データベースシステム 101－8
（1995. 1. 26）

生物分類樹データベースにおける履歴推論

北上 始*, 舘野義男**, 五條堀 孝**

*広島市立大学

**国立遺伝学研究所

近年、生物学の急速な発展に伴って、生物分類樹データベースの構築が急がれている。生物分類樹データベースは、生物種に関する系統学的な教育だけてなく、DNAのような分子レベルのデータと比較研究などを行う上でも大変重要である。しかし、現在の生物分類樹データベースは、DNA国際データバンクなどの二次的な業務として構築されており、構造が複雑なわりには十分な管理が行われていない。本稿では、無矛盾な生物分類樹データベースを構築するために有用な履歴推論について提案する。ここで扱う生物分類樹データベースは複雑で大規模な木構造データであり、二項関係モデルを用いて表現されている。再帰質問検索を用いて木構造データの局所的および大局的な探索が行われる。ここでは、構築および修正さた木構造データが時間論理を用いて管理される。履歴推論は、再帰質問検索に時間論理を組み込むことにより達成されている。履歴推論で用いられる時間論理では、状態の持続性や時制の一致などの概念が重要な役割を演ずる。

Historical Reasoning for Taxonomy Databases

Hajime Kitakami*, Yoshio Tateno** and Takashi Gojobori**

* Hiroshima City University
151-5 Ozuka, Numata-Cho, Asa-Minami-Ku, Hiroshima-Shi 731-31, Japan
** National Institute of Genetics
1111 Yata, Mishima-Shi, Shizuoka-Ken 411, Japan

Taxonomy databases represented by a tree structure are powerful electronic dictionaries which aid in biological research by computer. These databases are actively constructed with the DNA databases of the international DNA data banks in biological advancements. The taxonomy databases are, however not consistently constructed with a relational format. This paper proposes a historical reasoning method to be useful for constructing consistent taxonomy databases. The taxonomic tree is represented by a binary relation in this paper. Recursive query processing plays an important part in searching the tree structure. The historical reasoning is needed temporal logic concepts and the query processing which includes a local and global search functions for the tree structure. We use both duration for time of fact and checking of tense in the historical reasoning.

1. はじめに

近年、生物学の急速な発展に伴って、生物分類樹データベースの構築が急がれている。生物分類樹データベースは、生物種に関する系統学的な研究や教育だけてなく、DNAデータから推論される分子進化系統樹と比較研究などを行う上でも大変重要である。地球上には、動植物に限っても130万種以上もの生物が生息していると言われているが、生物分類自身の研究や整理は一元的に行われているのではなく、生物学の多くの専門家によって個別に行われている。このような生物分類によって作り出されたデータは電子化され、生物分類樹データベースと呼ばれている。その代表としては、DNA国際データバンクなど[1,2,3,4]で二次的な業務として構築されている生物分類樹データベースがあるが、それらには矛盾（エラー）がかなり含まれている。

第1の原因としては、近年の生物学の急速な発達に伴って発生する矛盾が挙げられる。この発達は形態学に基礎を置いて整理された生物分類樹を大きく変化させている。しかしながら、このデータベースは、それを必要とする組織で個別に構築・維持されているに過ぎない。即ち、それらを統一的に整理しデータベースを一元化するような組織的活動をするまでには至っていない。実際、現存するデータベースのいくつかを相互比較すると、それらは互いに矛盾していることが判る[5]。

第2の原因としては、生物分類樹データベースの構築者は、情報科学について十分な知識をもっていないことが挙げられる。このため、彼等は矛盾回避の重要性をはっきりと自覚しておらず、その手段を的確に表現する方法も知らない。これは、データ構築時に多くの矛盾が混入する原因になっている。たとえ、矛盾の存在に気がついていてもその修正は場当たり的に行われており、その修正方針はあいまいなままである[5]。

各データベース間に、共通部分と非共通部分のデータがあるとすれば、前者の矛盾は共通部分、後者は非共通部分に発生しやすい。

本論文では、両者の矛盾を効率良く除去するために有用な履歴推論方式について提案する。前者の矛盾については、データベース間の構築・修正履歴を比較することによってどのデータが正しそうか判断することができる。後者の矛盾については、構築・修正履歴を分析することにより有害な構築パターンを見つけることができる。例えば、時間を遡って修正履歴を調べれば、同じような失敗の繰り返しの発見が可能であろう。

著者らは、構築者自身にとって、矛盾発生の原因を最初からはっきりと自覚しているものではなく、履歴推論を利用することによって明確になると考えている。生物分類法の最新動向について、「対応が遅い」構築者が構築・修正したデータは、他のデータベースに比べて多くの矛盾を含むことになるであろう。この他に、構築者のある種の「思い込み」や「情報不足」によって生ずる矛盾も存在するであろう。いずれも、データベース構築者間でのきめの細かい相互理解や自動的な情報獲得手段があればこのような問題は生じないが、研究目的を始めとして思想や生活習慣が異なる組織間でそれを達成することは極めて困難である。その達成には多大な時間や労力さらには膨大な費用を必要とするのである。ここでは、履歴推論の立場からその問題を解決しようとしている。

2. 再帰質問処理

生物分類樹データベースには大規模な木構造が格納されているが、この二次元的な広がりを持つ木構造全体を眺めるような大型ディスプレーを考えることは非現実的である。著者らは、通常のディスプレー上で着目すべき部分構造を効果的に知る方法として、再帰質問処理[5]に基づく局所検索と大局検索を提案[6,7]してきた。局所検索は、ある基準ノードの周囲の二次元的な接続状況を知るために考えられた検索であり、大局検索は、ある基準ノード以下の部分木の構造を一次元的に眺めるために考えられた検索である。生物分類樹は子ノードと親ノードが再帰的に結合された木構造である。大局検索において、基準ノードを頂点とすれば、大規模な木構造全体を一次元的に表示できる。もし、その検索結果を印刷出力すれば、その接続状態を落ち着いて眺めることができる。

以下では、木構造を二項関係モデルを用いて表現するために、これを子ノード"X"と親ノード"Y"の組"(X,Y)"で表現する。ただし、"X"および"Y"は、ある同じ領域"D"から選ばれた値である。例えば、人間の生物分類について考えて見よう。人間の種(species)と属(genus)は、各々"sapiens"と"homo"である。これによると、人間のデータ表現は"taxonomy(sapiens, homo)"とするのが妥当だが、生

物の分類階級(level/ranking)も同時に表現したほうが分かりやすい。以後、これを"taxonomy(species(sapiens), genus(homo))"と表現する。

2.1 局所検索

局所検索の基本は、ある基準ノードから木構造を上下探索する処理である。以下では、この上下探索処理をそれぞれ"up"や"down"と呼ぶ。ここで提案する上下探索では、指定されたノード数だけの探索（移動）が可能である。また、この探索処理では基準ノードを"0"として、基準ノードからの探索ノード数も計算している。

以上をまとめると、上方向の探索処理については、次のような演繹データベースの再帰ルールとして表現できる。

```
up(Predicate,X,MAX,MAX,[[MAX,Y]]):-
        execute(Predicate,X,Y).
up(Predicate,X,MAX,CNT,[[CNT,root]]):-
        execute(Predicate,X,root).
up(Predicate,X,MAX,CNT,[[CNT,Y]|W]):-
        MAX>CNT,CNT1 is CNT+1,
        execute(Predicate,X,Y),
        up(Predicate,Y,MAX,CNT1,W).
```

"up"は、５つの引き数から構成されている。第１引き数には、生物分類樹データベースの名前を示す述語名が与えられる。第２引き数には、基準ノードを与える。第３引き数には、たどるべきノード数の最大値を与える。もし、木構造の頂点"root"までたどりたいのであれば、ここに、木構造の高さよりも大きな値を与えておけば良い。第４引き数は、探索ノードの番号を数え上げるための領域であり、利用者は初期値として"０"を設定する。この番号が第３引き数の値と一致すれば、探索処理を終了する。第５引き数は、探索された全ノードを探索結果として返すための領域である。検索結果はリストで表現され、リスト中の各要素は探索されたノード"Y"とその番号"CNT"で構成されている。

上記の３ルールの１、２番目は、探索終了条件を表わしている。１番目は、探索ノードの番号が指定した最大値に一致したときの終了条件であり、２番目は、基準ノードから上方向を探索し、木の頂点"root"に到達したときの終了条件である。どちらの終了条件も満足しないとき、最後の３番目のルールで次のノードが探索される。３つのルールはいずれも"execute"と名付けられた述語を含んでいる。ルール中の述語"execute(Predicate,X,Y)"は"Predicate(X,Y)"を実行することを意味する。以下、"down"についても、似たような演繹データベースの再帰ルールとして表現できる。

```
down(Predicate,X,MAX,MAX,[[CNT,Y]]):-
        execute(Predicate,Y,X).
down(Predicate,X,MAX,CNT,[[CNT,leaf]]):-
        execute(Predicate,leaf,Y).
down(Predicate,X,MAX,CNT,[[CNT,Y]|W]):-
        MAX>CNT,CNT1 is CNT+1,
        execute(Predicate,Y,X),
        down(Predicate,Y,MAX,CNT1,W).
```

"up"の処理と比較してみると、"down"は木構造の下方向を探索する処理なので、ルール中の"execute"では子ノードと親ノードの関係が反対になっている。また、"CNT"の解釈は、"up"と違い、基準ノードからの探索ノードまでの深さ（段数）を意味する。また、"up"の探索終了条件は頂点"root"への到着であるのに対して、"down"では葉"leaf"への到着になっている。

以上の２つのルールを用いて、著者が既に提案している３つの局所検索を表現してみよう。

（１）上位系統検索

これは、基準ノード"X"から頂点"root"までのノードを検索する処理である。これを、"lineage"と名付けると、"up"述語を用いて次のように表現できる。

```
lineage(Predicate,X,Y):-
        large_number(MAX),
        up(Predicate,X,MAX,1,Y).
```

ここで、"lineage_number"は、木構造の高さよりも大きな数を与える述語である。著者らは、これを関係データベース[8]でのストアードプロシージャとしてインプリメントする方法も開発してきている[6]。以下の２つの局所検索も同様なインプリメントが可能である。

（２）下位系統検索

これは、基準ノードの子ノードを探索する処

理である。これを、"posterity"と名付けると、"down"述語を用いて次のように表現できる。

```
posterity(Predicate,X,Y):-
        MAX=1,
        down(Predicate,X,MAX,1,Y).
```

これをN段下の子孫ノードまで探索するように拡張することは、極めて容易である。著者らは、関係モデルを用いて、これを木構造の幅優先探索として捕えて実現する方法も開発している。

（3）同一分類検索

基準ノードと同一分類に属するノードを探索する処理である。これを、"homology"と名付けると、"up"と"down"述語を用いて次のように表現できる。

```
homology(Predicate,X,Y):-
        MAX=1,
        up(Predicate,X,MAX,1,[[N,Z]]),
        down(Predicate,Z,N,1,Y).
```

これは、基準ノードの1段上の親ノードから見た同一分類ノードである。MAX=n（＞1）とすることによって、これをn段上の親ノードから見て同一分類のノードを探索するように拡張可能である。

以上の局所検索を用いれば、基準ノードが木構造全体の中のどの位置にあるべきかを知るための位置情報を計算することは容易である。例えば、(a)頂点"root"から基準ノードまでの深さ、(b)基準ノードと同一階級にあるノード数、(c)基準ノードから葉"leaf"までの高さなどがこの情報として挙げられる。(a)については、"lineage"を用いて、取得されたノード数を数えることによって達成される。(b)については、拡張された"homology"を用いて、取得されたノードを数えることによって達成される。(c)については、拡張された"posterity"を用いて取得されたノードの中で基準ノードから一番遠い葉ノード"leaf"までのパスに含まれるノード数を計算することによって達成される。

2.2 大局検索

大局検索は、ある基準ノード以下のノードの繋がり具合を一次元的に捕えるために考えられた検索である[7]。データベース内に格納される二項関係を直接表示（又は印刷）したのでは、ノード間の親子関係を利用者自身が探すことになり、利用者にとって大変な努力を要する。前述の"posterity"のインプリメンテーションで触れたような幅優先探索の方法は既にいくつか研究されているが、この方法では同一レベルのノードを連続的に表示するのに都合がよくても、ノード間の親子関係を連続的に表示するには有用でない。

ここでは、木構造を深さ方向に探索（ある種の深さ優先探索）しながらノードを連続的に表示する方法を提案する。これを"su_btree"と名付けると、以下のような演繹データベースの再帰ルールとして表現することができる。

```
sub_tree(Predicate, X, Y):-
        subtree(Predicate, X, 0, Y).

subtree(Predicate,leaf,CNT,Y).
subtree(Predicate,X,CNT,Y):-
        CNT1 is CNT+1,MAX=1,
        setof(Z,down(Predicate,X,MAX,1,[Z]),S),
        next_nodes(Predicate,S,CNT1,Y).

next_nodes(Predicate,[],CNT,[]).
next_nodes(Predicate,[[N,X]|Y],CNT,W):-
        subtree(Predicate,X,CNT,U),
        next_nodes(Predicate,Y,CNT,V),
        append([[CNT,X]|U],V,W).
```

"sub_tree"は述語"subtree"で表現されている。"subtree"は、4つの引き数から構成されている。第1引き数には、生物分類樹データベースの名前を表現する述語名が与えられる。第2引き数には、基準ノードを与える。第3引き数には、基準ノードから探索ノードまでの番号（深さ）を数えるための領域であり、利用者は初期値として"0"を設定する。第4引き数は、探索したノードを探索結果として返すための領域である。検索結果はリストで表現され、リスト中の各要素は探索されたノードとその番号"CNT"で構成されている。"subtree"は2つのルールで構成されている。1番目のルールは、この探索が葉"leaf"にたどりついたときの探索終了条件である。2番目のルールは、前述の下方探索述語"down"とその全解を抽出するメタ述語"setof"を用いて、基準ノードに対する全ての子ノードを

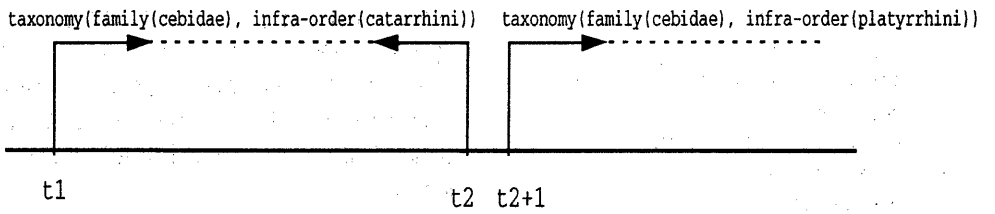

図１．データベース変更における時間表現

探索している。探索された子ノードは、１件ずつ"next_nodes"を用いて処理され、深さ方向への探索が進められる。このようにして探索された結果を次のような方法で表示すれば、ノード間の親子関係の情報が失われず、利用者は木構造の接続関係を容易に理解することができる。

（１）探索結果の表示（印刷や画面表示）において、１行で表示可能なノード数は１つだけとする。
（２）表示されるノード名は、常に着目ノードからの深さ分だけ右へシフトさせる。これにより、子ノードの表示においては、直前の親ノードよりも一文字右へシフトされる。また、同じレベルのノードを表示する場合は、直前の同レベルノードと同じ位置に出力される。親ノードの表示では、直前の子ノードよりも一文字左へシフトされる。

著者らは既にこれを関係データベースのストアードプロシージャとして実現する方法も提案している[7]。

3．時間論理の導入

データベースのログ情報は、データベースの障害回復に有用であるが、これを用いて過去から現在までのデータベースの変化の様子を質問検索するには大変な手間がかかる。ここでは、AllenやKowalskiなどの研究[9, 10, 11]で代表される時間論理を応用し、そのような質問検索を可能にする。以下では、データベースに格納されるファクト"P"を次のような二表現に限定する。

（１）ある時刻"T"から、ファクト"P"が成立する。
```
P since T.
```
（２）ある時刻"T"まで、ファクト"P"が成立する。
```
P till T.
```

上記の（１）はデータ挿入の代りとしてデータベースに追加され、（２）はデータ削除に代って追加される。時刻"T"でデータの更新があると、"P till T"および"P since T+1"が追加される。例えば、図１に示すように、時刻 "t1"で"taxonomy(family(cebidae), infra_order(catarrhini))"が挿入されたが、時刻 "t2"で"taxonomy(family(cebidae), infra_order(platyrrihini))"に変更されたとすると、データベースに次のようなデータが追加される。

```
taxonomy( family(cebidae), infra_order(catarrhini)) since t1.
taxonomy( family(cebidae), infra_order(catarrhini)) till t2.
taxonomy( family(cebidae), infra_order(platyrrihini)) since t2+1.
```

一般のデータベースでは、このような時間に関する情報を管理しておらず、暗黙のうちに現時刻"Tp"における質問"query(P):- P at Tp."を処理していることになる。後述するが、時間論理を用いて履歴管理されたデータベースでは、過去のデータベースに対する高度な問合わせが可能になる。"P at T"の時間オペレータは、時間の持続性という性質を用いて、オペレータ"since"や"till"と容易に関係づけることができる。以下では、データベースの直接検索に最も基本的な時区間オペレター"search"について述べる。表現"search(P,T1,T2)"を"P"は時刻"T1"から時刻"T2"の間まで成立すると解釈する。ただし、"P"は"taxonomy(_,_)"のようなデータベースに格納されるファクトデータを表現する述語である。オペレター"search"は、次のような演繹データベースのルールとして表現できる。

```
search( P, T1, T2):-
        P since T3, P till T4,
        present_time( Tp),
        T3=<T4, T3=<Tp, T4=<Tp,
        no_event( P, T3, T4),
        substitute( T1, T2, T3, T4).
search( P, T1, T2):-
        P since T3,
        present_time( Tp), T3=<Tp,
        no_event( P, T3, Tp),
        substitute( T1, T2, T3, Tp).
```

ここで、"T1","T2"は、区間の時刻を表わし、前者は後者以前の時刻を表わす。ルール中の"present_time"は現時刻を知るための述語であり、"no_event"は、与えられた時間内にデータ"P"に関する更新がなかったことを判定する述語である。"substitute"は、"T1"又は"T2"が変数のとき、それに妥当な時刻を与える述語である。

4. 履歴推論

ここでは、生物分類樹データベースの構築やエラー修正にとって有用な履歴推論方式について提案する。ここでは、2節の再帰質問処理に3節の時間論理を組み込むことによって履歴推論を可能にする。前述の"lineage(P, X, Y)", "posterity(P, X, Y)", "homology(P, X, Y)", "sub_tree(P, X, Y)"などを"P"で表現し、次のような時区間オペレータ"between"を伴った問合わせを考えてみよう。

```
query( P, T1, T2):- P between( T1, T2).
```

時刻"T1"及び"T2"が定数（以下では定数時区間と呼ぶ）の場合は、再帰質問処理に時間論理を組み込むことは容易である。即ち、指定された時区間を満足するデータだけを用いて推論を行えば、答えを計算することができる。しかし、時刻が変数（以下では変数時区間と呼ぶ）の場合は、推論結果に矛盾しないような時区間をデータベースから旨く選び出さなければならないという問題を含んでいる。ここでは推論中にループが生じないという前提を置き、遅延時制一致方式を提案する。この方式では、最初に時間を無視して可能な推論を行う。ここでは、複数解が得られる。その中から選択された解を分析し、推論に用いられた各データの時区間に矛盾が生じないかどうか調べる。それに矛盾があれば、次の候補について同じ様な時区間の矛盾判定を行う。この2段階処理により、各データの時制が一致するような結果を選びだすことができる。本節では、上位系統検索"lineage"にだけ着目するが、ここで述べる方法論は、他の再帰質問処理についても同様に適用可能であり一般性がある。

4.1 定数時区間

時刻"T1"及び"T2"が定数の場合の質問処理について述べる。前述のように"lineage"に時間論理を導入すると、この構成要素の"up"は次のように書き換えなければならない。

```
up(Predicate,X,MAX,MAX,[[CNT,Y]]) between(T1,T2):-
        execute(Predicate,X,Y,T1,T2).
up(Predicate,X,MAX,CNT,[[CNT,root]]) between(T1,T2):-
        execute(Predicate,X,root,T1,T2).
up(Predicate,X,MAX,CNT,[[CNT,Y]|W]) between(T1,T2):-
        MAX>CNT,CNT1 is CNT+1,
        execute(Predicate,X,Y,T1,T2),
        up(Predicate,Y,MAX,CNT1,W) between(T1,T2).
```

"up"では、"between(T1,T2)"の情報を"execute"に伝えるだけの処理であるが、"execute"の処理において初めて3節のルールと関係をもつようになる。

4.2 変数時区間

時刻"T1"又は"T2"が変数の場合の質問処理について述べる。最初に時区間を無視した推論を行うが、その後、推論中に利用された全てのデータについて時制一致に関する処理を行う。このために、各データがどんな時区間で成立するのかを収集しておくことが重要である。ここでは、時区間の情報を推論結果を返す部分に追加している。以上を整理すると、前述のルール"up"は以下のようになる。

```
up(Pred,X,MAX,MAX,[[CNT,Y,T3,T4]]) between(T1,T2):-
        execute(Pred,X,Y,T1,T2,T3,T4).
up(Pred,X,MAX,CNT,[[CNT,root,T3,T4]]) between(T1,T2):-
        execute(Pred,X,root,T3,T4).
up(Pred,X,MAX,CNT,[[CNT,Y,T3,T4]|W]) between(T1,T2):-
        MAX>CNT,CNT1 is CNT+1,
        execute(Pred,X,Y,T1,T2,T3,T4),
        up(Pred,Y,MAX,CNT1,W) between(T1,T2).
```

次に、推論に使われた各データ間で時間的矛盾が生じないかどうか判定する処理に移るが、それは以下の"check_tense"にて行われる。

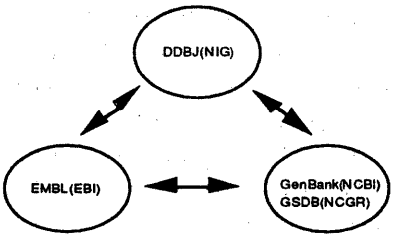

図2．国際DNAデータバンク

表1．再帰質問検索における時間の扱い

時区間 (T1—T2)	局所検索			大局検索 (sub_tree)
	上位系統検索 (lineage)	下位系統検索 (posterity)	同一分類検索 (homology)	
定数時区間による質問での制限	T1≤T2	T1≤T2	T1≤T2	T1=T2
変数時区間による質問での時刻決定	時区間の積集合	時区間の和集合	時区間の和集合	なし

```
query(Predicate,X,Y,T1,T2):-
        lineage(Predicate,X,Y) between(T1,T2),
        check_tense(Y,T1,T2).

check_tense(Y,T1,T2):-
        present_time(Tp),
        intersection(Y,0,Tp,MaxSince,MinTill),
        MaxSince=<MinTill,
        assign_interval(MaxSince,MinTill,T1,T2).
```

"check_tense"は推論結果"Y"を分析し、推論に用いられた時区間付きデータがお互いに矛盾しないかどうか判定するルールである。時区間の共通部分は"intersection"により算出される。時区間の共通部分がない推論結果は、時制一致のルールに反する。共通時区間の開始及び終了時刻の計算結果はそれぞれ変数"MaxSince"及び"MinTill"に保存される。このとき、"MaxSince=<MinTill"を満足していなければ、その推論結果は矛盾するので、その"Y"は答えから除外される。そして、次の候補が"lineage(Predicate,X,Y) between(T1,T2)"により探索され、その推論結果は"check_tense"により判定される。この判定で無矛盾な解答"Y"が発見されると、"assign_interval"により、履歴推論結果としての時区間"T1"及び"T2"が設定される。

5．適用例

ここでは、国際DNAデータバンクで構築されている生物分類樹データベースを取り上げ、その構築やエラー修正について考えてみよう。現在、図2に示すように国際DNAデータバンクは、日本、米国、欧州の三つの組織[1, 2, 3, 4]から構成されており、密接な情報交換をもとにDNAデータベースの共同構築が行われている。各データバンクのデータベースは、それぞれ複数の構築者により構築されている。しかし、生物分類樹データベースには、(1)データベース自身の矛盾、(2)データバンク間の矛盾などが含まれており、この矛盾を取り除く努力が積極的に行われている[6, 7]。前者の矛盾の代表は、スペルミスが挙げられるが、これはデータ登録や更新時の不注意によって発生する。また、構築者の「思い過ごし」や「認識不足」により生ずるミスは、構築者の個性が反映されている場合が多く、これにより生ずる矛盾を発見することはなかなか難しい。後者の矛盾は、主に生物の形態学の発達に伴って生ずる矛盾である。形態学の発達は、生物分類木におけるノードの分割や統合、さらには、ノード名の命名法の変更などにまで影響する。現状では、データベース構築・修正は、各組織で独立に実施されているので、データベース間の矛盾がどのデータバンクの原因によるものなのかを知ることが難しい。

データ登録や修正の状況を時間軸にそって追跡・監視できるツールがあれば、過去の修正履歴を容易に知ることができる。これにより、データベース管理者は、構築者にとって犯しやすいエラーパターンや、形態学の発達により生じたデータベース間の変化の様子などを容易に把握できる。

以下では、参考文献[12]の例を用いて、これまでに提案してきた履歴推論機能を用いて生物分類樹データベースの修正履歴に関する質問検索について述べる。この例を示すためのシステムは、Prolog[13]で実現されており、このシステムで利用可能な再帰質問検索は、"lineage", "posterity", "homology", "sub_tree"の4機能である。また、これらは時間の扱いについて、それぞれ表1に示すような制約をもっている。

紙面の都合上、実行トレースを示すことができないので、発表時にその詳細を述べる。ここでは、便宜上、時刻の始まりをT=0とし、修正があるたびに時刻を1ずつ進めるものとする。

6．まとめ

本稿では、無矛盾な生物分類樹データベースを構築するために有用な履歴推論システムの構成法について述べた。履歴推論は、データベースの構築・修正による変化を時間論理で表現し、生物分類樹の再帰質問検索に組み込むことにより可能になった。ここでは、主として、"lineage", "posterity", "homology", "sub_tree"の4機能の再帰質問検索の時間処理について述べた。

ここでは触れなかったが、データベース間のミスマッチを検出する場合、履歴推論機能を用いれば、時間を考慮したもっと柔軟な検出法も可能になる。従来は履歴推論機能がなかったために、現在時刻のミスマッチだけにしか関与できなかった。たとえば、ある時刻以後、自分のデータベースに最新情報を反映したとしても、他のデータベースが修正されない限り、データベース間のミスマッチは残ったままである。履歴推論機能を用いれば、この部分を従来のミスマッチから除外することができる。これにより、エラー修正の対象を限定することができる。

さて、無矛盾な生物分類樹データベースを国際協力で構築するためには、まず、データバンク間でデータベースに基づくきめの細かい相互理解をはかることが重要である。ここに提案した履歴推論は生物分類樹データベースを空間的かつ時間的にとらえるものであり、その相互理解を自動的に行うために重要な1機能であるともいえる。この相互理解をはかるための1つとして、実務者会議が年1回開催されてはいるが、画像などのイメージを用いてお互いのデータベースの状況を必要に応じて表示する機能も重要である。また、誰がデータ構築や修正をしたのかを履歴情報として残しておくことも重要と思われる。これらは、いずれも関係モデルやオブジェクト指向モデルへの実装を含めて今後の課題であろう。

参考文献

[1] Hajime Kitakami, Yukiko Yamazaki, Kazuho Ikeo, Yoshihiro Ugawa, Tadasu Shin-I, Naruya Saitou, Takashi Gojobori, and Yoshio Tateno: Building and Search System for a Large-scale DNA Database, Frontiers in Artificial Intelligence and Applications, Vol.22, Edited by S.Schulze-Kremer, Advances in Molecular Bioinformatics, IOS Press or Ohmsha Ltd., pp.123-138, 1994.

[2] Desmond G. Higgins, Rainer Fuchs, Peter J. Stoeher and Graham N. Cameron: The EMBL Data Library, Nucleic Acids Research, Vol.20, Oxford University Press, pp.2071-2074, 1992.

[3] Michael J. Cinkosky, James W. Fickett, Paul Gilna, and Christian Burks: Electronic Data Publishing and GenBank, Science,Vol. 252, pp.1273-1277,1991.

[4] Research News, Managing the Genome Data Deluge, Science, Vol.262, No.22, pp.502-505, 1993.

[5] Francois Bancilhon and Raghu Ramakrishnan 1986, An Amateur's Introduction to the Recursive Query Processing Strategies, Proceedings of the ACM SIGMOD '86, Washington D.C., pp.16-52.

[6] Hajime Kitakami, Yoshio Tateno and Takashi Gojobori: Toward Unification of Taxonomy Databases in a Distributed Computer Environment, Proc. of the second International Conference on Intelligent System and Molecular Biology, AAAI Press, pp.227-235, 1994.

[7] 北上始、森康真、有川正俊、舘野義男、五條堀孝：生物分樹類データベースに対する再帰質問検索、信学技報、DE94-60, pp.39-46, 1994.

[8] SYBASE Transact-SQL User's Guide, Sybase Inc., May 1989.

[9] Allen, J. F.: Towards a General Theory of Action and Time, Artificial Intelligence,Vol.23, No. 2, 1984.

[10] Kowalski, R. and Sergot, M. : A Logic-based Calculus of Events, New Generation Computing Vol. 4, 1986.

[11] 原裕貴, 北上始, 中島淳: 時間概念の表現とデフォルト推論, 人工知能学会誌, Vol.3, No.2, pp. 216-223, 1988.

[12] Saitou N. and Nei M.: The Neighbor-Joining Method, Mol.Biol.Evol. 4: pp. 406-425, 1987.

[13] Prolog by BIM(BIM_Prolog) Reference Manual, BIM (Belgium), 1990.