データベース・システム 65－1
（１９８８．５．１９）

DBMS上のスキーマ診断システム GSC

黒瀬博靖，安次富伸浩
（株）リコー
ソフトウェア研究所

データベース管理システムＧ－ＢＡＳＥの核はＤＭ－Ｌｉｓｐなる Ｌｉｓｐインタプリタである。本論文では，このＤＭ－Ｌｉｓｐ上に構築したルール駆動型プロダクションシステムＧＳＣ（G-BASE Schema Consultant）の概要を述べる。ＧＳＣはユーザが定義したデータベーススキーマを評価し，それを改善するための助言をおこなうことができる。この目的のため，ＧＳＣには，Ｇ－ＢＡＳＥマニュアルの内容に相当するスキーマ設計のための知識を搭載した。ＧＳＣはＤＢＭＳの核を兼ねるＬｉｓｐの上で動作しているため，データベースを参照するプロダクションルールを扱うことが容易である。

GSC - A Schema Consultant on a DBMS

Hiroyasu KUROSE, Nobuhiro AJITOMI

Software Research Center
RICOH Co., Ltd

1-1-17 Koishikawa, Bunkyo-ku, Tokyo 112, Japan

G-BASE is a DBMS whose kernel is a Lisp interpreter named DM-Lisp. This paper describes a rule-based production system GSC (G-Base schema Consultant) implemented on DM-Lisp. GSC can examine a database schema made by a user and suggest the user how to refine it. GSC contains knowledge for schema design written in one of G-BASE manuals. Since GSC is implemented on DM-Lisp, a DBMS kernel, it is easy to handle production rules that access to databases.

1．はじめに

当リコー・ソフトウェア研究所で開発したＧ－ＢＡＳＥ〔文献１〕は，関係データモデル〔文献２〕を拡張したグラフデータモデル〔文献３〕に基づく独特のＤＢＭＳである。このＧ－ＢＡＳＥのカーネルは ＤＭ－Ｌｉｓｐ（Data Manager Lisp）と呼ばれるＬｉｓｐのインタプリタになっている。ＤＭ－ＬｉｓｐはＣｏｍｍｏｎＬｉｓｐに似た言語仕様を持ち，データベースにアクセスする機能を備えたＬｉｓｐシステムである。Ｇ－ＢＡＳＥの一部（検索時のオプティマイザなど）はＤＭ－Ｌｉｓｐで記述されている。

我々は，このＤＭ－Ｌｉｓｐ上に，Ｇ－ＢＡＳＥ内のデータベーススキーマを診断・評価するためのエキスパートシステム，ＧＳＣ（ G-Base Schema Consultant）を構築した。ＧＳＣは与えられたデータベーススキーマを調べ，そのスキーマの問題点を指摘することができる。

ＤＢＭＳ分野におけるＡＩ技術の応用は，自然言語による問い合わせに関するものが中心であるが，ユーザのデータベース設計を支援するシステムもいくつか提案されている。たとえば〔文献４〕では，スキーマの論理的構造を対話的に作成・改善する方法を論じている。〔文献５〕や〔文献６〕では予想される問い合わせからこの種の構造を自動的に生成しようとしている。また〔文献７〕のように，ＤＢＭＳ専門家の行動をシミュレートする知的システムを構築しようとする試みもある。より実際的な効果をねらったものとしては，すでに存在するスキーマの評価と改善提案をおこなうシステムがあり，この例としてはＣＯＤＡＳＹＬ型ＤＢＭＳを対象とした〔文献８〕があげられる。

我々のＧＳＣは〔文献８〕と同様のアプローチをとっている。ＧＳＣの特徴は，第一に（前述のようにＤＢＭＳを兼ねるＬｉｓｐシステムの上に実現されたため）ＤＢＭＳとの結合が自然な形でなされていること，第二にユーザマニュアルと並行に開発されマニュアルと同等の知識（その中にはグラフデータモデルに特有の知識も含まれている）を搭載していることである。本稿では，このＧＳＣの概要について報告する。

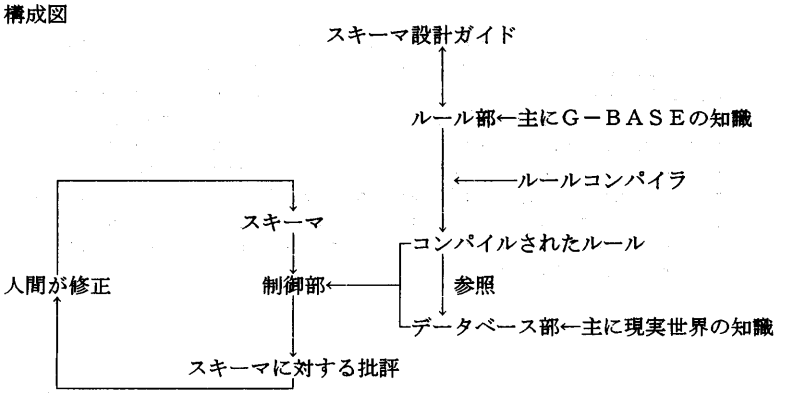
2．開発の意図

我々がＧＳＣの開発によって意図したのは，第一にデータベース設計者の支援である。データベースの設計，特にスキーマの作成は，ＤＢＭＳを導入し利用を開始する際に避けては通れないことだが，この作業は適用業務とＤＢＭＳの双方に関する詳細な知識を必要とする。ＧＳＣの目標は，この種の作業を助け，かかる労力をいくらかでも軽減することである．

特にＧ－ＢＡＳＥの場合は，グラフデータモデルという他のＤＢＭＳには見られないデータモデルを採用していること（具体的には「リンク型」という独特のデータ型を持つ），便利ではあるが濫用するとシステムに非常な高負荷をあたえる機能が備わっていること（例えば，フィールドひとつの中に数十Ｋｂｙｔｅにもおよぶデータを格納できる，等々）などの理由により，計算機資源を効率よく使うスキーマの作成には，ＤＢＭＳ一般に関する知識のほかにＧ－ＢＡＳＥシステム固有の知識が必須である。我々は，ＧＳＣにこのようなＧ－ＢＡＳＥ独特の知識を搭載し，データベース設計者のＧ－ＢＡＳＥに関する知識・経験不足をできるだけ補うことを試みた。

ＧＳＣ作成のもうひとつの狙いは，「マニュアルレス」システムの実験である。前述のように，Ｇ－ＢＡＳＥのスキーマ設計にあたって知っておくべきことは多いので，Ｇ－ＢＡＳＥのマニュアルの一部としてデータベース設計者のために「Ｇ－ＢＡＳＥスキーマ設計ガイド」〔文献１１〕が用意されている。これは確かにユーザのための配慮ではある。しかし，読むべき文書が一冊増えているわけであり，結局ユーザの負担を増やしていることになる。もっとも望ましいのは，マニュアルを読まずとも使うことができる，いわゆるマニュアルレス・システムであろう。我々は，マニュアルの内容を知識ベース化することにより，Ｇ－ＢＡＳＥをマニュアルレス化する実験をおこなった。ＧＳＣはその実験の成果であり，このシステムの中には前記「Ｇ－ＢＡＳＥスキーマ設計ガイド」の内容に相当する知識が含まれている。

構成図



なお，ＧＳＣの作成には，ＧＳＣと並行して（やはりＤＭ－Ｌｉｓｐ上に）開発された　ＧＩＥ（G-BASE Inference Engine）システムを利用した。ＧＩＥは，ｉｆ－ｔｈｅｎ型のプロダクションルールを実際に前向き推論をおこなうための（ＤＭ－Ｌｉｓｐの）プログラムに変換する，ＲＥＴＥアルゴリズム〔文献９〕に基づくルールコンパイラである。ＧＩＥが用いているＲＥＴＥアルゴリズムには若干の拡張が加えられており，ルールのｉｆ部に限量記号（全称∀及び存在∃）つきの条件式を書くことが可能になっている。

3．システム構成及び入出力

ＧＳＣはルール部と制御部，そしてデータベース部から構成されている。

まずルール部は，Ｇ－ＢＡＳＥスキーマ設計ガイドの内容をｉｆ－ｔｈｅｎルールに直したものである。この部分はルールコンパイラＧＩＥのためのルール記述言語（Ｌｉｓｐの式に近い形式）で書かれている。なお，ＧＩＥ自身はＤＭ－Ｌｉｓｐのプログラムである。

データベース部はアプリケーション（Ｇ－ＢＡＳＥを適用しようとしている業務）に関する知識を格納したデータベースを参照する部分であり，ＤＭ－Ｌｉｓｐで書かれている。いくつかのルールのｉｆ部ではこのデータベース部を参照している。

制御部はＧＳＣのメインルーチンにあたる。ルール部をＧＩＥに与えると，推論のための一連のＤＭ－Ｌｉｓｐ関数が得られるが，これらの関数を呼び出してスキーマ情報を与え，スキーマに関する批評を出力するのが制御部である。この部分はＤＭ－Ｌｉｓｐで書かれている。

ＧＳＣの通常の使い方は次のようになる。まずデータベース設計者がＧ－ＢＡＳＥのデータ定義言語〔文献１２〕を用いてスキーマを作成する。また，業務に関する知識を「カテゴリ」と呼ばれるかたちでＧＳＣのためのデータベースに入力しておく必要がある。（この情報はＧＳＣデータベース部が参照する。）この状態でＧＳＣを動かすと批評メッセージが得られる。この批評にしたがってスキーマを修正した後，再度ＧＳＣを起動する。一般には，満足できるスキーマが得られるまで，この「修正→ＧＳＣ起動」をくりかえすことになる。

以下に続く各節では，制御部，ルール部，データベース部のそれぞれについて説明する。

4．制御部

スキーマ評価のためのｉｆ－ｔｈｅｎ形式のルールをルール・コンパイラＧＩＥに与えると，ＯＰＳ５〔文献９，１０〕と同様の前向き推論を行なうための一連のＤＭ－Ｌｉｓｐ関数が得られる。このＧＩＥが生成したＤＭ－Ｌｉｓｐコード（この部分は現在約３００個のＤＭ－Ｌｉｓｐ関数から成っている）にスキーマ情報を与え，実際に推論をおこなうのがＧＳＣの制御部である。制御部は比較的単純な部分で，実際にＧＳＣの振舞いを決めるのは制御部自身ではなくＧＩＥが生成したコードの方である。そこで，以下ではＧＩＥについて論ずることにする。

ＧＩＥで用いているＲＥＴＥアルゴリズムには，条件部に通常の述語以外に全称記号（∀）や存在記号（∃）を書けるように拡張を施してある。〔文献１４〕この拡張のおかげで ルール作成者が一階の述語論理で書ける条件をより自然に記述できるようになり，Ｇ－ＢＡＳＥスキーマ設計ガイドの内容をルール化する作業が大変楽になった。

例１

```
(r r1 r2 f1 f2)
(
  (sco-is-record r) (sco-is-record r1) (sco-is-record r2)
  (equal (sco-no-of-field r) 2)
  (not (eq r r1)) (not (eq r r2))
  (sco-is-field f1) (sco-is-field f2)
  (sco-field-in-record f1 r) (sco-field-in-record f2 r)
  (not (eq f1 f2))
  (equal (sco-subcat f1) (sco-subcat (sco-key-of r1)))
  (equal (sco-subcat f2) (sco-subcat (sco-key-of r2))))
(
  (sco-mesg1 "レコード" r "は" r1 "と" r2 "の間のリアルリンクにしましょう。")
  (sco-mesg2 "レコード" r "に属するフィールドは" r1 "と" r2 "の主キーと"
        "同一カテゴリに属していますから，このレコードは" r1 "と" r2
        "の関係を表現する為に作られたものだと推定されます。"
        "そのような場合はリアルリンクを使用をお勧めします。"))
```

例１にルールの例を示す。ここで，最初のリストがこのルールの中に現れる自由変数の宣言，次のリストが条件（ｉｆ部），最後のリストが条件が成立したときに実行されるべきアクション（ｔｈｅｎ部）をそれぞれ表わしている。例１の意味するところは「フィールド二つから成るレコードでその各々が他のレコードの主キーになっている場合，そのレコードにはいる情報は実リンクで実現すべきである」ということで，これは実リンクの使用法に関するもっとも基本的な戦略である。

ルールのｉｆ部における原子論理式は，ＧＳＣ制御部実行の際に，そのままＤＭ－Ｌｉｓｐインタプリタによって評価される。したがって，ＤＭ－Ｌｉｓｐの関数で記述できることはすべてルールのｉｆ部に書くことができる。ＤＭ－Ｌｉｓｐにはデータベース（スキーマ情報が格納されているデータ辞書を含む）にアクセスするための関数がいくつか用意されているが，それらを自由にルールのｉｆ部で呼び出せることになる。この機能は，今回のシステムのようにデータベースを参照するプロダクションルールを多用する場合には，大変有用である。また，論理式で書くと煩雑すぎるような条件については，適当な（その条件を評価するための）ＤＭ－Ｌｉｓｐ関数を定義し，ルールのｉｆ部からその関数を呼び出すようなことが可能であり，実際ＧＳＣではこのような使い方をしている。

ＧＩＥはルールをコンパイルする時に，ＯＰＳ５同様，ｉｆ部を評価するプログラムの共有化を図っている。　すなわち，いくつかのルールのｉｆ部が（あるいはひとつのルールのｉｆ部中であっても）共通な部分式を持つときには，その部分式を評価するコードを二重に生成しないようにしている。そのほか，最適化のための工夫をいくつかしているが，生成されるＤＭ－Ｌｉｓｐプログラムの効率は，現状では，残念ながらあまりよいものとは言えず，なお改善の余地が残っている。ＧＩＥの改良については現在検討中であるが，通常の最適化のほかに，個々の原子論理式の成立しやすさや，評価のためのコストを考慮にいれて論理式の評価の順序を決定することも考えている。〔文献１５〕

最後に，ルール間の競合解消の方法について述べておく。同時にふたつ以上のルールのｉｆ部が成立した場合，その中から実行すべきルールをひとつ選ぶために競合解消の戦略が必要となる。この戦略に関してはＬＥＸやＭＥＡ〔文献１０〕などがあるが，ＧＩＥでは，各ルールにあらかじめ優先度を与えておき，優先度が高いほうを採用するという単純な方法をとった。具体的には，優先度は自然数で与える（より小さい数がより高い優先度を表わす）仕様になっている。このような方法をとった場合，ルール作成者が各ルールの優先度を定めなくてはならず，特にルールが多いときにはルール作成作業時の手間が大幅に増えてしまうおそれがある。幸い，ＧＳＣのルールセットは小規模なものであったため，あまり問題は生じていない。むしろ，ルール作成者がＧＳＣの動作を細部まで制御できるという効果のほうが大きかったようである。

5．ルール部

現在ルール部は約３０個のルールから成っている。このルールには「Ｇ－ＢＡＳＥスキーマ設計ガイド」〔文献１１〕に書かれている内容が記述されている。

このルールによって出力される批評は以下のように分類される。

1　データベースの正規性に関する勧告
2　フィールドに索引をつけることに対する勧告
3　フィールドの型に関する勧告
4　実リンクに関する勧告
5　仮想リンクに関する勧告
6　レコード型の重複に関する勧告
7　レコード型の主キーに関する勧告

実リンクと仮想リンクはＧ－ＢＡＳＥに固有のものである。リンクはレコード型間における二項関係である。リンクの概念及び使用法については〔文献１，３〕に詳述されている。

例２

```
(r f)
(
  (sco-is-record r) (sco-is-field f)
  (sco-field-in-record f r)
  (sco-virtuallinked f) (not (sco-indexed f)))
(
  (sco-mesg1 "レコード" r "のフィールド" f "には索引を付けましょう。")
  (sco-mesg2 "仮想リンクの両端のフィールドには索引を付けたほうがよいのです。"))
```

Ｇ－ＢＡＳＥに特有なルールとしては例２がある。これは仮想リンクの両端のフィールドには索引をつけるべきであるというルールを表現している。また例３のようにＤＢＭＳ一般について適用出来るルールもある。これはあるレコードの全てのフィールドが無制限可変長の場合にメッセージを出力する。全てが無制限というのは設計者の怠慢を意味する場合が多いからである。例３の４行めに出現する“ａｌｌ”は全称記号（∀）を表現している。この条件を論理式で書くと次の形になる。

```
sco-is-record(r)∧
((∀ f)
  ((sco-is-field(f)∧
    sco-field-in-record(f,r))
      ⇒ (sco-length-type-of(f) = UNLIMITED)))
```

sco-is-record(r): r がレコードであるという述語
sco-is-field(f): f がフィールドであるという述語
sco-field-in-record(f,r): f が r のフィールドの一つであるという述語
sco-length-type-of(f): f の長さのタイプ属性を得るための関数
UNLIMITED:無制限可変長を表わす定数

データベース部にある情報を用いて起動されるルールもある。データベース部には使用されることの多いフィールド（人名，住所，など）の情報が入っている。データベース内の情報はカテゴリと呼ばれており，カテゴリ名という名前で参照される。カテゴリ名はフィールドの一般的な意味や属性を指示するために用いられる。ＧＳＣを用いるユーザはスキーマの各フィールドに対しカテゴリ名を一つ与える。ユーザはフィールドの現実との対応をある程度は知っているため，「ｐｅｒｓｏｎレコードのｎａｍｅというフィールドは人の名前である」というようなことを指示できる。ＧＳＣはこの指示を用いてデータベース部にアクセスし，現実世界の情報を引き出す。データベース内の情報によって，たとえば長さは２文字以上で１０文字以下であり，文字型で検索の対象となりやすいということがわかる。この情報を用いてルールが選択され，このフィールドに対する批判が出力される。またこのカテゴリからの情報によってデータ独立性もチェックする。カテゴリを用いたルールとしては例４がある。フィールドの属するカテゴリに選択性と検索性が書かれている。Ｇ－ＢＡＳＥでは無制限可変長のフィールドには索引をつけることが出来ない。例４は，選択性と検索性の高いフィールドは長さを制限して索引を付けるべきである，というルールを表現している。

例３

```
(r)
(
  (sco-is-record r)
  (all (f) (imply
             (and (sco-is-field f)
                  (sco-field-in-record f r))
             (eql (sco-length-type-of f) UNLIMITED))))
(
  (sco-mesg1 "レコード" r
             "のフィールド全てが無制限可変長というのはおかしくありませんか。")
  (sco-mesg2 "フィールドの長さをもう少し真面目に管理しましょう。"))
```

例４

```
(r f)
(
  (sco-is-record r) (sco-is-field f)
  (sco-field-in-record f r)
  (sco-search (sco-subcat f)) (sco-select (sco-subcat f))
  (equal (sco-length-type-of f) UNLIMITED))
(
  (sco-mesg1 "レコード" r "のフィールド" f
         "は長さを制限して索引を付けましょう。")
  (sco-mesg2 "このフィールドは選択性が良いうえに検索に使われる可能性が高いため，"
         "索引を付けることが出来れば検索の効率が上がると思われます。"))
```

6．データベース部

ここはカテゴリ毎の属性が表となって入る。データベースのスキーマの一般形がはいっているとみなして良い。各々の応用分野毎のスキーマの形態の差はここの知識の変更で吸収できる。ユーザが新しいカテゴリを導入することによって知識を増やすことができる。現在フィールドの属性として，ベースタイプ，長さの上下限，選択性，一意性，検索の条件のなりやすさ，等が設定されている。

この属性のうち，長さの上限などは本来あいまいなものであるが，現在はしきい値を決めてそれより長いものをチェックしている。このしきい値はアプリケーションの種類によって異なってくることが予想される。またしきい値そのものをファジー化することも検討中である。

カテゴリの例として人名のカテゴリを挙げる。この情報を用いて人名のフィールドに関する索引の付加や一意性のチェックを行う。

カテゴリ名	ｈｕｍａｎｎａｍｅ
ベースタイプ	文字型
長さ	２文字以上１０文字以下
一意性	なし
選択性	高い
検索の頻度	高い
親レコード	人間に関するレコード
キーへの成り易さ	非常に成り易い

7．マニュアル

「スキーマ設計ガイド」〔文献１１〕を書くことによってルールの整理も同時に行なった。このガイドはルール化を意識して書かれており，構成的にスキーマを作成する部分と，それをより良くするためのアドバイスの部分から成っている。文法やスキーマ作成の実際の作業は「応用プログラマガイド」〔文献１３〕に記述されている。設計者は〔文献１３〕を読んだ後に〔文献 １１〕 を読めばより良い設計が行なえるようになっている。

〔文献１１〕作成の上で留意した点は，ルールを作りやすいように内容を書くということである。マニュアルのアウトラインの作成法はいろいろあるが〔文献 １６，１７，１８〕，このガイドは ルールの種類ごとに章立てを行い，一つのルールに一節を費やした。作成したスキーマをチェックしたい人はＧＳＣをそのまま使用すればよく，Ｇ－ＢＡＳＥのスキーマ設計全般にかかわる知識を得たい人はマニュアルを読めばよい。ＧＳＣとマニュアルは相補的な関係にある。

ＧＳＣ作成はマニュアルレスの実験としては半分成功といえる。確かにマニュアルの内容は項目毎に全てルールにすることが出来た。しかしエキスパートシステムだけでは，ルールを学習することには困難が伴う。マニュアルを完全に廃止するためには，スキーマ作成を学習するためのシステムが必要と思われる。

8．今後の展開

本節では，ＧＳＣ／ＧＩＥの今後の課題について論ずる。まず，ルールコンパイラＧＩＥについてはルール記述法の強化やそれに伴う推論アルゴリズムの改良が必要である。

・現在のところ，ｉｆ部に書くことができるのは，一階述語論理における式の一部にすぎない。ｉｆ部の条件式として，より広範囲の論理式を扱えるようにすることが望ましい。

・変数のデータ型という概念を導入すると，ルールの読みやすさ，生成コードの実行効率ともに向上する可能性がある。どのような形で導入するかを現在検討している。

・またルールの二項関係としてルールの優先順位を規定した場合，その二項関係が常に線型順序になるのかどうかというのも問題である。順序関係にならない場合は現行の競合解消ルールを考え直す必要がある。

次にＧＳＣの今後の課題をあげる。現在のところ，ＧＳＣルール部のルールはすべてｉｆ部でデータ辞書内のスキーマ情報と「カテゴリ」データベース内のアプリケーション情報を参照し，ｔｈｅｎ部で評価メッセージを生成するという形をしている。しかしこれではＧ－ＢＡＳＥの機能を全て利用しているとは言えない。

・Ｇ－ＢＡＳＥはデータ自動再構成機能を持っているので，この機能をルールのｔｈｅｎ部で呼び出すことができれば自動的にデータベース再構築を行なえる可能性がある。この発展としてスキーマ自動修正システムが考えられる。

・カテゴリを与えなかった場合にもなんらかの形でアプリケーションに関する知識を獲得ないしは推論し，動作できることが望ましい。現在のＧＳＣはカテゴリ情報なしでも動作はするが，その場合にはデータの内容にまで言及したメッセージ（たとえば「ｘは人名ですからｙ文字をこえることはめったにないはずです」等々）は一切出力できない。

9．おわりに

今回の成果を総括する。我々はＤＢＭＳを兼ねるＬｉｓｐシステム DM－Ｌｉｓｐ上にデータベース設計を支援するエキスパートシステムＧＳＣを作成し，試用した。その際マニュアル「Ｇ－ＢＡＳＥスキーマ設計ガイド」執筆とＧＳＣ開発を連係して進め，互いに同等のマニュアルと知識ベースからなるユーザ支援体系を構築することができた。また，ＧＳＣ作成のためにルールコンパイラＧＩＥをＤＭ－Ｌｉｓｐ上に開発したが，その過程でＲＥＴＥアルゴリズムを拡張し，限量記号を持つ論理式を扱うことに成功した。

参考文献

1 “特集／スーパーリレーショナル・データベース・システム”，コンピュトロール１７号，コロナ社，１９８７

2 E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”, Communications of ACM, Vol.13, No.6, 1970

3 H. S. Kunii, “Graph Data Language: A High Level Access-Path-Oriented Language”, Ph.D Dissertation, Univ. of Texas, 1983

4 上林，古川，国分，黄，“モデル間の変換によるデータベース設計システム”，情処学会第３４回全国大会，１９８７

5 S. Lanka, “Automatically Inferring Database Schema”, Proc Int Jt Conf Artif Intell, Vol.9, 1985

6 溝口，磯本，角所，“データベース論理設計支援エキスパートシステム”，ＩＣＯＴ研究論文１９８４

7 川口，溝口，山口，角所，“データベースの論理設計を支援する知的インタビューシステム”，情処学会知識工学と人工知能研究会，１９８６

8 馬場，中野，池田，“データベース性能診断システム”，情処学会データベースシステム研究会，１９８５

9 C.L.Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, Vol.19, 1982

１０ L. Bronston, R. Farrell, E. Kant, N. Martion, "Programming Expert Systems in OPS5", 1985

１１ “Ｇ－ＢＡＳＥ スキーマ設計ガイド”，リコー，１９８７

１２ “Ｇ－ＢＡＳＥ ＤＤＬ言語仕様書”，リコー，１９８７

１３ “Ｇ－ＢＡＳＥ 応用プログラマガイド”，リコー，１９８７

１４ 安次富他，“ＤＢＭＳを兼ねるＬＩＳＰシステム上のエキスパートシステムの構成”，情報処理学会第35回全国大会，7Cc-10，１９８７

１５ 黒瀬他，“ＤＭ－Ｌｉｓｐを用いた推論システム（１）－－ 知識表現－－”，情報処理学会第36回全国大会，3R-9，１９８８

１６ 海保，加藤，堀，原田，“ユーザ・読み手の心をつかむマニュアルの書き方” 共立出版，１９８７

１７ Brad M. McGehee, The Complete Guide to Writing Software User Manuals, 1984

１８ 高橋，“わかりやすいマニュアルの作成法”，日経マグロヒル，１９８５