Title	初心者におけるLOGOの学習過程
Author(s)	奥野，正義；上谷，宣正
Citation	北海道教育大学紀要．第一部．C，教育科学編，36(2)：61-69
Issue Date	1986-03
URL	http://s-ir.sap.hokkyodai.ac.jp/dspace/handle/123456789/5005
Rights	

初心者におけるLOGOの学習過程

奥野正義・上谷宣正

I．問　題

ここで検討しようとする問題は3つある．

1．教育と学習過程について

2．LOGOは，学習過程を分析するときに有効な学習材料となりうるか．

3．誤りは，学習においてどんな意味をもつのか．

1．教育と学習過程について

教育にとって大きな問題の一つは，学習者の学習過程である．学習者は，どのようにして新しい知識を既存の知識構造の中に同化し，それに合わせて教識構造を調節していくのであろうか．学習過程は，このような知識構造の同化と調節の一連の変化の現れとみることができる．良い教育をするためには，学習者がどのように学習を進めていくのか，その学習過程に関する理解が不可欠である．教育と学習は表裏一体の関係にあるといっていいだろう．

ところで，人間における学習をテーマとしてきたのは学習心理学であったが，そこでは，人間の頭の内部で知識がどのように学習されていくかという問題は直接扱うことを避けてきた．人間をブラックボックスとしてとらえ，入力する刺激とそこから出力する反応との間の関数関係を明らかにする形で学習を扱ってきた．人間をブラックボックスとして考えず，学習者が学習するとき，頭の中で生じていると思われる学習過程を直接扱おうとするのは，最近の認知心理学ないし認知科学（Cognitive Science）以後の問題意識と考えられる．そこでは，学習過程の問題は，次のようにいいかえられるだろう．

①　学習者がすでに持っている知識はいかなるものであるのか．その知識はどのように表現することができるのか．又，その知識の表現をコンピュータで実現するにはどのような方法があるのだろうか．

②　学習者の学習過程において，新しい知識はどのように獲得され，理解され，表現されるのだろうか．これをコンピュータでどのように実現すればよいのか．

③　既存の知識は，新しい知識によってどのように調節されるのだろうか．これは，コンピュータによってどのように表現されるのだろうか．

2．LOGOについて

本研究では，学習材料としてLOGOを用いた．LOGOは，コンピュータ言語の一つであり，1978年にMITのSeymour Papartらによって開発された．パパート自身が語っているように，Piaget

の発生的認識論の影響のもとに，子どもが使えることを意図してつくられ，又，数の処理にとどまらず，機械により知的な処理をさせるための機能を備えた教育用言語である．[1]

LOGOの特徴を挙げておく．[2]

①高度にインタラクティブ（対話的）な言語である．
②モジュラー型のプログラミング言語である．
③リスト処理を基本命令として持つ．
④タートル・グラフィックスを持つ．
⑤再帰的に手続きを定義できる．
⑥ローカル変数を持つ．
⑦インタープリタをコマンドとして持つ．

他のプログラミング言語にくらべて，LOGOは，広い年齢にわたって習得しやすい言語である．大人，学生はもちろんのこと，子どもでも習得可能である．障害児に対してLOGO学習を試みた例もある．[3]

これは，LOGOがタートル・グラフィックと手続き型言語という特徴を持っているためであろう．又，1つのコマンドを入力すると，その実行結果がタートルの動きとしてフィードバックされ，意図していた結果かどうかすぐに分かるという優れた点もある．このシングルステップでの実行結果の確認の容易さは，我々が学習過程を調べるために学習材料としてLOGOを選んだ理由でもある．

LOGOは，LISPから発展した言語であるので，リスト処理が可能な命令をもち，再帰的に手続きを定義することができる．このため，LISPやPROLOGで書かれている人工知能的なプログラムをLOGOで書くことも可能である．IntelligentなCAIや学習者の学習過程をコンピュータでシミュレーションするような学習過程のモデル作成にも役立つであろう．

LOGOのようなコンピュータ言語を用いて学習過程を調べるメリットは何であろうか．

学習過程は，前にも述べたように，知識の同化と調節の一連の変化の現れと考えられる．学習過程を明らかにするためには，学習のプロセスができるだけ細かく，1ステップずつ明確になるとよい．各ステップで，新しい知識がどのように習得され，既存の知識がいかに変化していくかを細かく調べていくことによって，知識の同化と調節の問題が明らかになっていくのではないか．又，各シングルステップでの実行結果の確認の容易さという条件も満たす必要がある．LOGOは，インタープリタであり，タートル・グラフィックを持つので，上記の2つの点をみたしていると考える．

3．学習者の誤りについて

結果を分析する観点としては，特に学習者が犯す誤りに注目したい．

一般に，テストで誤答したり，質問に答えられなかったりすることは，望ましいこととはされていない．誤りは学習者の知識の不完全さを示すものであり，誤りからなんらかの学習効果が得られるとは考えられていない．しかし，現在のCAIや認知科学の分野では，学習者が犯す誤りの持つ意味は重要視されている．[4，5，6]

これは，学習者の誤りが気まぐれや不注意によるものばかりでなく，誤りの中には学習者なりの論理があって，そのために一貫してまちがう場合もあること，そして，それはその対象分野に関する学習者の知識を反映していることが明らかになってきたためである．

II. 方　法

1. **被験者**：北海道教育大学幼稚園課程の学生
   男 9 人，女 15 人計 24 人．全員，コンピュータに触るのは全く初めてである．但し，今回分析の対象としたのは，このうちの 4 人（男 2 人，女 2 人）のデータである．これは，学習過程を分析するために，各個人の学習記録をすべてプリンターに打ち出す必要があったが，プリンターの数が限られていたためである．
   被験者　T：男　21 歳
   被験者　A：女　20 歳
   被験者　S：女　20 歳
   被験者　K：男　20 歳
2. **学習材料**：LOGO は市販のものを利用した．(ACCESS LOGO PC−8801 版)初心者用の LOGO のテキストを用意した．LOGO のテキストの内容と学習する順序，初出のプリミティブを表 1 に示す．
3. **実験期間**：昭和 60 年 4 月－5 月．1 週間に 1 回ずつ，連続して 5 週間おこなった．制限時間は特に設けないで，自由に好きなだけコンピュータを操作させた．
4. **実習場所**：北海道教育大学函館分校　共同利用電算機ルーム．
5. **手続き**
   ①実験に参加した学生は，今回初めてコンピュータに触れる者ばかりであった．そこで，まず，キーボードに慣れるために既存のキーボード練習プログラムで練習をさせた．あわせて，コンピュータの立ち上げ方，電源の切り方，フロッピーディスクの扱い方等基本的な操作方法を説明した．時間は約 1 時間である。
   ②LOGO のテキストを用意した．4 つの章からできている．その日に学習すべき部分のテキストをわたし，学習者はそのテキストに沿って学習を進めていく．さらに，その日のテキストを終えると課題が与えられる．実験者は，学習者に対しできるだけ助言を与えないようにした．

表 1　LOGO のテキストの内容と初出のプリミティブ

セッション 0　キーボードの練習
セッション 1　タートルを動かすために必要な基本的プリミティブを学ぶ。
課題：逆三角形を 2 つ平行に描く。
[SHOWTURTLE, FORWARD, BACK, RIGHT, LEFT, REPEAT, PENUP, WRAP, SETPC, CLEARSCREEN, FENCE, PENERASE, PENDOWN, WINDOW]
セッション 2　手続き（プロシジャー）の作り方を学ぶ。引数のつけ方，手続きの修正，格納，削除の仕方についても学ぶ。
課題：前回の課題をプロシジャーで定義する。
[TO　手続き名，引数，END, EDIT, OPEN, ERASE, SAVEALL, CLOSE, SAVE, LOAD]
セッション 3　円や半円を描くプロシジャーをつくる。
課題：にじをかくプロシジャーをつくる。
[HIDETURTLE, SETPOS, SETX, SETY, SETHEADING, LOCAL, MAKE]
セッション 4　再帰の考え方と再帰を利用した複雑な図形の作り方について学ぶ。
課題：顔を描く。 [再帰，IF−STOP]

表2　学習記録（被験者T セッション2のはじめの部分）

```
BB
? >  seih    seihoukei          to    TO SEIHOUKEI
>>   REPEAT  4 [FD 80 RT 90]                          ┐ 正方形のプロシジャーを定義
>>   END                                              ┘
SEIHOUKEI DEFINED
? >  SEIHOUKEI
UNDEFINED PROCEDURE NAME: REPEAT -- IN SEIHOUKEI ; エラーメッセージ（「定義されていない
? >  TO SEIHOUKEI                                                名前を用いた」）
SEIHOUKEI ALREADY DEFINED                          ; エラーメッセージ（「すでに定義されている名前を用いた」）
? >  ERASE  "SEIHOUKEI
? >  TO SEIHOUKEI                                   ┐
>>   REPEAT 4 [FD 80 RT 90]                         │ あらためて正方形のプロシジャーを定義
>>   END                                            ┘
SEIHOUKEI DEFINED
? >  SEIHOUKEI                                     ; 実行
? >  TO SEISANKAKU                                 ┐
>>   RT 30                                         │
>>   REPEAT 3 [FD 80 RT 120]                       │ 正三角形のプロシジャーを定義
>>   END                                           ┘
SEISANKAKU DEFINED
? >  SEISANKAKU                                    ; 実行
? >  SEIHOUKEI                                     ┐
? >  PENUP                                         │
? >  RT 90                                         │
? >  FD 100                                        │ 正方形の横に正三角形を書く
? >  LT 90                                         │
? >  PENDOWN                                       │
? >  SEISANKAKU                                    ┘
? >  CS                                            ; 画面クリア
? >  TO SEISA  IHOUKEI 1 : HEN                     ┐
>>   REPEAT 4 [FD : HEN RT 90]                     │ 変数をもつ正方形のプロシジャーを定義
>>   END                                           ┘
SEIHOUKEI 1 DEFINED
? >  SA  SEIHOUKEI 1 50  130                       ; 実行
? >  SEIHOUKEI 1 50                                ; 実行
? >  TO SEISANKAKU : HEN
SEISANKAKU ALREADY DEFINED                         ; エラーメッセージ（「すでに定義されている名前を用いた」）
? >  TO SEISANKAKU 1 : HEN                         ┐
>>   RT 30                                         │
>>  REPEAT 3 [FD : HEN RT 120]                     │ 変数をもつ正三角形のプロシジャーを定義
>>   END                                           ┘
SEISANKAKU 1 DEFINED
? >  SEISANKAKU 1 140
? >  SEISANKAKU 1 6 70
? >  SEISANKAKU 40
UNDEFINED PROCEDURE NAME: SEISANKAKU 40 ; エラーメッセージ（「定義されていない名前を
? >  SA  CS                                                          用いた」）
? >  SETPC 5                                       ; 色をつける
? >  TO BOX : TATE : YOKO                          ┐
? >  R FD : TAY TE                                 │
>>   RT 90                                         │
>>   FD : YOKO                                     │
>>   RT 90                                         │
>>   FD : TATE                                     │ 長方形のプロシジャーを定義
>>   RT 90                                         │
>>   FD : YOKO                                     │
>>   RT 90                                         │
>>   END                                           ┘
BOX DEFINED
? >  BOX    TO BOX ; :  : TAY TE : YOKO
BOX ALREADY DEFINED                                ; エラーメッセージ（「すでに定義されている名前を用いた」）
? >  BOX 1 100 150
? >  BOX  BOX 99 149
? >  BOX 98 148
? >  BOX 97 147
? >  ED  EDT  IT 2 "BOX                            ; BOX というプロシジャーをエディットする。
```

6．記録の仕方

学習過程の分析するために，各個人の学習の記録をすべてプリンターに打ち出した．但し，タートルが描いたグラフィックは出力されない．（表２参照）

Ⅲ．結　　果

1．学習時間

それぞれのセッションで学習に要した時間を表３に示す．但し，学習時間には，各セッションで学習者に与えた課題を解く時間も含まれている．

セッション毎の比較では，セッション４が一番時間がかかり，セッション１が一番少ない．又，セッション２と４の学習時間が多く，セッション１と３が少ないのは各セッションでの学習内容の難易度に差があるためと考えられる．セッション２ではエディターを使ってプロシジャーを書くことが要求され，セョション４では再帰という概念が導入される．両方とも初心者の学習者にとって初めてのやり方や考え方であるため，学習に時間を要したのであろう．

各セッション毎の最大学習時間をみると，セョション２から４までは被験者Ｔであり，最小学習時間ではセッション２から４まで被験者Ａになっている．これは，一種の学習のタイプを現していると思われる．被験者Ｔはじっくり学習を進めるタイプであり，被験者Ａはすばやく，ないし，要領良く学習を進めるタイプである．これは，LOGO を学習している時の様子の観察からもいえる．被験者Ｔはテキストの例を一度実行するとそれをいろいろ変えて試してみるのに対し，被験者Ａはテキストの例を１回実行しただけで次に進んでいくのである．

2．単位時間当たりに打ち込んだコマンド数

表４に各被験者毎の単位時間当たりに打ち込んだコマンド数を示す．この測度は１分間にどのくらい多くのコマンドを入力したかを示している．ここで，コマンドというのは，横一列に並べた命令のことを意味する．(表２参照) どの被験者も第１セッションで一番多くコマンドを入力している．又，第２セョション以降は，どの被験者もだいたい同じぐらいのペースでコマンドを入力している．第１セョションでコマンド数が多かったのはやさしくて簡単なプリミティブを学習したためであると考えられる．第２セッション以降は，考えながら学習を進めたために入力数が少なくなったのであろう．

表３　学習時間（分）

	1	2	3	4
T	45	135	92	195
A	55	118	70	113
S	75	120	73	139
K	38	120	91	167
平　均	53.25	123.25	81.5	153.5

表４　単位時間当たりに入力したコマンドの数

	1	2	3	4	平　均
T	3.0	1.04	1.05	1.6	1.67
A	1.78	1.09	1.06	1.48	1.35
S	2.26	1.11	1.0	1.39	1.44
K	3.48	0.96	0.75	0.83	1.38
平　均	2.72	1.05	0.97	1.33	

3．誤答について

3－1．誤答率

図1は，各被験者の誤答率をグラフ化したものである．この誤答率とは，エラーメッセージをだした命令を誤答とし，それをプリミティブとプロシジャーとを足した数で割ったものである．図1を見ると各被験者とも回を追うごとに誤答率が減少するような傾向はみられない．むしろ，各回の学習内容の難易度の差が誤答率に大きな影響を与えたと考えられる．確かにセッション2ではプロシジャーと変数の定義の仕方，セッション4では再帰と日常生活とはなじみのうすい考え方が導入され，困難度が増している．

3－2．誤答の内容及び原因とその修正方法

エラーメッセージは，入力されたプリミティブないしプロシジャーがLOGOのシステムからみて解釈できない場合にだされる．したがって，エラーメッセージがでたものはLOGOを学習する際の誤答と考えられる．このエラーメーセージの内容となぜ誤答が生じたのかという原因，及び学習者がどのように原因を推測して修正したかを調べることによって，LOGOを学習していく際の過程が明らかになると考えられる．

表5は出現したエラーメッセージを頻度の多いものから順番にならべたものである．この順序で10ケ以上ある誤答の原因とその修正方法について詳しく検討する．

1）UNDEFINED PROCEDURE NAME（ワードがプロシジャーとして定義されていない）

エラーをおこした原因で最も多いのは，このエラーメッセージの本来の意味である「ワードがプロシジャーとして定義されていない」場合である（48.4%）．次に多いのは，「RT 90のように英字と数字の間のブランクをあけなかったために新しい名前と解釈されてしまった」場合（36.8%）で，最後は「RRRIGHTのようなタイプミス」の場合（14.7%）であった．ころらは，ほとんど単純ミスと考えられる．

被験者は，これらのエラーをどのように修正しているだろうか．エラーを無視して次へ進んだの

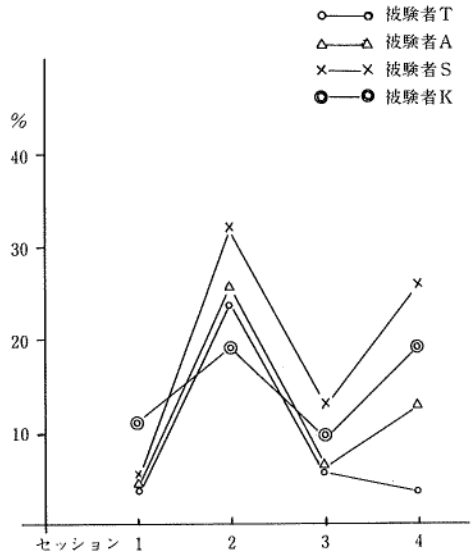

図1　誤答率の推移

表5　出現したエラーメッセージとその頻度

1.	UNDEFINED PROCEDURE NAME	95
2.	～ ALREADY DEFINED	30
3.	INPUT EXHAUSTED	26
4.	I DON'T KNOW WHAT TO DO WITH ～	16
5.	～ HAS NO VALUE	10
6.	～ ISN'T TRUE OR FALSE	3
7.	～ DIDN'T OUTPUT	2
8.	ILLEGAL PROCEDURE-DEFINITION	2
9.	) EXPECTED	1
10.	～ ISN'T A PROCEDURE	1
11.	～ ISN'T A LIST	1
12.	～ FILE NOT FOUND	1

は 8.4%で，大部分の場合(91.6%)は，修正をしてから再度試みて正しいかどうかを確認している．

2）～ALREADY DEFINED（すでにプロシジャー名が定義されている．）

このエラーメッセージは 3 つの場合に出現している．1 つは，すでに定義してある同じ名前でプロシジャーを定義した時である（73.1%）．次は，LOAD をした場合である（23.1%）．最後は，あるプロシジャーを定義する際に誤ってその中で別のプロシジャーを定義しようとした場合(0.04%)である．これらの誤答の原因は 3 つに分けられた．

①知識の欠如（53.8%）
②理解不足　（34.6%）
③単純ミス　（11.5%）

又，46.2%がエラーを無視して次に進み，53.8%は正しい修正をして再度確認している．エラーを無視して次に進んだのは，なぜ誤答になったのかとかエラーメッセージの意味が分からないなど知識の欠如によるものと考えられる．

3）INPUT EXHAUSTED（入力がたりない）

このエラーメッセージが出現するのは，80.8%が「LT など引数が必要な時に入力していない」場合である．逆に「EX 1 などのように引数を定義していないのに入力した」場合が 3.8%ある．残りは，なぜこのエラーメッセージが出現したのか不明の場合で 15.4%ある．原因をみてみると，

①知識の欠如（7.7%）
②知識の適用の誤り（3.8%）
④単純ミス（57.7%）
⑤不明（19.2%）

エラーを無視して次に進んだ場合が 30.7%．正しい修正をして確認をしているのが 69.2%である．

4）I DON'T KNDW WHAT TO DO WITH～(オブジェクトをどう処理するかわからない)

今までのエラーメッセージは，原因が比較的単純で，修正することも簡単であった．しかし，このエラーでは，原因が初心者にとって複雑で，修正も試行錯誤を繰り返している．そこで，一つひとつを詳しく検討してみることにする．

1：K－1（これは，被験者Tがセッション 1 でこのエラーをだしたことを示す．以下同様である．）

[原因]　PENDOWN　100 と入力してこのエラーをだした．PENDOWN では引数は必要がない．にもかかわらず，100 と引数を付け加えたためにエラーとなった．FORWARD や RIGHT では，長さや角度を示すために引数が必要であるが，この規則を拡大して適用したことが原因と考えられる．(知識の適用の誤り)

[修正]　次で正しく修正して結果を確認している．

2：A－2

被験者Aは，このエラーを続けて 3 回だしている．そのたびにいろいろ工夫しながら修正を試みている．問題となったのは，HOUSE というプロシジャーをつくることであった．これについて少し説明をしておくことにする．HOUSE は 4 つのプロシジャーで構成されている．(表 6 参照)HOUSE は，WALL と ROOF の 2 つのプロシジャーを呼び出している．又，WALL は BOX とい

表6　プロシジャーHOUSEの持つ構造

```
TO HOUSE : WIDTH
WALL : WIDTH
ROOF : WIDTH
END

TO WALL : WIDTH
BOX : WIDTH * 2/3 : WIDTH
END

TO ROOF : WIDTH
PENUP
FORWARD : WIDTH * 2/3
PENDOWN
SEISANKAKU : WIDTH
END

TO BOX : TATE : YOKO
REPEAT 2 [FORWARD : TATE RIGHT 90 FORWARD : YOKO RIGHT 9(
END

TO SEISANKAKU : HEN
RIGHT 90
REPEAT 3 [FORWARD : HEN RIGHT 120]
END
```

うプロシジャーを，ROOFはSEISANKAKUというプロシジャーを呼び出す．つまり，HOUSEは4つのプロシジャーが完成した時，はじめてエラーなしに実行される．このように，できるだけ小さな単位に分けてプログラムし，それらを積みかさねるように作っていくのはLOGOの特徴なのだが，初心者には理解が困難であるように思われる．

2－1：

［原因］ HOUSEで呼び出すROOFの中で，：WIDTHと2/3の間に掛け算の印である＊を忘れたためにエラーが生じた．これは単純ミスであろう．

［修正］ EDITでROOFをみたが，修正をしていない．

2－2：

［原因］ 再度試みたが，修正をしていないため同じエラーが起きた．これは，エディターで修正する方法がよくわからなかったためではないか．しかし，すでにその方法は学んでいるので，理解不足による誤りであろう．

［修正］ 間違いを修正した新しいプロシジャーROOF1を定義した．

2－3：

［原因］ ROOF 1は正しく定義されているが，ROOF 1をHOUSEに組込んでいないので同じエラーがおきた．エディターで修正する方法がわからなかったので，新しくプロシジャーを定義するという別の方法を用いたが，それをHOUSEに組込むことをしなかった．これは，その必要性を知らなかったという知識の欠如による誤りと考えられる．

［修正］ HOUSEそのものを新しくHOUSE 1に定義しなおした．今度は成功．

被験者Aのエラーは，HOUSEというプロシジャーが持っている構造を理解していないために起きたと考えられる．同じような誤りは，他の2人の被験者にもみられた．

5）～HAS NO VALUE（～が値をもっていない）

このエラーメッセージは，プロシジャーの中で引数を定義していないのに，実行する時RINKAKU 10のように引数を入れたりすると起きる．これらの誤答の原因は次のようであった．

①理解不足　　10%

②単純ミス　90%

修正に関しては，修正をせずに次に進んだのが 10%，修正をして再度実行し確認をしたのが 90% である．

IV. 考　察

「学習者は，どのようにして新しい知識を既存の知識構造に同化させ，それにあわせて知識構造を調節していくのか」というこが基本的問題であった．これを明らかにするために，LOGO を学習材料として用いて初心者の学習過程について検討した．学習過程は，知識構造の同化と調節の一連の変化の現れととらえることができる．LOGO は，シングルステップの実行結果の確認の容易さという特徴をもっているが，これによって，同化と調節の一連の変化を調べることができると考えた．これは，特に誤答の原因と修正を分析することによって明らかにすることができた．たとえば，

「被験者Aは，掛け算の印である＊を忘れるという単純ミスで誤答になった．彼女は初めエディターを使って修正しようとしたが，その方法をよく理解していなかったので，次のステップで ROOF 1 という名前で新しく定義した.」これは，新しい知識であるエディターによる修正を試みたが（調節）失敗し，今までの知識を用いて新しく名前を定義した（同化）と解釈することができる．このような誤答の分析が可能になったのは，学習材料として LOGO を用いたからである．

誤答の分析で注目すべきもう一つの点は，学習者の誤答に対する修正方法の正確さである．今回の実験ではエラーメッセージの説明はしていないし，また，学習中において学習者を援助することは極力避けてきた．それにもかかわらず，多いもので 91.6% (UNDEFINED PROCEDURE NAME)，少ないものでも 53.8% (～ALREADY DEFINED) の修正が正しかったことである．これは，学習者がなんらかの手掛かりからエラーの意味を推測したのではないかと考えられる．学習者は，正答よりも誤答からさまざまの情報をとりこんでいる可能性がある．誤答のもつ効果を明らかにすることが今後の課題である．

引 用 文 献

[1] パパート，S. 著，奥村貴世子訳　1982　「マインドストーム」　未来社

[2] 上谷宣正，奥野正義　1984　LOGO による教育実践(1)―理論的検討と予備実験―北海道教育大学人文論究　第 44 号　pp.95－107

[3] 上谷宣正，奥野正義，山崎正吉　1985　障害児への LOGO の適用　信学技報　Vol.85 No.80 pp 9－12

[4] 平賀譲　1985　エラーの認知モデル　数理科学　No.266 pp 18－23

[5] 溝口理一郎，豊田順一　1985　ICAIと認知科学　Computer Today Vol. 1 (5) pp 32－38

[6] 野島久雄　1982　手続き的バグの診断・生成・除去　波多野誼余夫（編）認知心理学講座第 4 巻　学習と発達　第 6 章III　東京大学出版会

[7] Access　LOGO　1984　Guide Book & Reference Manual　日本ソフトバンク

（奥野正義　本学講師　函館分校・上谷宣正　本学助教授　函館分校）