ICOT Technical Memorandum: TM-0501

TM-0501

プログラム自動生成のための日本語仕様記述言語

和田　孝，土田賢省，杉山高弘，阪田全弘，
富田兼一，宮下洋一

May, 1988

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

プログラム自動生成のための日本語仕様記述言語

和田 孝　土田 賢省　杉山 高弘　阪田 全弘　富田 兼一　宮下 洋一
日本電気株式会社

現在、大規模なアプリケーションソフトの開発においては、システムエンジニアがモジュール毎に詳細設計を行なって仕様を主に自然言語を用いて記述し、それをプログラマがプログラムに翻訳するのが一般的である。我々は将来のプログラマ不足を解決する一つの試みとして、プログラマの知的作業を計算機で自動化し、モジュール仕様書からプログラムを自動生成する実験システムＰＧＥＮを開発した。ＰＧＥＮへの入力となるモジュール仕様書は、ある程度あいまいさや省略の許された日本語と表を用いて記述することができる。本稿では、ＰＧＥＮの仕様記述言語の特徴、およびその言語で記述された仕様を理解するためのＰＧＥＮにおける手法について述べる。

Japanese-like Specification Language for Automatic Programming

Takashi WADA, Kensei TSUCHIDA, Takahiro SUGIYAMA, Masahiro SAKATA,
Ken-ichi TOMITA and Youichi MIYASHITA

NEC Corporation
4-14-22, Shibaura, Minato-ku, Tokyo 108, Japan

We have developed an automatic programming system called PGEN. In the design phase of a typical software development process, systems engineers usually write specifications for component modules and give them to human programmers. PGEN generates business application programs in COBOL from module specifications, automating programmers' intellectual activities. This paper presents the features of the specification language which PGEN accepts as input. Specifications in our language consist of Japanese language sentences, which may include ambiguities to some degree, and of tables expressing certain conditions and actions under each condition. We also describe how PGEN can understand such specifications and generate codes from them.

1．はじめに

現在、大規模なアプリケーションソフトの開発では、要求分析、機能設計、基本設計、詳細設計、製造などの各フェーズ間の情報のやりとりは、自然言語で記述された仕様書を通じて行なわれるのが普通である。詳細設計の結果作成されるモジュール仕様書は、システムエンジニアの意図をプログラマに伝え、プログラマは仕様書を理解してプログラムに翻訳する。したがってモジュール仕様書は、あいまいさ、省略を含んだ自然言語と表の組み合わせで簡潔に記述されることが多い。モジュール仕様書の理解しやすさは、プログラマに対してのみならず、仕様の査閲や後の保守のためにも非常に重要である。

我々は予測される将来のプログラマ不足を解決する一つの試みとして、プログラマの知的作業を計算機で自動化し、モジュール仕様書からプログラムを自動生成する実験システムＰＧＥＮを開発した。入力となるモジュール仕様書は、ある程度制限された、しかしある程度あいまいさや省略の許された自然言語（日本語）と表を用いて記述することができる。本稿では、ＰＧＥＮの仕様記述言語およびそれを理解するためのＰＧＥＮにおける手法について中心に述べる。

2．モジュール仕様書の特徴

我々は、はじめに、仕様記述言語を定める手がかりとして、現実のシステム開発において人間向けに書かれたモジュール仕様書を調査した。それは銀行オンラインシステムのモジュール仕様書であり、したがってＰＧＥＮで扱った例題も銀行オンライン業務をもとにしている。

以下に実際のモジュール仕様書における記述の特徴を挙げる。

（１）業務用語が非常に多く使われる。モジュール仕様書に現れる名詞のほとんどは業務用語である。これらは、おおよそデータ名、データ型名（「通帳残高」、「普通預金」、「操作モード」など）である。

（２）使われる動詞、形容詞などの用言の数が絞られる。用言の大多数は、処理や動作を表現する語（「代入する」、「繰り返す」など）と述語を表現する語（「大きい」、「等しい」など）である。

（３）日本語の記述の中に数式が混入することがある。数式内には日本語の表現が含まれる。たとえば、「通帳残高＞マル優限度額ならば」のような記述が現れる。

用語に関する制限は以下で述べる仕様記述言語にも受け継がれ、したがってその処理系は制限を利用して、より深い意味解析が可能であると言える。

3．仕様記述言語

仕様記述言語は、できる限り文法を意識せずに記述できることが望ましい。上述の通り、現実の仕様書における日本語と同様に語彙の制限された日本語であるが、ある程度のあいまいさと省略を許し、モジュール仕様を記述するのに十分な自由度はある。また、条件および対応する動作をまとめた表と、日本文中の数式表現を組み合わせて簡潔に記述することができる。

モジュール仕様を記述する前に準備として、データ型を定義しておく必要がある。これはシステム全体に共通の定義であり、各モジュール仕様書の変数宣言で参照できる。以下では、はじめにデータ型定義について触れ、次にモジュール仕様の記述言語について例をもとに説明する。

3．1．データ型定義

データ型定義では、データ型の構造や定数を定義するとともに、業務用語との対応づけをおこなう。データ型は以下の３種類からなる。

（１）基本的なデータ型

現在ＰＧＥＮはＣＯＢＯＬのプログラムを生成するが、ＣＯＢＯＬの基本的なデータ型を業務用語と対応させることができる。

（２）構造体

既に定義されたデータ型を部分データとする構造体を定義することができる。部分データの繰り返しも記述できる。定義の例を図１（ａ）に示す。

（３）コード型

コードの名前（業務用語）と実現における値をまとめて定義し、これらのコード値からなるデー

```
%TYPE   T-nyuushukkin-meisai
%NNAM   /入出金明細エリア/;
%PNAM   NYUUSHUKKIN-MEISAI-A;
%STRC
    01  NYUUSHUKKIN-MEISAI-A;
        02  MEISAI              times 4 /明細/ /入金明細/;
            03  KINSHU          :T-kinshu;
            03  KINGAKU         :T-kingaku /入金額/;
        02  KUCHISUU            pic 9(1) /口数/ /入出金口数/;
        02  SOUGAKU             :T-kingaku /総額/ /総金額/
                                          /入金総額/;
```

（ａ）構造体

```
%TYPE   T-sousa_mo-do;
%ATTR   pic X(1);
%NNAM   /ＣＢＩ/ /操作モード/;
%CODE   @       /指示なし/;
        B       /本日締後/;
        D       /前日締後/;
        F       /翌日締後/;
        A       /後送/;
        C       /後送本日締後/;
        E       /後送前日締後/;
        G       /後送翌日締扱い/;
%CCOD   /後送モード/      A C E G;
        /通常モード/      @ B D F;
        /本日締後/        B C;
        /前日締後/        D E;
        /翌日締後/        F G;
```

（ｂ）コード型

図１．データ型定義の例

タ型として定義する。いくつかのコードをまとめて集合的な名前（業務用語）をつけることもできる。図１（ｂ）に例を示す。

３．２．モジュール仕様の記述

モジュール仕様はデータ宣言と処理記述の２つの部分からなる。データ宣言ではモジュールに現れる変数を宣言し、データ名として業務用語を対応させ、処理記述においてはモジュールの処理内容を日本文と表を用いて記述する。

３．２．１．データ宣言

データ宣言は、モジュールにローカルなデータと、モジュール間の情報受け渡しのための共有データを宣言する。

宣言の形式は、図２の例のようにＣＯＢＯＬのそれに類似している。データの構造と各データの名前（業務用語）が定義できるほか、あらかじめ定義されたデータ型も参照できる。これらの業務用語は処理記述の日本文や表の中で参照される。

３．２．２．処理記述

日本文と表による処理記述の例を図３に示す。この例に従って処理記述の特徴を述べる。

（１）節・段落構造

処理記述は節で構成され、通常の文章のように節を入れ子にすることができる。いくつかの日本文の集まりは段落をなす。節・段落構造は仕様記

レベル	データ名	日本語名	PIC/TYPE	TIMES
01	CHUUI-CODE-A	注意コードエリア		
02	CHUUI-KUBUN	区分	9(1)	
02	CHUUI-CODE-NUM	コード数　注意コード数	9(2)	
02	CHUUI-CODE-G			5
03	CHUUI-CODE	コード	9(2)	
01	TEKIYOU	摘要	:T-tekiyou	

図２．データ宣言の例

〔1〕入金総額の計算
　金種が"００"でない入金明細について、以下を繰り返す。
　（1）総額に入金額を加える。
　（2）入出金口数をカウントアップする。
〔2〕監査キーコードエリアのセット
　有帳のとき、以下の処理を行なう。
〔2．1〕
　区分が1のとき、以下を行なう。
　（1）コード数をカウントアップする。
　（2）コード数が5以下なら、コード数番目のコードを"３２１"とする。
　0のとき、後送以外なら、エラーコードを"３２１"として、リターン。
〔3〕注意コードエリアの　セット
　有帳のとき、以下の処理を行なう。
〔3．1〕
　区分が1のとき、以下を行なう。
　（1）コード数をカウントアップする。
　（2）コード数が5以下なら、コード数番目のコードを"３２１"とする。
　0のとき、後送以外なら、エラーコードを"３２１"として、リターン。
〔4〕
　科目コードが"２０"のとき、摘要コードに、"Ｎ０"をセットし、
　"２２"のときは、"Ｎ１"をセット。
〔5〕
　通帳がマル優限度額より大きいとき、表1に従い摘要コードに値をセットしてリターンする。
〔6〕
　流動性預金のとき、表2の組み合わせ以外なら、エラーを４０９としてリターンする。

表1．

科目	処理区分	摘要コード
普通	有帳	"Ｎ０"
	無帳	"Ｎ１"
職員	有帳	"Ｍ０"

表2．

操作モード	処理区分	操作メディア
通常モード	有帳	"Ｇ","Ｋ"
後送モード	有帳	"Ｃ","Ｏ"
	無帳	

図3．処理記述の例

述言語専用のエディタによって入力する。

節・段落構造は、たとえば「以下を繰り返す」における「以下」が指示している範囲を同定するのに利用される。

（2）節見出し

節には見出しを書くことができる。見出しは通常、節についてのコメントとして扱われるが、節の内容に関する補足的な役割を果たすこともある。

たとえば、仕様記述例の〔2〕節と〔3〕節は見出しを除いて日本文が全く同じであり、「コード数」の意味があいまいである。記述例のように見出しがあれば、ＰＧＥＮはデータ型定義をチェックして〔2〕節の場合は「監査キーコードエリア」、〔3〕節は「注意コードエリア」のサブフィールドの「コード数」と判定する。

（3）処理条件の簡略表記

「有帳のとき」という表現は主語が省略されており、「処理区分が有帳のとき」が完全な表現である。「有帳」はコード名であり、データ型「処理区分」の変数が取り得る一つのコード値であるとデータ型定義において定義されている。この情報から主語を補完できる。コード値に関してはこのような主語の省略が可能である。

（4）主語や目的語の省略

上述した主語の省略の他、文型に着目した主語や目的語の省略もできる。たとえば、記述例の〔4〕節の中の「"２２"のときは」は主語「科目コードが」が省略され、「"Ｎ１"をセット」で

は目的語「摘要コードに」が省略されている。この場合は直前の文の文型と似ているので、PGENは省略語を直前の文から補完できる。

また、「AとBを加え、Zに代入する」のような記述も許される。この文では何を「Zに代入する」のかが明示的に記述されていない。しかし、局所的な文脈を理解することにより「AとBを加えた結果」であることが分かる。

（5）繰り返しデータについてのループ

繰り返しデータ（配列）についての処理ループ（配列の各要素について繰り返す処理）を簡潔に記述することができる。記述例の〔1〕節にある「入金明細について、以下を繰り返す」がその例である。ここで「入金明細」は繰り返しデータであり、繰り返し回数（配列の大きさ）はデータ型定義あるいはデータ宣言に記述されている。

繰り返し処理の記述中にもあいまいな業務用語が記述できる。たとえば、「入金額」があいまいであっても、それが「入金明細」のサブフィールドであれば、「入金明細の入金額」という意味になる。

（6）表

記述例の〔5〕節では、条件の一部を日本文で表現し、残りの条件は表にまとめている。また、処理の大筋は日本文で述べているが、「セットする」値は表の中に記述されている。表は、条件と対応する処理についての情報（条件のみの場合もある）を記述するが、それを参照する日本文と組み合わせて完全な仕様となる。

（7）数式混入の日本文

数式を日本語化した形で日本文の中に記述することができる。たとえば、「入力の通帳残高＝元帳の通帳残高でなければ」のような表現が許される。

4．PGENのシステム構成と処理概要

図4にPGENのシステム構成を示す。

処理記述の日本文と表は構文解析、構造解析を行なって、それぞれNネット、Tネットと呼ばれるネットワーク形式の表現に変換する。構文的にあいまいな日本文の構文解析では、はじめにすべての可能な解析結果を求め、文節間の係り受けの良さを評価し、最も良い構文木を選択している。

データ型定義は開発対象システムの全モジュールに共通であり、定義が終わればただ一度だけ辞書作成ツールによって、内部処理用辞書の項目に変換される。同様に、データ宣言はモジュール毎に辞書作成ツールによって辞書項目に変換される。

内部処理用辞書は構文解析用と意味処理用の2種類である。上述のように追加されるデータに関する辞書項目の他、PGENに固有な項目が初めから含まれている。それは用言など日本語に関する知識である。

意味処理は、ネットワークによるモジュール仕様の表現を自然言語に近い形からプログラムを生成できる形へ徐々に変換してゆく過程である。意味処理で扱うモジュール仕様のネットワーク表現をMネットと呼ぶ。意味処理部は、NネットとTネットから初期Mネットを生成し、意味処理用辞書を参照しながらMネットを変形してゆく。

最後に、変換の終了したMネットからプログラムを生成する。

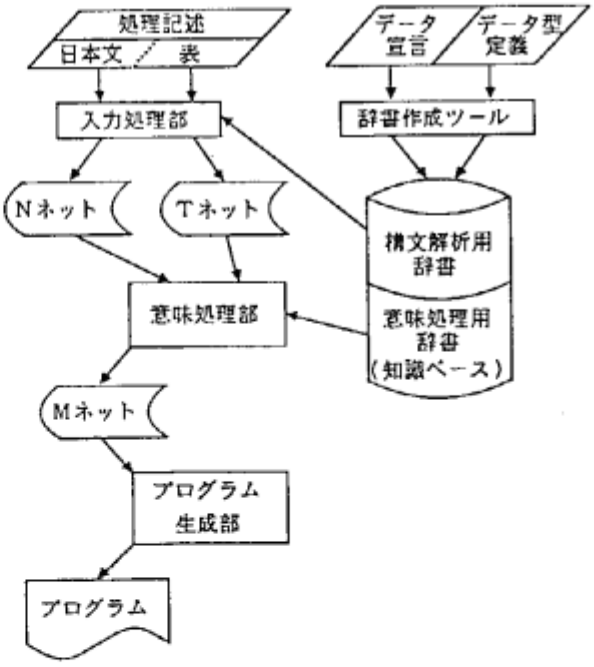

図4．PGENのシステム構成

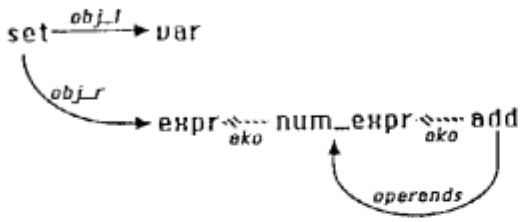

図５．Ｍモデルの一部

５．仕様記述の意味処理

上述の通り、意味処理ではモジュール仕様を表現したＭネットを変換して、あいまいさを解消し省略を補い、プログラムを生成可能な完全な形にする。Ｍネットの変換に関する規則は意味処理用辞書に書かれている。意味処理用辞書およびＭネット変換の規則について以下に述べる。

５．１．意味処理用辞書

意味処理用辞書はＭネットを変換するための知識の集まりであり、知識表現法の一つであるフレームの考え方に沿って記述されている。この記述をＭモデルと呼ぶ。図５はＭモデルの一部の例である。

Ｍモデルはフレームの階層構造になっており、下位概念のフレームから上位概念のフレームへスロット "ako" で関係付けられる。この関係に沿ってフレームの性質の継承が行なわれる。図５の例では、"num_expr"（数式）は "expr"（式）の一種であると表現されている。

Ｍネットの各ノードはＭモデル内のあるフレームのインスタンスである。Ｍネットのノードが持つアークの種類と、その先にくるべきノードのタイプがＭモデルに記述されている。図５の例によれば、「代入する」を表現する "set" なるＭネットのノード（正確には "set" なるフレームのインスタンスであるノード）は、"obj_l"（代入の左辺）と "obj_r"（右辺）なるアークを持ち、その先はそれぞれ変数と式を表わしたノードでなければならない。

このようなチェックは、意味処理のはじめにＭネットを生成する時点で行なわれる。このとき、あいまいさが解消される、もしくは、減少する可能性がある。用言やその主語・目的語が意味的にあいまいで、それぞれについて複数のノードが生成された場合、組合せの中のいくつかは上述のチェックにより捨てることができるからである。

この他、Ｍモデルのフレームは、そのインスタンスであるＭネットのノードをどのように変換するかを記述した規則を持つ。この規則のいくつかについて次節に述べる。

５．２．Ｍネットの変換規則

意味処理は６フェーズに分かれ、各フェーズ毎にＭネットをトラバースし変換を行なう。トラバースのしかたはフェーズ毎に異なる。各フェーズの主な処理は以下の通りである。

（１）Ｍネットの生成
（２）並列句の展開
（３）繰り返しループの生成
（４）表の展開
（５）省略語の補完
（６）不要な入れ子ブロックの消去

次に、並列句の展開と省略語の補完についてやや詳しく述べる。

５．２．１．並列句の展開

「ＸがＡ、Ｂと等しい」のような並列句が含まれる場合は、プログラムを生成しやすい形に展開しなければならない。この例では「ＸがＡと等しい、またはＸがＢと等しい」と展開すべきである。ネットワークの形で展開を図示すると図６のようになる。

「Ｘ」に当たる部分が並列句になっているか否かによって展開のしかたが異なり得る。また並列句に関わる用言（この例では「等しい」）にも依存する。したがって、展開規則はその用言のＭモデルに記述される。

表１に、その他の並列句の例を展開したときの意味を示す。

５．２．２．省略語の補完

前に述べたが、省略語の補完には主として文型による補完と文脈による補完がある。

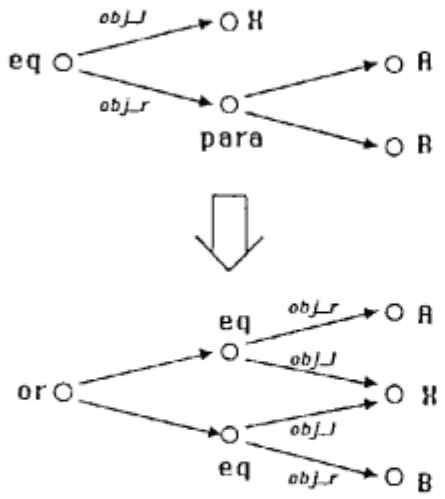

図６．並列句の展開

日本文の中に省略があった場合、Mネット上では省略語のあるべき位置に特殊なノードが省略語の代わりに置かれる。図７の中のノード"＄＄"がそれである。第５フェーズのトラバースでノード"＄＄"に出会うと、補完の処理を開始する。

文型による補完は、ノード"＄＄"を含むMネット中のサブネットと相似なサブネットをMネット中に探す。探す範囲は、問題にしている省略を含んだ日本文より前に書かれている日本文を表現したMネットの部分である。相似なサブネットが見つかれば、ノード"＄＄"を、見つかったサブネット中で"＄＄"に対応する位置にあるノードで置き換えればよい。

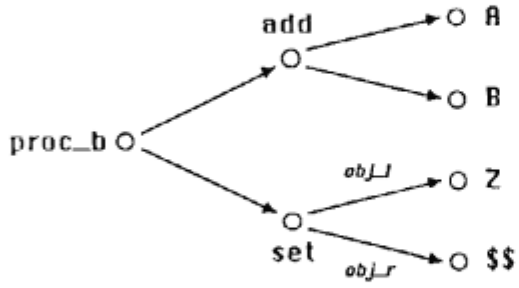

図７．省略語を含んだMネット

日本語表現	意味
X、YがAと等しい	X = A and Y = A
X、YがA、Bと等しい	X = A and Y = B
XにA、Bを加える	X = X + A + B;
X、YにAを加える	X = X + A; Y = Y + A;

表１．並列句の意味

文脈による補完は、ある用言の目的語の位置にノード"＄＄"が置かれていたときに、直前の文で置き換えることを試みる。「AとBを加え、Zに代入する」では、図７に示したように「代入する」の目的語の位置にノード"＄＄"がある。図２のMモデルによれば、"set"（「代入する」）の目的語は"expr"でなければならない。一方、"add"（「加える」）は"num_expr"、さらには"expr"の一種であるから、「AとBを加え」を「代入する」の目的語とすることができる。置き換えた後のMネットを図８に示す。

６．仕様記述言語の拡張について

ＰＧＥＮは現在、実験用の小規模な辞書類を用いており、あらゆる処理記述が可能なわけではない。ＣＯＢＯＬで記述可能なすべての処理をＰＧＥＮの仕様記述言語で記述できることが要請される。それが満たされれば、ビジネスアプリケーションに関して仕様記述言語の記述能力は十分であると言える。

我々は仕様記述言語を拡張するため、まずＣＯ

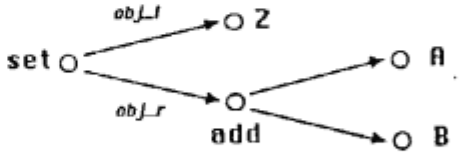

図８．文脈による補完の結果

ＢＯＬの各命令を日本語で言い替えることを試みた。日本語表現には以下の性質を要求した。

（１）日本語として正しい。

（２）あいまいさがない。

（３）表現が簡潔である。（たとえば、命令のパラメータそれぞれについて、その日本語表現が複数箇所に出現しない。）

以上を満足する日本語の言い替えを、すべてのＣＯＢＯＬの命令について求めることができた。あらゆる言い替えを求めたわけではないが、低レベルな記述から高レベルな記述の合成、ただし書きや指示代名詞の扱いなどの問題を含んだ表現がかなりみられた。ＰＧＥＮの処理方式を拡張してその問題を扱うのは難しいので、プログラミング言語的な記法やプログラム構造図を導入するなど、日本語の自然さをある程度犠牲にする必要はあろう。

７．おわりに

我々は、モジュール仕様の実際的いくつかの例題を記述し、それからプログラムが生成できることを確認した。

ＰＧＥＮをアプリケーションソフトウェアの開発に適用した場合、大量の業務用語をデータ型定義において定義する必要がある。したがって、かなり大規模の業務（銀行オンライン業務など）に適用する場合にＰＧＥＮは有効である。

参考文献

［１］岩元，西谷，和田，庄司，土田「ＰＧＥＮ－１：仕様書理解をベースとしたプログラム自動生成システム」情報処理学会第３２回全国大会５Ｍ－６

［２］和田，西谷，岩元「数式を含む日本文の構文解析」情報処理学会第３２回全国大会４Ｓ－４

［３］岩元，西谷，和田「プログラム自動生成システム」ＮＥＣ技報 vol. 40, no. 1, pp. 35-38 (1987)