B4　プログラミング言語のトランスレータ

久原　由美子　　大槻　説乎　（九州大学中央計数施設）
工藤　奈津子　　竹下　節子　（九州大学大型計算センタ）

1. はじめに

今から10年余り前まではまだコンパイラができていなかったので，電子計算機の利用者は機械語やアセンブラ言語という計算機ごとに全く異なる言語を使ってプログラムを書くのが常識であった．その後コンパイラの技術が開発され，機械に独立なプログラミング言語という概念が発生して，世界中でFORTRAN，ALGOL，COBOLなどというような共通言語を使ってプログラムを記述できるというプログラミングの世界で特筆すべき革命がおこり，それを一つの契機として計算機はそれまでのいわゆる「計算する機械」から「情報を処理する機械」へと急速な発展をとげてきた．

現在日本でもFORTRANをはじめとして，いくつかのプログラミング言語が広く使われている．これらの言語は機械語やアセンブラ言語との比較においてはたしかに機械に独立ではあるが，コンパイラをも含めた計算機システムに対しては同じ名前の言語でさえも，実際は処理系によって違った仕様を含んでいる上，ＪＩＳ規格などでレベルが違っている場合もある．大多数の利用者はこのように処理系ごとに定められた言語仕様を習得して自分のプログラムを記述せねばならない．最近のように計算機の設置台数が増えると違った計算機を使う機会が多くなり，そのたびに計算機の言語仕様に合わせて自分のプログラムを書き直さざるを得ない．プログラムが専門でない大多数の利用者には，このプログラムの書き変えや言語仕様をおぼえる手間が大きな負損になっている．しかも計算機は4，5年ごとに新しいものに入れ替えられるために莫大なプログラムの書き変えの必要もおこってくる．

このような問題を避ける方法として，言語仕様そのものを計算機にデータとして入力することにより任意の言語を任意の計算機に理解させ，計算機システムに独立な言語を自由に使えるようにしようという考え方がある．われわれのトランスレータの実験もこのような考え方から出発したもので，あるプログラミング言語Li　を別のプログラミング言語Ljに自動変換する問題を扱っており，あらかじめLi，Lj　のシンタクスとセマンティクスを計算機に入力して必要な情報処理を行なう部分と，Li　を使って書かれたプログラムを入力して解析し，同じプログラムをLj　を使って再構成する部分とからなっている．以下Li　は変換されるべきソース言語を，Lj　は生成されるべき目的言語を表わすことにする．

2. トランスレータの構成

トランスレータは図1に示す処理部分と図2に示すファイル部分とから成る．＊で始まる語はトランスレータの制御指令であり，図1のように5種類の指令に従ってそれぞれ新しいファイルの開設，シンタクス処理，セマンティクス処理，翻訳および処理の終了のルーチンに制御を移す．

＊NAME　　新しく言語仕様を入力する場合は先づこの制御文とその言語の名前とを入力する．処理ルーチンはこの名前をつけたファイルを新しく開設する．

＊SYNTAX　　新しい言語仕様のシンタクス部分を記述したデータを入力して処理し，名前のついたファイルに翻訳のための情報を格納するルーチンを呼び出す．

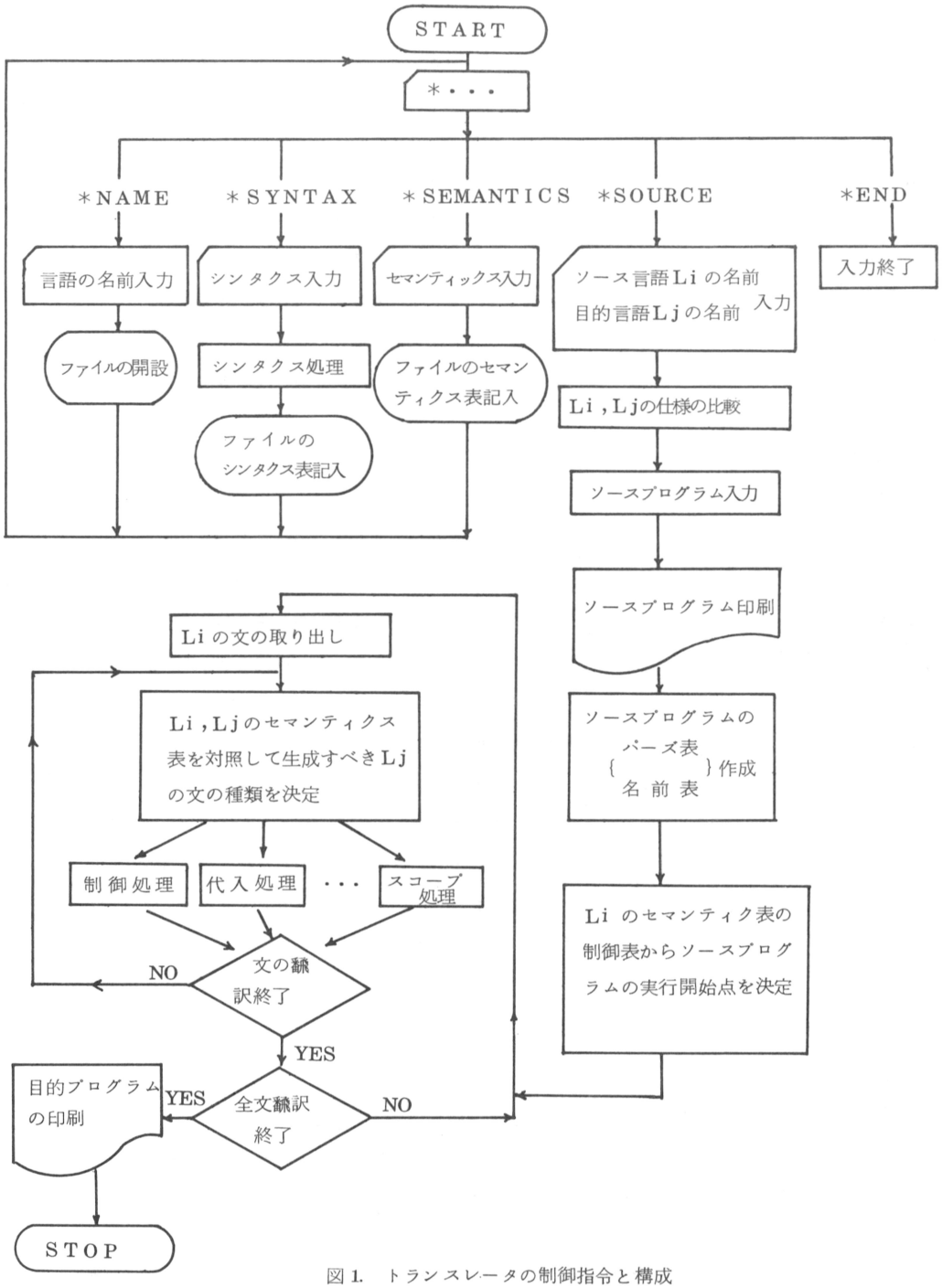

図1. トランスレータの制御指令と構成

＊SEMANTICS　新しい言語仕様のセマンティクス部分を記述したデータを入力してファイルに格納するルーチンを呼び出す．

＊SOURCE　翻訳ルーチンを呼び出す制御文で，この文につづいてソースプログラムの言語Li と目的プログラムの言語Lj を指定するデータがつづき，さらにソースプログラムの文がつづく．Li とLj は共にその名前がファイルに登録済のものでなければエラーとなる．翻訳ルーチンはこれらのプログラムを読んだのちLi とLj のファイルを比較して異っている部分のチェックを行う．仕様が一致する部分は翻訳の必要がないからである．勿論仕様によっては機械的に翻訳できない場合もおこる．この場合はメッセージを印刷する．ソースプログラムのパージング，名前の処理，スコープの処理を行ったのちセンテンスごとに翻訳をおこなう．

＊END　新しい言語仕様を入力する時の最後のデータ又はソースプログラムの終点を指示するための指令文である．

1 名　　前
　2 シンタクス表
　　3 始　点　表1
　　3 始　点　表2
　　3 始　点　表3
　　3 語　順　表
　　3 記号列表
　2 セマンティクス表
　　3 ターミナル表
　　3 スコープ表
　　3 制　御　表
　　3 入出力形式表
　　　⋮

図2.　言語仕様を記入するファイル

われわれのトランスレータの内容を説明するために非常に簡単な2種の言語L1とL2を作ってこの報告の全節にわたって例題として使用することにする．L1とL2を使って書いた「N個の数を大きい順に並べるプログラム」を図3および図4に示す．

```
「モシA(I)>A(J)ナラバ
      W=A(J),
  A(J)=A(I),
  A(I)=W0
タダシI=J+1   カラ   N   マデ
    J=1       カラ   N-1マデトスル」.
```

図3.　L1で書いたプログラム

```
    (START)  10
10  〔J=1〕  20
20  〔I=J+1〕30
30  <A(I)>A(J)>40, 50
40  〔W=A(J), A(J)=A(I), A(I)=W〕50
50  <I=N>60, 90
60  <J=N-1>70, 80
70  (STOP)
80  〔J=J+1〕20
90  〔I=I+1〕30
```

図4.1　L2で書いたプログラム．この言語は下の流れ図と1対1に対応している．

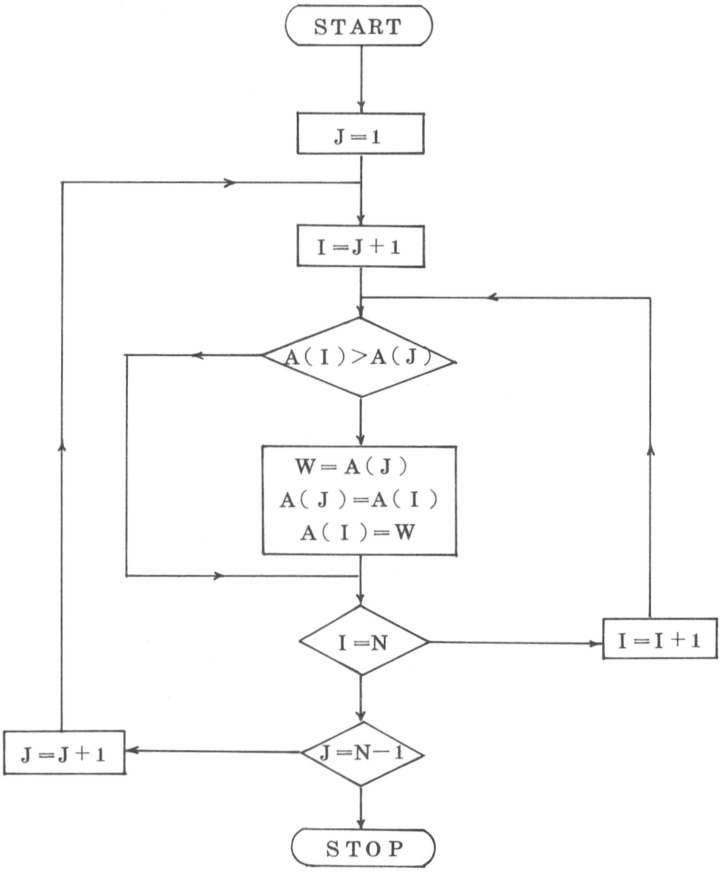

図　4.2

3. シンタツクスの記述とその処理

3.1 シンタックスの記述

前に述べたように言語仕様はシンタックスとセマンティクスに分けて記述される．シンタクス部分はＢＮＦで記述し，次の５つの制約に従わねばならない．

1　メタネイムには必ず＜プログラム＞という部分がなくてはならない．これはソースプログラムをパージングする際のパージングの木のＴＯＰになる．

2　ソースプログラムのセンテンス単位で目的プログラムの生成の可能性を調べる（一般にソース言語と目的言語は１対１対応しない)ので，メタネイムにはセンテンスの種類全部を総称するものが要る．

3　すべての種類のセンテンスはメタネイムを持たねばならない．

4　セマンティクス表に引用されるターミナル以外の名前は必ずメタネイムとして記述しなければならない．

5　式（算術式，関係式，論理式，添字式など），数値，名前は全部セマンティクス表で記述する．

例題の言語Ｌ１とＬ２のシンタクスは図５および図６に示されている．図中下線のある語は上の制約(5)に従ってセマンティクス表に記述されているもので，ＢＮＦによる記述ではターミナルと同じように扱われる．右端の括弧の中の数字は上の制約のどの部分に該当しているかを示したものである．数字のない行はＢＮＦの本来の機能すなわち語順の記述（特にくり返しの記述），分類，類の間の結合の強さの順位などのいずれかを表わすために必要なものである．上記の制約(5)はＢＮＦ記述の常識からは相当外れているように思う．特に算術式などはＢＮＦで記述した構文解析の例題としてたいていの教科書に載っている．しかし次節で述べるようなターミナル記述の中にこれらの仕様を埋めた方が，ＢＮＦで記述するよりもはるかに実用的で詳細な記述が得られる上，言語間の仕様比較が簡単に行えるのであえてシンタックス記述から落としたものである．

＜プログラム＞::＝＜文＞｜＜文＞＜プログラム＞　(1)

＜　文　＞::＝＜コメント文＞.｜＜条件文＞.｜＜代入文＞.｜＜繰り返し文＞.　(2)

＜コメント文＞::＝コメント　文字列　(3)

＜条　件　文＞::＝＜条件部＞＜実行部＞　(3)

＜実　行　部＞::＝＜文＞｜＜文＞，＜実行部＞　(4)

＜条　件　文＞::＝比較式ナラバ｜論理式ナラバ｜モシ＜条件部＞

＜代　入　文＞::＝変数＝算術式　(3)

＜繰り返し文＞::＝「＜本文＞タダシ＜繰り返し節＞トスル」　(3)

＜本　　文＞::＝＜文＞｜＜文＞＜本文＞　(4)

＜繰り返し節＞::＝変数＝算術式カラ算術式マデ｜　(4)

変数＝算術式カラ算術式ゴトニ算術式マデ｜

＜繰り返し節＞＜繰り返し節＞

図５．　Ｌ１のシンタクス

＜プログラム＞::＝＜開始記号＞文番号＜文＞｜＜プログラム＞文番号＜文＞　(1)

＜　文　＞::＝＜条件文＞｜＜複文＞｜＜終止記号＞　(2)

＜条　件　文＞::＝　＜比較式＞　文番号，文番号　(3)

＜複　　文＞::＝〔＜本文＞　(3)

＜本　　文＞::＝＜代入文＞|＜代入文＞，＜本文＞

＜代 入 文＞::＝変数＝算術式　(3)

＜終止記号＞::＝（ＳＴＯＰ）　(3)

＜開始記号＞::＝（ＳＴＡＲＴ）文番号　(4)

図6.　Ｌ２のシンタクス

3.2　シンタクス処理

シンタクスの処理ルーチンは次の部分からできている．

ルーチン名	内　　容	エラー検出
ＢＮＦ読み込み	ＢＮＦを入力して次の４種のグループを作る (a) 3.1の制約(1)，(2)によるもの (b) 3.1の制約(3)によるもの (c) 3.1の制約(4)によるもの (d)その他	①ＢＮＦ記入形式エラー ②ターミナルばかりで構成されるメタネイムが無い ③3.1の制約から外れている
再帰的定義の処理	Ａ::=Ｃ|ＡＢ，Ａ::＝Ｃ｜ＢＡ， Ａ::=Ｄ|ＢＡＣの形のＢＮＦを反復記号〔　〕を用いてそれぞれＡ::＝Ｃ〔Ｂ〕，Ａ::＝〔Ｂ〕Ｃ，Ａ::＝〔Ｂ〕Ｄ〔Ｃ〕に変形する．	①Ａ::＝ＢＡＣの形のエラー．ただしＢ，Ｃは空系列も含む．
代　　入	右辺に含まれるメタネイムをその値（そのメタネイムを左辺とする右辺を指す）で置きかえる．ただしグループ(a)と(b)の間の代入は行わない．	①同じグループの間でサイクルができる時は上のエラーと同じ形になる．
キー語の選定	(b)グループの各文でキー語になるターミナルを決定する．	①全く等しい右辺を持つメタネイムが２つ以上存在して異ったセマンテックス表から引用されている
シンタックス表の作成	・ターミナルを集めて記号表を作る ・右辺を集めて語順表を作る． この中にエラー処理のポインターを含んでいる． ・(a)，(b)，(c)グループのメタネイムと語順表へのポイターを含む表をそれぞれ始点表１２，３とする．	

4.　セマンテイツクス表とその意味

セマンティックス表は，Ｌi やＬj で用いられる記号，シンタックスとの関係を表わすメタネイム，記号列，セマンティックス表の中だけで使われる記号などにより表わされ，一つ一つの意味はセマンティックス番号と一対一に対応づけられており，セマンティックス表の任意の意味を参照することができる．

セマンティックスとして記述せねばならない部分はとりあつかう言語によって異るので，セマンティックス表の任意の部分を拡張することができるようにした．表中｛　｝でくくった部分はその意味を表わすセマンティックス番号を表わし，＜＞はシンタクスのメタネイムを表わす．

4.1　ターミナル表

式や数の意味は，普通数学で用いられる式や数の概念にLi やLj で用いられる式や数の意味をそれぞれ対応づけて表わすことにより，Li とLj の式や数の意味を対応させる．式の意味は，数学で用いられる演算記号に対応づけてプログラムで用いられる演算記号，その演算記号がprefixかinfixかsuffix かの区別，演算子の優先順位，被演算子の型，結果の型，優先順位を変更するための記号の指定から構成されている．

数，名前の意味は，表に示される種々の記号を用いることにより記述される．

L1，L2のターミナル表は表1で表わされる．

L1のターミナル表

意味		数学上の記号	Pre., in., Su. の別	terminal	優先順位	被演算子の型 第1の型	被演算子の型 第2の型	結果の型
1. 演算子	1.1 算術演算子	1.1.1 ＋	in.	＋	4	注1）		
		〃 2 －	in.	－	4	〃		
		〃 3 ×	in.	＊	3	〃		
		〃 4 ÷	in.	／	3	〃		
		〃 5 巾	in.	＊＊	1	注2）		
		〃 6 ＋	pre.	＋	2	注3）		
		〃 7 －	pre.	－	2	〃		
	1.2 比較演算子	1.2.1 ＞	in.	＞	5	｛算術式｝	｛算術式｝	｛論理値｝
		〃 2 ＜	in.	＜	5	〃	〃	〃
		〃 3 ＝	in.	＝	5	〃	〃	〃
		〃 4 ≧	in.	≧	5	〃	〃	〃
		〃 5 ≦	in.	≦	5	〃	〃	〃
	1.3 論理演算子	1.3.1 ∧	in.	∧	7	｛比較式｝	｛比較式｝	｛論理値｝
		〃 2 ∨	in.	∨	8	〃	〃	〃
		〃 3 ¬	pre.	¬	6	〃	〃	〃
	1.4 優先順位の変更	（　）		（，）	0			
2. 代入			in.	＝		｛整数｝ ｛実数｝	｛整数｝ ｛実数｝	｛整数｝ ｛実数｝
3. 数	3.1 単純変数	3.1.1 整数	〔I｜J｜K｜L｜M｜N〕			＃：＝1		
		3.1.2 実数	α			＃：＝1		
	3.2 10進定数	3.2.1 整数	ε^*			＃：＝10		
		3.2.2 実数	$\varepsilon^*.\varepsilon^*$			＃：＝7		
	3.3 添字付変数	3.3.1 整数	｛整数｝（e｛整数｝）					
		3.3.2 実数	｛実数｝（e｛整数｝）					
	3.4 論理定数		TRUE｜FALSE					
	3.5 文字列		δ^*					

L 2のターミナル表

意味		数学上の記号	Pre., in, Su. の別	terminel	優先順位	被演算子の型 第1の型	被演算子の型 第2の型	結果の型
1. 演算子	1.1 算術演算子	1.1.1 ＋	in.	＋	4	注1）		
		〃 2 －	in.	－	4	〃		
		〃 3 ×	in.	＊	3	〃		
		〃 4 ÷	in.	／	3	〃		
		〃 5 巾	in.	＊＊	1	注2）		
		〃 6 ＋	pre.	＋	2	注3）		
		〃 7 －	pre.	－	2	〃		
	1.2 比較演算子	1.2.1 ＞	in.	＞	5	{算術式}	{算術式}	{論理値}
		〃 2 ＜	in.	＜	5	〃	〃	〃
		〃 3 ＝	in.	＝	5	〃	〃	〃
		〃 4 ≧	in.	≧	5	〃	〃	〃
		〃 5 ≦	in.	≦	5	〃	〃	〃
	1.3 論理演算子	1.3.1 ∧	in.					
		〃 2 ∨	in.					
		〃 3 ¬	in.					
	1.4 優先順位の変更	（ ）		（ , ）	0			
2. 代入			in.	＝		{整数} {実数}	{整数} {実数}	{整数} {実数}
3. 数	3.1 単純変数	3.1.1 整数	〔 I | J | K | L | M | N 〕 β^*					
		3.1.2 実数	$\alpha\beta^*$ — {整数}					
	3.2. 10進定数	3.2.1 整数	ε^*					
		3.2.2. 実数	$\varepsilon^*.\varepsilon^*$		$\mid\varepsilon^*.\varepsilon^*_{10}$〔＋1－1 λ〕ε〔ε | λ〕#:＝7			
	3.3 添字付変数	3.3.1 整数	{実数}（e{整数}）					
		3.3.2 実数	{実数}（e{整数}）					
	3.4 論理定数		TRUE | FALSE					
4. 文のID			ε^* #:＝4					

注1）

第1の型	第2の型	結果の型
{整数}	{整数}	{整数}
{整数}	{実数}	{実数}
{実数}	{整数}	{実数}
{実数}	{実数}	{実数}

注2）

第1の型	第2の型	結果の型
{整数}	{整数}	{整数}
{実数}	{整数}	{実数}
{実数}	{実数}	{実数}

注3）

第1の型	結果の型
{整数}	{整数}
{実数}	{実数}

記号の意味　α：英大文字
β：英数字
γ：英数字，特殊記号
ϑ：英数字，特殊記号，カタカナ
ε：数字
右肩の＊は0回以上のくりかえしを表わす．
eは算術式，eのつぎの｛　｝はセマンティックス番号で結果の型を表わす．
入：空系列
＃は最大桁数を示す．桁数は1桁以上最大桁数まで許される．
A｜BはAまたはBを表わす．
〔A｜B〕〔C｜D〕はAC｜AD｜BC｜BDを表わす．
他の記号はterminal記号を表わす．

表1.　L1，L2のターミナル表

4.2　スコープ表

ソース言語Liで書かれたプログラムから目的言語Ljで書かれたプログラムへ翻訳する場合，ソースプログラムや目的プログラムに現われる名前（変数，文のIDなど）についてその有効範囲（スコープ）をLiやLjのセマンティックスとして与える必要がある．名前と有効範囲について記述することを目的として，現段階では，シンタックスのメタネイムや記号列を用いることによりテーブルの形式（スコープ表）で与えている．一般にプログラミング言語はその言語がブロック構造をなしているか，ブロック構造をなしていないかによって，名前の有効範囲が局所的であるか大域的であるかが定められる．シンタックスとの関係はメタネイムや記号列だけとし，実際の範囲の決定はトランスレータのスコープ処理で行う．スコープ表は変数と文のIDに分けられ，変数に対しては，そのセマンティックス番号，ブロック構造かブロック構造でないかの別，宣言があるかないか，ブロック構造の場合はそれに対応するメタネイムか記号列を，宣言がある場合にはそれを示すメタネイムか記号列をセマンティックス番号と対応させて記述する．文のIDに対しては，ブロック構造であるかないかの区別を記述する．L1，L2のスコープ表は表2のように記述される．

<table>
<tr><th>意　味</th><th>セマンティックス番号</th><th>ブロック構造か，ブロック構造でないか</th><th>ブロックを示すメタネイム又は記号列</th><th>宣言の有無</th><th>宣言を示すメタネイムまたは記号列</th></tr>
<tr><td rowspan="3">5.1
変

数</td><td>5.1.1
｛整変数｝</td><td rowspan="3">ブロック構造
でない</td><td rowspan="3">な　し</td><td rowspan="3">な　し</td><td rowspan="3">な　し</td></tr>
<tr><td>5.1.2
｛実変数｝</td></tr>
<tr><td>5.1.3
｛添字付変数｝</td></tr>
<tr><td>5.2
文のID</td><td>5.2.1
｛文のID｝</td><td>ブロック構造でない</td><td>な　し</td><td colspan="2"></td></tr>
</table>

表2.　L1，L2のスコープ表

4.3　制御表

制御表は始点終点表，制御の変更表，繰り返し表の3つから成る．

(1)　始点終点表

始点終点表には，プログラムの実行の開始を示す始点とプログラムの実行の終りを示す終点についてのセマンティックスが記述される．これらに関して，シンタックスのどの文に対応しているか，始点の次に実行すべき文が指定さ

れているか否か，指定されているとすれば行先はシンタックスの何で表わされているか，について記述する必要がある．一般にこれらを指定する文がなければ，プログラムの実行は最初の文（ＴＯＰ）から始まり，最後の文（ＤＯＷＮ）で終了するものと考えられる．

Ｌ１，Ｌ２の場合は表３の様になる．

Ｌ１

意　味	文のメタネイム	次に実行すべき文	
		指定の有無	行　先
6.1 始　点	ＴＯＰ（注１）	０（注３）	
6.2 終　点	ＤＯＷＮ（注２）		

Ｌ２

意　味	文のメタネイム	次に実行すべき文	
		指定の有無	行　先
6.1 始　点	＜開始記号＞	１（注３）	文番号
6.2 終　点	＜終止記号＞		

（注１）　ＴＯＰ　……　プログラムの最初の文を示す．
（注２）　ＤＯＷＮ　…　プログラムの最後の文を示す．
（注３）　０　……　次に実行すべき文の指定がないことを示し，制御は次の文へ移る．
　　１　……　指定がある事を示し，「行先」に示されている文へ制御が移る．

表３　Ｌ１，Ｌ２の始点終点表

⑵　制御の変更表

この表では無条件変更，条件は変更などの制御の変更に関する文のセマンティックスが記述される．①シンタックスのどの文に対応しているか，②制御の変更が行われる場合に条件があればその条件を，③制御の変更が行われるならばその変更先，などについて記述する必要がある．②は更に，その条件はシンタックスのどの部分に記述されているか，その条件を表わすものはセマンティックス上でどの範囲か（例えば論理式，比較式，算術式など），これらの条件がどの様な値を持つ場合か，などについて記述する．③は制御の変更が行われるか否か，行われるとすればそれは自分自身の文の中での変更か，文の外への変更かについての区別と，その行先がシンタックスの何で表わされているかについて記述する．一般に制御の変更がない場合は次の文へ制御が移ると考えられる．

Ｌ１，Ｌ２の場合は表４の様になる．

Ｌ１

意　味	文のメタネイム	条　件			制御の変更	
		シンタックス表との対応	セマンティックス	値	文の中か外か	行　先
7.1 無条件変更	０（注１）					
7.2 条件付変更	＜条件文＞	＜論　理　式＞	｛論理式｝（注３）	Ｔ	－１（注４）	＜実行部＞
		＜論　理　式＞	｛論理式｝	Ｆ	０（注４）	

L2

意　味	文のメタネイム	条件			制御の変更	
		シンタックス表の対応	セマンティックス	値	文の中か外か	行　先
7.1 無条件変更	<複　文>	0（注2）			1（注4）	文番号
7.2 条件付変更	<条件文>	<比較式>	{比較式}（注3）	T	1	文番号（注5）
		<比較式>	{比較式}	F	1	文番号（注5）

（注1）　文のメタネイムが0の時は該当する文がない事を示す．

（注2）　条件の，シンタックス表との対応が0の時は条件がない事を示す．
従って無条件変更文のこの欄は常に0

（注3）　ターミナル表のセマンティックス番号で示される．

（注4）　制御の変更のこの欄はそれぞれ次の意味をもつ．

- −1 …… その文の中で制御の変更がおこなわれることを示す．この時の「行先」はその文の中の実行すべき部分を示す．
- 1 …… 文の外へ制御が移されることを示す．
- 0 …… 変更は行なわれず，次の文へ制御が移ることを示す．従って，この時「行先」は意味を持たない．

（注5）　1つの文のシンタックスで同じメタネイムがいくつか現われる時は，この様に左から順に1，2，3…と順番をつけて区別する．

表4.　L1，L2の制御の変更表

(3)　繰り返し表

プログラムのある部分を繰り返すための繰り返し文についてのセマンティックスが記述される．一般に繰り返し文はきざみ指定型（例えばFORTRANのDO文や，ALGOLのstep until 型など）や，ALGOLにおけるwhile型などの様に，繰り返しの制御の仕方によって，かなりセマンティックスの記述も異なってくる．

ここでは，まず繰り返しの制御の型の区別を記述し，それぞれの型に応じて繰り返し制御表の記述形式を変えている．

この表では，①シンタックスのどの文に対応しているか，②繰り返しの範囲（シンタックス上で），③繰り返しの制御がシンタックスのどの部分に記述されているか，④繰り返しの制御の型の区別，⑤繰り返しの実際の制御方法について記述している．

L1の場合は表5の様になる．

意　味	文のメタネイム	繰り返しの範囲	繰り返し制御										
			シンタックス表との対応	繰り返し制御の型	制御変数		初期値		きざみ			最終値	
					シンタックス表との対応	セマンティックス	シンタックス表との対応	セマンティックス	指示の有　無	シンタックス表との対応	セマンティックス	シンタックス表との対応	セマンティックス
8.1 きざみ指定型繰り返し文	<繰り返し文>	<本　文>（注1）	<繰り返し節>	きざみ指定型	変　数	整変数	算術式1（注2）	{整数}	有	算術式2（注2）	{整数}	算術式3（注2）	{整数}
							算術式1（注2）	{整数}	無		1	算術式2（注2）	{整数}

（注1）　繰り返しの範囲がシンタックスで1つのメタネイムで表わされている時は，この欄は1つのメタネイムをもつ．
そうでなければ，繰り返しの範囲の始めと終りを示すメタネイムを2つもつ．

（注2）　表4の注5と同じ．

表5　L1の繰り返し表

4.4　言語の入出力形式表

言語の入出力記述形式は，free formatとformat 指定のあるものの2つに分けられる．free formatの場合は文の区切イ記号を指定すればよいが，入力媒体がカードの場合は，本文が記述されるカラムを指定する必要がある．format 指定のあるものは，文のＩＤや本文が記述されるカラムを指定しなければならない．

Ｌ1，Ｌ2の記述形式は表6で表わされる．

Ｌ1

	コメント文	本文	文のＩＤ	継続行	文の区切り	ＩＤ SEQ
カラム	第1～第72	第1～第72	なし	指定なし	。	第73～第80

Ｌ2

	コメント文	本文	文のＩＤ	継続行	文の区切り	ＩＤ SEQ
カラム	なし	第6～第72	第1～第4	指定なし	なし	第73～第80

表6.　Ｌ1，Ｌ2の入出力形式表

5.　制御について

5.1　始点終点の処理

始点を示す文がＬi，Ｌjのどちらか一方にある場合には，新しく始点の文をＴＯＰの前に生成したり，文のいれかえをしたりする必要がある．

終点についても，どちらか一方に終点を示す文がある場合には，新しく終点の文を生成したり，終点の文を無条件変更文に置き換えたりする必要がある．

5.2　制御の変更の処理

(1)　無条件変更分の変換

ここで問題になるのは文のＩＤの変換であるが，一般に新しい文のＩＤを生成する場合も含めてスコープの問題と関係が深いので，文のＩＤの変換および生成に関してはスコープ処理および名前の生成ルーチンにまかせている．

(2)　条件付変更文の変換

Ｌi に条件付変更文があり，Ｌj にない場合には変換不能である．従って，Ｌi，Ｌj 共に条件は変更文がある時に変換がおこなわれる．

条件付変更文の変換は一般に次の3つの部分から成る．①文のＩＤの生成，②条件文の変換，③Ｌi，Ｌjの少なくと

もどちらか一方で，制御の変更がその文の中でおこなわれる場合には，実行すべき部分の変換又は生成．

Li とLjの制御の変更表を比較し，条件が異なる場合は条件部の変換がおこる．これは，場合によっては複数個の条件文に変換されることもある．（例えば，条件が論理式のものから比較式または算術式のものへの変換など）

5.3　繰り返し文の処理

大きく分けると，繰り返し文をもつもの（Li）からもたないもの（Lj）への変換，Li，Lj共に繰り返し文をもつ場合の変換の2つの場合がある．前者の場合に，Lj が条件付変更文，無条件変更文を持っていれば変換可能である．

繰り返し文の変換は，繰り返しの制御部分の生成，繰り返しの範囲の変換，文のＩＤの生成の3つの部分から成る．

6. スコープについて

(1) パージングでのスコープ処理

スコープは，セマンティックス表（スコープ表）のブロックを示す記号列がプログラムに現われたときその深さが決定される．

文のＩＤ名，変数名のスコープはソースプログラムＩＤ表（図7参照）のスコープの深さを示す値を参照することにより，従来のコンパイラと同じようなやり方で決定される．

(2) 目的プログラムの生成でのスコープ処理

目的プログラムにおけるスコープも目的プログラムのスコープ表を参照することによりブロックを示す記号列を生成するとき決定しなければならない．

ソースプログラムに現われる名前に対応する名前を生成する場合には，ソースプログラムでの名前は同じであっても異ったスコープをもつ名前に対しては，ソース言語も目的言語もブロック構造の場合を除いて，異った名前を生成する．名前を生成する際に，実際にそこで名前を確定することができないような場合が起り得るが（例えば，ソースプログラムに一対一に対応しない全く新しい制御変更のための文を生成するような場合，行先を表わす名前は一般には確定できない），そのときは目的プログラムＩＤ表（図7参照）へのポインタを名前のかわりに使って生成し，同時に目的プログラムファイルへのポインタに関する情報を目的プログラムＩＤ表に書きこんでおき，すべての文の生成が終了した後，もう一度走査し名前を確定する．

宣言の生成に関しては，ソース言語がブロック構造で，目的言語がブロック構造でない場合であって，宣言の必要があればプログラムの先頭に，必要がなければセマンティックス表の記述にしたがって名前を生成する．ソース言語がブロック構造でなく，目的言語がブロック構造の場合は1ブロックのみのプログラムとして生成する．宣言はすべての生成が終了した後，ブロックの先頭に宣言する．

7. 式について

式はパージングによりsuffix記法を用いた式に変換される．ソース言語の式がinfix 表現の場合は，セマンティックス表1に示された演算子の優先順位を用いてパージングする．ソース言語の式がprefix表現の場合には，式の右から左へと逆順に並べかえればsuffix表現が得られる．ソース言語の式がsuffix表現の場合は式に現われる名前だけをパージングすればよい．

式の中に現われる名前はセマンティックス表1を用いて，各々の記号の意味に従ってパージングし，どのセマンティックス番号（数や名前の種類）にあてはまるかを決定する．誤りがなければセマンティックス表の被演算子の型と比較し，一致すればパージングの過程で結果の型に還元される．一つの式が正しくパーズされたときは，パーズ表に｛名前｝

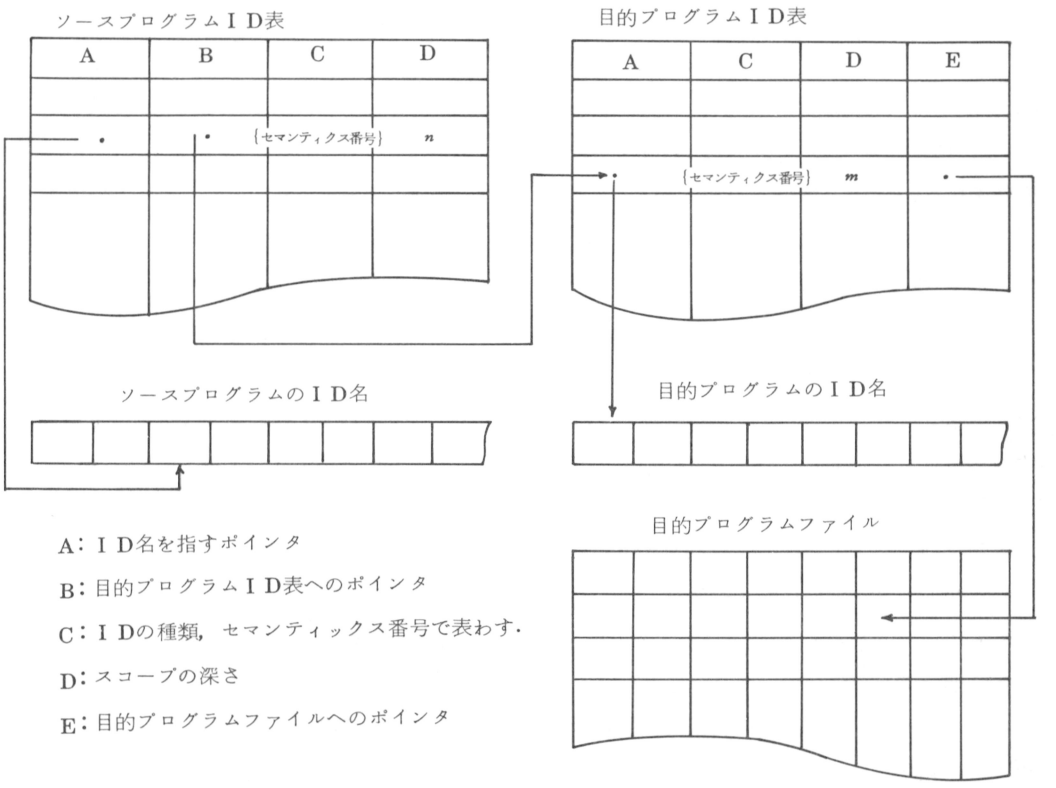

図７　スコープ処理で用いられるＩＤ表

｛名前｝，｛演算子｝…｛演算子｝がセマンティックス番号とソースイメージの両方で書かれる．

次にパージングの結果出力されたsuffix表現の式を目的プログラムのセマンティックス表に示された表現形式にしたがって生成する．目的言語の式がinfix 表現の場合は，suffixで表わされた式を，一番左側の演算子から右の演算子へと順次その左の名前を参照しながら生成する．名前は名前生成ルーチンとスコープ処理ルーチンにより生成される．演算子はセマンティックス表を参照して対応する演算子を生成する．その際，目的言語のセマンティックス表を参照して，次に現われる演算子より優先順位が低い場合は，セマンティックス表に示された優先順位変更のための記号を参照し，その記号でくくって生成する．目的言語の式がprefix表現の場合には，右から左へと逆順に生成してゆけばよい．

ソースプログラムに一対一に対応しないような全く新しい式を生成する必要があるときも名前や演算子を対応するセマンティックス番号を用いてsuffix表現で与えることにより同じような方法で生成することができる．

8. おわりに

われわれのトランスレータの概略についてのべた．このこゝろみはまだ実験の段階であって検討すべき多くの問題を含んでいる．特にセマンティックスの記述形式については今後実験を重ねて改良すべき点が多いと考えている．

本 PDF ファイルは 1965 年発行の「第 6 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html