Vol. 29 No. 4 情報処理学会論文誌 Apr. 1988

要求仕様における日本語表現と形式表現間の相互変換†

西田 富士夫†† 高松 忍†† 谷 忠明††

最近，制限付き自然言語で書かれた要求仕様を論理的にチェックしたり，これを詳細化してプログラムを自動的に生産したり，これらのプログラムを能率的に更新するソフトウェア生産や保守の自動化が盛んに研究されている．この論文は，要求仕様に用いられる簡単な日本語表現の文法的パターンを与えるとともに，この表現で書いたユーザのプログラム仕様を構文解析により形式表現に変換する能率的な方法を提案している．日本語で書いたユーザの仕様の構文解析は，対象分野のフレームの知識を用いて組織的に行い，内部表現を構成する．それから，この結果を後につづく詳細化処理の能率化のために形式表現に変換する．また，ライブラリモジュールを援用して初めのプログラム仕様を半自動的に詳細化して得られる形式表現を読みやすくするために，形式表現から日本語表現へ変換する方法を上述の変換の逆変換として提案している．

1. まえがき

ここ数年来，プログラム生産要求件数の激増とその大型化に伴い，要求仕様技術の確立が急がれている[1]．要求仕様はわかりやすく，フレキシブルに表現しやすいことが望ましい．この立場からは，各専門分野で日常用いている自然言語や図式などを用いた表現が適している[2]．他方，仕様の内部矛盾を機械が直接チェックしたり，ライブラリモジュールなどを用いて仕様を詳細化して半自動的にプログラムを生成するためには，仕様表現を文法構造が明確で機械処理がしやすい形式表現に変換することが必要である．このような観点から，日本語仕様などを形式仕様などに変換し，プログラムを生成する研究がいくつか行われているが[4),5),10]，初期の段階にあり，対象とする日本語仕様表現に関する構文上の制限，すなわち文法や，これを用いて日本語仕様を構文解析しその内部表現から組織的に形式仕様を構成する方法などを明らかに示していない．

仕様の具体的内容は分野により多様である．しかし，その基本的内容とその構文は分野とほぼ無関係に，手続きや手順を表す手続き文と，変数の定義や入出力関係を表す性質関係文からなるものと考えられる．本稿では，これらの文を生成する基本的な構文規則をあげて，日本語仕様の解析や，詳細化出力に対応する日本語文を生成する一つの方法を示している．また，仕様には表や数式などを用いることが多いが，本稿では読みやすさと機械処理に便利なように，前者は仮名漢字を用いたレコード表現，後者は，パスカル風の中置形式表現を採用し，これに制限している．また，日本語仕様の形式仕様への変換は，内部表現を手続き表現，制御表現，問題記述表現などのパターンに分類し，各パターンごとに設けた基本的な変換規則により日本語仕様と形式仕様との間で組織的に相互に変換する方法を述べている．日本語表現の仕様を構成する各語は情報処理などの専門分野を対象としているためユニークなものであり，このため構文解析や形式表現への変換はこれらの分野のフレーム構造を用いることにより能率的に行うことができる．また表やレコードなどの構造をもつデータのインフォーマルな仕様表現についても，従来の読みやすさを保存しながら機械可読の形でタイプ表現などの形式表現に変換する表現法を述べている．また，ライブラリモジュールなどを援用してユーザが与えた仕様を半自動的に詳細化して得られる形式表現を，読みやすい日本語表現に組織的に変換する方法を与えている．

2. 日本語仕様表現の構文

要求定義や仕様の記述形式は現在のところ標準的なものは見あたらないが，要求仕様の内容は基本的には問題の記述と手続きの記述よりなると考えられる．問題の記述は，与えられた性質や構造をもつ入力データや入力条件に対して，出力データの満たすべき条件を規定するもので，処理の前後におけるデータ間の入出力関係などを通して求めるべき処理システムを簡潔に定義する．これに対し手続きの記述は，マクロな手続き表現や制御表現を用いて，入力データから所要の条件を満たす出力データを生成する概略の手続きを表す．いま，これらの手続き表現や制御表現を，ライブ

† Transformation between Japanese Expressions and Formal Expressions in Program Specifications by FUJIO NISHIDA, SHINOBU TAKAMATSU and TADAAKI TANI (Department of Electrical Engineering, College of Engineering, University of Osaka Prefecture).

†† 大阪府立大学工学部電気工学科

ラリモジュールなどに登録されたいくつかの基本的な手続きの具象的な例として分けることができるものとする．このとき，これらの部分手続きに対する問題的記述をつなぎ合わせて，ユーザが与えた要求仕様のマクロな手続きの記述が問題の記述に対する解になっているかどうかを機械的にチェックすることなどができる．実際上の要求仕様では問題的記述または手続き的記述の中，いずれか一方のみが，全体または部分ごとに与えられることも多いが，ともかく問題の記述や手続きの記述が要求仕様の基本である．

さて，人間が機械に与える仕様表現や，機械から出力されるコメントなどは，まぎれが少なく機械処理が容易であるとともに，書きやすく見やすい表現であることが望ましい．この章では簡単で標準的な日本語表現を用いて問題や手続きなどからなる仕様の内容を表す方法について述べる．

表1は，このような日本語仕様の記述に必要な一つの基本的な構文規則を正規表現を併用してニーモニックな形で示したものである．表に示すように，問題記述の部分は入出力データの性質や入出力データ間の関係を規定する述語表現が主なものであり，〈システム構成文〉，〈GIVEN 文〉，〈型定義文〉，〈性質関係文〉などからなる．手続きの記述はマクロなプロセジュアや制御の表現や代入文などの表現が主なものであり，〈手続き文〉，〈条件分岐文〉，〈繰り返し文〉，〈代入文〉などからなる．

表1において，角括弧〈，〉で囲んだ記号は非終端記号を表す．〈$A|..|B$〉や($a|..|b$)は複数個の記号や記号列の中の一つを表す．sをストリングや記号，cをコンマのような記号とするとき，$s*$はsの1回以上の繰り返しを表し，($s\ c$)*- は ($s\ c$)*s，すなわち s の記号列の間に c を内挿した記号列 $scsc..scs$ を表すものとする．nl は復改記号を表し，また，“.”は文の終止記号を表すとともに復改をも指示するものとし，次の行の文字の先頭位置を，現在の行と同じ文字の先頭位置に設定するものとする．indent 記号は一定字数だけ字下げを行い，ret は文字の開始位置を，最も新しく indent を行った直前の行の文字の開始位置まで復帰することを指示する．また，[,] で囲まれた部分はその内容が省略可能なことを示す．

ここに(2)(5)(13)(14)は複数個の文からなる一つのテキストを記述している．また，(12)の2番目の構文規則は手続き操作とその結果の置数を一つにまとめた文を表している．〈性質関係文の仮定形〉などは文の終止形や終止符の代わりに，対応する活用形を代置したり助詞などを補ったものである．また，(9)は例えば“A, Bを実数型，Cを文字型とする．”というような文を表す．

表1 日本語仕様の構文規則
Table 1 Syntactic rules of Japanese specifications.

(1) 〈要求仕様〉::=〈問題記述部〉|〈手続き記述部〉
(2) 〈問題記述部〉::=〈問題入力文〉*
このとき nl
〈問題出力文〉*
(3) 〈問題入力文〉::=〈システム構成文〉|〈GIVEN 文〉
|〈性質関係文〉|〈型定義文〉
(4) 〈問題出力文〉::=〈OBTAINED 文〉|〈性質関係文〉
(5) 〈手続き記述部〉::=(〈型定義文〉
|〈手続き記述文〉)*
(6) 〈手続き記述文〉::=〈手続き文〉|〈条件分岐文〉
|〈繰り返し文〉|〈代入文〉
(7) 〈システム構成文〉::=〈対象名〉は(〈対象名〉,)*-
よりなる.
(8) 〈GIVEN 文〉::=(〈ターム名〉,)*- が与えられる.
|(((〈ターム名〉が〈変数名〉に),)*-
与えられる.
(9) 〈型定義文〉::=(((〈ターム名〉,)*- を〈データ型名〉),)*-
とする.
(10) 〈性質関係文〉::=(〈ターム名〉〈助詞〉)*
〈性質関係述語〉
|〈性質関係文の仮定形〉
〈性質関係文〉|〈算術論理式〉
(11) 〈OBTAINED 文〉::=〈ターム名〉が[〈変数名〉に]求められる.
(12) 〈手続き文〉::=[〈性質関係文の連体形〉]〈変数名〉を
[〈変数名〉から][〈変数名〉に]
[〈手続き名〉により][〈フォーマット〉で]
〈手続き述語の終止形
|手続き述語の命令形〉
|〈手続き文の連用形〉，その結果を
〈変数名〉に入れる.
(13) 〈条件分岐文〉::=〈性質関係文の仮定形〉 nl
indent〈手続き記述文〉* ret
[そうでなければ nl
indent〈手続き記述文〉* ret]
(14) 〈繰り返し文〉::=〈性質関係文の連体形〉間 nl
indent〈手続き記述文〉* ret
|〈変数名〉が〈変数名|定数〉から
〈変数名|定数〉まで
[〈変数名|定数〉ずつ]変わるとき
nl indent〈手続き記述文〉* ret
(15) 〈ターム名〉::=〈定数名〉|〈変数名〉|〈関数名〉
|[〈レコード変数〉の]〈レコード属性名〉
(16) 〈データ型名〉::=〈データ値域型名〉
|[〈データ値域型名〉の]
〈データ構造型名〉
(17) 〈変数名〉::=[〈データ型名|レコード属性名〉[の]]
〈変数記号名〉
(18) 〈関数名〉::=[(〈定数|変数名|関数名〉〈助詞〉)*]
〈関数記号名〉

3. フレーム構造と構文解析

3.1 フレーム構造

自然言語の構文解析は格フレームを用いて行われることが多い．格フレームを用いた構文解析は，述語詞に対する文の構成要素の役割と，述語詞の表す状態や動作に対する実世界の構成要素の役割との対応をあらかじめ与えておき，これに基づいて解析を行うものである．すなわち，文の意味を実世界の要素間の基本的な関係知識に基づいて同定するもので，係受けなど文の意味を確定する上で有用である．特にプログラムの要求仕様の場合には，記述の対象が限定されており，対象と名前とは一対一に対応するのみでなく，手続きや状態を表す動詞語の動作や関係を詳しく規定する構成要素の語のカテゴリとその役割とは必須格などとしてほぼ決まっており，格フレームの利用は有用である[6)-8)]．

表2に要求仕様に現れる二，三の動詞語の格フレームを示す．

表2において，第1行目は動詞の格フレームにおける構成要素の表層（文法）上の役割名を代表的な後置詞を用いて示している．2行目以下は各動詞語について，それぞれの格に入る語の代表的な意味カテゴリ名と格ラベル（意味上の役割名）の組を示す．表よりわかるように，格に入る語のカテゴリは情報処理などに関する専門分野のものであり，手続き動作を確定する上に必要な格の個数も一般の日本語文の表現に用いられるものより多い．このため一般の日本語文に比べ要求仕様文の構文解析における係受けの同定を，より確定的に行うことができる．

例えば次のような動詞の連体修飾を含む文について考えよう．

カードファイルから

読んだ N 人の科目 A の成績 $A(1..N)$...

この文において“読む”が修飾する語句として，破線および実線の矢印で示すように，“N 人”，“科目 A”，“成績 A $(1..N)$”が文法上考えられる．これらの語句を単語辞書と構文規則を用いて解析し意味カテゴリを求めると，“成績 A $(1..N)$”のカテゴリだけが表2の“読む”の格フレームにおける *OBJect* 格のカテゴリ“ファイル名”に属することから，実線の矢印で示すように係受けを確定できる．

表2 格フレーム表
Table 2 A table of case frames.

	から格	を 格	に 格	で 格
読む	入力装置名 *SOurce*	ファイル名 *OBJect*	変 数 名 *GOal*	フォーマット *MODE*
書く		変 数 名 *OBJect*	出力装置名 *GOal*	フォーマット *MODE*
検索する	ファイル名 *SOurce*	レコード名 *OBJect*	レコード名 *GOal*	

これは次のような名詞句においても同様である．

$t_1 j_1 t_2 j_2 \cdots j_{n-1} t_n$

$j_1, j_2, \cdots, j_{n-1}$ は所有格などの助詞であることが多く，動詞語の“もつ”や“である”を，これらの助詞などで置き換えたものと考えられる．この場合にもターム $t_1, t_2, \cdots, t_n$ に関連する専門分野において，対象と属性，属性値間の関係などの知識を利用することにより，係受けを確定できることが多い．

例1

“在庫レコードというレコードが属性として品名と在庫量をもち，品名と在庫量の属性値がそれぞれ文字列と整数で与えられる”というレコードの構造を，

RECORD (*OBJ*：在庫レコード，品名：文字列，
在庫量：整数)

のような形で表そう（4章式(13)参照）．このような知識を用いることにより

“品名 ABC の在庫レコードの在庫量”

なる名詞句は“品名 ABC”が属性 *-OBJ* の関係で“在庫レコード”に係り，これらが *OBJ-* 属性の関係で“在庫レコード”に係ってこの名詞句の内部表現（4章式(2)参照）は

在庫量 (*ATTR*：＊，
OBJ：在庫レコード (*OBJ*：＊，
ATTR：品名 (*OBJ*：＊, *VAL*：ABC)))

と構成される．ただし＊はこれを囲む括弧対の左隣の語を表し，この語を限定修飾する括弧対で囲まれたフレーム内でのこの語の役割を明らかにしている．

3.2 構 文 解 析

構文解析は3.1節で述べた動詞の格フレームと**表3**のような引数部をもつ構文規則を用いて行う．

表3において，引数部は左の非終端記号で表される表層表現部分列の内部表現を表す．K：t は格ラベルとその格に入るタームの対を表し，$\boldsymbol{K}$：$\boldsymbol{t}$ はこれらの対のいくつかの列を表す．表の(1)は条件分岐文のような2つ以上の述語詞の語を含む文の構文を，(2)は述語詞の語を1つだけ含む文の構文を表す．

表 3 引数部をもつ構文規則
Table 3 Syntactic rules with arguments.

(1) 〈文 (*PRED*: *t*0, ***K*1**: *t*1, ***K*2**: (*PRED*: *t*02, ***K***: *t*))〉
::= 〈述語節 (*PRED*: *t*02, ***K***: *t*)〉〈後置詞 (*K*2)〉
〈文 (*PRED*: *t*0, ***K*1**: *t*1)〉

(2) 〈文 (*PRED*: *t*0, ***K***: *t*)〉::= 〈述語節 (*PRED*: *t*0, ***K***: *t*)〉
〈終止記号〉

(3) 〈述語節 (*PRED*: *t*0, ***K*1**: *t*1, *K*2: *t*2)〉
::= 〈名詞句 (*t*2)〉〈後置詞 (*K*2)〉
〈述語節 (*PRED*: *t*0, ***K*1**: *t*1)〉

(4) 〈述語節 (*PRED*: *t*0, *K*1: *t*1)〉
::= 〈名詞句 (*t*1)〉〈後置詞 (*K*1)〉〈動詞 (*t*0)〉

(5) 〈名詞句 (*t*1 (*PRED*: *t*0, *K*1: ∗, ***K*2**: *t*2)〉
::= 〈述語節の連体形 (*PRED*: *t*0, *K*1: ∗, ***K*2**: *t*2)〉
〈名詞句 (*t*1)〉

(6) 〈名詞句 (*t*1 (*K*1: ∗, ***K*2**: *t*2, *K*3: *t*3)〉
::= 〈名詞句 (*t*3)〉〈後置詞 (*K*3)〉
〈名詞句 (*t*1 (*K*1: ∗, ***K*2**: *t*2)〉

(7) 〈名詞句 (*t*1)〉::= 〈名詞 (*t*1)〉

構文解析システムは，日本語仕様文を左から走査し，表2の動詞の格フレーム表や構文規則を参照して，上向きに並列に構文解析を行う[8]．動詞語が現れると，表3の(3)，(4)によりその左側の部分還元列の中からこれに係る語を求め，これらの語のカテゴリと後置詞が格フレームと整合するように語の格ラベルを求める．構文規則の右辺に対応する記号列と格ラベルとをまとめて左辺の非終端記号に還元しその内部表現を構成する．このため使用する語の文法カテゴリ，意味カテゴリや動詞語の格フレームなどを語ごとに単語辞書に記載しておく．なお，表1(9)のような省略形の文に対しては“とする”のような述語を必要な個数だけ補って内部表現を構成する．

また，代入文や算術論理式は，ユーザが与える日本語仕様や，詳細化出力を自動変換してつくる日本語表現では，読みやすさのためパスカル風の中置形で表すが，形式表現では中置形を前置形に変換し，リスプ言語などの任意の言語への変換が一般的に行えるようにしている．ただしこれらの中置形と前置形との相互変換は，原理的には公知のものと同じであるので，ここでは省略する．

4. 内部表現から形式表現への変換

日本語仕様を構文解析して得られる内部表現（の命題部）は，手続き文や入出力述語文のような述語詞を中心とする文では，格文法による一般の文の構文解析の慣用に従って

$$(PRED: t, K_1: t_1, \ldots, K_n: t_n) \qquad (1)$$

のような形に設定している．また，関数名のような名詞句は

$$t(OBJ: *, K_1: t_1, \ldots, K_n: t_n) \qquad (2)$$

のような形の内部表現に変換する．

他方，従来，使用されているC言語やリスプなどのいろいろの高級言語の多くは，仕様やプログラム表現において手続き名や関数名などのキーワードを見出しとして前置している．したがって仕様やプログラム表現において

$$key\text{-}word\ (K_1: t_1, \ldots, K_n: t_n) \qquad (3)$$

のように，*key-word* を前置した形を個々の表現別に形式表現として定め，仕様の内部表現を形式表現に変換し，これを必要に応じて詳細化した後，指定したプログラム言語に組織的に展開するのが能率の上から望ましい．以下，内部表現から形式表現へ変換する手法について仕様の表現別に述べる．

4.1 手続き表現

この表現は

手続き名（格ラベル 1：ターム 1，…，
格ラベル *n*：ターム *n*） (4)

の形をとる．手続き名は手続きや操作の動詞語を用いることが多く，格ラベルにはそれらの処理対象である *OBJect* 格や方法を示す *MEANS* 格などがある．

手続き名はモジュール検索の見出しとして用いるためユニークで簡潔であることが望ましい[3),9)]．ところが“見出す”や“ソートする”などに対応する述語“find”や“sort”などは手続き名としては一般的すぎるため，*OBJect* などの種類によりモジュールを別々に構成することが多い．このような場合には日本語の内部表現の *OBJ* 格や *MEANS* 格の語を，形式表現では式(5)のように手続き名に組み入れて手続き名をユニークなものにする．

$$(PRED: FIND, OBJ: function\ (OBJ: x))$$
$$\rightarrow FIND\text{-}function\ (OBJ: x)$$
$$(PRED: procedure, OBJ: x, \ldots, MEANS: method)$$
$$\rightarrow procedure\text{-}BY\text{-}method\ (OBJ: x, \ldots) \qquad (5)$$

例2

(a) “x の階乗を求める”

$$(PRED: FIND, OBJ: FACTORIAL\ (OBJ: x))$$
$$\rightarrow FIND\text{-}FACTORIAL\ (OBJ: x)$$

(b) “1次元配列 a から2分割探索法により x を検索する”

$$(PRED: RETRIEVE, SO: a, OBJ: x,$$

MEANS : *BINARY-SEARCH*)
→*RETRIEVE-BY-BINARY-SEARCH*
(*SO* : *a*, *OBJ* : *x*)

また，表1 (12)の右辺後半部が表す重文の内部表現は

(*PRED* : ***procedure***, *OBJ* : *t*, ...,)
AND (*PRED* : 入れる，[*OBJ* : その結果]，
GO : *z*) (6.1)

となるが，これを

procedure (*OBJ* : *t*, ..., *GO* : *z*) (6.2)

のような一つの手続き表現の形にまとめる．

4.2 制御表現

表1の(13) (14)のような制御文は構文解析の結果次のような内部表現

(*COND* : (*OBJ* : *x*, *PRED* : *t*),
THEN : (*PRED* : ***procedure***, *OBJ* : *y*))
(*COND* : (*OBJ* : *x*, *PRED* : *t*),
THEN : (*PRED* : ***procedure*** 1, *OBJ* : *y* 1),
ELSE : (*PRED* : ***procedure*** 2, *OBJ* : *y* 2))
(*DURation* : (*OBJ-REL* : *t* (*OBJ* : *x*,
PRED : ∗),
PRED : 成立する),
PRED : ***procedure***, *OBJ* : *y*)
(*TIME* : (*PRED* : 変わる，
OBJ : *i*, *FROM* : *m*, *TO* : *n*,
[*STEP* : *k*,]),
PRED : ***procedure***, *OBJ* : *x*) (7)

に変換される．これらに対する形式表現は，制御の種類を表す記号を見出しとする手続き表現と類似の形

IF-THEN (*COND* : *t* (*OBJ* : *x*),
OBJ : ***procedure*** (*OBJ* : *y*))
IF-THEN-ELSE (*COND* : *t* (*OBJ* : *x*),
OBJ : ***procedure*** 1
(*OBJ* : *y* 1),
ELSE : ***procedure*** 2
(*OBJ* : *y* 2))
WHILE (*COND* : *t* (*OBJ* : *x*),
OBJ : ***procedure*** (*OBJ* : *y*))
FOR (*INDEX* : *i*, *FROM* : *m*, *TO* : *n*,
[*STEP* : *k*,]
OBJ : ***procedure*** (*OBJ* : *x*)) (8)

に選定する．

式(7)の各内部表現から式(8)の各形式表現に変換するにはそれぞれの内部表現がどの種類の制御表現に属するかを同定した後，対応する形式表現を格ごとに構成すればよい．

4.3 問題記述部の表現

入出力のデータの関係などを規定する問題記述部に対する内部表現は，表1の(2)から，例えば

(*PRED* : 与えられる，
OBJ : *x* (*PRED-PROP* : ***property***,
OBJ : ∗), *LOC* : *u*)
COND : このとき
(*PRED* : 求められる，*OBJ* : *q*(*x*), *LOC* : *z*)
(9)

のような形に求められる．これに対応して形式表現を入力文 *IN* と出力文 *OUT* の区別を表す記号を前置して次のような表現

IN : *GIVEN* (*OBJ* : *x*, *LOC* : *u*),
property (*OBJ* : *x*)
OUT : *GIVEN* (*OBJ* : *q* (*x*), *LOC* : *z*) (10)

に設定する．上式の表現は手続き表現と同様に述語記号を前置し，後続の推論処理が便利なようにこれらの述語記号をともに *GIVEN* というキーワードに置き換え，また *OBJect* *x* の性質などは性質や関係の述語の連言形で表したものである．式(9)から式(10)への変換は手続き文のそれと同様に述語のカテゴリや格ラベルや格に入るタームを参照して行われる．また，必要に応じて式(10)から式(11)のようなモジュールの入出力表現のクローズ形を求める．

GIVEN (*OBJ* : *q* (*x*), *LOC* : *z*) ∨
¬ *GIVEN* (*OBJ* : *x*, *LOC* : *u*) ∨
¬ ***property*** (*OBJ* : *x*) (11)

4.4 タイプ表現

'*x* を正の整数とする' というような型定義文や '…正の整数 *x*…' のような句表現の内部表現は表1の(9)や(17)からそれぞれ

(*PRED* : とする，*OBJ* : *x*,
APPOSition : 整数 (*OBJ* : ∗, *ATTR* : 正))
…*x* (*OBJ* : ∗, *APPOSition* : 整数 (*OBJ* : ∗,
ATTR : 正))…

のように表される．これらの形式表現は入出力述語表現と同じくタイプ名の述語を見出しとして前置し

INT (*OBJ* : *x*), *POS* (*OBJ* : *x*)

のように設定する．

データがレコードなどのように構造をもつ場合，一般にはインフォーマルに，その構造を表などの見やすい形で指定している．しかし見やすさを保持しながら機械可読に便利なようにするため，この論文ではその

構造を次のように表す.

レコード

　データ名：レコード名

　属性名 1：属性値，または属性値の値域名［と桁数］，またはレコード名

　……

　属性名 n：属性値，または属性値の値域名［と桁数］，またはレコード名

ファイル

　データ名：ファイル名

　成　　分：レコード

配　列

　データ名：配列名

　TYPE：値域名

　SIZE：($l1..l2, m..n$)

　ATTR：(*attr-m*, …, *attr-n*)　　(12)

上の表現ではデータの構造を表すのに名前と値をコロン：で句切っている．レコードにおいてレコード名を属性値に選ぶ場合にはいろいろなレベルのレコードが定義され，インデントなどを用いてその木構造を直接，表現することができる．しかし，構造が簡単な場合や紙面を節約したい場合には改行せず，コンマ“,”や括弧( )を用いればよい．

式(12)のような機械可読であるがインフォーマルなタイプ表現の形式表現は次のように設定する．

RECORD (*OBJ*：レコード名,
　ATTR-1：*value*-1, …,
　ATTR-n：*value-n*)

FILE (*OBJ*：ファイル名,
　COMP：*RECORD* (*OBJ*：レコード名,
　　ATTR-1：*value*-1, …,
　　ATTR-n：*value-n*))

ARRAY (*OBJ*：配列名, *TYPE*：値域名,
　SIZE：($l1..l2, m..n$),
　ATTR：(*attr-m*, . . , *attr-n*))　　(13)

インフォーマルなタイプ表現とその形式表現とは構造的に対応しており，変換を容易に行うことができる．

例3

表4に対する(12)のレコード表現は

　職員レコード

　　職員番号：整数

　　氏　　名：文字列

　　入社年月：年月

表4　職員レコード
Table 4　An employee's record.

職員番号	氏　名	入社年月	
整　数	文字列	年	月
		整　数	整　数

　　年：整数

　　月：整数

その形式表現は

RECORD (*OBJ*：職員レコード,
　職員番号：整数，氏名：文字列,
　入社年月：*RECORD* (*OBJ*：年月,
　　年：整数，月：整数))

となる．

5. 形式表現より日本語表現への変換

ユーザが与えた日本語表現による仕様は，前述のように形式仕様に変換される．形式仕様の中に含まれるnon-primitiveな表現は，ライブラリモジュールのOP部（手続き詳細記述部）を具象化したものを副プログラムとしてこの表現の部分から呼ぶか，またはこれで置き換えて（展開して）詳細化し，non-primitiveな表現がなくなるまでこれを繰り返す．しかし，このようにしてえられた詳細化の結果は，形式表現で読みにくい．そこでこの形式表現を機械により読みやすい日本語表現に変換することが考えられる．

詳細化過程で形式表現が生成されると，4章とは逆の方向に，形式表現を内部表現に変換する．内部表現から日本語表現への変換は表3の書換え規則を構文解析のときとは逆に下向きに適用して行われる．すなわち変換すべき内部表現の格構造パタンならびに前置した品詞記号から適用すべき書換え規則を選択し，日本語表現を生成する．算術代入文などは前置形から中置形に変換する．ただし日本語表現を読みやすくするために，変換の途中で次のような処理を行っている．

5.1 語　　順

述語詞語に対し，後置詞句の標準的な語順は日本語でもほぼ決まっており，述語詞語のカテゴリごとに後置詞句の語順をその格名により単語辞書に記載している．しかしある文Sでその直前の文の語を引用するような場合には，これを文Sの文頭にもってきて代名詞化するほうがわかりやすい．例えば連続する手続き表現

proc 1(..., *GOal*：*z*)　　(a)

proc (K : z, ...) (b) (14)

において，手続き表現(a)の処理結果zが次の手続き表現(b)の *OBJ* 格，*SOurce* 格や *INSTRument* 格のタームとして用いられるとする．このとき，手続き表現(b)からの日本語的表現の生成において，Kが例えば *INSTR* 格のときは，“その結果を用いて”のようにタームzを代名詞化するとともに，これを文頭へ移動する．

5.2 置き換えと注釈化

詳細化において，条件を細かく指定したり任意格に標準的なデフォルト値をいれた形式仕様からは，長くてわかりにくい日本語的表現が生成されることが多い．このようなときには，長くて複雑なタームはとりあえず代名詞などを用いて主要部を記述しておき，続いて注釈として詳細部を生成する．

例えば多くの格を含む手続き表現

proc ($K_1 : t_1, \ldots, K_m : t_m, K_{m+1} : t_{m+1}, \ldots, K_n : t_n$) (15)

を日本語的表現の一つの単文に変換しようとすると，長くてわかりにくい文になる．このような場合，$K_1, \ldots, K_m$ を必須格，$K_{m+1}, \ldots, K_n$ を非必須格とするとき，まず(15)の必須格部分

proc ($K_1 : t_1, \ldots, K_m : t_m$) (15.1)

の日本語的表現を生成し，続いてただし書きとして残りの非必須格部分の日本語的表現を次のように生成する．

‘この *proc* において
　K_{m+1} は t_{m+1} であり，
　.....
　K_n は t_n である.’ (15.2)

6. 実験例とシステム

前章までに述べた方法に基づいて日本語的仕様と形式仕様とを相互に変換するシステムを作成し，これを仕様の詳細化システム MAPS[9),12)] に組み込んだ．システムはリスプ言語で書かれ，変換システムの主な部分は，日本語的仕様から形式仕様への変換システム，形式的表現から対応する日本語的表現への変換システム，単語辞書（約500語），文法規則（約30個），変換規則などからなる．

処理時間は，日本語から形式表現への変換に，ACOS-850 のリスプインタプリタモードで，1語あたり約 0.2 秒である．また，形式仕様から約 50 行のプログラムを生成するまでに要する時間は，詳細化仕様に対応する日本語的表現の生成を含めて約 30 秒程度である．

例4に，変換システムの変換過程の一例として自然言語処理における構文解析プログラムの生成に関する実験例を示す[11)]．(a)の日本語的表現から(b)の形式仕様に変換し，これをライブラリモジュールを用いて詳細化し，得られた形式表現(c)から対応する日本語的表現(d)を生成する．ここで，(a)のタイプ表現における＊印はその右のレコードの列を示す．

例4

(a) 日本語的仕様

問題記述部：

書換え規則レコードとハンドル記号列が与えられる．

　書換え規則レコード
　　規則名：文字列
　　左　辺：構文要素レコード
　　右　辺：構文要素レコード列
　　　＊構文要素レコード
　ハンドル記号列
　　＊構文要素レコード
　構文要素レコード
　　非終端記号名：文字列
　　下位文法情報：文字列
　　モーダル情報：文字列
　　意味カテゴリ情報：文字列
　　⋮　　⋮

このとき

還元記号列が求められる．

.....

手続き記述部：

ハンドル記号列を書換え規則の右辺と単一化し，その結果を代入リスト列に入れる．

代入リスト列が NULL なら

　フェイルを返す．

書換え規則の左辺を TEMPORARY にコピーする．

TEMPORARY を代入リスト列を用いて具象化し，その結果を還元記号列に入れる．

還元記号列を返す．

(b) 形式仕様

```
IN : GIVEN (OBJ : HANDLE, REWRITING-RULE)
OUT : GIVEN (OBJ : REDUCED-SYMBOL)
TYPE : RECORD (OBJ : REWRITING-RULE,
                RULE-NAME : STRING,
                LEFT-SIDE : SYNTACTIC-COMP,
                RIGHT-SIDE : SYNTACTIC-COMP-SEQUENCE)
       SEQUENCE (OBJ : SYNTACTIC-COMP-SEQUENCE,
                  COMP : SYNTACTIC-COMP)
       SEQUENCE (OBJ : HANDLE,
                  COMP : SYNTACTIC-COMP)
       RECORD (OBJ : SYNTACTIC-COMP,
                NON-TERMINAL-SYMBOL-NAME : STRING,
                LOWER-GRAMMER : STRING,
                ..... )
PROC :
UNIFY (OBJ : HANDLE,                                          }
       PARTIC : RIGHT-SIDE                                    }
                (OBJ : REWRITING-RULE, ATTR : *),             } A
       GO : SUBSTITUTION-LIST);                               }
IF-THEN (COND : NULL (OBJ : SUBSTITUTION-LIST),
         OBJ : RETURN (OBJ : FAIL));
COPY (OBJ : LEFT-SIDE (OBJ : REWRITING-RULE,
      ATTR : *), GO : TEMPORARY);
INSTANTIATE (OBJ : TEMPORARY,
             INSTR : SUBSTITUTION-LIST,
             GO : REDUCED-SYMBOL);
RETURN (OBJ : REDUCED-SYMBOL)
```

(c) 第一段階詳細化 (Aの部分の副プログラム化)

```
PROC : UNIFY (OBJ : HANDLE, PARTIC : RIGHT-SIDE,
              GO : SUBSTITUTION-LIST)
IN : GIVEN (OBJ : HANDLE, RIGHT-SIDE)
OUT : GIVEN (OBJ : UNIFIER (OBJ : HANDLE,
                            PARTIC : RIGHT-SIDE),
             LOC : SUBSTITUTION-LIST)
TYPE : .....
OP : FIND-DISAGREEMENT-SET (OBJ : HANDLE,
                            PARTIC : RIGHT-SIDE,
                            GO : SUBST-PAIR);
     WHILE (COND : AND (NOT (UNIFIED (OBJ : SUBST-PAIR)),
                        NOT (UNUNIFIABLE
                             (OBJ : SUBST-PAIR))),
            OBJ : APPEND (OBJ : SUBST-PAIR,
                          GO : SUBSTITUTION-LIST);
            INSTANTIATE (OBJ : RIGHT-SIDE,
                         INSTR : SUBST-PAIR,
```

```
                                    GO : RIGHT-SIDE);
                        FIND-DISAGREEMENT-SET (OBJ : HANDLE,
                                               PARTIC : RIGHT-SIDE,
                                               GO : SUBST-PAIR)
                    );
        IF-THEN-ELSE (COND : UNIFIED (OBJ : SUBST-PAIR),
                      OBJ : RETURN (OBJ : SUBSTITUTION-LIST),
                      ELSE := (SUBSTITUTION-LIST, NIL);
                            RETURN (OBJ : SUBSTITUTION-LIST)
                    )
```

(d) コメント文

PROC：ハンドルを右辺と単一化し，その結果を代入リストに入れる．

IN：ハンドルと右辺が与えられる．

OUT：ハンドルと右辺の単一化代入が代入リストに求められる．

TYPE：……

OP：ハンドルと右辺の不一致集合を求め，その結果を代入対に入れる．

代入対が UNIFIED でなくかつ UNIFIABLE でない間

代入対を代入リストにアペンドする．

右辺を代入対を用いて具象化し，その結果を右辺に入れる．

ハンドルと右辺の不一致集合を求め，その結果を代入対に入れる．

代入対が UNIFIED なら

代入リストを返す．

そうでなければ

代入リスト := NIL

代入リストを返す．

6. むすび

要求仕様の具体的内容は対象分野により多様である．本論文では要求仕様の基本的，抽象的内容は問題部と手続き部とからなるものとし，これに対する簡単で標準的な一つの日本語表現と形式表現を設定し，それらの間の変換法を述べた．実験はプログラミングテキストに現れる問題のような小規模のものから，機械翻訳システム[11]のような大規模のものを対象として行ったが，ほぼ所期の結果を得ることができた．今後はさらに，見出しをもつ節やパラグラフなどの導入による仕様のテキスト的構造化や図形情報の形式表現への導入が必要と思われる．また，仕様の見やすい日本語的表現の拡充や，慣用的表現と形式表現との相互変換法の拡充は，ソフトウェアの能率的な生産や保守に極めて効果的であり重要と思われる．

参考文献

1) 大野 豊：ソフトウェア工学の背景と展望，情報処理，Vol. 28, No. 7, pp. 845-852 (1987).
2) マーチン，J. ほか著，国友ほか訳：ソフトウェア構造化技法，近代科学社，東京 (1986).
3) Horowitz, E. and Munson, J. B.: An Expansive View of Reusable Software, *Trans. IEEE*, Vol. SE-10, No. 5, pp. 477-487 (1984).
4) 杉山，秋山，亀田，牧之内：対話型自然言語プログラミングシステムの試作，電子通信学会論文誌，Vol. J 67-D, No. 3, pp. 297-304 (1984).
5) 原田，篠原：部品合成によるプログラム自動生成システム ARIES/I，情報処理学会論文誌，Vol. 27, No. 4, pp. 417-423 (1986).
6) 大西，阿草，大野：要求定義のための要求フレーム，情報処理学会論文誌，Vol. 28, No. 4, pp. 368-374 (1987).
7) Nishida, F.: English-Japanese Machine Translation System and the Application, *JARET*, Vol. 7, pp. 163-172 (1983).
8) 高松，藤田，谷，西田：日英機械翻訳における内部表現の部分的トランスファと英文の生成，情報処理学会論文誌，Vol. 26, No. 5, pp. 788-797 (1985).
9) 西田，藤田：ライブラリモジュールを用いたプログラムの半自動的詳細化，情報処理学会論文誌，Vol. 25, No. 5, pp. 785-793 (1984).
10) 西田，藤田，高松：日本語による仕様記述からのライブラリモジュール援用プログラムの半自動生成，情報処理学会，プロトタイピングと要求定義シンポジゥム論文集，pp. 111-120 (1986).
11) Nishida, F., Fujita, Y. and Takamatsu, S.: Construction of a Modular and Portable Translation System, *Proc. of COLING 86*, pp. 649-651 (1986).

12) 西田，藤田，高松：ライブラリモジュールのリンクの手法による仕様の詳細化と誤りの検出，情報処理学会論文誌，Vol. 28, No. 5, pp. 489-498 (1987).

(昭和 62 年 9 月 14 日受付)
(昭和 63 年 1 月 19 日採録)

西田富士夫（正会員）

大正 15 年生．昭和 25 年京都大学工学部電気工学科卒業．現在，大阪府立大学工学部電気工学科教授．工学博士．自然言語処理，プログラムの仕様と詳細化，問題解決システムなどの研究に従事．著書「言語情報処理」など．電子情報通信学会，電気学会，計測制御学会，IEEE 各会員．

高松　忍（正会員）

昭和 23 年生．昭和 46 年大阪府立大学工学部電気工学科卒業．昭和 48 年同大学院工学研究科修士課程修了．工学博士．現在，大阪府立大学工学部電気工学科講師．自然言語処理，知識情報処理の研究に従事．電子情報通信学会，人工知能学会，日本認知科学会各会員．

谷　忠明（正会員）

昭和 30 年生．昭和 57 年大阪市立大学文学部卒業．現在，大阪府立大学工学部電気工学科教務技師．自然言語処理の研究に従事．