH w] $N_{H:s}$
/* 名詞句→冠詞 一般名詞 */
[S10] DET[AGR NP[ATT[NFORM NORM, PER 3, -PLU, CASE γ]],
WH NIL] → a
[S11] DET[AGR NP[ATT[NFORM NORM, PER 3, -PLU, CASE γ]],
WH NIL] → every
[S12] DET[AGR NP[ATT[NFORM NORM, PER 3, PLU α, CASE γ]],
WH NIL] → the
[S13] NP[+APP, ORD ord]/NIL→
NP_H[-APP, ORD ord1] NP_H[-APP, ORD ord2]
/* 同格 */
[S14] NP[WH NIL]/NIL→NL_H[TYPE type]
/* NL[TYPE type]はデータタイプを表わす名詞 */
[S15] NP[ATT f, WH NIL]/NIL→
NL[ATT f, TYPE unit] PPNAME[ATT f]
/* NL[TYPE unit]は単位を表わす名詞 */
[S16] NP[WH w]/NIL→PRON[WH w]
/* PRONは代名詞 */
[S17] NP[WH NIL]/NIL→PPNAME
/* PPNAMEは固有名詞 */
[S18] N→$N_{H:S}$ PP[PFORM NIL]
/* 前置詞句による名詞の修飾 */
[S19] N→N $N_{H:s}$ [S20] N→AP $N_{H:s}$
/* 名詞の形容詞的用法 */ /* 形容詞の名詞の修飾 */
[S21] N→$N_{H:S}$ VP[PAS] [S22] N→$N_{H:s}$ VP[PRP]
/* 分詞による名詞の修飾 */
[S23] N/NIL→N_H VP[INF]/NP
/* 不定詞の形容詞的用法 */
[S24] N[ATT f]→$N_{H:S}$ S[WH [R, ATT f]]
[S25] N→$N_{H:S}$ SUB[such that] S[VFORM FIN, COMP NIL, WH NIL]
/* 関係代名詞 */
[S26] AP→ADVP $AP_{H:s}$
/* 形容詞句→副詞，形容詞句 */

[S27] VP/NIL→VL_H[1] (die)
[S28] VP→VL_H[2] NP (love)
[S29] VP[+AUX]→VL_H[7] XP_S (be)
[S30] XP→$VP_{H:s}$[PRP] [S31] XP→$VP_{H:s}$[PAS]
[S32] XP→$AP_{H:s}$ [S33] XP→$NP_{H:s}$
[S34] XP→$PP_{H:s}$[PFORM NIL]
[S35] VP[INF]→VL_H[12] VP_H[BSE]$_S$ (to)
[S36] VP[AGR NP[there, plu]]→VL_H[22] NP[plu]$_S$ (be)
[S37] VP→VL_H[25] PP[of]$_S$ (consist)
[S38] AP→SUB[29] $A_{H:s}$
[S39] A/NIL→AL_H[45] (big)
[S40] N/NIL→NL_H[36] (death)
[S41] N→NL_H[37] PP[with]$_S$ PP[about]$_S$ | (argument)
NL_H[37] PP[about]$_S$ PP[with]$_S$
[S42] P[WH w]→PL_H[44] NP[WH w]$_S$ (in)
[S43] S[WH w]→S[CONJ c_0]$_{H:s}$ S[CONJ c_1]$_{H:s}$
[S44] NP[+ORD]→NP[CONJ c_0, -ORD]$_{H:s}$ NP[CONJ c_1, -ORD]$_{H:s}$
[S45] X→X[CONJ c_0]$_{H:s}$ X[CONJ c_1]$_{H:s}$
ただし <c_0, c_1>∈{<both, and>, <either, or>,
<neither, nor>, <NIL, but>}
X∈{VP, VL, AP, A, AL, N, NL, PP}
[S46] S[WH w]→S[WH w]$_{H:s}^{+}$ S[CONJ c]$_{H:s}$
[S47] NP[+ORD]→NP[-ORD]$_{H:s}^{+}$ NP[CONJ c_1, -ORD]$_{H:s}$
[S48] X→$X_{H:s}^{+}$ X[CONJ c]$_{H:s}$
ただし c∈{and, or}
X∈{VP, VL, AP, A, AL, N, NL, PP}
[S49] X[CONJ c]→SUB[c] X[CONJ NIL]$_{H:s}$
ただしX∈{S, VP, VL, AP, A, AL, NP, N, NL, PP}
[S50] SUB[and]→ and [S51] SUB[or]→ or

(非終端記号) は，以下のとおり．
(1) x, y, …等は，タイプ指定がVARの変数．すなわち，擬似自然語文や論理式自身に現れる変数を表す．
(2) α, β, …等は，素擬似名詞句を構成する句を表す変数．そのタイプ指定は，公理毎に記述する．αのタイプ指定がNであるとき，α: Nと記述する．
(3) s, s_1, s_2, …は，タイプ指定がSである変数．すなわち，それらは擬似自然語文を表す．
(4) s', s_1', s_2', …は，タイプ指定がBOOLである変数．すなわち，それらは論理式を表す．

以下で，3.で記述した構文則[Sn]に対応する変換則にはラベル[Tn]を付している．

[変換則T9] α: N (一般名詞)
s≈s' , ok_sbst_for x in s >>
[1] s{{x/a α}} ≈
if GEN(s, x) then ∀x[x∈α <=> s']
else ∃x[x∈α ∧ s']
[2] s{{x/every α}} ≈ ∀x[x∈α ⊃ s']
[3] s{{x/no α}} ≈ ¬∃x[x∈α ∧ s']
[4] s{{x/the α}} ≈ s'{x/α}
[5] s{{x/α}} ≈ s'{x/α}

[T17] β: PPNAME (固有名詞)
s≈s' , ok_sbst_for x in s >>
s{{x/β}} ≈ s'{x/β} □

ここで，s{{x/α}}は，式sの最初に現れるxにαを，それ以降のxに適当な指示代名詞を代入して得られる式．ただし，s中でxが主語となっているとき，αを生成する非終端記号の引数値PLU(数)，PER(人称)の値に従って，sの動詞句を変形する．s'{x/α}は，s'の全てのxにαを代入して得られる式を表す．s{{x/α}}，s'{x/α}のASL／＊による定義は省略する．[T9]の[1]において，GEN(s, x)は，文sにおいて，xにa αの形の名詞句を代入したとき，それが，an even number is an integer which can be divided by 2. における先頭の an のように 'generic' な意味をもつ (全称記号に対応する) か (値はtrue)，または，存在記号に対応するか (false) を表す述語である．GEN(s, x)の定義は，s中の動詞句を構文的に分類する (例えばxが現在進行形の主語ならばfalse) こと等によって定める．詳細は検討中である．[T9]の[1]は，擬似自然語文sと論理式s'が対応することが既に知られており，かつsの変数xに擬似名詞句の代入を行えるとき，GEN(s, x)がtrueならば，s{{x/a α}}は∀x[x∈α <=> s']と対応し，falseならば，∃x[x∈α ∧ s']と対応することを表す．[T9]の[4]，[5]及び[T17]は，the を冠詞として持つ名詞句，抽象名詞，固有名詞については，論理式においても，これらを"項"とみなして代入すること等を表す．冠詞 some 及び形容詞句 at least one 等については上の[1]のelse部，all, each, any については[2]と同様に定義する．

なお，s{{x/α}}の代りに，2つめ以降のxに，指示代名詞ではなく，αの“核”となる名詞（αの構文木において，根から（構文則において）インデクスHのついた部分句のみをたどって得られる終端記号）にtheを付けた名詞句（例えばα＝every record of a fileならば，the record）を代入する場合も同様に定義する．

〔例〕should be updated∈E(VP)，かつ
record，file∈E(N)とする．

s_1＝x should be updatedは，擬似自然語文であるから，s_1はs_1自身に対応する．

s_2＝s_1{{x/every record of y}}
＝every record of y should be updated　は
s_2' ＝∀x[x∈record of y⊃x should be updated]
に対応する．　([T9]の[2]より)

s_2{{y/a file}}
＝every record of a file should be updated　は
∃y[y∈file∧∀x[x∈record of y⊃
x should be updated]]
に対応する．　([T9]の[1]より)□

上の例で，“is”は非論理的要素とみなしている．isの定義については5.を参照されたい．

以上の変換則では，擬似名詞句に現れる関係節や分詞による修飾句はそのまま対応する論理式中に現れる．以下では，関係節に関する変換則を与える．

[T24] α: N, β: S[WH [R]]
norm(x, β)≈s' , x is_not_in α , x is_not_in β
>>
x∈αβ <=> x∈α ∧ s'　□

norm(x, β)は，関係節β中，先行詞の抜けた場所に変数xを補い，関係代名詞を含む句が文頭に移動している場合には，それをもとの位置に戻して得られる文．norm(x, β)の定義は省略するが，引数SLASH(3.参照)の値を利用することにより定義できる．分詞による修飾（[S21]，[S22]），不定詞の形容詞的用法([S23])，such thatを用いた関係節([S25])についても同様に定義する．

〔例〕x is an integer which is less than 10 は
∃y[y∈integer which is less than 10 ∧ x is y]
に対応する．　([T9]より)
上の式中の句 y∈integer which is less than 10 は
y∈integer ∧ y is less than 10
に対応する．　([T24]より)□

次に，接続詞で結ばれた名詞句と，同格に関する変換則を定義する．名詞句以外の句が接続詞で結ばれる場合については，後述する．

[T48-1] α_1: NP[-ORD]，…，α_{n-1}: NP[-ORD]，
α_n: NP[CONJ π，-ORD]
$s\{\{x/\alpha_1\}\}\approx s_1'$ ，…，$s\{\{x/\alpha_{n-1}\}\}\approx s_{n-1}'$ ，
$s\{\{x/\alpha_n\}\}\approx s_n'$, ok_sbst_for x in s >>
$s\{\{x/\alpha_1\cdots\alpha_{n-1}\alpha_n\}\}\approx$
s_1' trans(π)…s_{n-1}' trans(π) s_n' □

ここで，trans(π)は，接続詞πに対応する論理結合子で，例えば，trans(and) == ∧，trans(or) == ∨と定義する．構文則[S45]に対応する変換則も同様に定義する．

〔例〕s＝z is givenとする．
s{{x/an integer}}　は
∃z[z∈integer ∧ z is given]　に対応し　([T9]より)
s{{x/a character}}　は
∃z[z∈character ∧ z is given]
に対応する．　([T9]より)
trans(and)＝∧
であるから，s{{z/an integer and a character}}
＝an integer and a character are given　は
∃z[z∈integer ∧ z is given] ∧
∃z[z∈character ∧ z is given]
に対応する．　([T48-1]より)□

次に，同格に関する変換則を述べる．

[T13] α: NP[-APP]，β: NP[-APP]
s{{x/α}}≈s' , ok_sbst_for x in s >>
s{{x/αβ}} ≈ s' ∧ β is α　□

〔例〕s_1＝x is given, an integer∈NP[-APP]，I∈PPNAME⊆NP[-APP]とすると，
s_2＝s_1{{x/y I}}＝y I is given　は
s_2' ＝ y is given ∧ I is y
に対応する．　([T13]より)
従って，s_2{{y/an integer}}＝an integer I is given は
∃y[y∈integer ∧ [y is given ∧ I is y]]
に対応する．　([T9]より)□

最後に，接続詞で結ばれた動詞句に関する変換則を示す．

[T48-2] α_1: VP[-ORD]，…，α_{n-1}: VP[-ORD]，
α_n: VP[CONJ π，-ORD]
x $\alpha_1\approx s_1'$ ，…，x $\alpha_{n-1}\approx s_{n-1}'$, x $\alpha_n\approx s_n'$ >>
x $\alpha_1\cdots\alpha_{n-1}\alpha_n\approx$
(s_1' trans(π) …s_{n-1}' trans(π) s_n')□

〔例〕x is recorded and is updated は
x is read ∧ x is updated
に対応する．([T48-2]より)□

接続詞で結ばれた文についても同様に定義する．

4.2　省略を含む文の意味定義について

自然語によるプログラム仕様ではプログラムの入力パラメタに対応する引数を省略することが多い．ここでは，省略を含む自然語文の意味定義を次のように行う．

(1)省略される可能性のある名詞句の集合を生成する非終端記号の集合SN_{abr}を指定する．

(2)与えられた自然語原文（あるいは一般に自然語文の系列）$\bar{s}$に対し，$\bar{s}$において省略されている可能性のある名詞句の集合E[$\bar{s}$]を定義する．例えば，$\bar{s}$の中に現れ，かつSN_{abr}に属する非終端記号から生成される名詞句の集合をE[$\bar{s}$]と定義する．

(3)4.1での各変換則を次のように変更する．すなわち，自然語文$\bar{s}$，擬似自然語文s，及び変数xに対し，β∈E[$\bar{s}$]ならば，s中の全てのxに対して（あるいは，2番目以降のxに対して），xにβを代入せず単にxをsから消去する．（xの直前に前置詞がある場合には，それも同時に消去する）

例えば，変換則[T9]において，条件部に，a αがE[$\bar{s}$]に属することを表わす条件を追加し，≈の左辺に現れる
s{{x/a α}}の意味を上記(3)のように変更する
(s{{x/a α}}の意味を定義する公理を変更する)．

〔例〕$\bar{s}$＝an input file is given and each record is to be updated とする．
s_1＝y is given and each record of y is to be updatedは
s_1' ＝y is given ∧
∀x[x∈record of y ∧ x is to be updated]
に対応する．　([T9]の[2]等より)
上の変換則により，（“of y”が消去され）$\bar{s}$が
∃y[y∈input file ∧ s_1']　に対応する．□

命令文については，主語の省略が起こる場合の変換として定義する．すなわち，命令文の主語となり得るような名詞句（例えば，the reader, the program under consideration）を生成する非終端記号NP_{imp}が指定されているとする．変換則において，式sの変数xに句αを代入する際，α∈E(NP_{imp})ならば，sの主部に位置するxにはαを代入せず，

表3　自然語による仕様の例（文献(3)より）

[J1] An input file of card images is to be analysed. [J2] There are three card types, T1, T2 and T3, with the values 1, 2 and 3 respectively, in position 1 of the card. [J3] The required analysis is as follows:
[J4] Count the cards preceding the first T1 (count A);
[J5] Display the first T1;
[J6] Display the last card, which is always the first T2 following the first T1;
[J7] Count the batches following the first T1, where a batch is either an uninterrupted succession of one or more T1 cards or an uninterrupted succession of one or more T3 cards (count B);
[J8] Count the T1 cards after the first T1 card (count C);
[J9] Count the batches following the first T1 card which consist of T3 cards (count D);
[J10] All counts are to be displayed following the display of the last card. [J11] The file is known to be in correct format; that is, there is at least one T1 card, the last card is a T2, and no T2 intervenes between the first T1 and the last card.

表4　変換の対象となる仕様

[N1] An input file of card images is to be analysed.
[N2] There are three card types, T1, T2 and T3.
[N3] A T1 card is a card with 1 in position 1.
[N4] A T2 card is a card with 2 in position 1.
[N5] A T3 card is a card with 3 in position 1.
[N6] Count the cards preceding the first T1.
[N7] Display the first T1.
[N8] Display the last card.
[N9] The last card is always the first T2 following the first T1.
[N10] Count the batches following the first T1.
[N11] A batch is either an uninterrupted succession of one or more T1 cards or an uninterrupted succession of one or more T3 cards.
[N12] Count the T1 cards after the first T1 card.
[N13] Count the batches following the first T1 card which consist of T3 cards.
[N14] All counts are to be displayed following the display of the last card.
[N15] There is at least one T1 card, the last card is a T2, and no T2 intervenes between the first T1 and the last card.

xをsから消去し，sの動詞を原形に変える（sが助動詞をもつ場合はそれも消去する）．

5. 論理式への変換と語句の定義の例

ここでは，例として，文献(3)の例題の一つを取り上げ，実際に変換則を適用した結果得られる論理式と，仕様の記述に用いられる語句の定義を示す．また，これらの公理（論理式）から，簡単な性質をいくつか導いてみる．

表3に，例題原文を示す．ここでは，表3を次の(1)～(3)のように変更した仕様（表4）を，変換の対象とする．

(1) 原文[J2]を，[N2]～[N5]の4文に分離した．これは，現行の変換則では，副詞 'respectively' の係り受け（T1と1，T2と2，T3と3）の機能が定義されていないからである．

(2) [J3]の内容は，[J4]～[J9]で記述されているので省略した．同様に，[J12]の前半部も省略した．

(3) [J6]を[N8]と[N9]の2文に分離した．[J6]での関係代名詞 which は，主文とは独立に，先行詞に関する条件を述べる関係節を形成しており，現行の変換則では，ここでの用法に対応する意味定義を行っていないからである．同様に，[J7]も[N10]と[N11]に分離した．

変換の結果得られる論理式を表5に示す．ただし，文[Ni]に対応する論理式にはラベル[Li]を付けている．[N1]に対応する論理式において，変数Inputの有効範囲は，[L1]～[L15]の論理式全てにわたっているので，Inputに対する全称記号 '∀' による束縛は省略している．[N6]以下の命令文は，4.2で述べた省略に関する変換を行い，主語に対応する定数を the program としている．また，the first T1, the last card, There is at least one T1 card には，in Input が省略されているとして，変換を行った．

次に，本例題の記述に用いている語句の定義を，表6に示す．新たに導入する語句には下線を引いている．また，見易さのため，論理式

$\forall x1[x1 \in d1 \wedge \forall x2[x2 \in d2 \wedge \cdots \forall xn[xn \in dn \wedge [\alpha]\cdots]]]$

を，$\forall x1 \in d1, \forall x2 \in d2, \cdots, \forall xn \in dn \ \alpha$ と略記し，また，論理式 $\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$ を，

(1) α_1 (2) α_2 $\cdots$ (n) α_n と略記している．

"∈" は，データタイプに関する基本的な演算で，x∈d は，xがデータタイプdの式であることを表わす([D1]～[D3])．"is" については，合同関係を表わす場合([D6])，特定のデータタイプに関して，元の存在を表わす場合([SQ12])等，非論理的要素として意味が定義されている．なお，表5の[L14]に関しては，(1) 分詞構文の意味上の主語が主文全体になっており，また，(2) follow はdisplayに関する時間の前後関係を述べている．この種の構文，語句の意味定義に関しては現在詳細を検討中である．

論理式[L6]と表6の[PR3]より，次が成り立つ．

the number of the cards
preceding the first T1 in Input ∈
output of the program ⋯(*)

語句 the number of は，集合に関する演算として，[ST3]で定義している．[ST2]は，データタイプ d の部分クラス d1 に対し，the d1 は，d1 で表される条件を満たす全ての d の元からなる集合を表すと定義しているが，これより，

x∈the cards preceding the first T1 in Input <=>
x∈card preceding the first T1 in Input

が成り立つ．さらに，[L6]と，表6の[SQ7]より，

x∈card preceding the first T1 in Input <=>
x∈card ∧ [the position of x <
the position of the first T1 in Input]

が成り立つ．ここで，語句 the first・in・は，[SQ6]で定義されている．このようにして，上の(*)の意味が，データタイプに関する基本的演算を用いて定義されることがわかる．

6. 処理系の概要

自然語による仕様の構文解析と論理式への変換を行い，また，語句の意味定義の蓄積を支援するシステムを開発中である．システムの全容については文献(5)を参照されたい．ここでは，構文解析部と論理式への変換部の概要を述べる．

6.1 構文解析部

3. で示した構文規則をDCGのルールに手作業で変換した後，BUPトランスレータ[4]（DCGで記述されたルールを縦型上昇構文解析プログラムに変換するプログラム）を用いて構文解析プログラムを作成した．

構文規則のカテゴリ名及び引数に，DCGの述語記号，引数をそれぞれ対応させると，制約スキーマが引数の簡単な制限という形で実現でき，DCGへの変換はたやすく行える．また，BUPトランスレータを用いているので，DCGから構文解析プログラムへの変換が自動的に行える上，得られたプログラムが左再帰的なルールがあるために停止しないという，DCGを直

表5　表4より変換によって得られる論理式

[L1] Input∈input file of card images∧
Input is to be analysed
[L2] There are three card types ∧
∃x[x∈card type ∧ T1 is x] ∧
∃x[x∈card type ∧ T2 is x] ∧
∃x[x∈card type ∧ T3 is x]
[L3] ∀x[x∈T1 <=>
∃y[y∈card with 1 in position 1 ∧ x is y]]
[L4] ∀x[x∈T2 <=>
∃y[y∈card with 2 in position 1 ∧ x is y]]
[L5] ∀x[x∈T3 <=>
∃y[y∈card with 3 in position 1 ∧ x is y]]
[L6] the program should count the cards preceding the first T1 in Input ∧
∀x[x∈card preceding the first T1 in Input<=>
x∈card ∧ x precedes the first T1 in Input]
[L7] the program should display the first T1 in Input
[L8] the program should display the last card in Input
[L9] The last card in Input is always the first T2 following the first T1 in Input ∧
∀x[x∈T2 following the first T1 in Input<=>
x∈T2 ∧ x follows the first T1 in Input]
[L10] the program should count the batches following the first T1 in Input ∧
∀x[x∈batch following the first T1 in Input <=>
x∈batch ∧ x follows the first T1 in Input]
[L11] ∀x[x∈batch <=>
∃y[y∈uninterrupted succession of one or more T1 cards ∧ x is y] ∨
∃y[y∈uninterrupted succession of one or more T3 cards ∧ x is y]]
[L12] the program should count the T1 cards after the first T1 card in Input
[L13] the program should count the batches following the first T1 card in Input which consist of T3 cards ∧
∀x[x∈batch following the first T1 card in Input which consist of T3 cards <=>
x∈batch following the first T1 card in Input∧ x consists of T3 cards] ∧
∀x[x∈batch following the first T1 card in Input<=>
x∈batch ∧ x follows the first T1 card in Input]
[L14] ∀x[x∈count ⊃
x is to be displayed following the display of the last card]
[L15] ∃x[x∈T1 card ∧ there is x in Input] ∧
∃x[x∈T2 ∧ the last card in Input is x] ∧
¬∃x[x∈T2 ∧ x intervenes between the first T1 in Input and the last card in Input]

表6　表5で用いられる語句の定義

(データタイプに関する基本的定義)
[D1] ∀d∈data type
(1) class of d∈data type (2) d∈class of d
[D2] ∀d∈data type, ∀d1∈class of d [d1∈data type]
[D3] ∀d∈data type, ∀d1, d2, d3∈class of d, ∀e∈d
(1) d1 is a subclass of d2∈bool
(2) d1 is a subclass of d (3) d1 is a subclass of d1
(4) d3 is a subclass of d2 ∧ d2 is a subclass of d1 ⊃ d3 is a subclass of d1
(5) (d2 is a subclass of d1 ∧ e∈d2) ⊃ e∈d1
[D4] ∀d1∈data type [d1 type∈data type]
[D5] ∀d1∈data type, ∀d2∈d1 type
d2∈class of d1 ∧ [d2 d1 == d2]
[D6] ∀d∈data type, ∀x, y∈d [x is y <=> x == y]

(集合に関する定義)
[ST1] ∀d∈data type, ∀e∈d
(1) set of d∈data type
(2) ∀st∈set of d [e∈st∈bool] (3) { }∈set of d
(4) ∀st∈set of d [st∪e∈set of d]
[ST2] ∀d∈data type, ∀d1∈class of d, ∀e∈d
(1) (all) the d1∈set of d
(2) ∀e∈d[e∈the d1 <=> e∈d1]
[ST3] ∀d∈data type, ∀st∈set of d, ∀e∈d
(1) the number of { } == 0
(2) the number of st∪e ==
if e∈st then the number of st
else (the number of st) + 1
(系列に関する定義)
[SQ1] ∀d∈data type [sequence of d∈data type]
[SQ2] ∀d∈data type, ∀sq∈sequence of d
(1) the length of sq∈positive integer
(2) ∀p∈positive integer less than the length of s
p th d of sq∈d
[SQ3] ∀d∈data type, ∀sq∈sequence of d
subsequence of sq∈class of sequence of d
[SQ4] ∀d∈data type, ∀sq∈sequence of d,
∀subsq∈subsequence of sq
(1) subseq∈sequence of d
(2) the starting position of subseq∈positive integer
(3) the terminating position of subseq∈positive integer
(4) 1≦the starting position of subseq≦the terminating position of subseq≦the length of sq
(5) ∀i∈positive integer, ∀j∈positive integer
1≦i≦j≦the length of sq ⊃
(5.1) the subsequence of sq from i to j∈ subsequnece of sq
(5.2) the length of the subsequence of sq from i to j == j - i + 1
(5.3) ∀k∈positive integer
not greater than the length of sq[i, j]
the k th d of sq[i, j] == the (i+k-1) th d of sq
where sq[i, j] := the subsequence of sq from i to j
[SQ5] ∀d∈data type, ∀csq∈class of sequence of d,
∀p∈positive integer, ∀e∈d
(1) csq with e in position p∈class of csq
(2) ∀sq∈csq
sq∈csq with d in position p <=>
the p th d of sq = d
[SQ6] ∀d∈data type, ∀sq∈sequence of d
(1) occurrence of sq∈class of d
(2) ∀o∈occurrence of sq
the position of o∈positive integer not greater than the length of sq
(3) ∀d1∈class of d
(3.1) ∀k∈positive integer
the k th d1 in sq∈occurrence of sq
(3.2) ∀o∈occurrence of sq
[o∈d1 ∧ the number of the d1 preceding o = k-1] ⊃ the k th d1 in sq == o
(3.3) the first d1 in sq == the 1 th d1 in sq
(3.4) the last d1 in sq == the (the number of the d1 in sq) th d1 in sq
[SQ7] ∀d∈data type, ∀sq∈sequence of d,
∀o1, o2∈occurrence of sq, ∀subsq∈subsequence of sq
(1) o1 follows o2 <=>
the position of o1 > the position of o2
(2) o1 precedes o2 <=>
the position of o1 < the position of o2
(3) subsq follows o2 <=>
the starting position of subsq > the position of o1
(4) subsq precedes o2 <=>
the terminating position of subsq < the position of o2

```
[SQ8] ∀d∈data type, ∀sq∈sequence of d,
      ∀subsq∈subsequence of sq, ∀d1∈class of d
  subsq∈uninterrupted succession of d1 <=>
  (1) ∀p∈positive integer
                  not greater than the length of subsq
        the p th d of subsq∈d1
  (2) (the starting position of subseq = 1) ∨
      ¬ sq[the starting position of subsq - 1]∈d1
  (3) (the terminating position of subseq =
           the length of sq) ∨
      ¬ sq[the terminating position of subsq + 1]∈d1
[SQ9] ∀d∈data type, ∀sq∈sequence of d,
      ∀o1∈occurrence of sq, ∀d1∈class of d
        o1∈d1 after o2 <=>
            o1∈d1 ∧ o1 follows o2
[SQ10] ∀d∈data type, ∀sq∈sequence of d, ∀d1∈class of d
     sq∈consists of d1 <=>
  ∀p∈positive integer not greater than the length of sq
       the p th d of sq∈d1
[SQ11] ∀d∈data type, ∀sq∈sequence of d,
       ∀o1, o2∈occurrence of sq, ∀d1∈class of d,
       ∀e∈d
  e intervenes between o1 and o2 <=>
    ∃p∈positive integer
      [the position of o1≦p≦the position of o2 ∧
       the p th d of sq = e]
[SQ12] ∀d∈data type, ∀sq∈sequence of d, ∀e∈d
  there is e in sq <=>
    ∃p∈positive integer
                  not greater than the length of sq
       the p th d of sq = e
(ファイルに関する定義)
[FL1] ∀d1∈data type
  (1) file of d1 image∈class of sequence of d1
  (2) input file of d1∈class of file of d1
(カードに関する定義)
[CD1] card∈class of sequence of character
[CD2] ∀c∈card[the length of c = CARDLEN]
(プログラムに関する定義)
[PR1] program∈data type  [PR2] the program∈program
[PR2] ∀pr∈program  (1) input of pr∈data type
                   (2) output of pr∈data type
[PR3] ∀data type∈d, ∀st∈set of d, ∀pr∈program
  pr should count st <=>
    the number of st∈output of pr
```

接PROLOGの処理系で処理したときの問題が解決される．

この方法で得られたプログラムは効率の点で問題があるが，システムが試作段階であること，及び，システムの性格上，変更，拡張が容易でなければならない事などから，その問題は重要視していない．

また，構文解析時に複数の構文木が得られる場合は，利用者に選択してもらうという方法をとる。

現在，システムはVAX 750 UNIX のCPROLOG VERSION 1．5上で稼働している．プログラムサイズは，DCGレベルで約300行である．工数は約１人月で，作業の大半は，構文規則の修正及び拡張についやされた．また，BUPトランスレータは(財)新世代コンピュータ技術開発機構の松本氏から提供を受けたものを，CPROLOG用に移植して用いた．

6.2 論理式への変換部

ここでは，与えられた自然語文$s \in L_{NS}$の構文木を，sに対応する論理式，すなわち，$s \approx s'$を満たす論理式s'に変換する．実際には，４．での定義に従い，与えられた文sに対し，(1) まず，sの名詞句をすべて変数に置き換えて得られる素擬似自然語文s_0を作り，(2) s_0の変数に対して素擬似名詞句の代入を行ってsを得る過程に対応する論理式の変形操作をs_0にほどこす．ただし，(イ) s中のどの名詞句に同一の変数を割り当てるか，及び，(ロ) 各変数の依存関係をどう決めるか，すなわち，各変数を束縛する量記号を，論理式中のどこに付けるかは，４．の変換則の定義に従って決定する．すなわち，(イ) については，次の(a)，(b)のいずれも満たす名詞句同志のみが，同一変数を割り当てられる可能性があるとする．

(a) 名詞句の非終端記号の引数のうち，PLU(数)及びPER(人称)の値が一致している．

(b) every record of a file と the record における'record'のように，'核'となる名詞(４．参照)が等しい．

また，(ロ) についても，条件

(a) 述語ok_sbst_for_in_(５．参照) を常に真にするような代入の順をとらねばならない．

(b) 名詞句αの部分として現れる名詞句に割り当てられた変数を束縛する量記号は，αのそれよりも外側に現れる．

を考慮する．なお，$s \approx s'$を満たすs'が複数存在するという意味であいまいさが残る場合は，ユーザに指定してもらう．

現在，言語Cを用いてプログラムを作成中である．

７．あ と が き

より実用的な仕様として，６．で述べたシステムを用いて，ＯＳＩの自然語仕様を解析する予定である．

謝辞 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する．また，プログラム作成に御協力いただいた，本学学生の大野泰樹氏，松村享氏に深謝する．

文　献

(1) D. R. Dowty, R. E. Wall and S. Peters: "Introduction to Montague Semantics", D. Reidel Publishing Company (1981).
(2) G. Gazdar, E. Klein, G. Pullum and I. Sag: "Generalized Phrase Structure Grammar", Basil Blackwell (1985).
(3) M. A. Jackson: "Principles of Program Design", pp. 49-50, Academic Press (1975).
(4) Y. Matsumoto, H. Tanaka, H. Hirakawa, H. Miyoshi and H. Yasukawa: "BUP: A Bottom-Up Parser Embedded in Prolog", New Generation Computing, Vol. 1, No. 2, pp. 145-158 (1983).
(5) 石木，並河，関，杉山，藤井，鳥居，中小路: "プログラム仕様に用いる自然語の処理システム　一代数的仕様への変換と知識処理一"，ソフトウェア工学研報，SW-52-6 (1987-2)．
(6) 嵩，谷口，杉山，関: "代数的言語ＡＳＬ／＊　一意味定義を中心に一"，信学論(D)，Vol. J69-D, No. 7, pp. 1066-1074 (1986-7)．
(7) 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について"，信学技報，COMP86-2 (1986-5)．
(8) 並河，関，藤井，嵩: "プログラム仕様記述に用いる自然語の形式的意味定義"，ＬＡシンポジウム (1986-7)．