Vol. 27 No. 2 情報処理学会論文誌 Feb. 1986

# Horn 集合反証に基づく会話型データベース問い合わせ言語†

高木利久†† 松尾文碩††† 牛島和夫††

Tsuno は Horn 集合反証機構に基づく会話型のデータベース問い合わせ言語である．Tsuno は推論関係型データベース管理システム Adbis の利用者インタフェースとして設計開発した言語であり，次のような特徴をもつ：(1)仮想関係定義が行える，(2)応用プログラムとの結合が容易である，(3)情報検索型インタフェースを備えている．(4)プログラミング言語として利用できる．Tsuno は，現在九州大学大型計算機センターの FACOM M-382 OS IV/F 4 上で構築された二つのデータベースシステムの検索に利用されている．本稿では Tsuno の言語仕様およびその実現方式について述べる．

## 1. まえがき

今日，大学などの学術研究機関において実験・観測データなどの数値データを集積し，そのデータを解析するための実用的なデータベースシステムは欠かせない．筆者らは，そのようなデータベースシステムの構築を支援するために **Adbis**[2] と呼ぶ関係型データベース管理システムを開発した．Adbis は，Horn 集合 (Horn Sets[4]) の反証 (refutation[7]) に基づく推論機能をもつシステムである．Adbis の構成図を**図1**に示す．

図1の **DMP** (Data Manipulation Primitives) は関係型データベースの構築・管理・検索などを行うための，**HSI** (Horn Set Interpreter) は反証を行うための，モジュールである．これらのモジュールは，利用者にその機能をサブルーチンプログラムの形式で提供する．開発当初は，Adbis を DMP と HSI の二つのモジュールで構成していたため[2]，データベースの問い合わせなどを行うには，利用者はこれらのモジュールを呼び出すプログラムを作成する必要があった．そこで，端末利用者と HSI とを仲介するモジュール **Tsuno** を開発し，上記の機能を会話型で使えるようにした．さらに，Horn 集合の記述に不慣れな利用者のために，集合操作を基本とした伝統的な情報検索型のコマンドによりデータベースの問い合わせが行えるインタフェースを開発した．これが **Tsuno-Kakushi** である．

Tsuno の主な役割は Horn 節の入力，Horn 集合の管理，各種ファイルの管理，メモリの確保，答えの整形出力などであるけれども，端末利用者にとっては，HSI を処理系の中核とする会話型のデータベース問い合わせ言語 (Query Language) とみなすことができる．

問い合わせ言語としての Tsuno は次のような特徴をもっている：(1)仮想関係 (virtual relation) 定義が行える，(2)応用プログラムとの結合が容易である，(3)情報検索型インタフェース (Tsuno-Kakushi) を備えている，(4)プログラミング言語として利用できる．

Tsuno と Tsuno-Kakushi は現在九州大学大型計算機センターの FACOM M-382 OS IV/F 4 上で構築された二つのデータベースシステム[3),6)] の検索に利用されている．

本稿では Tsuno の言語仕様，Tsuno-Kakushi の言語仕様とその開発の背景およびこれらの言語処理系を構成する各モジュール (Tsuno, Tsuno-Kakushi, HSI, DMP) の役割とその実現について述べる．このうち，HSI と DMP の開発の背景，基本機能，理論的な基盤についてはすでに報告しているので[1),2)]，この二つのモジュールに関する説明は，言語処理系の中核を成す HSI の概要とその具体的な実現方式以外は割愛する．

## 2. Horn 集合と HSI

本稿では論理式を次の節形式 (clausal form) で表現する．

$$A_1, \cdots, A_m \rightarrow B_1, \cdots, B_n$$

Horn 節 (clause) とはたかだか 1 個 ($0 \leq n \leq 1$) の正リテラル (positive literal) しかもたない節であり，Horn 節の集まりを Horn 集合と呼ぶ．$m$ と $n$ の値

† Interactive Database Query Languages Based on Horn Sets by TOSHIHISA TAKAGI (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University), FUMIHIRO MATSUO (Computer Center, Kyushu University) and KAZUO USHIJIMA (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科
††† 九州大学大型計算機センター

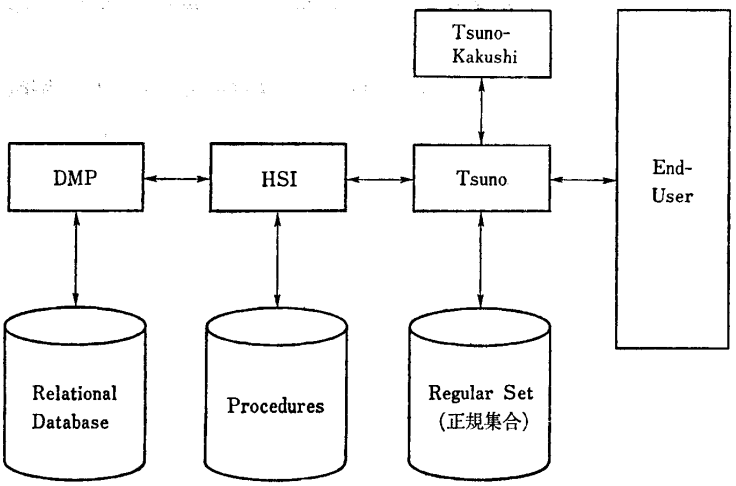


**図 1** Adbis のシステム構成図
Fig. 1 System configuration of Adbis.

の組み合わせによって Horn 節は次の 4 種類に分類できる.

$m=0$, $n=1$ の場合　正節 (positive clause);
$m>0$, $n=1$ の場合　混合節 (mixed clause);
$m>0$, $n=0$ の場合　負節 (negative clause);
$m=0$, $n=0$ の場合　空節 (empty clause).

正節と混合節とを正規節 (regular clause) と呼び, その集合を正規集合と呼ぶ. また, 変数を含まない正節を正基礎節 (positive ground clause) と呼ぶ.

HSI は三つの基本的な機能をもっている. 一つは反証機能である. これは利用者から与えられた負節形式の質問に対して, 正規節を使って下降型 (top down) の推論を行い空節を得ることである. HSI は, 導出原理 (resolution principle[7]) と呼ばれる, ただ1個の推論規則を用いてこの推論を行う. いま, $A, B$ をリテラル; $\Gamma, \Delta, \Lambda$ をリテラルの有限系列または空系列; $\eta, \sigma$ を代入 (substitution[7]); 表現 (expression[7]) $E$ への代入の結果を $E\sigma$ で表すことにすれば, 推論規則は次のように書ける.

[推定規則]
前提 1 : $\Delta, A, \Lambda \rightarrow$　　(負節)
前提 2 : $\Gamma \rightarrow B$　　(正規節)
結果 : $\Delta\sigma, \Lambda\sigma, \Gamma\eta\sigma \rightarrow$　　(負節)

ここで, $\eta$ は前提1と前提2が同じ変数をもたないようにするための改名代入 (renaming substitution[7]); $\sigma$ は $A$ と $B\eta$ の mgu[7] (most general unifier) である.

HSI の二つめの機能は, DMP を使って関係型データベースを検索しデータベースの関係名と関係の組 (tuple) とをそれぞれ述語記号 (predicate symbol) と述語引数 (term) とする正基礎節の集合に変換することである. なお, Horn 節を使って定義される関係とデータベース上の関係を区別するためにデータベース上の関係を基本関係と呼ぶ.

三つめは Fortran 77 などの高級手続き型言語で書かれたサブルーチン形式のプログラム (以降プロシジャと呼ぶ) を実行し, サブルーチン名とそのサブルーチンの引数の値の並びをそれぞれ述語記号, 述語引数とする正基礎節に変換する機能である.

上の推論規則において前提1のリテラル $A$ の述語記号がデータベースの関係名またはプロシジャ名と一致すれば HSI は上記の第2, 第3の機能を使って自動的に前提2に相当する正基礎節を作成する. そのため利用者はデータベースの問い合わせやプロシジャの呼び出しのための特別な操作を必要とせず, それらを反証の一過程として統一的に捉えることができる.

## 3. 言語仕様

### 3.1 入力形式

Tsuno は利用者から空節以外の Horn 節を受けつける. 負節が入力されると, Tsuno はそれと正規節とを使って反証を行い, その結果を利用者に返す. 正規節が入力されると, それを正規集合 (図1) に蓄える. Tsuno では二つの入力モードを設けて負節と正規節の入力を区別している. Tsuno が促進記号として端末に "." を出力したモードのときは負節のみを受けつける. 促進記号として行番号を出したモードのときは, 正規節を受けつける. このように二つのモードを設けることにより述語記号の局所性と仮想関係のモジュール性とが高まるため, Tsuno の利用者は容易に正規集合を管理することができる.

Tsuno では, 正リテラルを矢印の右辺に, 負リテラルを符号を除いたアトム (atom) の並びにして左辺に置いた形式で Horn 節を表現する. **図2**に Tsuno における Horn 節の例を示す.

ATOMP, KY, AUTH, $DB などは述語記号であり, これは英字 (英大文字, 英小文字, $) で始まる8文字以内の英数字列で表す. また, X, Y, ID など

```
(正節)
  -> ATOMP (32768, 23, 62150)
  -> KY (32768, 'TRIAZOLE')
(混合節)
  KY (X, Y), AUTH (X, Z) -> KYAUTH (X, Y, Z)
  BIB (ID, CN, MF, BI, CO, YR) -> BIBCNMF(CN, MF)
(負節)
  $DB ('CXDB') ->
  KY (ID, 'TIAZOLE'), AUTH (ID, 'WERNER, P. -E.') ->
  KY (ID, 'XYLITOL') -> $A (ID)
```

**図 2** Horn 節
Fig. 2 Horn clauses.

は変数であり，これも述語記号と同じ形式で表す．32768, 23, および，'TRIAZOLE', 'CXDB' などは定数である．引用符で囲まれた文字列は文字定数であり，数字列は数値定数である．定数の表記法は Fortran 77 の文字定数および算術定数の表し方と同じである．アトムとアトムの間にカンマまたは空白を置いてもよいし，何も挿入しなくてもよい．矢印は，"->" または ">" で表すが，省略してもよい．矢印を省略しても，Tsuno は入力モードによりそれが負節，正規節のどれであるかを区別することができるからである．

なお，図2中の $A (ANSWER の略) という述語記号は Tsuno があらかじめ用意している述語記号 (組み込み述語と呼ぶ) であり，その役割は $A の引数として指定された変数に代入された値を反証終了時に出力することである．Tsuno は $A を含む節を，推論規則の適用上は矢印の右側に $A がない節とまったく同じように扱う．

### 3.2 Tsuno-Kakushi

Tsuno-Kakushi は，Horn 節の記述に不慣れな利用者に集合操作を基本とした伝統的な情報検索システム[8] の機能を提供するために開発したインタフェースである．以下に Tsuno-Kakushi のコマンドの例を示す．

```
FIND KW=TRIAZOLE AU=WERNER, P. -E.
AND KW=XYLI
DISPLAY CN MF
```

上の FIND コマンドの意味は，KW の値が TRIAZOLE で，かつ，AU の値が WERNER, P. -E. であるものを検索することを指示している．このコマンドは図2の第2番目の負節に相当する．AND コマンドは FIND の結果集合のうちで KW の値が XYLI であるものの検索．DISPLAY コマンドは検索結果の出力である．

このインタフェースを開発した背景を以下に述べる．

1) いくつかの科学データベースシステムの構築の経験から，科学データは数値データだけではなく，その数値データを掲載している文献に関するデータを含んでいることがわかった．文献データの検索には伝統的な情報検索システムが適している．

2) Horn 集合の反証ではデータの授受を統一化 (unification[7]) の機構を用いて行う．統一化においては述語引数の順序だけが重要であり，変数名は同じ変数であるか否かを示す役割しかもたない．一方，データベース上の関係においては関係を構成する属性名 (attribute name) が重要であり，その順序は意味をもたない．そのため，Tsuno を使ってデータベースの問い合わせを行う場合，利用者は常に基本関係の属性名を述語引数の順序に翻訳することが必要となる．

3) データベースの利用者の多くは，述語論理よりも，集合操作に基づく情報検索システムの方が馴染みやすい．実際，Tsuno-Kakushi 開発後は，利用者は Tsuno より，このインタフェースの方を好んで使っている．Tsuno-Kakushi は現在七つのコマンドを用意している．Tsuno-Kakushi のコマンドは負節に変換して処理されるが，その処理方式については 4.2 節に述べる．

### 3.3 出力形式

図2の例にも示したように負節には次の二つの形式がある．組み込み述語 $A を右辺に含まない負節を閉質問 (closed query) といい，この質問に対しては Tsuno は肯定・否定を答えとして返す．一方，$A を含む質問を開質問 (open query) といい，この質問には空節に到達した推論の過程で，$A の引数として指定された変数に代入された値の組の集合を答えとして返す．質問に対する答えは "=" のあとに出力する．閉質問に対しては "=T" (真) または "=F" (偽) を答えとして出力する．開質問の場合は答えが複数個存在しうるので，"=" のあとに順番を示し，そのあとに変数名とその値を対にして (=で結ぶ) 出力する．

### 3.4 組み込み述語

Tsuno は以下の組み込み述語を用意している．

1) $D (define)：仮想関係の定義を行う．この述語が入力されると，Tsuno は負節を受けつけるモードを正規節を受けつけるモードに切り換える．

2) $DB (database)：この述語はデータベースをオープンし，その定義情報を Tsuno の処理系に読み

込む.

3) $P (procedure): プロシジャ定義ファイル (次節参照) から, 定義情報を Tsuno の処理系に読み込む.

4) $TSS: 引数に記述された TSS コマンドを実行する.

5) $T (trace): 推論の経過を表示する.

6) $C (cut): 反証を途中で打ち切るための述語である. 打ち切り条件には CPU 時間, 推論規則の適用回数, 推論の深さ, 答えの個数の四つが指定できる.

7) $S (save) と $L (load): $S は正規集合を利用者のファイルに保存するための述語であり, $L はそれを復元するためのものである.

8) $A (answer): $A は 3.1 節に述べた機能のほかに, 推論結果を利用者ファイルに出力することができる. $A の最後の引数として定数を書くと, その定数を名前とするファイルを確保し, その中に答えを書く.

9) Fortran 77 の組み込み関数: Fortran 77 の組み込み関数がそのまま組み込み述語として利用できる.

Tsuno にはこのほか, ファイル処理, 数値演算, リスト処理などを行う組み込み述語が備わっている. これらの組み込み述語を使えば, Horn 集合反証を手続的に解釈[5] することにより, Prolog のような論理型プログラミング言語として Tsuno を使用することもできる. 現在 Tsuno が用意しているプログラミングのための組み込み述語の数は多くはないけれども, プロシジャ呼び出し機能を使うことにより, 利用者は任意のデータ構造とそれを扱うプログラムを作成することが可能である.

### 3.5 応用プログラムとの結合

Tsuno は次の二種類の方法で応用プログラムとデータベースとを結合できる.

1) 組み込み述語 $TSS を使う結合

これは応用プログラムを TSS コマンドの形式で作成しておき, それを組み込み述語 $TSS を使って起動する方法である. この方法では, 応用プログラムとデータベースとの間のデータの受渡しはファイルを経由して行う. まず, 組み込み述語 $A を利用してデータベースの検索結果をファイルに出力する. 次に $TSS を使ってこのファイルを処理するコマンドを起動すればよい.

2) プロシジャ呼び出し機能を使う結合

この方法は応用プログラムをプロシジャの形式で作成しておき, それを先に述べたプロシジャ呼び出し機能を使って実行する方法である. この方法の場合, プロシジャはそのまま Horn 節のリテラルとして記述できるためデータの授受は述語引数を経由して行うことができる. しかもプロシジャの実行は Tsuno が自動的に行うため, 結合のための特別な記述を利用者が行う必要はない. しかし, Tsuno がプロシジャを呼び出すのに必要な情報をあらかじめ Tsuno に与えなければならない. Tsuno が必要とする情報は, プロシジャ名, プロシジャの引数の個数, 引数の属性 (型, 長さ, 入力引数か出力引数かの区別) である. あらかじめ利用者はこれらの情報を利用者ファイルに作成し (これをプロシジャ定義ファイルと呼ぶ), このファイル名を組み込み述語 $P の引数に指定することが必要である.

### 3.6 会 話 例

Tsuno は現在九州大学大型計算機センターの FACOM M-382 OS Ⅳ/F4 上で構築された結晶構造データベースシステム **XDT**[3], 核酸塩基配列データベースシステム **GENAS**[6] の問い合わせ言語として利用されている.

**図3** は XDT を Tsuno を用いて検索した会話例である. 図の下線で示した部分が利用者の入力であり, READY 以外は Tsuno の出力を表している. READY は TSS のメッセージである.

利用者が TSUNO というコマンドを入力すると Tsuno は促進記号 "." を出力し, 利用者が負節を入力するのを待つ. 利用者は組み込み述語 $DB を使って, データベースのオープンを指示する. CXDB は XDT のデータベース名である. 次行の "=T" は閉質問 $DB ('CXDB') -> の反証が成功した, つまり, データベースのオープンが成功したことを示す. 次の負節は, 'TRIAZOLE' というキーワード (KY) をもち, かつ, その化合物の解析結果を報告した論文の著者 (AUTH) が 'WERNER, P. -E' である化合物を求め, 化合物名 (CN), 分子式 (MF), 文献情報 (BI) を出力することを指示したものである. CN, MF, BI は変数名であり, 任意の文字列で構わない. ID, CO, YR も変数名でこの例ではそれぞれ XDT の参照番号, 誌名コード, 論文の発行年に対応する. Tsuno は上記の検索条件を満足する化合物を全部で3個見つけ, 最初の答えを "=1" 以下に出力している. 同様に2番目, 3番目の答えをそれぞれ "=2",

```
READY
TSUNO
.$DB('CXDB')->;
=T
.KY(ID,'TRIAZOLE'),AUTH(ID,'WERNER,P.-E.'),BIB(ID,CN,MF,BI,CO,YR)->$A(CN,MF,BI);
=1
CN=4-AMINO-3-(2,6-DICHLOROBENZYLIDENE-HYDRAZINO)-1,2,4-TRIAZOLE HYDROCHLORIDE TR
IHYDRATE
MF=C9 H9 CL2 N6 +, CL1 -, 3(H2 O1)
BI=L.GOTHE,S.SALOME,P.-E.WERNER: CRYST.STRUCT.COMMUN.,8,1,1979,11.32.10
=2
CN=4-AMINO-3-(2,6-DICHLOROBENZYLIDENEHYDRAZINO)-5-METHYL-1,2,4-TRIAZOLE HYDROCHL
ORIDE
MF=C10 H11 CL2 N6 +, CL1 -
BI=P.-E.WERNER: CRYST.STRUCT.COMMUN.,6,69,1977,9.32.33
=3
CN=4-ACETOAMIDO-3-(1-ACETYL-2-(2,6-DICHLOROBENZYLIDENE)HYDRAZINE)-1,2,3-TRIAZOLE
MF=C13 H12 CL2 N6 O2
BI=P.-E.WERNER: CRYST.STRUCT.COMMUN.,5,873,1976,9.32.43
.END
READY
```

**図 3** Tsuno の会話例
Fig. 3 Example of Tsuno.

"=3" のあとに出力している．なお，この例では，KY, AUTH, BIB はすべて基本関係であり，仮想関係は使用していない．Tsuno の処理を終るには，END を入力する．

## 4. 実　　現

図1の Tsuno, Tsuno-Kakushi, HSI について，その役割と実現上の要点を述べる．これらのモジュールはすべて Fortran 77 を用いて実現している．

### 4.1 Tsuno

Tsuno は負節を受け取ると，省略されたカンマや矢印を補い，それを文字列の形式で HSI に渡すとともに，HSI に実行を委ねる．一方，Tsuno は正規節を受け取ると，それを正規集合（図1参照）に蓄えるとともに，その内容をやはり文字列の形式で HSI に送り，HSI に実行権を渡す．正規集合に蓄えるのは後の修正に備えるためである．負節，正規節以外の入力の場合，これを情報検索型コマンドとみなして Tsuno-Kakushi に渡す．このように，Tsuno の入力か，Tsuno-Kakushi の入力かは Tsuno が判断するので利用者は二つの入力時期を区別する必要がない．負節と正規節の入力モードの切り換えは HSI からの指令によって行う．

HSI の反証が終了すると HSI は実行権と反証結果を Tsuno に返す．Tsuno は反証結果を 3.3 節で述べた形に整形して端末利用者に返す．

このほか，Tsuno は，他のモジュールで使用する各種のファイルやメモリの確保，割り込み処理などを行う．

### 4.2 Tsuno-Kakushi

このモジュールは，利用者から Tsuno を経由して情報検索型コマンドを受け取り，それを負節に変換して Tsuno に返す．それ以降の Tsuno の処理は利用者が直接 Tsuno に負節を入力した場合と同じである．反証結果の出力は Tsuno から直接利用者に返す．

**図 4** に Tsuno-Kakushi のコマンドから Tsuno の負節への変換例を示す．この図はコマンド (C 1), (C 2) をそれぞれ Horn 節 (H 1), (H 2) に変換したこ

```
(C 1)  FIND KW=TRIAZOLE
(C 2)  DISPLAY CN MF
                  ↓
(H 1)  KY (ID, 'TRIAZOLE') -> $A (ID, 'WORKFILE')
(H 2)  $MEMBER ('WORKFILE', ID),
           BIB (ID, CN, MF, BI, CO, YR) -> $A (CN, MF)
```

**図 4** Tsuno-Kakushi コマンドから負節への変換例
Fig. 4 Example of transformation in Tsuno-Kakushi.

とを示している．(H 1) の $A (ID, 'WORKFILE') は反証によって求まった ID の値を WORKFILE というファイルに出力することを指示している (3.4 節の(8)参照)．(H 2) の $MEMBER は 'WORKFILE' から ID の値を1個ずつ取り出すための組み込み述語である．

このような変換を行うには，Tsuno-Kakushi に，KW が関係 KY の2番目の述語引数であり，CN, MF は BIB のそれぞれ2番目と3番目の述語引数であり，さらに，KY と BIB の1番目の引数が同じ意味をもつという情報を与える必要がある．Tsuno-Kakushi はこれらの情報を利用者から得るのではなく，データベースの定義情報から自動的に抽出する方法を採っている．具体的には，データベースがオープンされたとき，HSI が DMP を使ってデータベースの定義情報を入手し，それをもとに変換のための情報を作成し，それを Tsuno-Kakushi に渡す仕組みになっている．自動抽出の方法を採用しているため Tsuno-Kakushi のコマンドのオペランドに指定できるのは基本関係の属性名に限られる．

### 4.3 HSI

(a) 内部形式

HSI は Tsuno から渡された Horn 節をすべて内部形式と呼ぶ形式に変換し，それに対して推論規則(導出原理) を適用する．内部形式は推論の高速化と記憶領域の節約のために Horn 節を次のような形式に符号化したものである．いま，利用者の入力した Horn 節を

$$P_1(T_{1\,1}, \cdots, T_{1\,n_1}), \cdots, P_{m-1}(T_{m-1\,1}, \cdots, T_{m-1\,n_{m-1}}) \rightarrow P_m(T_{m\,1}, \cdots, T_{m\,n_m})$$

とすると，この節を次の内部形式に変換する．

$$mp_1 \cdots p_m n_1 \cdots n_m t_{1\,1} \cdots t_{1\,n_1} \cdots t_{m\,1} \cdots t_{m\,n_m}$$

ここで，$m$：節内のリテラルの個数；$p_i$：リテラル $P_i$ の述語記号を符号化したもの；$n_i$：リテラル $P_i$ の述語引数の個数；$t_{ij}$：$P_i$ の述語引数 $T_{ij}$ を符号化したもの．$m, p_i, n_i, t_{ij}$ はすべて2バイトの長さをもっている．$p_i$ はリテラル管理表 (後述) へのポインタ，$t_{ij}$ はもしそれが定数ならば定数管理表 (後述) へのポインタとなっている．述語引数が変数であるか定数であるかは $t_{ij}$ の先頭1ビットで区別する．変数の場合，$t_{ij}$ の2ビット目は改名 (renaming[7]) 処理のために使う．

このような内部形式を用いることにより，導出原理適用に必要な改名や代入の処理を高速に実行できる．また，深い推論を行う場合，一般に大量の Horn 節が生成されるけれども，内部形式を利用することにより記憶領域の消費を減らすことができる．

(b) 節選択アルゴリズム

内部形式への変換は推論規則そのものを高速化するための手段であるが，HSI では以下に述べる手法を用いて推論規則の適用回数を減らす工夫も行っている．

2章で示した推論規則を適用するには次の2種類の選択を行う必要がある．

1) 負節から一つのリテラルを選ぶ．これは前提1のリテラル $A$ を決めることに相当する．

2) 正規節の集合から一つの正規節を選ぶ．これは前提2としてどの節を選ぶかを決定することである．

空節に到達するまでの推論規則の適用回数はこの二つの選択をいかに行うかによって大きく左右される．この二つの選択に関して，選択の情報は，すべて外部から与えるか，HSI が自動的に選択するか，あるいはそれらの中間型が考えられるが，HSI では自動選択の方法を採用している[1]．HSI の自動選択の基本的な戦略は1段の先読み (look-ahead) を行って，上記2) の選択時になるべく正節を選ぶようにすることである．これは本質的には単一優先戦略 (unit preference strategy)[7] であり，この戦略は今日知られている汎用戦略のうち効率的には最良のものの一つである．

具体的には次のアルゴリズムを用いて節を選択する．

[前処理1] 反証を開始する前に次の処理をする．まず，正規集合内のすべての正規節に，その節内の負リテラルの個数 (矢印の左側のリテラル個数) をもち点とする点数をつける．この点数を節点数と呼ぶ．正規節が正節の場合，節点数は0となる．

[前処理2] 次に同じ述語記号を正リテラルにもつ正規節の集合を求め，それらの正規節の節点数のうちで最も低い点数をそのリテラルの点数とする．これをリテラル点数と呼ぶ．

[前処理3] 節点数が1で，かつ，その正規節内の負リテラルのリテラル点数が0の場合，節点数を 0.5 に変更するとともに，この点数に基づいてリテラル点数を再計算する．この処理が1段の先読みに相当する．

[選択1] 推論規則の前提1のリテラル $A$ としてリテラル点数が最も低いリテラルを選ぶ．もし，同じ点

数のものが複数個存在する場合は，述語引数のうち定数であるものの割合が大きい方のリテラルを選ぶ．この割合も等しいときは最も左端のリテラルを選ぶ．

［選択2］ 上の方式で選んだリテラルと同じ述語記号を正リテラルにもつ正規節の中から最も節点数が低いものを前提2として選ぶ．等しい場合は正規集合において，より上にある正規節を選ぶ．

（c） 管 理 表

HSI は次の管理表を使う．

1) データベース定義表：この表はデータベースで定義されている基本関係の個数に相当するエントリをもち，各エントリにはデータベースの問い合わせに必要な，関係名，関係を構成する属性名，キイ，属性間の関係，属性の型と長さ，などの情報が入っている．

2) プロシジャ定義表：ここには，3.5 節で述べたプロシジャの定義情報を格納する．この表もプロシジャの個数に対応するエントリをもつ．

3) 正規節管理表：ここには，正規節の内部形式とその節の節点数が入っている．

4) リテラル管理表：この表には，述語記号，述語引数の個数，リテラル点数，および，このリテラルを正リテラルとする正規節の個数とその正規節へのポインタが入っている．もし，そのリテラルが基本関係であればポインタはデータベース定義表のエントリを，プロシジャであればプロシジャ定義表のエントリを指す．正規節として定義されていれば，ポインタは正規節管理表のエントリを指すことになる．なお，内部形式における $p_i$ はこの表内の対応するエントリのアドレスを表している．

5) 定数管理表：この表には反証で使われる定数の実体を格納している．内部形式における $t_{ij}$ はこの表内の対応するエントリのアドレスを表している．

6) 負節管理表：ここには反証過程で生成される負節の内部形式を格納する．

（d） メモリの管理

HSI は管理表をメモリ上に連続して割り付ける．各管理表の大きさは Tsuno から与えられたメモリの大きさに従って HSI が自動的に決めるけれども，必要に応じて変更できるようになっている．管理表のうち，データベース定義表，プロシジャ定義表，リテラル管理表は反証前に大きさが決まり，その後も大きさは変わらない．

一方，HSI はデータベースを検索するとその結果を正基礎節の集合に変換するので，検索の度に正規節管理表は書き換えられ，大きくなる．このとき，定数も増えるので，定数管理表も大きくなる．このことは，プロシジャを呼び出したときも起きる．また，負節管理表は推論規則の適用の度に大きくなる．これらの三つの管理表は，後戻りのとき，その領域を返却するように設計している．

（e） 反 証

HSI の処理は入力が正規節であるか負節であるかによって二つに分けられる．

1) 正規節が入力された場合

入力された正規集合に対して前処理1-3を行い，節点数，リテラル点数を計算するとともに，正規集合を内部形式に変換する．次に正規節管理表，リテラル管理表，定数管理表を作成する．

2) 負節が入力された場合

HSI は Tsuno から負節を受け取ると，リテラル管理表をもとにそれを内部形式に変換する．もし，リテラル管理表にないリテラルが負節内にあれば，そのリテラルは未定義となり反証は行われない．次に，節選択アルゴリズムに従い，推論規則の前提1，前提2を決め，推論を行う．もし，推論結果が空節であれば答えを Tsuno に返す．負節が閉質問の場合反証を終える．開質問ならば後戻りして別の答えを見つける．これを後戻りできなくなるまで続ける．もし，推論結果が空節でなければ，新たに作成された負節に対して推論を行う．もし，推論規則の適用に失敗すれば (mgu[7] が存在しないとき) 節選択の時点に後戻りし，前提2を変更し，推論を行う．これにも失敗すれば今度は前提1のAを変更して推論を行う．もはや後戻りできなければ，その旨の答えを Tsuno に返して処理を終える．

（f） データベースへの問い合わせ

データベースの問い合わせを行うには，利用者は負節の入力前に組み込み述語 $DB をあらかじめ実行するだけでよい．$DB が入力されると HSI はデータベースをオープンし，その定義情報を読み，データベース定義表を作成するとともに，データベース上の関係を正基礎節とみなしてリテラル管理表に登録する．なお，この正基礎節の節点数は0である．

節選択アルゴリズムによって選ばれた前提1のリテラルがデータベース上の関係と等しければ，HSI は推論規則の適用前に DMP を用いてデータベースを検索する．検索後，DMP の結果集合を内部形式の正規節に変換し，正規節管理表に登録する．ここで生成

された正規節は推論規則の前提2になる．これ以降の処理は Horn 節の反証と同じである．

（g） プロシジャの実行

プロシジャを実行するには，データベースの問い合わせ同様，組み込み述語 $P をあらかじめ実行するだけでよい．

節選択アルゴリズムによって選ばれた前提1のリテラルがプロシジャならば，HSI は推論規則の適用前にプロシジャを呼び出して正規節を生成し，正規節管理表に登録する．これ以降の処理は Horn 節の反証と同じである．

## 5. むすび

Tsuno および Tsuno-Kakushi を問い合わせ言語とした結晶構造データベースシステム XDT，核酸塩基配列データベースシステム GENAS はどちらも九州大学大型計算機センター利用者のみならず，大学間ネットワークを通して全国の利用者に使用されている．

今後は，XDT，GENAS の経験をもとに Adbis の機能強化，改善，洗練を図る予定である．とくに HSI と DMP とのインタフェースを改良し，データベース参照回数を減らすことによる効率改善を計画している．

一方，Horn 集合を知識ベース，反証機能を推論エンジンととらえることにより，Adbis を使ったデータベースシステムは知識ベースシステムとして機能させることができる．しかし，そのためには推論の柔軟な制御，推論の信頼度の扱い，質問応答インタフェースなどの機能が Adbis には欠けている．そこで現在，Waku と呼ぶ質問応答のインタフェースを Adbis に追加するとともに，このインタフェースを利用した質問応答システムを開発している．

## 参考文献


1) 松尾文碩，高木利久：データベース操作のための Horn 集合反証器，情報処理学会論文誌，Vol. 25, No. 3, pp. 458-464 (1984).
2) 松尾文碩，二村祥一，高木利久：推論関係型データベース管理システム Adbis，情報処理学会論文誌，Vol. 24, No. 2, pp. 249-255 (1983).
3) 松尾文碩，二村祥一，高木利久，河野重昭：Adbis による結晶構造データベースシステムの構築，情報処理学会アドバンストデータベースシステムシンポジウム論文集，pp. 39-48 (1982).
4) Henschen, L. and Wos, L.: Unit Refutations and Horn Sets, *J. ACM*, Vol. 21, No. 4, pp. 590-605 (1974).
5) Kowalski, R.: Predicate Logic as Programming Language, in Rosenfeld, J. (ed.): *Information Processing 74*, pp. 569-574, North-Holland, Amsterdam (1974).
6) Kuhara, S. et al.: GENAS: A Database System for Nucleic Acid Sequence Analysis, *Nucleic Acids Res.*, Vol. 12, No. 1, pp. 89-99 (1984).
7) Loveland D.W.: *Automated Theorem Proving: A Logical Basis*, p. 405, North-Holland, Amsterdam (1978).
8) Salton, G. and McGill, M.: *Introduction to Modern Information Retrieval*, p. 448, McGraw-Hill, New York (1983).


(昭和60年5月31日受付)
(昭和60年9月19日採録)