TM-1278他

情報処理学会　第47回全国大会論文集

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~ 5

Institute for New Generation Computer Technology

TM1278	モデル生成型証明器によるFGHCのモード解析	越村　三幸、長谷川　隆三
TM1280	ポータブルKL1処理系KLICの概要	仲瀬　明彦、藤瀬　哲朗、近山　隆
TM1281	遺伝的アルゴリズムを取り入れたタンパク質配列解析	戸谷　智之、石川　幹人、星田　昌紀（松下）、小長谷　明彦（日電）
TM1283	演繹オブジェクト指向データベース言語を用いた遺伝子知識ベースの記述	広沢　誠（日立）、田中　令子（情数研）、石川　幹人

モデル生成型証明器によるFGHCのモード解析

越村 三幸　　長谷川 隆三
新世代コンピュータ技術開発機構研究所

1　はじめに

モデル生成型定理証明器 MGTP による並列論理型言語 FGHC のモード解析手法を示す．FGHC(Flat GHC) プログラムのモード解析は不動点計算に帰着されるが，これは MGTP ではモデルを求めることに相当する．得られたモデルを解読することにより，対象プログラム中の変数のモードを知ることができる．また，対象プログラムにモードの不整合がある場合には，それを同定することができる．モード情報は，コンパイラの最適化のみならず，プログラムの静的バグ同定にも有効である．

2　モデル生成型証明器 MGTP

MGTP はモデル生成法に基づいた一階述語論理の自動証明器である．問題は節形式の集合として与えられ，MGTP はこの節集合のモデルの生成を行なう．

MGTP 節は次のように含意式の形で表現される．

$$A_1, A_2, \ldots, A_n \rightarrow C_1; C_2; \ldots; C_m$$

ここで，$A_i (1 \leq i \leq n)$ および $C_j (1 \leq j \leq m)$ は原子論理式である．$\rightarrow$ の左側を前件部，右側を後件部という．前件部は $A_1, A_2, \ldots, A_n$ の連言 (',')，後件部は $C_1, C_2, \ldots, C_m$ の選言 (';') である．$n = 0$ のとき，前件部を特に $true$ と書き，正節と呼ぶ．一方 $m = 0$ のとき，後件部を特に $false$ と書き，負節と呼ぶ．それ以外の節 $(m \neq 0, n \neq 0)$ は混合節と呼ばれる．また，MGTP 節がアトムの集合 M に置換 σ のもとで violated であるとは，$\forall i (1 \leq i \leq n) \sigma A_i \in M \wedge \forall j (1 \leq j \leq m) \sigma B_j \notin M$ であることをいう．

モデル生成法は，与えられた節集合に対するモデルを，空集合から始めて構成的に求める証明手法である．

モデル生成法には次の二つの規則がある．規則中 $\mathcal{M}$ は構成途中のモデルの集合を表し，各要素をモデル候補と呼ぶことにする．$\mathcal{M}$ の初期値は $\{\phi\}$ である．

- モデル拡張規則:　　混合節もしくは正節 $A_1, A_2, \ldots, A_n \rightarrow C_1; C_2; \ldots; C_m$ が，あるモデル候補 $M \in \mathcal{M}$ に置換 σ のもとで violated である時，M の代わりに各 $C_j\sigma$ を M に加えてモデル候補を拡張する $(\mathcal{M} := \mathcal{M} \cup \bigcup_{j=1}^{m} \{M \cup \{C_j\sigma\}\} \setminus \{M\})$.
- モデル棄却規則: 負節 $A_1, A_2, \ldots, A_n \rightarrow false$ が，あるモデル候補 $M \in \mathcal{M}$ に置換 σ のもとで violated である時，M を棄却する $(\mathcal{M} := \mathcal{M} \setminus \{M\})$.

二つの規則のいずれも適用できなくなった時点で，節集合の全モデルが $\mathcal{M}$ の要素として得られる．$\mathcal{M} \neq \phi$ なら節集合は充足可能であり，$\mathcal{M} = \phi$ とき，その節集合が充足不可能であることがわかる．

3　FGHC のモード解析

本稿で述べるモード解析は，[2] の形式化に基づく．[2] では，先ず述語毎にモード制約を論理式で表し，その論理式で表されている制約を有理木 (rational tree) を用いて表現する．プログラム全体のモード解析は，これら有理木の単一化に帰着される．

一方，ここで提案するモード解析法は，論理式で表されたモード制約を MGTP 節に変換する．そして，プログラム全体のモード解析は，この節集合のモデルを計算することに帰着される．本節では次に示すプログラムを例にして，モード制約の表現法の概略を述べる．詳細は [2] を参照されたい．

```
s([],          D    ) :- true | t(D).
s([push(X)|S],D    ) :- true | s(S,[X|D]).
s([pop(X)|S], [Y|D]) :- true | X=Y, s(S,D).
```

第一引数は，[] か cons セルに具体化されるのを待っているので，入力であることが分かる．これを，$m(<s,1>) = in$ と表すことにする．$<s,1>$ は述語 s の第一引数であることを表す．例えば，第一引数の cons セルの car 部は，$<s,1><.,1>$ ($<.,1>$ で car 部，$<.,2>$

A Mode Analyzer for FGHC Programs in a Model Generation Theorem Prover
Miyuki Koshimura and Ryuzo Hasegawa
ICOT Research Center
4-28, Mita 1-chome, Minato-ku, Tokyo 108 Japan

でcdr 部を表す) という具合に<>を連結して表す．この記法によって，項やアトムの任意の部分項を指定することができ，これをパスと呼ぶことにする．また，$m()$ は引数で示される位置のモードを表す．

第三節の第二引数より同様に $m(<s,2>) = in$ が分かる．第二節は，cons セルの car 部分に $push()$ が具体化されるのを待っている．これは，$m(<s,1><..1>) = in$ と表す．第三節からも同じ式が導かれる．また，第二節のボディ部で，s の第二引数を cons セルで具体化し呼んでいる．つまり第二引数は入力であり，$m(<s,2>) = in$ が分かるが，これは既に述べたように第三節のヘッド部からも導かれる．

さて，第一節の変数 D に着目してみよう．これは，ヘッド部とボディ部に現れている．これより，s の第二引数のモードは，t の第一引数のモードに等しいことが分かる．これを $m/<s,2> = m/<d,1>$ と表す．m/p は $\forall q((m/p)(q) = m(pq))$ を満たす関数である．同じく第二節変数 D，S，X に着目し，$m/<s,2> = m/<s,2><..2>$，$m/<s,1><..2> = m/<s,1>$，$m/<s,1><..1><u,1> = m/<s,2><..1>$ が分かる (u は，$push$ の略)．

第三節の X と Y は共にヘッド部に現れ，ボディ部でユニファイされている．これより，X と Y のモードは逆であると分かる．これは，$m/<s,1><..1><o,1> = \overline{m/<s,2><..1>}$ と表す (o は，pop の略)．$\overline{m()}$ でモードの反転を表す．

4 MGTP 上のモード解析器

本節では，前節で述べた制約式の MGTP 節への変換について述べる．

前節で述べた制約式は，$m() = in$(または，$m() = out$) と $m/p = m/q$(または，$m/p = \overline{m/q}$) という形式の二つに分けられる．大雑把にいうと MGTP では，$m() = in$ を $true \rightarrow m() = in$ なる正節に，$m/p = m/q$ を $m/p \rightarrow m/q$ と $m/q \rightarrow m/p$ の二つの混合節として表現する．

表1に前節の制約式を MGTP 節に変換した結果を示す．ここで，$m(Path, Mode)$ でパス $Path$ のモード $Mode$ を表すものとする．また例えばパス $<s,1><..1>$ はリストを用いて，$[s/1,../1]$ に変換されるものとする．

さらに，同一パスでモードが反転しているモードの不整合は，次のような負節によって発見することができる．$m(P,M), m(P,\overline{M}) \rightarrow false$.

表 1: 制約式 → MGTP 節変換

制約式	MGTP 節
$m(<s,1>) = in$	$true \rightarrow m([s/1], in)$
$m(<s,2>) = in$	$true \rightarrow m([s/2], in)$
$m(<s,1><..1>) = in$	$true \rightarrow m([s/1,../1], in)$
$m/<s,2> = m/<d,1>$	$m([s/2\|X], M) \rightarrow m([d/1\|X], M)$ $m([d/1\|X], M) \rightarrow m([s/2\|X], M)$
$m/<s,2> = m/<s,2><..2>$	$m([s/2\|X], M) \rightarrow m([s/2,../2\|X], M)$ $m([s/2,../2\|X], M) \rightarrow m([s/2\|X], M)$
$m/<s,1><..1><u,1> = m/<s,2><..1>$	$m([s/1,../1, u/1\|X], M) \rightarrow m([s/2,../1\|X], M)$ $m([s/2,../1\|X], M) \rightarrow m([s/1,../1, u/1\|X], M)$
$m/<s,1><..1><o,1> = \overline{m/<s,2><..1>}$	$m([s/1,../1, o/1\|X], M) \rightarrow m([s/2,../1\|X], \overline{M})$ $m([s/2,../1\|X], M) \rightarrow m([s/1,../1, o/1\|X], \overline{M})$

5 おわりに

MGTP を用いたモード解析法のアイディアを述べた．本方法では，局所的 (述語単位) にモードの制約式を求め，大域的解析を MGTP に任すので分割コンパイルとの相性もよい．また解析の仕組みが単純であるという点も，実用的解析器を作成する上での利点になるものと思われる．

現在，解析器のプロトタイプを開発中である．そして，小さな FGHC プログラムに対する実験により，幾つかの問題点も明らかになりつつある．

本稿では，モードの種類として in と out の二種類しか考慮しなかったが，この抽象化では粗過ぎて，解析できないプログラム特性もある．例えば，例プログラムの第二節のボディ部と第三節のヘッド部から同じ制約 $m(<s,2>) = in$ が得られたが，これを区別したくなる状況もある．モード in/out の意味について，再考の必要があるものと思われる．

参考文献

[1] H. Fujita and R. Hasegawa, A Model Generation Theorem Prover in KL1 Using Ramified-Stack Algorithm, In *Proc. of 8th ICLP*, 1991.

[2] 上田,「Moded Flat GHC のモード解析」, 情報処理学会研究報告 93-PRG-12, 1993.

ポータブル KL1 処理系 KLIC の概要

仲瀬 明彦、藤瀬 哲朗、近山 隆
(財) 新世代コンピュータ技術開発機構

1 はじめに

プログラミング言語 KL1[1] は FGCS プロジェクトの核言語であり、並列推論マシン [2] 上の OS である PIMOS[3] や応用プログラムを記述することで高並列処理への有効性を実証してきた.

しかしながら, KL1 で記述された数多くのプログラムは, 並列推論マシン上でのみ動作するものが多い. そこで第五世代コンピュータの研究基盤化の一環として, KL1 により実現されているプログラム動作環境を汎用計算機上に実現する. 実現にあたり本環境は, プログラム開発のためのシステムであると同時に開発したプログラムを走らせるための環境となることを強調したい. そのため本環境は並列推論マシンを目標に, 問題によってはそれ以上の性能をもち, 同時に高い移植性を保つことを必要とする.

これらの問題を解決するために KL1 プログラムを プログラミング言語 C で記述されたプログラムに変換する方式の処理系 (以下 KLIC と呼ぶ) を開発中である. 本稿では KLIC の概略について説明し, 試験的に実装した逐次版の性能評価結果についても示す.

2 KLIC の開発方針

KLIC では以下の技術を利用して開発を進める.

- 汎用計算機技術
  KLIC が搭載されるのはワークステーション, 並列 UNIX システム, それらを汎用ネットワーク結合したもの等の汎用計算機システムである. これらのマシンの多くが Unix もしくは Unix をベースとした OS をもち, C を中心とした開発環境を抱えている. このように、ハードウェアを隠蔽するプラットホームとして Unix/C を利用することは、移植性の面から見て大変有利である。また, 優れた最適化を行なう C コンパイラがある等, マシン依存の低レベルな最適化は汎用計算機技術に頼れることも重要である.

A portable KL1 language processor KLIC
Akihiko Nakase, Tetsuro Fujise, Takashi Chikayama
ICOT, Mita Kokusai Bldg.21F, 4-28 Mita 1-chome, Minato-ku, Tokyo 108 Japan

- 並列推論技術
  並列推論マシン上の KL1 の実装技術、並列処理プログラム開発環境さらに応用ソフトウェアは, そのまま汎用計算機上の KL1 処理系で使用できる技術も少なくない. 特に分散データ管理・分散実行管理技術や並列処理プログラム開発環境のデザインが生かせることは大変有利である。
- 汎用技術と並列推論技術の親和性を高める技術
  KL1 の C 言語への効率的変換方式や Unix 環境下での効率的並列分散方式等, 上述の 2 つの技術を結び付ける技術を開発する必要がある. KLIC では, 特に静的解析を中心とした並列論理型プログラミング言語の解析技術 [4], [5] により最適化を行なうことを重要視する.

3 実装の基本方式

KLIC では, 図 1 に示す通り KL1 プログラムを C プログラムに変換する基本方式を採用した. 実行手順例を以下に挙げる.

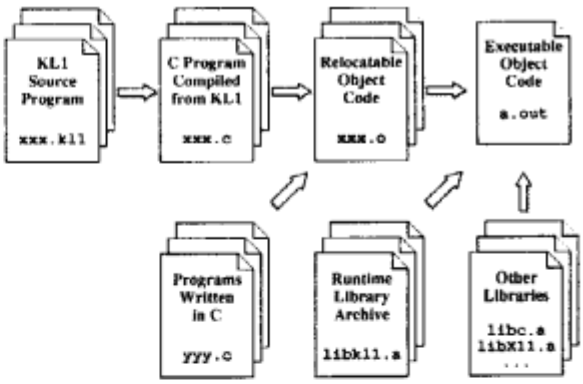

図 1: 実装の基本方式

1. KL1 プログラムを C プログラムに変換
2. C に変換されたプログラムを C コンパイラでコンパイル
3. オブジェクトと実行時ライブラリをリンクし、実行形式を作成
4. この実行形式オブジェクトを普通の Unix のプロセスとして実行

Lisp や Prolog 等のようにデバッガその他すべての機能を満載した実行環境を提供する方式ではなく，KLIC では実行される KL1 プログラムのみを C へ変換する環境を提供する．

一般に機械語への直接の変換より C に変換する方が，実行効率面で劣ることが予想される．しかしながら以下の面でメリットがある．

- 多様なシステムへ簡単に移植できる．
- 低レベル，マシン依存の最適化は C コンパイラに任せられる．
- 既存プログラムと容易にリンクして使える．

特に他言語プログラムとのリンク方式を提供することにより，KL1 の言語システムの自由な拡張や，他言語プログラムのプロセス群の KL1 による並列制御が可能になる．

4 試験的実装の性能評価

性能上鍵となる機能を中心に，一部の仕様のみ実現した逐次版試験実装により性能評価を行なった．naive reverse (30 要素,1000 回繰り返し) の結果を表 1 に示す．

表 1: 試験的実装の評価 (その 1)

System	Speed	Code Size
Symmetry S81	118 KLIPS	608 bytes
Sun-3/260	205 KLIPS	532 bytes
SparcStation 2	985 KLIPS	640 bytes
SparcStation 10/30	2,000 KLIPS	640 bytes
SparcCenter 2000	2,265 KLIPS	680 bytes
DEC alpha 7000	3,701 KLIPS	928 bytes
SICStus (compactcode)	482 KLIPS	320 bytes
SICStus (fastcode)	1,053 KLIPS	736 bytes

(KLIC: GCC -O2 オプション，GC 時間含む)
(SICStus Prolog: ver.2.1 on SS 10/30, GC 時間含まず)

速度性能に関してもまずまずの結果を得ている．また速度向上度に関して汎用計算機搭載の効果がみられ，また C 言語を中間言語とすることで核部分の高い移植性も確認できた．

次に，論理型言語で使用される他のベンチマークプログラムについて，fastcode でコンパイルされた SICStus Prolog プログラム (直接機械語が生成される) と比較した結果を表 2に示す．実行は SparcStation 10/30 上で行なった．

速度性能では，SICStus Prolog の 1.4 倍から 2 倍の性能が出ている．

コードサイズについては，通常のリスト処理のみを使うベンチマークについては SICStus Prolog と比較しても妥当と思われる．しかし，算術演算やガード部のインデキシングにおいて，若干最適化できてない部分があり，それらの処理を多く含んでいるベンチマークについては，SICStus Prolog より大きなコードが生成されてしまう．これらの最適化は順次行なってゆく予定である．

表 2: 試験的実装の評価 (その 2)

	実行時間		コードサイズ (bytes)		
program	KLIC	SICStus	KLIC	SICStus	S/K
qsort	1	1.41	1160	720	0.62
hanoi	1	1.43	504	496	0.98
prime	1	1.66	1488	768	0.52
lqueen	1	1.84	4344	1088	0.25
mgtp	1	1.68	24648	8448	0.34

5 まとめと課題

KLIC の開発方針と、暫定版処理系の評価について述べた。

KLIC と並列推論マシンとの開発方針に関する大きな違いは以下のことである．

並列推論マシン: 動的処理の効率化に重点
KLIC: 静的解析技術による効率化に重点

KLIC では並列推論マシンと比較してプロセッサが速くなったため，相対的にメモリは遅くなっている．このため，頻繁に利用するワーキングセットを小さくし，キャッシュヒット率を向上させるために generation GC を検討中である．

また KLIC の並列処理に於いては，並列推論マシンとは異なり専用ハードウェアや専用 OS をもたないため，プロセッサ間の通信性能，特にレスポンスが極端に悪くなることが予想される．このことに対処するためには，バッファリングやプログラム解析等による多レベル一括転送機能を実現し，通信頻度を低減する必要がある．

参考文献

[1] Ueda, K. et al.: Design of the kernel language for the parallel inference machine, *The Computer Journal*, December 1990.

[2] Goto, A. et al.: Overview of the parallel inference machine architecture (PIM), *Proceedings of FGCS'88*.

[3] Chikayama, T. et al.: Overview of the parallel inference machine operating system (PIMOS), In *Proceedings of FGCS'88*.

[4] Ueda, K. et al.: Moded Flat GHC and Its Message-Oriented Implementation Technique, submitted to New Generation Computing, 1993.

[5] Sastry, A. et al.: A Compile-Time Memory-Reuse Scheme for Concurrent Logic Programs, TR CIS-TR-91-24 of Univ. of Oregon, 1992.

遺伝的アルゴリズムを取り入れたタンパク質配列解析

戸谷 智之、石川 幹人、星田昌紀 *、小長谷明彦 **

(財) 新世代コンピュータ技術開発機構, 松下電器産業 (株)*, NEC(株)**

1 はじめに

我々は、タンパク質配列の類似性を解析する手法のひとつであるマルチプルアライメントの問題の解決を目標に研究を行なってきた。昨年、並列反復改善法を用いたマルチプルアライメントシステムを開発し、従来以上に、高品質なアライメント結果を短時間のうちに獲得できることを可能にした。今回は、さらなる解の高品質化を目指し、並列反復改善法に遺伝的アルゴリズムをとりいれ、並列反復改善法の発展を試みた。それについて、報告を行なう。

2 並列反復改善法

並列反復改善法 [1] は、Berger-Munson の反復改善法のアイデアをもとにしている。これは、従来から行なわれてきた手法と比較して、高品質の解を得られる有効な方法であるが、N 本の配列群に対して、2^N-1 通りのグループ分割が考えれ、実用規模（数十本）のアライメントを得るには莫大な時間を要してしまい、現実的とは言えなかった。我々は、それを並列化し、さらに独自のヒューリスティクスを導入することで、実用規模の問題を短時間のうちに解き、高品質のアライメントを獲得できるシステムに発展させた。（図 1）

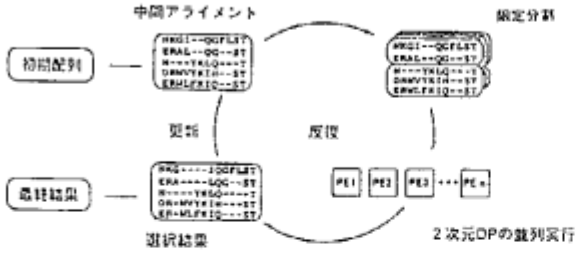

図 1: 並列反復改善法

(1) アライメントする配列群を 2 つのグループに分割する複数の組合せを生成する。
(2) それぞれの分割組合せで、分割された 2 つのグループ間でのグループ間 DP を行なうが、その処理を複数のプロセッサで並列に実行させる。

(3) 各グループ間 DP で得られた結果、アライメントの評価値として最も良いものを次のサイクルの配列群として、(1) の処理に戻る。

さらに、(1) の処理で、ヒューリスティクスを導入して、分割の場合の数を制限することで、計算時間の削減に成功した。それは、配列群を 2 つのグループに分割する際に、偏りをつけた分割の方が改善に効果があることが実験により確かめられたからであった。

このようにして、我々の並列反復改善法は、Berger-Munson の手法をより実用的な規模の問題で、質の高い解を短時間のうちに解決できるように発展させたのであった。ただ、実験を繰り返していくうちに、問題によって、比較的質の良くないアライメントが得られることがあった。解空間上で、局所解につかまってしまったと考えられるが、我々は、その解決に遺伝的アルゴリズムを取り入れることを考えた。

3 遺伝的アルゴリズムとそれを取り入れた並列反復改善法

遺伝的アルゴリズム（Genetic Algorithm 以下、GA）[2] は、生物の進化過程を発想のもとにした、組合せ問題の解決するための汎用の探索アルゴリズムである。 GA は、適応度の高い個体が生き残っていくように設計されたものであり、解空間上で複数の解（個体）の集団を操作の対象としている。GA では、“mutation”、“crossover”、“selection” の 3 つのオペレーションが用いられる。図 2 のように、これらのオペレーションを組み合わせて行なわれるのが GA である。我々は、この GA の発想をもとに、並列反復改善法を拡張することを考えた。従来の反復改善法では、

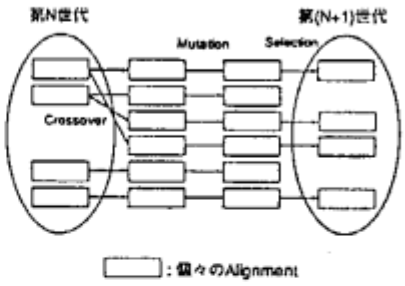

図 2: 遺伝的アルゴリズム

Parallel Iterative Aligner with Genetic Algorithm
Tomoyuki Toya, Masato Ishikawa, Masaki Hoshida*, Akihiko Konagaya**
ICOT, Matushita Electric Industrial Co.*, NEC Co.**

並列に実行を行なったグループ間DPの結果で、最も良い解だけを選ぶことを行なってきた。つまり、山登り法を行なってきたと言える。しかし、それでは局所解につかまることもあるので、それをGA的に発展させることで回避しようというものである。GAの適応度はアライメントの評価値に相当する。また、アライメントの解そのものを個体とした。我々は、GAにおける3つのオペレーションに相当するものを並列反復改善法では以下のように定義した。

mutation 個体の1部分を変化させるという意味で、解（配列群）の1本を抜きだし、その1本と抜かれた残りの配列群の間でDPによるアライメントを行なうこととした。GAの本来のmutationが持つ意味とは異なって、各個体を改善させるオペレーションとなっている。

crossover crossoverは、2つの個体間での解の混合と解釈できる。具体的には図3に示すようなオペレーションを行なう。選ばれた2つの個体（配列群）で、他の個体と“交換する配列”と“交換しない配列”をやはりランダムに決定する。そうして決定された“交換する配列”群を個体間で交換し、その後グループ間でDPを用いてアライメントする。

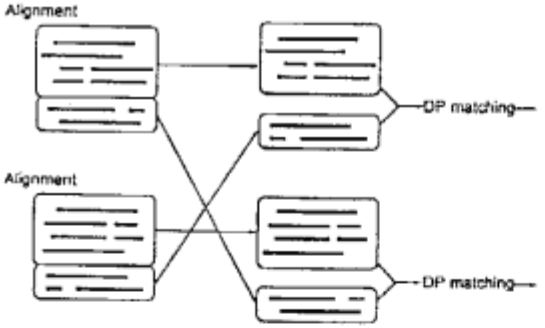

図 3: crossover

selection selectionは、今ある個体群から、指定した割合で、適応度の高い個体を残し、適応度の低いものは捨てる操作を行う。捨てられた分の個体数だけ、生き残った個体間でcrossoverを実行し、全体の個体数は一定を保つようにしている。

4 GA反復改善法の評価

我々は、並列論理型言語 KL1 を用いて、GA反復改善法のアライメントシステムを構築し、並列計算機PIM上で並列実行させた。

図4は、従来の並列反復改善法の結果とGA反復改善法の結果を比較するグラフである。問題は、いずれも80個のアミノ酸からなる22本の配列群であり、実用的な規模の問題である。

いずれも、256プロセッサをもつPIM上で実行した。

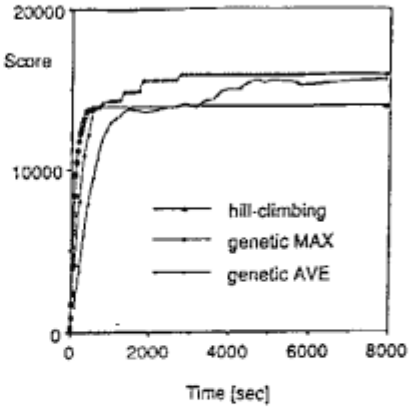

図 4: 実験結果

並列反復改善法では、ヒューリスティクスとして、分割する際に、“1本と残り”、もしくは“2本と残り”になる分割のみに制限する手法を用いている。この場合、22本の配列があるため、$253({}_{22}C_1 + {}_{22}C_2)$の分割ができるため、マスタープロセッサ1つを加えた、254プロセッサでの並列実行ということになる。25サイクル約11分で改善が見られなくなった時点で実行は終了し、得られた解の評価値は13903であった。

GA反復改善法では、256プロセッサ使用可能なので、全体の個体数を255として実行した。各世代の時間は80秒とし次の世代に生き残る個体を上位90%とした。MAXとあるのは、各世代で最も高い評価値をプロットしたものである。AVEとあるのは、各世代での全個体の平均評価値をプロットしたものである。MAXを見ると、山登り法の場合とほぼ同じぐらいの時間で同レベルの解に達していることが分かるであろう。GAはさらに実行が続き、MAXの評価値は15775、AVEも15121にまで達した。

5 まとめ

今回は、GAの発想をもとに並列反復改善法の発展させたマルチプルアライメントのシステムを実装し、より質の高いアライメント結果が得られたことを報告した。今後は、個体数、生存率、世代時間などの、パラメータと性能についての相関について、実験を行なっていきたいと考えている。

参考文献

[1] 星田昌紀，石川幹人，広沢誠，戸谷智之，十時泰：“並列反復改善法によるタンパク質配列のアライメント”，情報処理学会第27回情報学基礎研究会，13-24, 1992.

[2] Goldgerg,D.E.: “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley Publishing Company,Inc., 1989.

演繹オブジェクト指向データベース言語を用いた遺伝子知識ベースの記述 *

広沢 誠 †　　田中 令子　　石川 幹人
(株) 日立製作所 ‡　　IMS　　ICOT

1 はじめに

生物の遺伝子を解析することは、生命現象を解き明かすために必須の技術である。現在、バイオテクノロジーの進歩により、すでに検出され、解析されるべき遺伝子は急激に増加している。この大量の遺伝子を自動的に解析し、生物学的情報を抽出するためには、高速の解析アルゴリズムと共に、知識処理の導入が必要である。しかし、現在、遺伝子関係のデータは、知識処理に利用することを念頭に構築されていないため、高度の知識処理を行なうシステムを作ることは困難である。

我々は、生物学的知識を表現する方式について研究してきた。今回は、遺伝子のモチーフという情報を発見するという課題を念頭において知識表現の研究をすることにした。我々は、既にモチーフ発見システムを開発している [Hirosawa *et al.*1993]。この研究では、暫定的な知識表現を用いていたので、今回は、知識ベースを演繹オブジェクト指向データベースの言語である *QUIXOTE* [Yasukawa *et al.*1992] を用いて作成し、この知識ベースを基にシステムを再構築した。この結果、システムの性能が向上した。今回は、オブジェクト指向的な表現により生物学的概念の適切な表現が可能になったことを示す。

2 遺伝子の解析とは

生物が持つ遺伝子の解析は、生物学の分野だけではなく、医学では癌のメカニズムの解析にも必須である。遺伝子解析とは、新たな遺伝子が実験により同定された時、この遺伝子が生体内で担っている機能を推定することである。生物の遺伝子は、細胞の核の中に DNA としてコードされ格納されている。この DNA は必要に応じ mRNA に転写され、tRNA の等の働きによりアミノ酸の鎖に変換される。このアミノ酸の鎖は、適切な形に成形され蛋白質として機能するようになる。

蛋白質の鎖のユニットとして使用されるアミノ酸は２０種類ある。各々のアミノ酸を一つのアルファベットを用いて表わすと、蛋白質をアルファベットの配列で表現することができる。例えば、"GIVEQCCTSICSLYQL" は、インシュリンの一部を示したアミノ酸の配列である。ここで、G はグリシンというアミノ酸である。このような表現方法をとると、遺伝子の解析とは、アルファベットの配列として表現されている遺伝子のアミノ酸の配列から、この遺伝子がコードしている蛋白質の機能を推測することとなる。

遺伝子の解析のために必要な情報としてモチーフがある。モチーフとは、ある種類の蛋白質が共通に含むアミノ酸の配列パターンである。モチーフの中には、それに対応する蛋白質の部位が果たす機能 (ex. DNA に結合する) が判明しているものもある。このように、モチーフは蛋白質の機能を予測するのに重要な情報である。

3 遺伝子の解析システム

我々は、すでに発見されている遺伝子とそれに対応する蛋白質が持つモチーフが判明している時、これを事例として、入力された蛋白質の持つモチーフを発見するシステムを開発した。これは、入力された蛋白質の機能も推測する。システムの構成を第１図に示す。以下、その構成と動作を簡単に説明する。

まず、*Motif Finder* は、Aligner が入力配列に対して行なった 類似性解析結果を解析し、配列の共通部分を検出する。*Motif Generator* は、この解析結果を基に、モチーフの候補を可能性の高いものから順に生成していく。可能性の高いものというものは Biological Knowledge Base に含まれるモチーフの情報に基づくいて生成したモチーフ候補である。

具体的には、Motif Generator は、*Motif Rule Base* に含まれる ルールを、優先度にしたがって実行し、モチーフ候補を生成する。(詳しい記述は、[Hirosawa *et al.*1993] にある)。ルールは、必要であれば *Biological Knowledge Base* の *Prosite** と *User* を参照する。これらの知識は、演繹オブジェクト指向データベース言語である *QUIXOTE* により記述されている。*Motif Tester* は、生成されたモチーフが生物学的統計的基準を満たしているかを調べる。Motif Finder でモチーフとして容認された場合には、これを Biological Knowledge Base の *Discovery* に登録される。

4 Biological Knowledge Base

Biological Knowlege Base は、３つのサブデータベースに分かれている。各々は,Prosite*,User, Discovery である。Prosite* には、既存のモチーフデータベースとして代表的な Prosite[Bairoch 1991] に含まれているモチーフなどが階層的に表現されている。Prosite* におけるクラス分けは、Prosite で採用しているものを基本にしているが、これでは十分ではないため、生物学的な知識にしたがい再分類を行なっている (以前の分類方法に復元することも可能である)。

User は、ユーザーが文献などから得たモチーフを登録する場所である。また、Discovery は、システムが Prosite* や User に登録されている知識を利用して発見したモチーフを登録する場所である。User や Discovery が用いているクラス構造は Prosite* で用いているものと同じものである。

Figure 2 に Prosite* の一部を視覚化したものを示す。図では、キナーゼというリン酸基を転移する蛋白質 (kinaseGroup) は、４つのサブグループに再分類され、その１つがプロテイン・キナーゼ (protein-kinase) を含んでいる。それは、さらに、チロシン・キナーゼ (ty-kinase) とセリン／スレオニン・キナーゼ (s-th-kinase) とに分類されている。

各階層には、対応する蛋白質が持つモチーフが登録されている。プロテイン・キナーゼは、protein-kinase-general というモチーフのエントリーを持っており、

---

*Description of Genetic Knowledge Base using Deductive Object-Oriented Database Language

†Makoto Hirosawa *et al.*

‡Hitachi System Development Lab. 1099 Ohzenji Asao Kawasaki 215

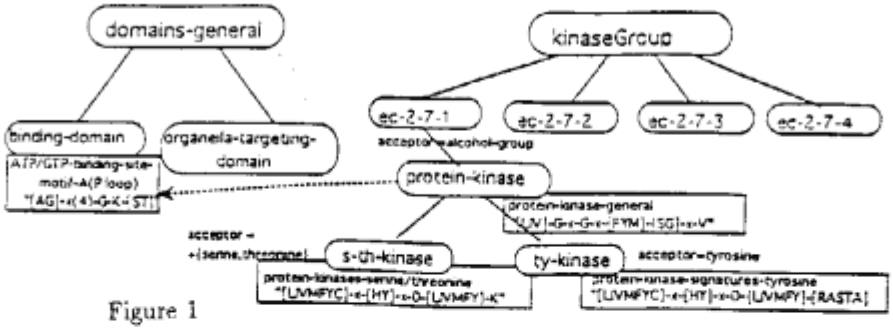

Figure 1

```
kinaseGroup >= ec-2-7-1 ;;
kinaseGroup >= ec-2-7-2 ;;
kinaseGroup >= ec-2-7-3 ;;
kinaseGroup >= ec-2-7-4 ;;
    ec-2-7-1 >= protein-kinase ;;
protein-kinase >= s-th-kinase;;
        protein-kinase >= ty-kinase;;
;
```

Figure 3

```
kinaseGroup::protein-kinase[name = "Protein kinases general"]/
      [pattern = "[LIV]-G-x-G-x-[FYM]-[SG]-x-V",
       source   = prosite*
       otherMotif = domains_general:binding_domain
         [name = "ATP/GTP-binding site motif A (P-loop)"]    ];;
kinaseGroup::protein-kinase[name = "Protein kinases ATP-bind"]/
           [pattern = "A-x-K",
            source   = user      ];;
```

Figure 4

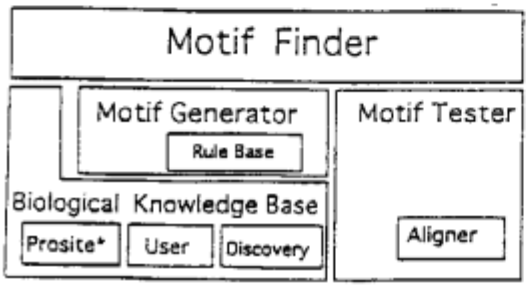

Figure 2

このパターンは "[LIV]-G-x-G-x-[FYM]-[SG]-x-V" である（ここで、[SG] は S と G のどちらかという意味であり、x はどのアミノ酸でもよいという意味である）。チロシン・キナーゼは、protein-kinase-signatures-tyrosine というモチーフのエントリーを持っている。また、チロシン・キナーゼ のモチーフとして、それ自体に登録されているものの他に、上位概念のプロテイン・キナーゼが持つモチーフも参照することができる。これは、*QUIXOTE* のメソッドを介して可能となる。

また、プロテイン・キナーゼは、Prosite では binding-domain というクラスに属している '[AG]-x(4)-G-K-[ST]' というモチーフパターンも持っている（Prosite では、この情報は自然言語で書かれていたために、計算機では読みとれなかったが、我々はこの情報をデータベースに陽に記述し計算機で読みとれるようにした）。

QUIXOTE を用いた記述を Figure 3 （知識の階層構造）と Figure 4 （モチーフ）を記述したものを に示す。Figure 4 には、2 つのモチーフのエントリーがある。この内、上のモチーフは Prosite* に属するものであり、下のモチーフは User に属するものである。どちらに属するかは source の属性値により記述する。source の属性値 としては、prosite*、user、discovery のどれかをとることができる。

Prosite* の プロテイン・キナーゼ（一番上のエントリー）には otherMotif という属性があるが、これは、Prosite では、他のクラス（domains-general:binding-domain）に含まれているモチーフを、システムがプロテイン・キナーゼのモチーフとしても参照可能にするためのものである。これを可能にするためのメソッドも、*QUIXOTE* で記述されている。このような多重継承は、演繹オブジェクト指向デ--タベースの一機能である。

チロシン・キナーゼとは、acceptor の属性値として、tyrosine を持つ。この二つのクラスの上位クラスであるプロテイン・キナーゼは、acceptor の属性値として、さらに上のクラスである ec-2-7-1 から、alcohol-group を継承している。したがって、蛋白質が発見された時、この蛋白質がプロテイン・キナーゼとのみ判明している場合には acceptor の属性値が alcohol-group であると推論する。しかし、さらに蛋白質がチロシン・キナーゼであると特定される場合には、システムはさらに正確な情報を提示することができる（tyrosine は、alcohol-group の一員である）。これも、演繹オブジェクト指向データベースの一機能である。

5 まとめ

生物学的情報であるモチーフに関する知識を、演繹オブジェクト指向データベース言語 *QUIXOTE* を用いて記述した。オブジェクト指向的な表現方法により生物学的な概念が有効に表現されることを示した。また、演繹オブジェクト指向データベースの演繹的側面が、論理型言語で記述したシステムと相性が良いので演繹オブジェクト指向データベースが知識処理を用いる遺伝子解析にも有効であることが判明した。

参照文献

[Bairoch 1991] Bairoch,A. Prosite : A dictionary of Protein site and pattern : User manual Release 7.00, May 1991.

[Hirosawa *et al.* 1993] Hirosawa,M. *et al.* Protein Multiple Sequence Alignment using Knowledge. *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*,Vol.1 pp803-812,1993.

[Yasukawa *et al.* 1992] H.Yasukawa *et al.* Objects, Properties, and Modules in Quixite. *Proc. of FGCS92, pp89-112, 1992*