ICOT Technical Memorandom: TM-1126

TM-1126

第五世代コンピュータ デモンストレーション一覧

佐藤　博

October, 1991

ICOT

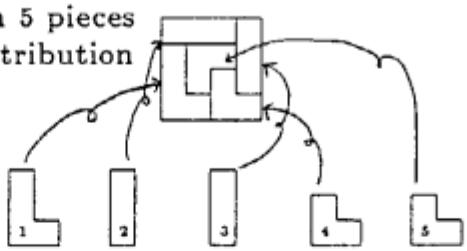
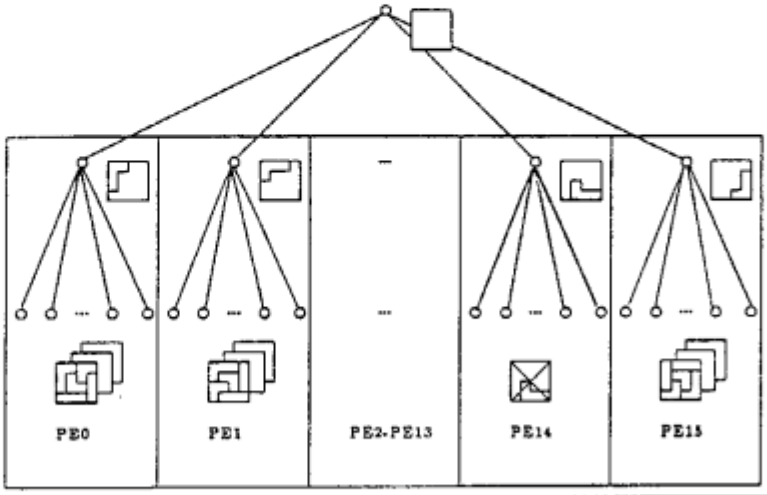
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

第五世代コンピュータ
デモンストレーション一覧

（平成２年10月31日現在）

Title	Pentomino — Packing Piece Puzzle Solver
Purpose	Dynamic load balancing scheme for OR-parallel search programs is studied. Multi-level load balancing scheme is proposed, and evaluated by implementing all-solution exhaustive search Packing Piece Puzzle (Pentomino) solver program.
Outline & Features	Packing Piece Puzzle is a puzzle, consisting of a rectangular box and a collection of pieces with various shapes. The problem is to find all possible ways to pack the pieces into the box. This puzzle is known as the Pentomino puzzle, when the pieces are all made up of 5 squares. This is a typical OR-parallel search program. A multi-level dynamic load balancing scheme is developed to highly utilize the processors. **Program structure:** An OR-parallel exhaustive search. **Load distribution:** Tasks are generated by a master processing elements (PE), and are distributed to idle PEs, in order to balance work loads. To overcome the task supply bottleneck at the master PE, multi-level load balancing is introduced.
System Configuration	Packing Piece Puzzle with 5 pieces Search Tree and Load Distribution  

1 Overview

In the demonstration, packing piece puzzle with 10 pieces (Fig.1) is solved with increasing number of processing elements (PEs), and speedup figures are shown.

The demonstration is carried out as follows.

- Program is executed on 16 processors with simple load balancing scheme.
- Load balancing can be observed real-time in the performance meter window.
- Program is executed on 64 processors with simple load balancing scheme.
- Task supply bottleneck can be observed in the performance meter window.
- Program is executed on 64 processors with multi-level load balancing scheme.
- Near-linear speedup is obtained.

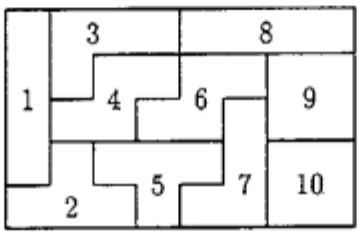


Figure 1: Packing Piece Puzzle

2 Description of the program

To solve this puzzle, the program starts with the empty box, and finds all possible placements of a piece to cover the square at the top left corner. Then, for each of those placement, it finds all possible placements of a piece (out of the remaining pieces) to cover the uncovered square which is the topmost leftmost, and so on until the box is completely filled. Each partly filled box defines an OR-node, where the possible placements of a piece to cover the uncovered topmost leftmost square define child nodes.

The program does a top-down exhaustive search of this OR-tree. Here, deepening the tree depth corresponds to place one piece in the box. The number of OR-nodes increases as the search level deepens, but since some OR-nodes are pruned when there are no more possible placements, number of OR-nodes decreases below a certain tree depth.

3 Load balancing scheme

Load balancing is done on master PE by partitioning a program into mutually independent subtasks (Subtask Generation), and by distributing subtasks to idle PEs so as to balance work loads (Subtask Allocation). To detect idle PEs, on-demand distribution method is

D e t a i l s (2/3)

utilized. When a PE becomes idle, it sends a message to the master PE, requesting a new subtask. Subtask generation is done until the search reaches the certain depth in the tree. However, as the number of processors increases, the rate of subtask execution eventually becomes larger than the rate of subtask supply. In other words, subtask generation becomes a bottleneck.

To overcome this bottleneck, we have introduced multi-level load balancing scheme. Each subtask generator is in charge of a certain fixed number of processors, which form processor groups (PG). N processors are grouped into M processor groups, therefore, each PG is composed with $\frac{N}{M}$ PEs and a certain PE in a PG is called group master PE.

In Fig.2, two-level load balancing scheme is shown. At the first level distribution, super-subtasks are distributed to idle PEs to balance the loads of PGs. At the second level, subtasks are distributed to idle PEs to balance the loads of PEs which belong to a PG.

This scheme is scalable to any number of processors because of this multi-level structure.

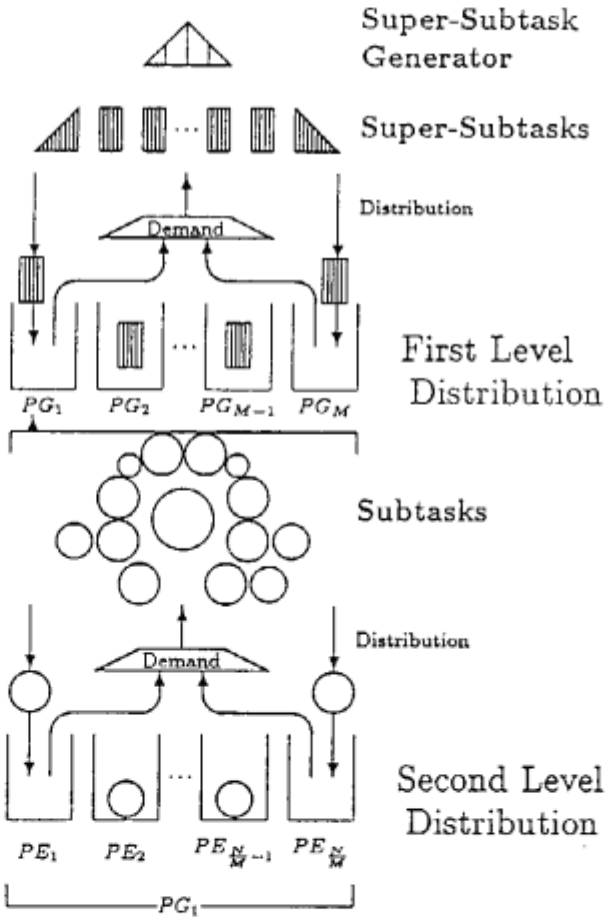


Figure 2: Structure of Multi-Level Load Balancing

4 Speedup Measurement

Execution times are measured for one-level load balancing and two-level load balancing. Speedup (S_N) is defined as the ratio of execution time on 1 PE (T_1) to N PEs (T_N), and calculated by $\frac{T_1}{T_N}$, and it is described in Figure 3.

Speedup of one-level load balancing becomes saturated because of the subtask generation bottleneck. However, it is improved by two-level load balancing, and near-linear speedups are obtained: 7.7 with 8 PEs, 15 with 16 PEs, 28.4 with 32 PEs, 50 with 64 PEs.

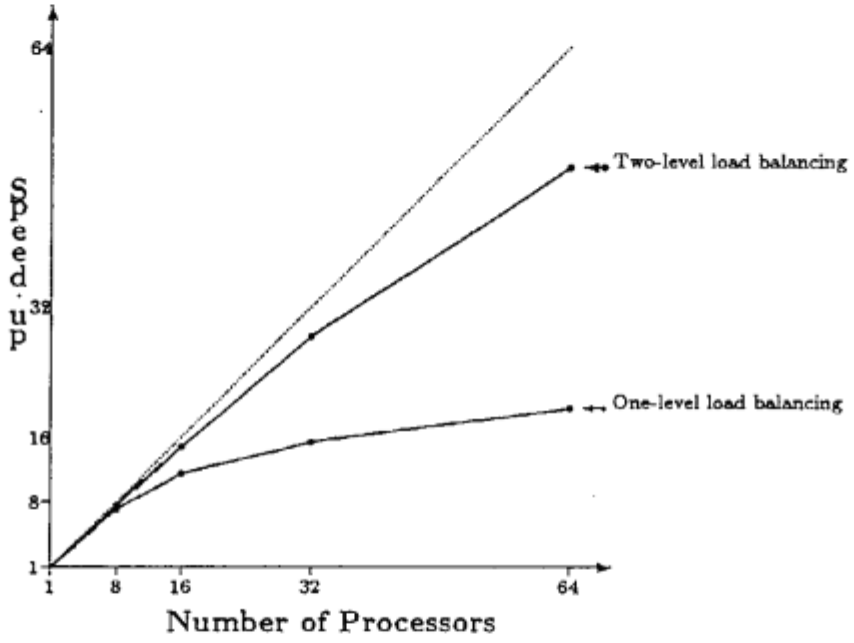
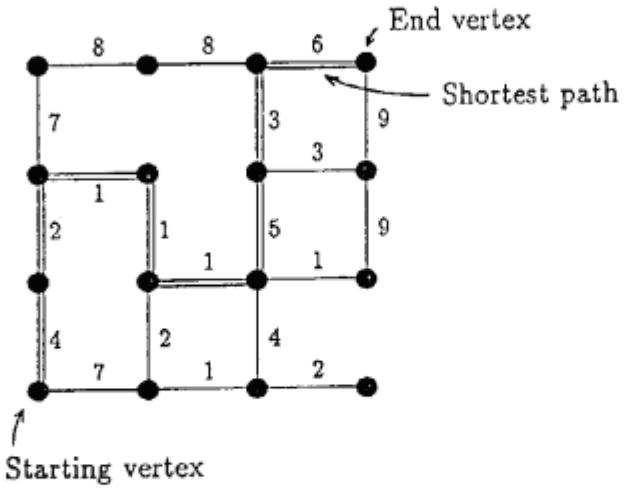


Figure 3: Speedups

5 Conclusion and Future Works

This scheme is efficient not only for OR-parallel search problems, but also applicable to general trees search problems including alpha-beta pruning problems, which does not involve frequent inter-processor communication.

This multi-level dynamic load balancing scheme is now availavle as a utility program to come with the operating system PIMOS.

Title	**Bestpath — Shortest Path Problem Solver**
Purpose	The problem of mapping intercommunicating processes on loosely-coupled multiprocessors is studied and evaluated by implementing a shortest path problem solver.
Outline & Features	**Problem** : The single-source shortest path problem is to find the minimum cost paths between a given starting vertex and all other vertices of a graph in which each edge has a non-negative cost. Large-scale grid graphs with tens of thousands of vertices are used in the demonstration. Edge costs are given by random numbers as the test data. **Algorithm** : Processes corresponding to each vertex exchange messages with each other. Each message contains path and cost from the starting vertex. A priority is attached to each message so that a message with lower cost is sent earlier than one with higher cost. Each vertex remembers the shortest path notified so far by the messages and its cost. **Load balancing** : Three different static mapping strategies are tried to get high processor utilization with low interprocessor communication.
System Configuration	Example 

Outline

The single-source shortest path problem is to find the minimum cost paths between a given starting vertex and all other vertices of a graph in which each edge has a non-negative cost. In the demonstration, the grid graph consists of forty thousand vertices. Edge costs are given by random numbers.

In the demonstrated program, processes are generated for each vertex in a graph and computation is performed by exchanging messages between processes. This algorithm requires less computation than the algorithm in which processes are forked for each candidate path. Priority control efficiently prunes the search branches, so the algorithm is as computationally efficient as Dijkstra's algorithm.

Algorithm

A message contains the path from the starting vertex to the receiver vertex and its cost. The computation is initiated by sending a message with an empty path and zero cost to the starting vertex. All the vertices remember the minimum cost to reach the vertex notified by the messages received so far. Initially, the cost remembered by each vertex is infinite (Figure 1).

When a message arrives at a vertex and the cost notified by the message is smaller than the minimum cost remembered in the vertex, the new cost is remembered and messages with better paths and costs are sent further to the adjacent vertices (Figure 2). If the message has a larger cost value than the known minimum, it is simply discarded.

Since a message is represented by a process, sending a message means creating a message process, while receiving a message means executing a message process. Each message process has a priority decided by the cost: a message with a lower cost is received earlier than one with a higher cost.

When there are no messages left in the graph, each vertex has the shortest path from the starting vertex and its cost.

Load Balancing

The heaviest part of the processing is communication, which is initiated at the starting vertex and propagates gradually to the whole graph like wave propagation. So, the processor utilization is expected to be higher as the grid is divided into smaller blocks.

Conversely, when the grid is divided into blocks for mapping, interprocessor communication arises at the boundaries of the blocks. So the more the grid is divided into smaller blocks, the more interprocessor communication occurs.

The program tries to attain a good compromise between communication localization and processor utilization.

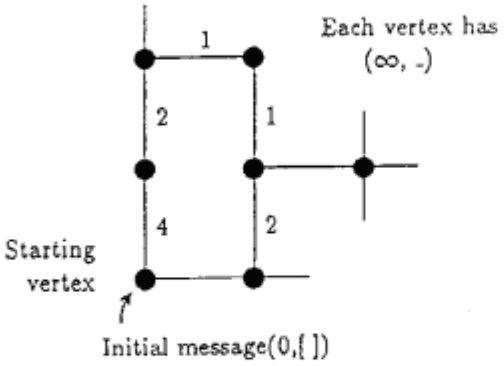


Figure 1: Initial state

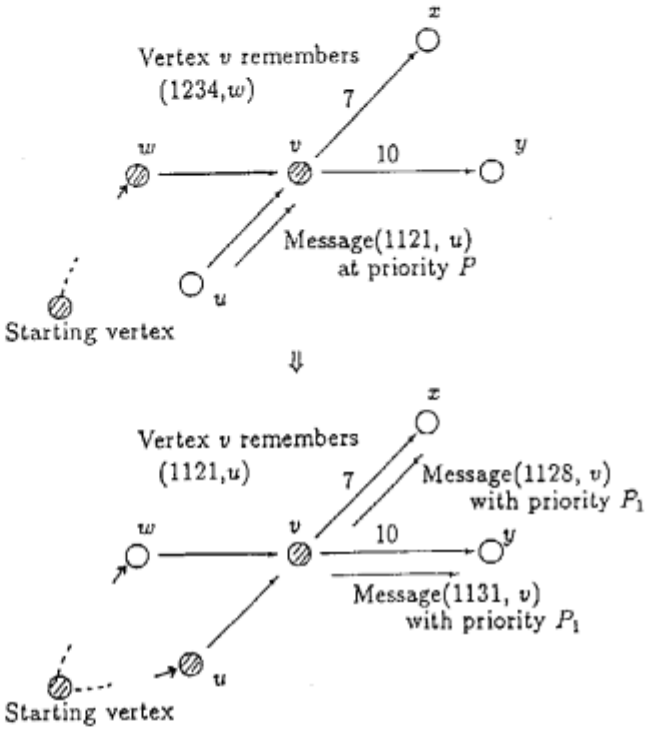


Figure 2: Message communication

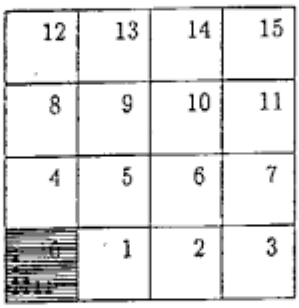
Mapping Strategies

The following three mapping strategies are tried. In each mapping, $p = q^2$ processors are employed.

Two-Dimensional Simple Mapping
Divide the grid into $q \times q$ blocks and map each block onto the corresponding processor.

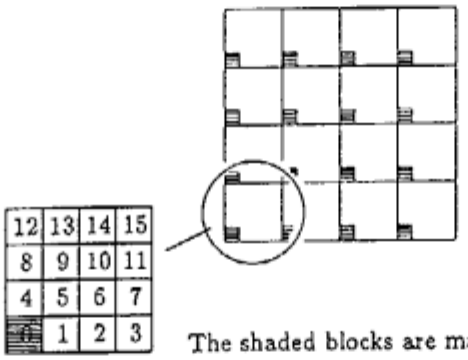
Two-Dimensional Multiple Mapping
Divide the grid into k super-blocks, each of which is again divided into p blocks just as in the two-dimensional simple mapping. Each processor is responsible for k blocks, one from each super-block.

One-Dimensional Simple Mapping
Divide the grid simply as p narrow rectangular strips and map them onto the processors.



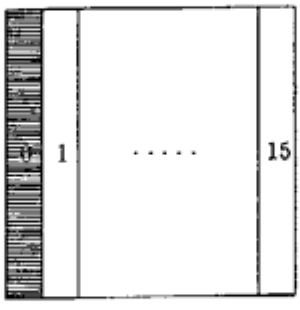
The shaded block is mapped onto processor 0.

Figure 3: Decomposition of a grid for two-dimensional simple mapping



The shaded blocks are mapped onto processor 0.

Figure 4: Decomposition of a grid for two-dimensional multiple mapping



The shaded block is mapped onto processor 0.

Figure 5: Decomposition of a grid for one-dimensional simple mapping

Performance Results for a 40,000-vertex grid

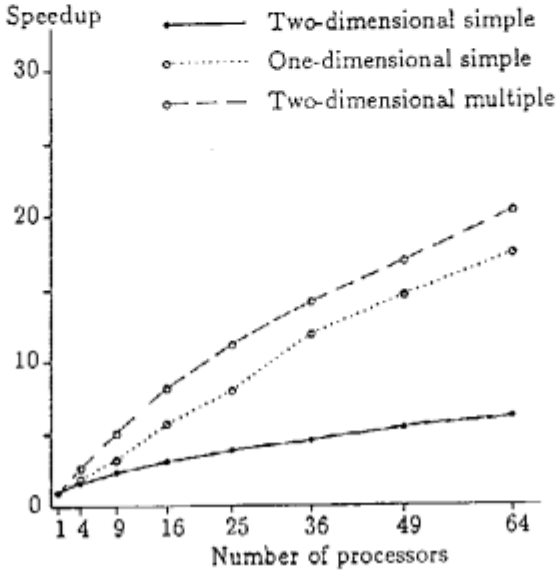


Figure 6: The speedup for various mappings and numbers of processors

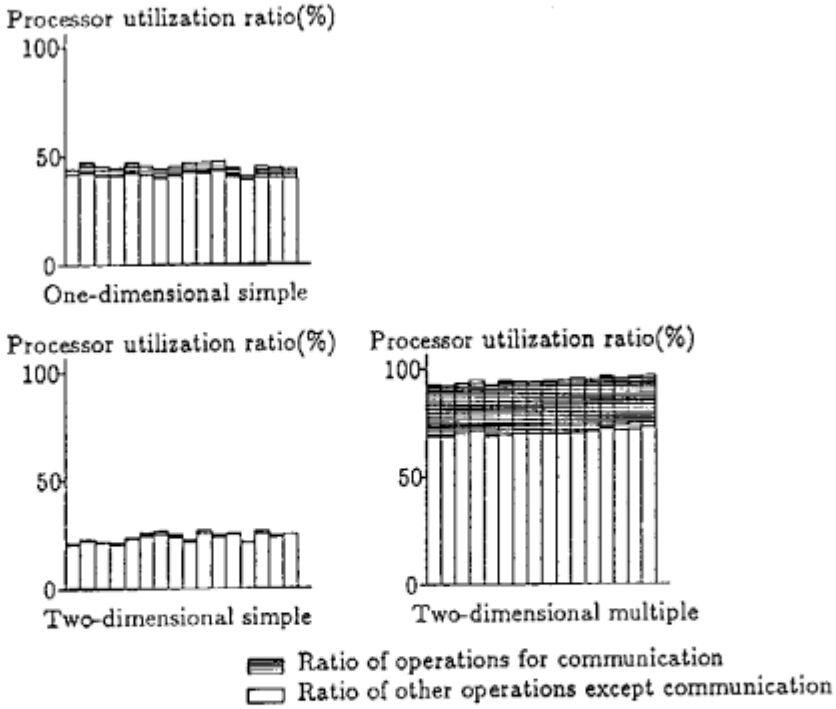


Figure 7: The processor utilization ratio for various mappings with 16 processors

Bestpath

— Shortest Path Problem Solver —

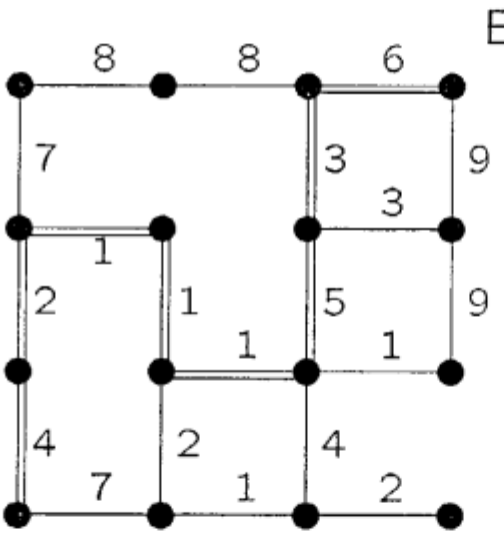
Kumiko Wada

ICOT 2nd Lab.

October 17th, 1990

Demonstration at the
Joint ICOT/DTI-SERC Workshop

What is the single-source shortest path problem ?



End vertex

Shortest path

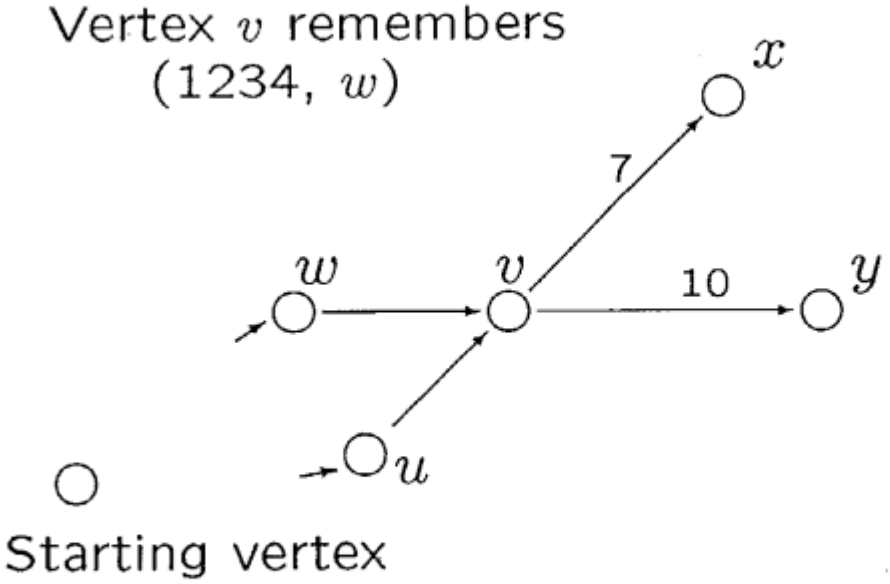
Starting vertex

The distributed algorithm

Each vertex process remembers the minimum cost path known so far

Initial values:

$(Cost_i,\ Path_i) = (\infty,$ undefined)

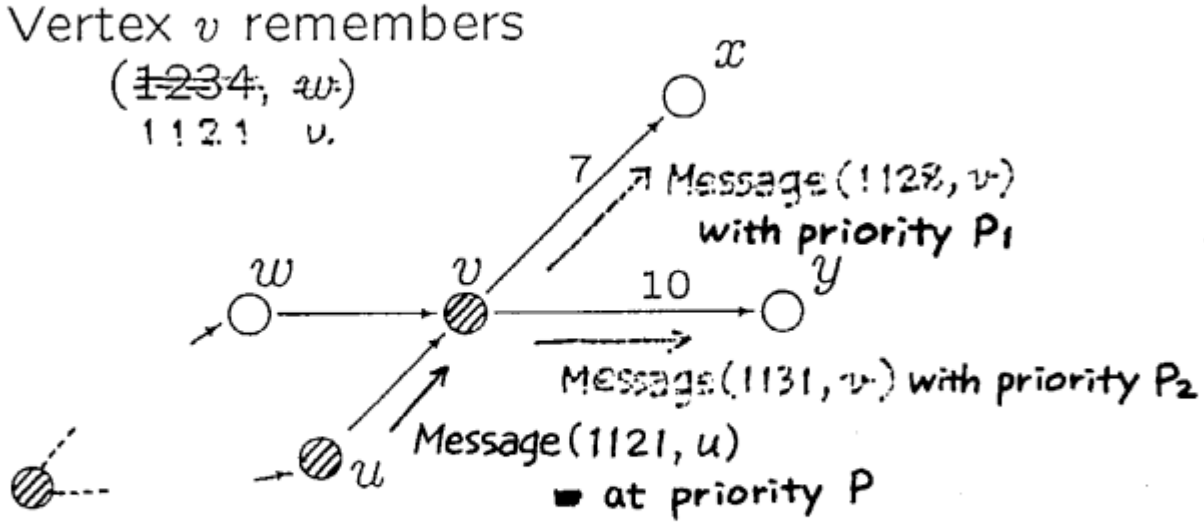


The distributed algorithm

Each vertex process remembers the minimum cost path known so far

Initial values:

$(Cost_i,\ Path_i) = (\infty,$ undefined)



Priority : (higher) $P \geq P_1 \geq P_2$ (lower)

Message communication with the priority

- Messages are given priorities

- Lower cost messages are given higher priorities

$\Rightarrow$ Efficient pruning
for the search space

Demonstration

Graph: 40,000-vertices grid

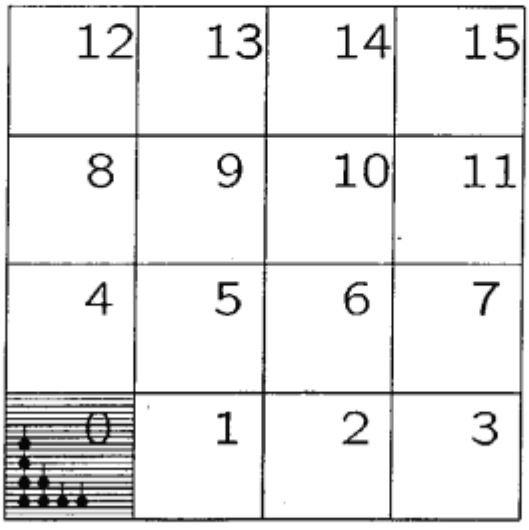
Mapping strategies:

- Two-dimensional simple mapping
- Two-dimensional multiple mapping
- One-dimensional simple mapping

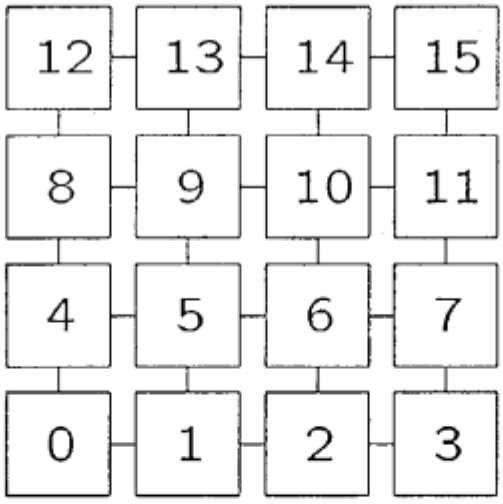
Processors: 16

Two-dimensional simple mapping(2s)

- For given $p = q^2$ processors, divide the grid into $q \times q$ blocks, map each block onto the corresponding processor
- preserves locality of a graph well, with the same number of vertices in each block
- Speedup of $\sqrt{p}$

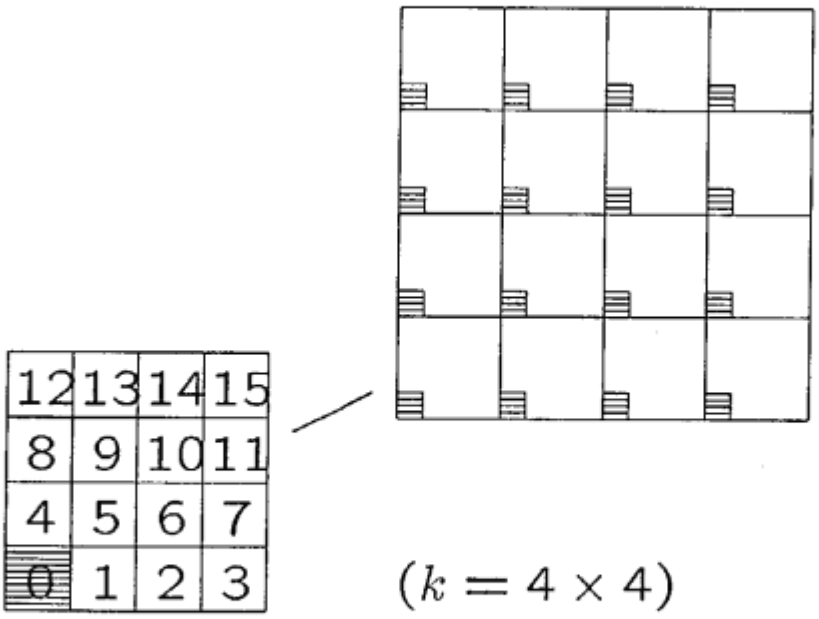


Logical processor configuration of the Multi-PSI (16PEs)



Two-dimensional multiple mapping(2m)

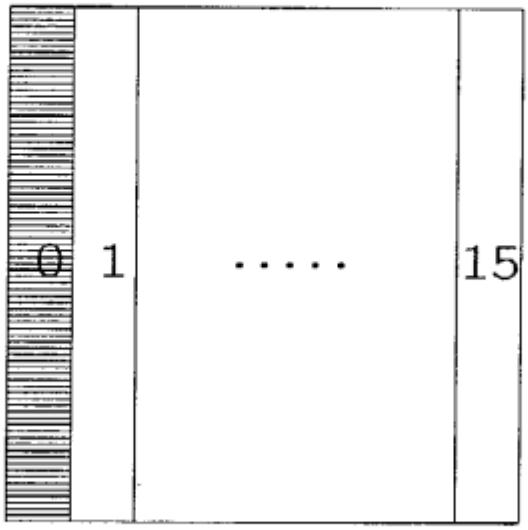
- Divide the grid into k super-blocks, then, each is again divided into p blocks just as in 2s mapping
- Higher processor utilization ?



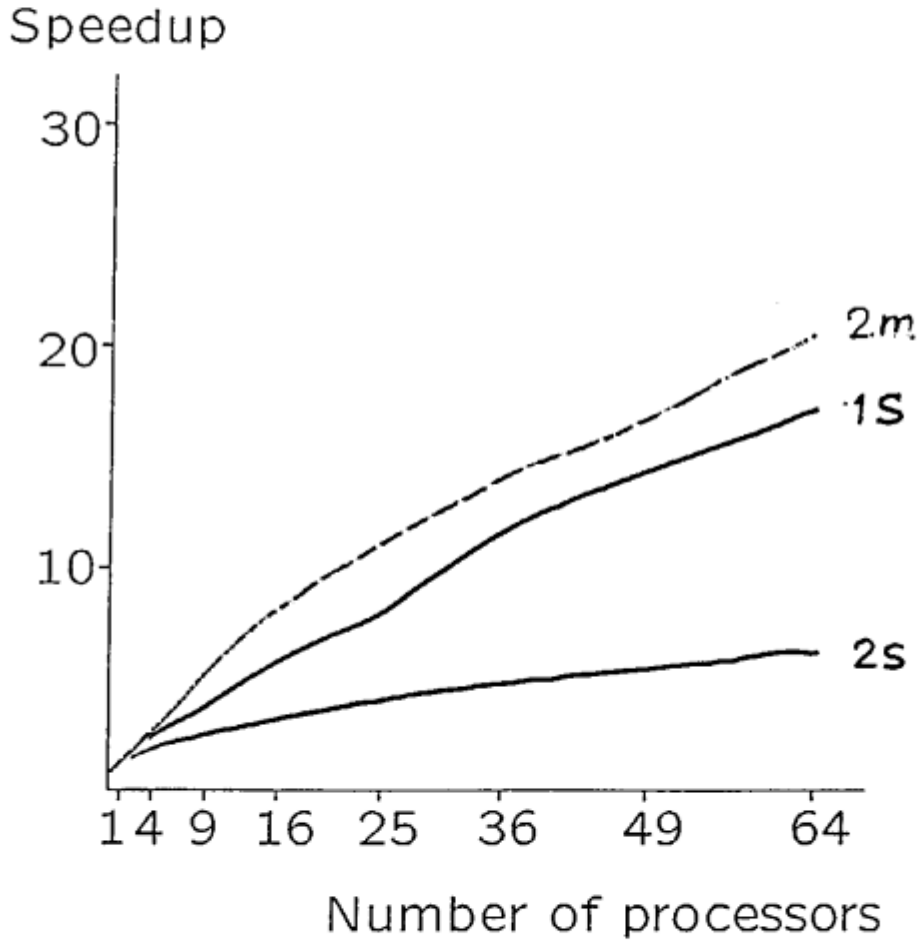
$(k = 4 \times 4)$

One-dimensional simple mapping(1s)

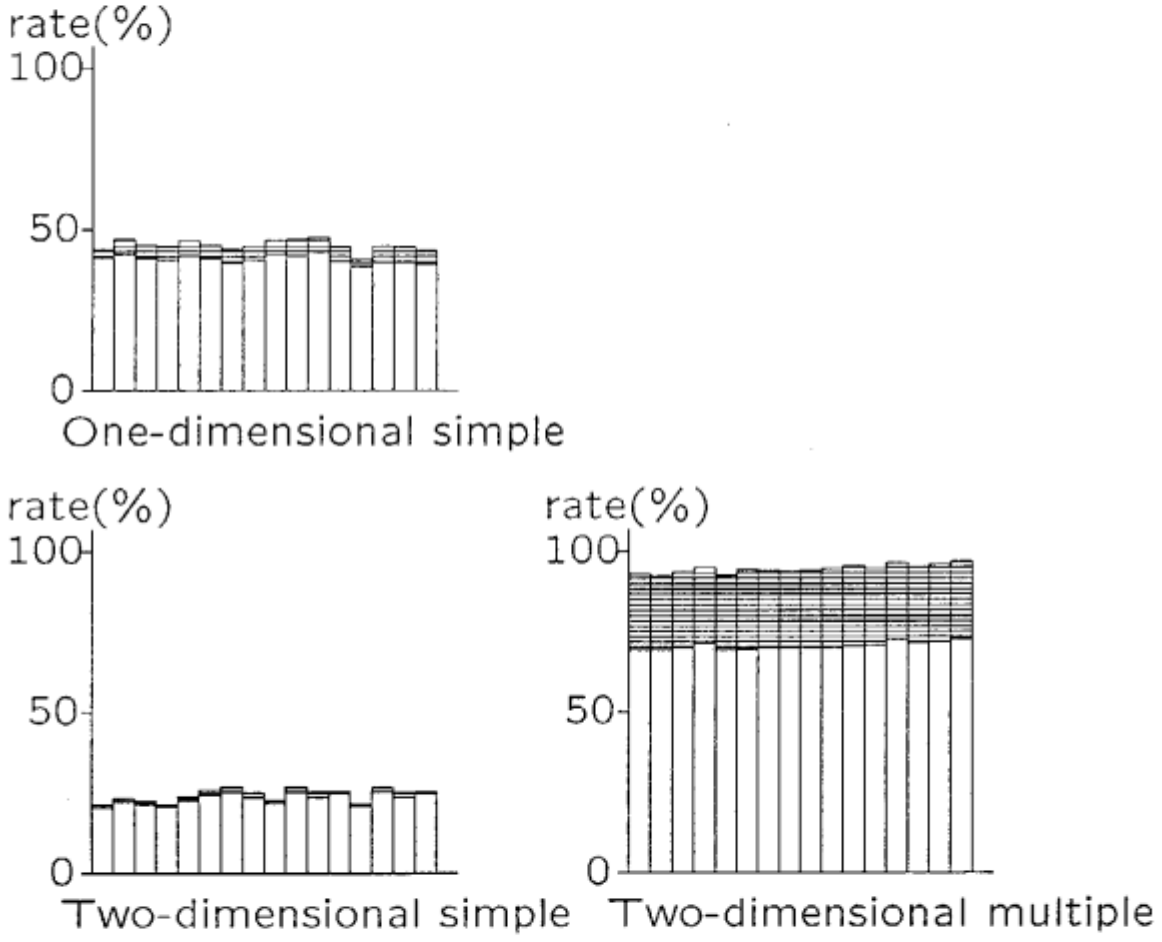
- Divide the grid into p narrow rectangular strips, and map them onto the processors
- High processor utilization by the minimum graph decomposition
- Speedup of $\frac{2}{3} \cdot p$



Speedup


Speedup
30
20
10
2m
1s
2s
1 4 9 16 25 36 49 64
Number of processors

Processor utilization



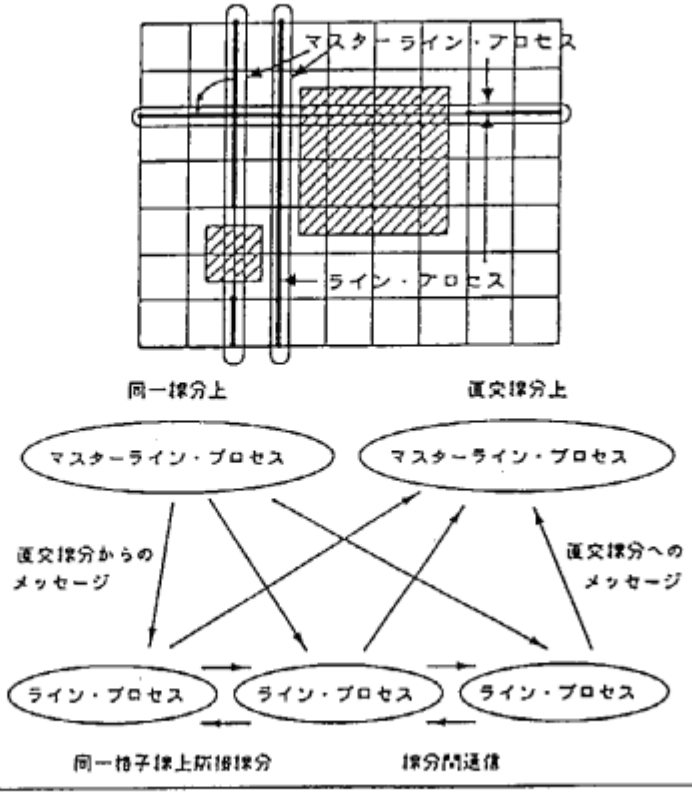
(with 16 processors)

Conclusion

2s: low processor utilization

2m: best performance;
high processor utilization but heavy communication overhead

1s: low communication overhead and not so high processor utilization

題　名	並列版 LSI-CAD 実験プログラム (1) LSI 配線
目　的	VLSI 設計の下流 CAD は多大な計算量を必要とする問題を多く含むが、その一つである実用規模の LSI チップの配線処理を行なう並列プログラムの設計を通して並列アルゴリズムおよび負荷分散手法を研究開発する。
概要及び特徴	【概要】 LSI 設計の下流工程の一つである LSI チップの配線処理をおこなう。スタンダードセルと呼ばれるモジュールのチップ上での配置が決定した後、各モジュールの接続されるべき端子間の接続経路を決定する。 【並列アルゴリズム】逐次アルゴリズムである予測線分探索法を基本アルゴリズムとして採用。配線処理の並列性は主に複数線分の同時配線により抽出する。 【プログラム手法】線分探索を基本アルゴリズムとしていることから、並列処理のプリミティヴとして配線格子の各線及びその上の線分をそれぞれプロセスとする。途中の配線状況は各線分プロセスの内部状態として表現される。配線過程では結線の追加や取り消しに対応して、線分のプロセスは動的に分裂・結合する。それらの線分プロセスは互いに通信し合って探索・配線を行なう。配線格子のプロセスはその通信を仲介する。
構　成	

デ　　モ　　内　　容　（1／3）

【配線問題】

配線問題とは、ゲートアレーやスタンダードセルやビルディングブロックなどのモジュールのチップ上での配置が決定した後、各モジュールの端子間の接続経路を決定する問題である。解法としては迷路法や線分探索法、チャネル配線法等がよく知られている。ここで扱う問題では配線層は２層を用い、縦方向・横方向で配線層を使い分け、配線は全てあらかじめ固定された配線格子上を通るものとする。配線の通過禁止、スルーホール禁止などの制約も扱う。

【予測線分探索法】

線分探索法を拡張した予測線分探索法[北沢 85]を基本アルゴリズムとして採用している。

予測線分探索法では、配線における線分探索において線分の通る期待位置を予測する先読みを行なっている。図１を例としてその基本処理を示す。Ｓから探索を開始してＴへ接続する場合、Ｓから下方に向かった線分がＡ点で折り曲げられたとすると、到達できるＴに最も近い点はａである。同様にして、Ｃ,Ｄで折り曲げられたとき到達できる点はc,dとなる。これらの点(a,c,d)をＳからの期待位置と呼ぶ。Ｂ点はスルーホール禁止であり、ここで折り曲げることはできないため、ｂ点は期待位置としない。水平方向に進んだ場合も同様にして期待位置(e,f,g,h)が得られる。これら期待位置(a,c,d,e,f,g,h)の中で最もＴに近い点はｃである。これより、S-C間を接続する。次にＣから水平方向に探索を行ない、Ｔに最も近い期待位置を求めるとｉが得られる。これより、C-I間を接続する。なおこの場合は、Ｉ点は前回の期待位置ｃとは異なっている。同様に探索を続けることによりS-T間を接続する一つの経路S-C-I-J-K-L-Tが求められる。

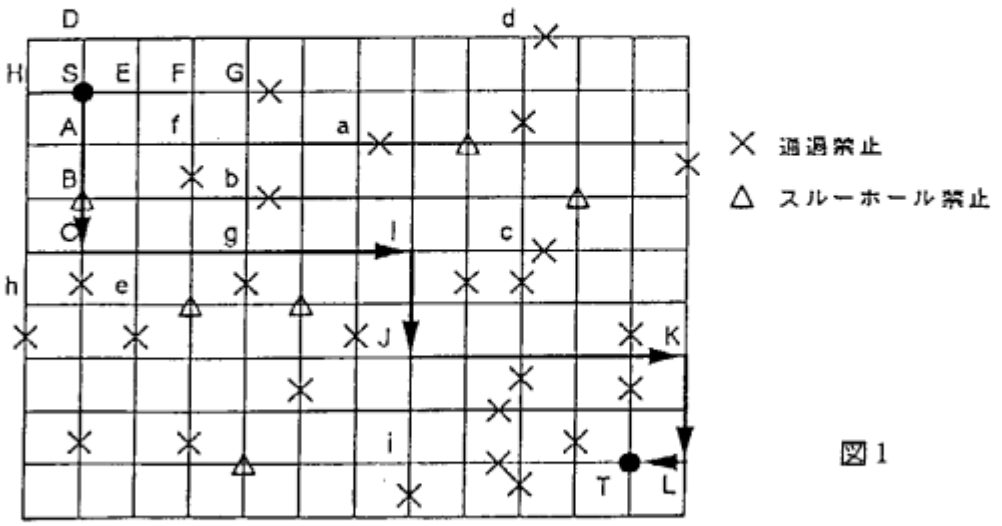


図１

更に二重探索を避けるために、期待位置として選択することを禁止するフラグ（IEP フラグ）を導入し、また期待位置が得られない時には最後に接続した線分を取り去り、一つ前の点に戻って同様な探索を続けることによって脱出路を求めるバックトラッキング処理を追加することによって、配線すべき 2 点間に経路が存在すれば必ず接続できることを保証している。

【並列化の手法】

線分探索を基本アルゴリズムとしていることから、配線格子の各線及びその上の線分領域をそれぞれプロセスとし、これらを並列処理のプリミティヴとする。前者をマスターライン・ライン・プロセスと呼び、後者をライン・プロセスと呼ぶ（図 2 参照）。

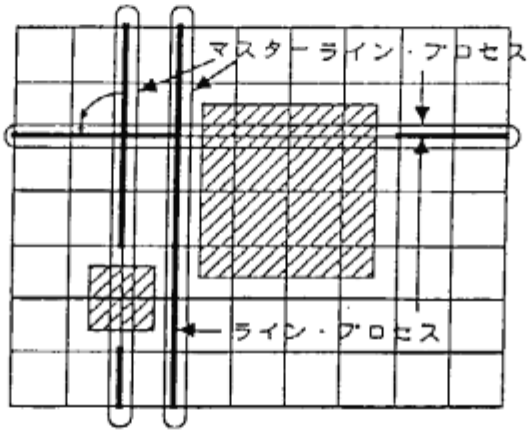


図 2

ライン・プロセスは処理の実行主体であり、ライン・プロセスは互いに通信し合いながら探索・配線処理を行なう。マスターライン・プロセスは直交する線分のライン・プロセス間のメッセージ通信を仲介する。

予測線分探索法は逐次配線アルゴリズムであるため、配線処理の並列性は主に複数ネットの同時配線つまり複数ネットを並列に探索することにより抽出する。ただしネットとは接続されるべき端子間の配線のことである。
1 ネットの配線では先読み部分を並列化している。すなわち一つのネットの探索において複数の期待位置予測を並列に行なうことにより並列性を引き出した（図 3(a)-(d) 参照）。

あるネットの 1 線分領域の配線処理は、それを含むライン・プロセスを、配線される線分領域のライン・プロセスと、残りの未配線の線分領域のライン・プロセスに分割する処理である。バックトラックを行なう場合には、一度配線した線分を再び未配線の状態に戻さなければならない。これは配線の逆過程であり、分割したライン・プロセスを再結合する処理である。このようにライン・プロセスは探索過程で動的に分裂 / 結合する。

デ　　モ　　内　　容　（3／3）

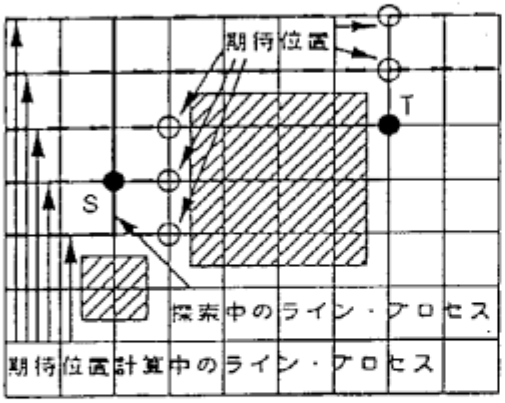


図 3(a)

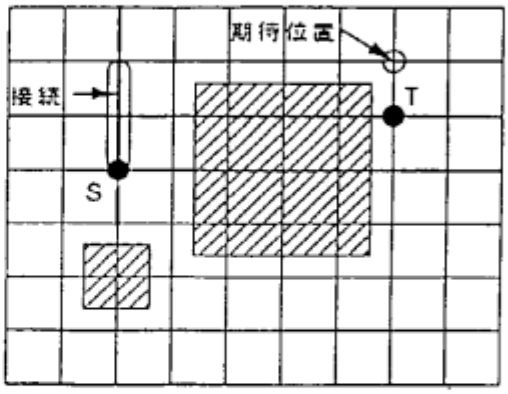


図 3(b)

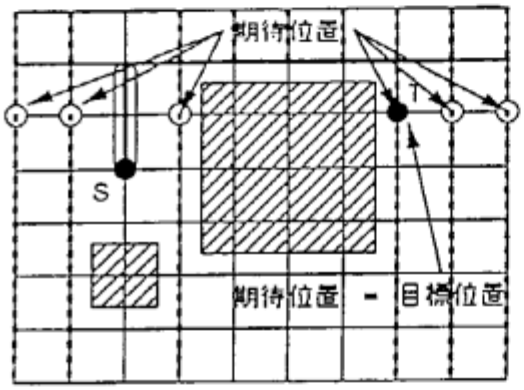


図 3(c)

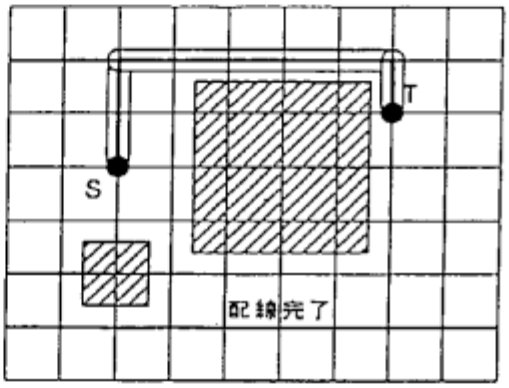


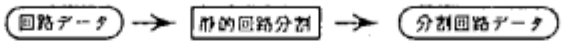
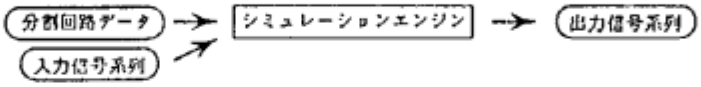
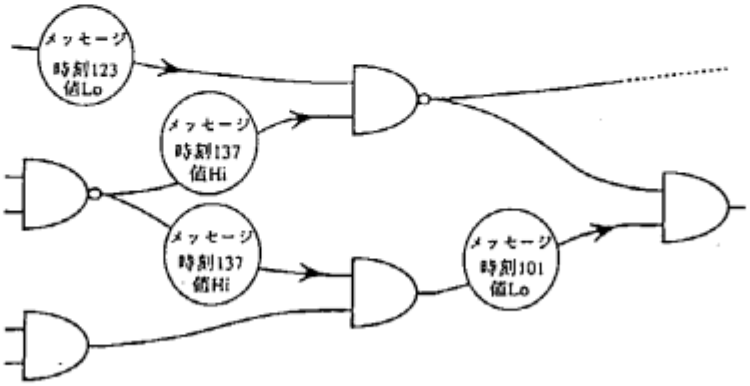
図 3(d)

上記の並列化アルゴリズムでは、メッセージ通信量がかなり大きくなることが予想され、マッピングについては、マルチ PSI や PIM のような通信コストの高い疎結合型並列計算機では通信の局所化が重要課題である。現在各種方式を検討中であるが、今回は初期評価用にプロセスを無作為に各 PE に割り付けるマッピング方式を用いて試作を行なった。

【デモの内要】

上記のような並列アルゴリズムに基づいて KL1 で試作したプログラムにより実用規模の LSI チップの配線処理をマルチ PSI 上で行なう様子とその結果とを表示する。

[北沢 85] 北沢 仁志, 高配線率線分探索の一手法, 「情報処理」第 26 巻第 11 号 ,1366-1375(1985).

題　名	並列版 LSI-CAD 実験プログラム (2): 論理シミュレーション
目　的	1万ゲートを超える実用規模 LSI 回路を扱えるゲートレベル並列版論理シミュレータを試作し、バーチャル・タイム方式と呼ばれる並列制御方式の評価を行なう。
概要及び特徴	本システムは、遅延を考慮したゲートレベル論理シミュレータをバーチャル・タイム方式と呼ばれる並列制御方式を用いてマルチ PSI 上に試作したものである。 • イベント駆動型のシミュレーション方式 – 信号値の変化 (イベント) 発生時のみシミュレートする。 • 並列制御方式 — バーチャル・タイム方式 – ゲート間で、イベント及び発生時刻を伝えるメッセージを通信することでローカルな同期のみをとる。 – メッセージ到着順序がイベント発生時刻順と異なった場合、ロールバック (過去の履歴の巻き戻し) によって処理のやりなおしをする。 – 同一プロセッサ内では、イベント処理のスケジュールをすることで、ロールバック発生回数を軽減する。 • 負荷分散方式 — プリプロセッシングによる静的負荷分散 – ロールバック発生回数を軽減すること – プロセッサ間通信を軽減すること
構　成	システム構成 前処理による静的負荷分散  並列シミュレーション実行  シミュレーションメカニズムイメージ 

デ　　モ　　内　　容　（1／3）

概要

論理シミュレーションは、現在のLSI設計において重要な位置をしめている。論理シミュレーションの目的は対象となる論理回路と入力信号系列に対し、シミュレーションによって出力信号系列をもとめ、設計された回路の論理、およびタイミング検証を行なうことである。このシミュレーションに費やされる時間は膨大なものであるため、高速なシミュレータに対する要求は強い。

本システムは、遅延を考慮したゲートレベルでの論理シミュレーションをバーチャル・タイム方式と呼ばれる並列制御方式を用いてマルチPSI上に試作し、並列化による処理速度向上の効果を検証するものである。

シミュレーション対象および方式

本システムでは、1万ゲートを超える実用規模回路(組み合わせ回路、順序回路)を対象とした、ゲートレベルでのシミュレーションをおこなう。

対象信号値として、Hi、Lo、X(不定)の3値を扱う(3値モデル)。また遅延値は、各素子の遅延値の最大公約数を単位とし、その整数倍を割り当てる(割り当て遅延モデル)。

シミュレーション方式としてはイベント駆動方式を採用している。これはゲートの入力線における信号値の変化という事象(イベント)の発生時にのみ出力値をシミュレートするものである。

並列制御方式

本システムでは並列制御方式としてバーチャル・タイム方式を採用した。バーチャル・タイム方式では各ゲート間でイベント発生とその時刻を伝えるメッセージを通信しながら、ローカルな同期のみによって処理を進める。各ゲートでは、メッセージが届いた順にその履歴を取りながら処理を行なう。もし、メッセージのもつ時刻が逆転して届けば、過去の履歴を巻き戻して(ロールバック)、処理のやりなおしを行なう。

これに対し、ディストリビューティッド・シミュレーション(distributed simulation)方式と呼ばれる並列制御方式では、各ゲートの全入力線へのメッセージ到着を待って処理を始めることで、処理の正当性を保証する。この方法はデッドロックの危険性を伴うものであり、デッドロックを避けるための処理コストは非常に高い。

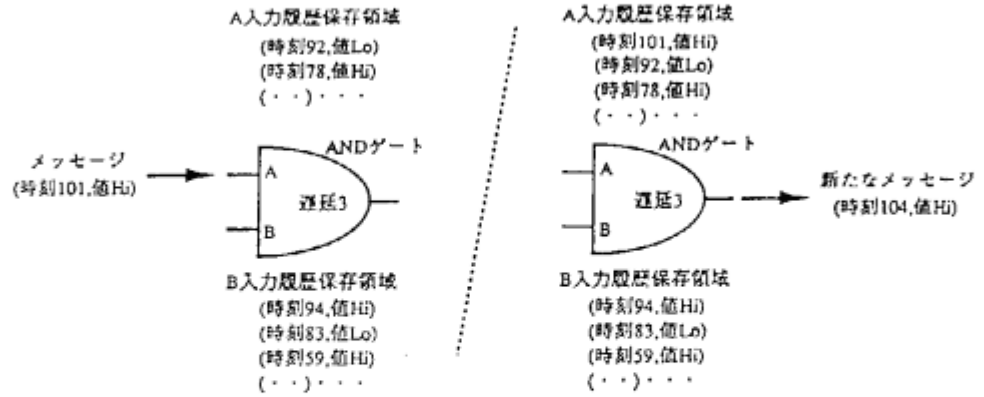
バーチャル・タイム方式は本質的にデッドロックの危険性がない点が大きな利点となっている。

1. ローカルな処理

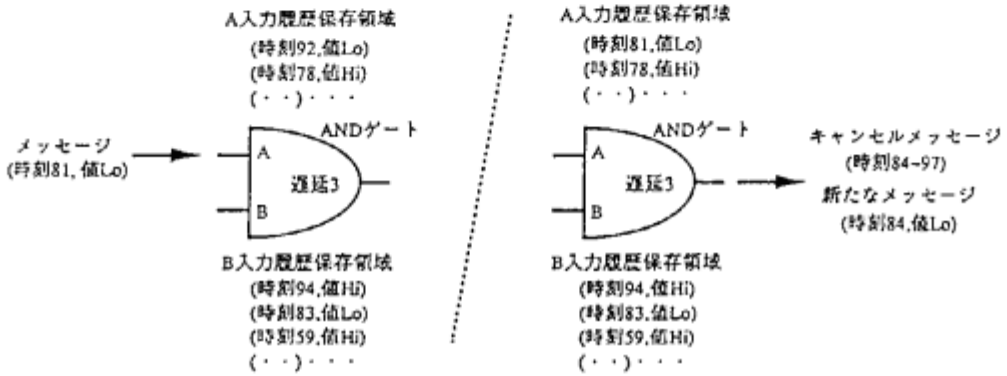
   (a) 時刻順にメッセージが届いた時の処理
   直前のゲートの状態とメッセージ情報から出力信号値を求め、イベントが発生すれば次段のゲートに対し新たにメッセージを送る。

   (b) 時刻の逆転が発生してメッセージが届いた時の処理
   メッセージが伝えた時刻の直前のゲートの状態を回復し、その時点からの処理をやり直す。また、次段のゲートに対し、誤って伝えたメッセージをキャンセルするメッセージを送る。

デ　　モ　　内　　容　（2／3）



(a)時刻順にメッセージが届いたときの処理



(b)時刻の逆転が発生してメッセージが届いたときの処理

2. グローバルな処理

バーチャル・タイム方式では、ロールバックに備えて各ゲートが入力信号の履歴を保存しなければならず、そのためのメモリ消費が問題になる。そこで、時々システム全体のグローバルなシミュレーション時刻を求めると、それ以前の状態にロールバックしないことが分かるため、その分の履歴が占有していたメモリ領域を回収している。

また、システム全体のグローバルなシミュレーション時刻がシミュレーション終了時刻に達した時点をもって、シミュレーション終了とする。

3. スケジューリング

一つのプロセッサ内では、可能な限り各ゲートにメッセージが正しい順序で届くようにすることで、ロールバックの発生を出来るだけ抑えることが出来る。そのために、1プロセッサではメッセージを集中管理し、時刻の若いメッセージから順に各ゲートに対して伝えるようなスケジューラを置く。

負荷分散方式

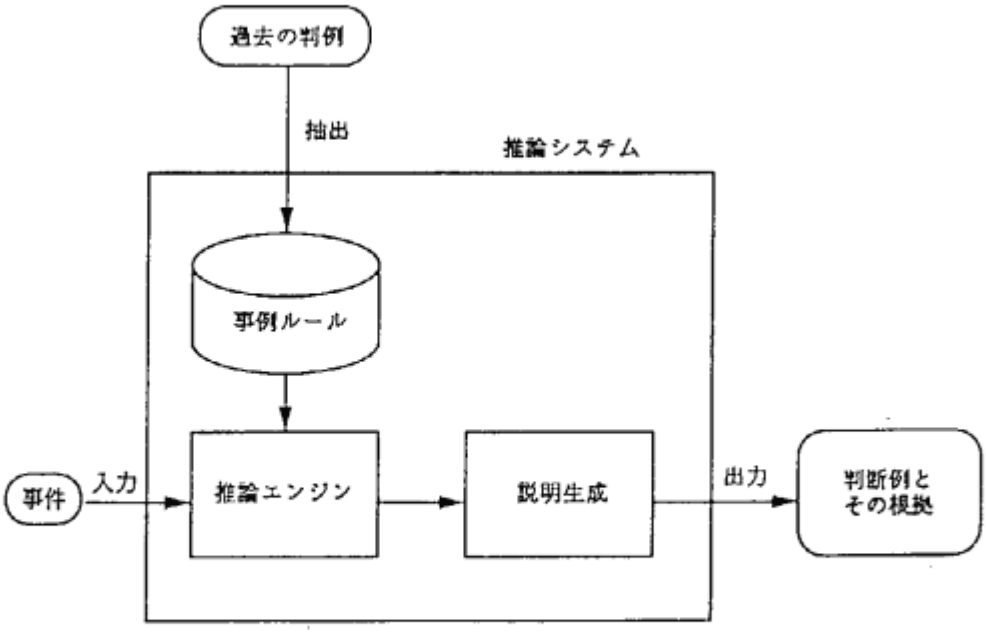
(1) 各プロセッサの時刻の進み方が出来るだけ均一になること、(2) プロセッサ間に渡るメッセージ通信が出来るだけ少なくなること、の2つを実現するように予め静的に回路を分割しておく。

デ　　モ　　内　　容　（3／3）

デモの内容

1. 1万ゲートを超える規模の順序回路に対し、クロック線以外はランダムに入力信号系列を与えた場合を1台のプロセッサのみを使ってシミュレートし、出力信号系列を求める。

2. 同じシミュレーションを32台のプロセッサを使っておこない、プロセッサ1台のみを用いたシミュレーションに比べた、処理速度の向上、および、履歴巻き戻し回数をみる。

題　名	事例に基づく並列法的推論実験システム
目　的	知識処理の１つの手法である〔事例に基づく推論〕を特徴に備えたシステムを、マルチＰＳＩ上に構築し、法的推論問題への応用を通して並列推論の実証的研究を行う。
概要及び特徴	• 事例に基づく推論は、過去の具体的な問題解決事例をもとにして、新たな問題の解決を行うものである。大量の事例の中から、新たな問題と状況が類似するものを高速に検索、利用するために並列化を行った。 • 応用として、事例に基づく法的推論システムを試作した。本システムに新たな事件の記述を入力すると、過去の判例の中から、新たな事件に関係する判断例と、その根拠となる説明が生成される。 • 本実験システムでは、判例から抽出した当事者の主張や裁判官の判断を意味ネットワークを用いた事例ルールの形式に表現している。事例ルールは、過去の事件の内容を具体的なレベルで記述するので、ルールにまとめるのがプロダクションルールに比べて容易である。 • 多くの事例ルールをＫＬ１プロセスのネットワークに展開し、新たな事件に類似する事例ルールの条件部を照合する処理を、並列に実現している。
構　成	

デ　　モ　　内　　容　（1／3）

【法的推論問題】

一般に法律の問題というと、法律の条文を論理的に捉え、三段論法などの形式的な推論法で解決可能なものと受け取られがちだが、それだけでは十分ではない。現実の裁判などは条文をもとにした形式的な推論だけではなく、解釈を伴う法的推論が行われている。解釈を行うときには、他の法律や学説や過去の判例などが参照される。なかでも判例は、より客観的な解釈を行ううえで、最も重要な情報源である。現に、判決文をみると、過去に起きた類似の事件で用いられた論理をそのままで、あるいは拡張や類推によって利用して、論理展開するものが認められる。その意味では法律の専門家も、事例を利用した推論を行っているということができる。このように法的推論問題は、事例に基づく推論の格好の応用問題分野と考えられる。

本システムでは、労働災害認定に関する過去の判例に現れた解釈・説明を、事例ルールの形で表現し、システム内部に準備している。ここに、新たな労働災害認定事件の事実関係を入力すると、当該事件に関する労災認定および非認定について、各々、可能性のある論理構成（説明）が、過去の事例に基づき生成される。ただし、この出力結果は、システムによる判決予測ではなく、あくまでも、専門家の判断を支援する材料のひとつとなる可能性を狙ったものである。

【事例の知識表現】

各事件に関する事実関係は｛対象，関係，値｝の三項構成（以下ファクトと呼ぶ）の集合で、（｛雇用関係 #1，従業者，山田氏｝，｛雇用関係 #1，雇用主，郵便局 #1｝，｛雇用関係 #1，職務，庶務｝，｛郵便局 #1，責任者，局長 #1｝，．．．）などと表現する。このファクトの集合は、いわゆる意味ネットワークと呼ばれる有向グラフを形成する。

判決文には、弁護士や判事の主張や、裁判官の判決理由が記述されている。その内容は、多くの部分において、複数のファクトの連言を条件部に持つ事例ルールのかたちに、抽象化して捉えることができる。たとえば「山田氏は郵便局で庶務の仕事に就いていたが、同時に保険勧誘が奨励されていたので、保険勧誘も職務の一環というべきである」という種類の記述が判決文にあれば、”IF｛雇用関係 #1，従業者，山田氏｝and｛雇用関係 #1，雇用主，郵便局 #1｝and｛雇用関係 #1，職務，庶務｝and．．．THEN (make｛雇用関係 #1，職務の一環，保険勧誘｝)”のように表現できる。

各判決例からは論点に応じて平均１０個程度の事例ルールが抽出でき、それらのルールを過去の事例としてシステムに蓄える。事例ルールはいわゆるプロダクションシステムが扱うルールのように、一般化された規則を表現したものではない。本システムのルールは、事例の具体的な条件判断をそのまま表したものであって、その条件部は類似性に基づいて評価される。

デ　　モ　　内　　容　（2／3）

【事例に基づく説明生成】

新たに吟味すべき事件例は、ファクトの集合のかたちでシステムに入力される。システムは蓄えている事例ルールのうち、入力ファクトと条件部が類似するルールの実行部を実行する。実行の結果、新しいファクトが生成され、それにより、また別のルールが照合して連鎖的にルールが実行される。すなわち、類似した事件には類似の論理展開が利用できる傾向を盛り込んだことに相当する。

事例ルールには，次の二つの手法を併用して類似度を考慮した照合を行っている。ここで、〔類似度〕は、事例や新たな事件を記述した２つの意味ネットワークにおいて、対応するノードやリンクの数で評価している。（1）ファクトの各項が一致しなくとも、各項の上位の概念が一致すれば、ファクト全体として一致すると判断する。これはいわゆるIS-Aリンクを用いた一般化に相当する。（2）ルールの条件部の各ファクトに、事例から推定できる相対的重要度を数値で付加し、各ファクトがすべて照合しなくとも相対的に重要なファクトが照合すれば、条件部全体として照合すると判断する。

事例ルールの実行連鎖は、結論に至る推論過程を成していると捉えられる。そのため、各ファクトについて、ファクト生成に要したルールの実行連鎖を表示すれば、その結論に対する根拠の説明を与えることができる。本システムでは、たとえば、最終的に｛事件関係，認定，労働災害｝かまたは、｛事件関係，認定，非労働災害｝かのファクトが（多くの場合両方が）生成される。いずれかのファクトについて、それに要したルールの実行連鎖をシステムに問えば、結論を正当化する可能性のある論理展開がシステムから得られる。

【並列処理方式】

本システムは、過去の事例が複数の事例ルールとして表現されているため、推論エンジンの処理はプロダクションシステムと同様の、条件照合-実行のサイクルで行われる。各事例ルールの条件部は、ＫＬ１のプロセスネットワークに展開される。このネットワーク上に、吟味すべき事件に関するすべてのファクトを流し、ファクトとルールの条件部との照合を行う。

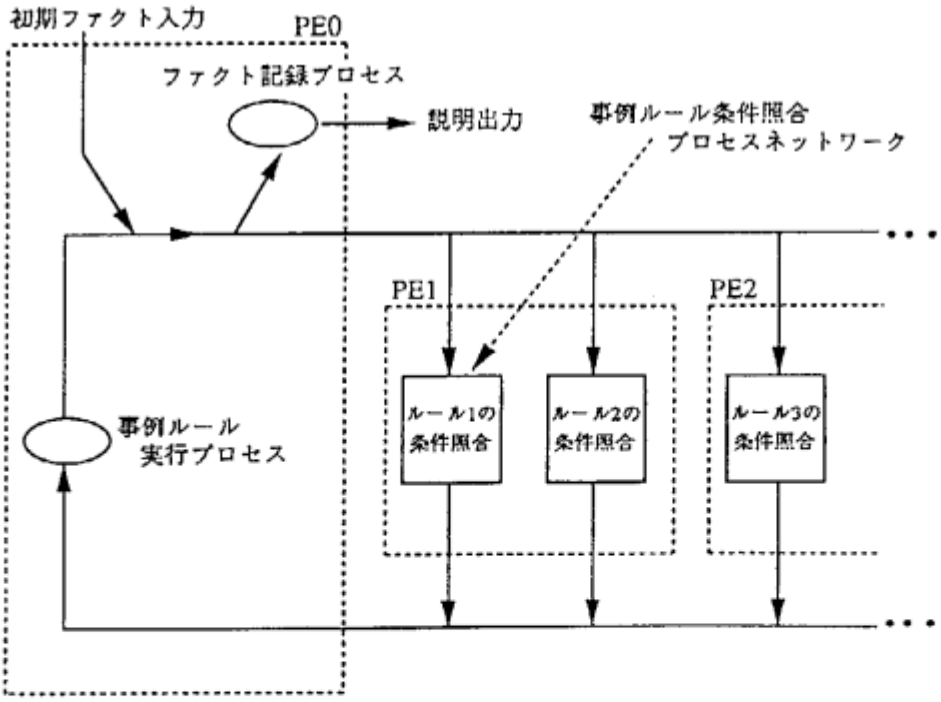
ひとつの事例ルールの条件部のプロセスネットワークでは、各条件の照合を行う部分が、それぞれ条件照合プロセスとして実現されている。そのプロセスは、入力からファクトを受信すると条件の重みを考慮した照合操作を行い、照合度を点数のかたちで付加して出力する。この条件照合プロセスの出力は、結合プロセスで次々に併合されて、最終的に照合判断プロセスに至る。そこでは照合度の演算をもとに、条件部全体の類似度を判断する。

デ　　モ　　内　　容　（3／3）

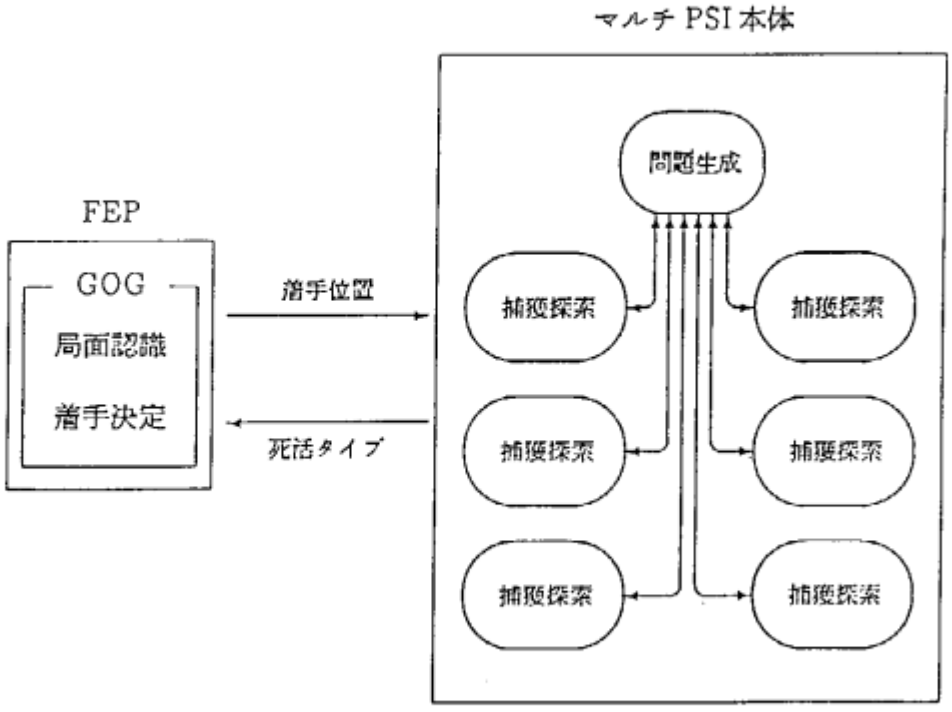
こうして条件部が類似したと判断されたルールは、ルール実行プロセスに移り、そこで、ルールの実行部の指定により新たなファクトを生成する。この新たなファクトは、その根拠（ファクトを生成したルールと、そのルールの条件部を満足させたファクト）とともに、記録プロセスにも記録され、後の説明生成に使用される。事例ルールは、ルールの条件部の照合に時間がかかるので、多数のルールが存在する場合には、ルールを複数のプロセッサに分割して並列に照合操作を行い、高速化を図っている。

【デモ内容】

労働災害認定に関する事件例を入力し、それが労災認定か、非労災認定かの判断例を生成させる。次に、それらの判断の根拠となる論理展開を、推論木のかたちで出力表示する。



並列推論エンジンの構成

題　名	並列版「碁世代」実験システム
目　的	大規模知識処理システムの１つである囲碁システムのマルチ PSI 上での開発を通じ、並列知識処理に関する問題を抽出し、新しい解決方法へのアプローチを探る。
概要及び特徴	[概要] ・逐次推論マシン上での囲碁システム（GOG）の並列版作成の中間成果 ・本研究は電子技術総合研究所との共同研究課題 [特徴] (1) 計算量の大きな処理の固まりで他の部分との独立性が高く、かつ同様の処理の個数の多い捕獲探索の、動的負荷分散方式による並列処理実験 (2)FEP(フロントエンドプロセッサ) 上に GOG を乗せ、捕獲探索をマルチ PSI 本体に処理させる構成
構　成	

デ　　モ　　内　　容　（1／3）

並列版囲碁システムの開発

これまでに、逐次マシン上で囲碁システム（GOG）の開発をしてきた。そのシステムの棋力の向上のために取り入れたい処理は、まだ沢山残っている。しかし、逐次マシンでは、人間との対局に耐え得る時間内に実行ができないので、それらの処理を取り入れることができない。そこで、システムの並列化によって処理時間の増加を抑えつつ高機能化を実現させることを目指す、並列版囲碁システムの設計開発を始めている。

本実験システムはその途中結果である。逐次版の処理の中から計算量の大きな処理の固まりで他の部分との独立性が高く、かつ同様の処理の個数の多い捕獲探索を取り出し、動的負荷分散方式を利用して並列処理を行う。最終的には、全ての処理を並列実行させる予定である。

また、本実験システムは、一部ではあるが逐次版の GOG では処理時間の制約により扱えなかった処理（高精度の捕獲探索及びある種の打着候補手の生成）も取り入れている。

捕獲探索処理

囲碁の対局を行なうための処理の流れは、次の通りである。

相手の打着 → 局面認識 → 打着手決定

この中の局面認識では、盤上の危険な状態にある全ての石の死活を判断しなければならない。盤面上の石の死活タイプ（活き / 死に / 中立）の認識具合によって打着手は大きく変化するからである。現在 GOG では、つながっている石のダメ数が３以下のものについて石の死活を判断する捕獲探索を行なっている。また、それぞれの捕獲探索問題は独立に実行可能であるという性質がある。

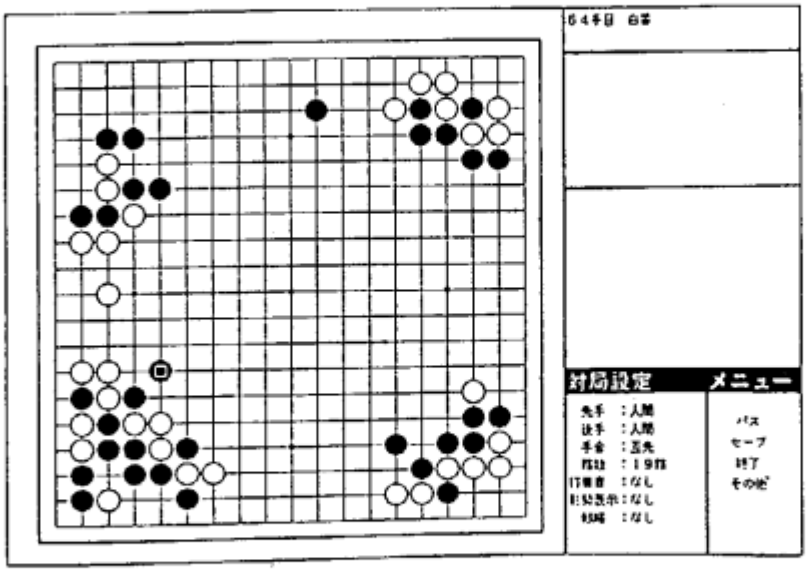


図１．捕獲探索問題が多発する例

デ　　モ　　内　　容　（2／3）

複数の捕獲探索問題の並列処理

複数発生した捕獲探索は以下のように並列実行される。

1．相手の着手によって発生するダメ数が３以下のつながっている石の捕獲探索問題（図２の捕獲１）を、全ての暇なプロセッサ（PE）に一つずつ割当てる。
2．全ての捕獲探索問題を割当て終わったら、処理の終わった暇な PE にはさらに高度な捕獲探索問題（ダメ数が４のつながっている石の死活タイプを求める、図２の捕獲２）を割当てる。
3．高度な探索問題も全て割当て終わったら、暇な PE には、各点に於ける周囲の状況を調べることによって、勢力を高めたり地を効果的に作る打着候補手を求める処理（図２の候補手）を割当てる。

1,2 の処理で割当てている探索問題は粒度が大きく、処理が終了する時刻は PE によって大きなばらつきを生じる。しかし、３の処理で割当てている候補手生成のための処理は粒度が小さいので、暇になった PE に割当てても、1,2,3 の全部の処理時間は、1,2 の処理のみを実行した場合と比較しても、大きな差はでない。これは、暇な PE に有益な処理を実行させていることになり、稼働率向上として現れる。

なお、各 PE は盤面全体の情報を常に保持している。これはデータの重複を招くが、一つの盤面情報を集中管理することによるボトルネックを防ぎ、PE 間通信量の低減にもつながっている。

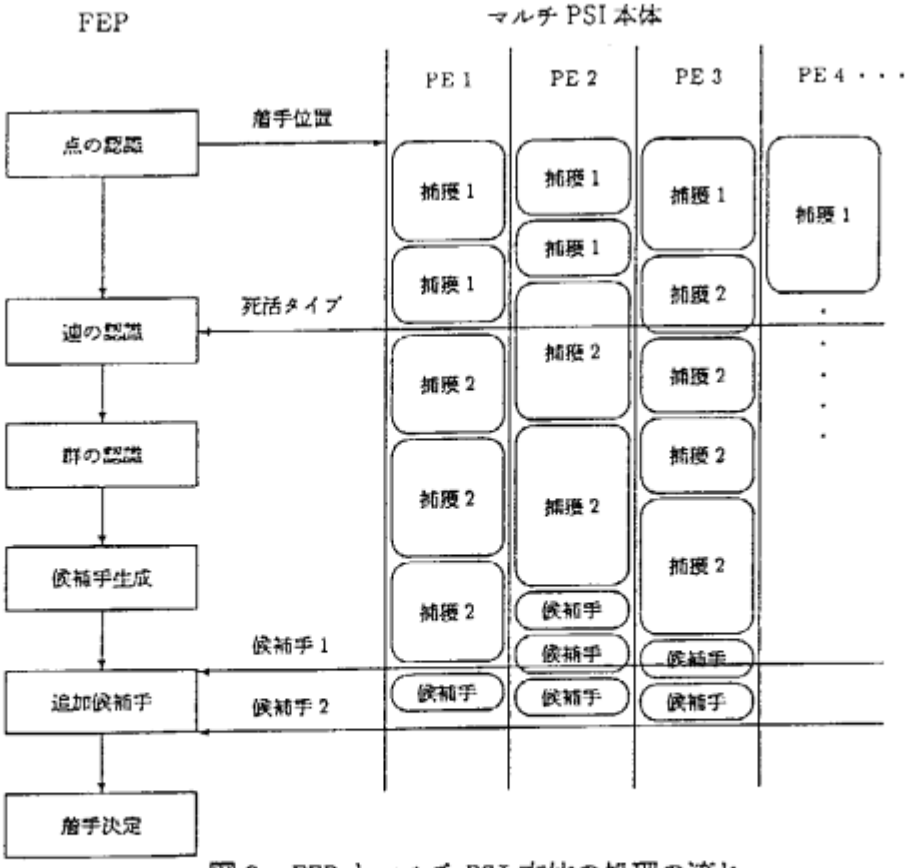


図２．FEP とマルチ PSI 本体の処理の流れ

デ　　モ　　内　　容　（3／3）

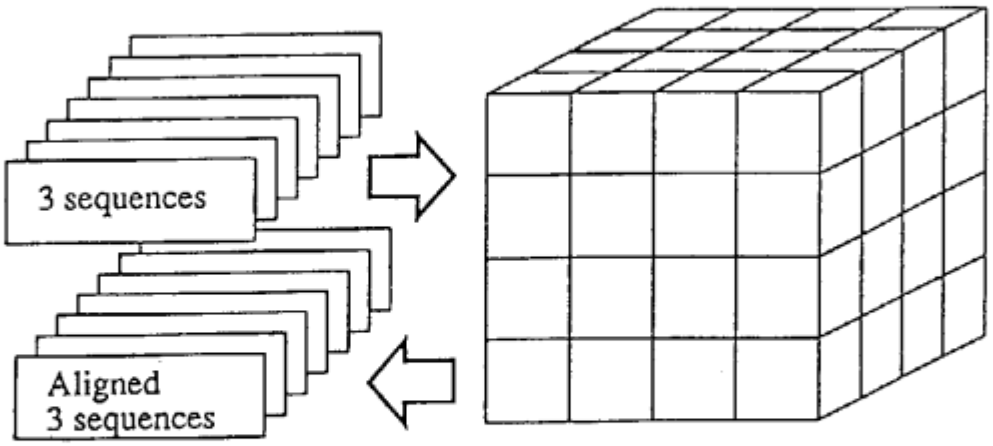
FEP(フロントエンドプロセッサ) 側の処理

FEP には、GOG(逐次の囲碁システム) が存在し、マルチ PSI 側と通信を行ないながら処理を進めている。相手の打着から自分の打着手を決めるまでの FEP の処理の流れは、以下の通りである。

1．相手の打着手をマルチ PSI 側に通信する。
2．マルチ PSI から返ってきた捕獲探索問題の結果である死活タイプに基づき、ある程度つながっていると思われる石の認識処理などの高度な認識を行ない、打着候補手の生成を行なう。
3．マルチ PSI から得られた、勢力を強めたり地を効果的に作る打着候補手と、FEP 内の GOG 自身が出した候補手を比較検討して、打着手を決定する。

デモの内容

1．ある局面に於ける認識時間を 1 台で実行した場合と、並列に実行した場合との時間の比較
2．実際の対局

Title	Genome Analysis Program (1): Multiple Sequence Alignment by 3-Dimensional DP-matching
Purpose	• Parallel programming on a large-scale problem in KL1. • The first step to genome analysis.
Outline & Features	[Outline] The system solves three-sequence alignment problems by 3-dimensional DP-matching. The DP-matching is executed by a prism network of KL1 processes. The network works as a parallel pipeline. [Feature] • Efficient DP-matching by parallel pipeline processing. • Quality improvement in three-sequence alignments.
System Configuration	 3D DP matcher made of KL1 process-network

1 What is multiple sequence alignment?

Biologists often align DNA and protein sequences in order to determine how similar they are. DNA is a chain of four kinds of nucleic acids and a protein is a chain of twenty kinds of amino acids, which are translated from a chain of nucleic acids. Strong similarities between sequences may result from a common evolutionary relationship, and these sequences may have almost same function.

Figure 1 shows a typical multiple sequence alignment. Twelve fractions of enzyme proteins are aligned. Each letter stands for an amino acid: D is aspartic acid, R is arginine, H is histidine, and P is proline. A good alignment has same or similar amino acids in each column. To make an alignment good, each sequence is shifted or gaps (dash characters) are inserted into the sequence.

```
---DRHP-IPNMDEILGKLGRC-NYFTTIDLAKGFHQIEMDPESVSKTAFS-------
---DAYN-LPNKDELLTLIRGK-KIFSSFDCKSGFWQVLLDQESRPLTAFT-------
---DIHPTVPNPYNLLSGLPPSHQWYTVLDLKDAFFCLRLHPTSQPLFAFEW-RDPEM
---L-FGPVQRGLPLLSALPQDWKLI-IIDIKDCFFSIPLYPRDRPRFAFTIPSLNHM
---P-FGAVQQGAPVLSALPRGWPLM-VLDLKDCFFSIPLAEQDREAFAFTLPSVNNQ
---DLSSSSPGPPDL-SSLPTTLAHLQTIDLRDAFFQIPLPKQFQPYFAFTVPQQCNY
---TLTSPSPGPPDL-TSLPTALPHLQTIDLTDAFFQIPLPKQYQPYFAFTIPQPCNY
---PIPALSPGPPDL-TAIPTHPPHIICLDLKDAFFQIPVEDRFRSYLSFTLPSPGGL
---D-FWEVQLGIPHPAGLKKKKSVT-VLDVGDAYFSVPLDEDFRKYTAFTIPSINNE
VHWPKF-AVPNLQTLANLLSTDLQWL-SLDVSAAFYHIPISPAAVPHLLVG-------
VSWPKF-AVPNLQSLTNLLSSNLSWL-SLDVSAAFYHIPLHPAAMPHLLVG-------
MRFPRY-WSPNLSTLRRILPVGMPRI-SLDLSQAFYHLPLNPASSSRLAVS-------
```

Figure 1 Multiple sequence alignment

2 Dynamic programming on sequence matching

Dynamic programming (DP) is a basic method to find an optimal alignment. The method is regarded as the best path search in the N-dimensional network. In the method, for example, if two sequences, ADHE and AHIE are given, we form a 2-dimensional network that has 25 nodes connected by arrows. A cost is assigned to each arrow. We search a path from the top left node to the bottom right node, minimizing the total cost of arrows. In this case, the set of arrows that connect white circle nodes is the best path. This best path corresponds to the optimal alignment, ADH-E and A-HIE (Figure 2.1).

D e t a i l s (2/3)

Costs on arrows should reflect similarity between compared characters. In the case of protein sequence alignment, Dayhoff's odds matrix (Figure 2.2) is the most popular way of obtaining the costs. The matrix was obtained by statistical analysis of mutation probability of amino acids.

Though DP-matching is an optimal method for alignment, it takes a lot of calculation time. DP-matching with more than three dimensions is too time-wasteful to be used for practical alignment. So DP-matching has been used for partial matching, when several sequences need to be aligned. For instance, we can produce all pairwise alignments of given sequences with 2-dimensional DP, then merge the alignments one by one.

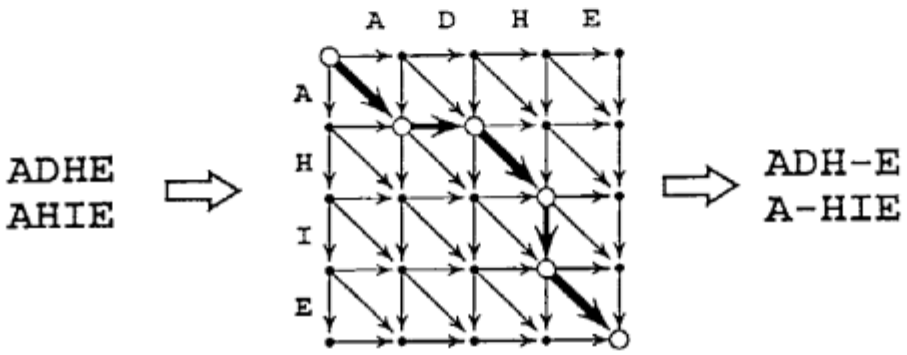


Figure 2.1 DP-matching method

```
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X
A -2
R  2 -6
N  0  0 -2
D  0  1 -2 -4
C  2  4  4  5-12
Q  0 -1 -1 -2  5 -4
E  0  1 -1 -3  5 -2 -4
G -1  3  0 -1  3  1  0 -6
H  1 -2 -2 -1  3 -3 -1  2 -6
I  1  2  2  2  2  2  2  3  2 -6
L  2  3  3  4  6  2  3  4  2 -2 -6
K  1 -3 -1  0  5 -1  0  2  0  2  3 -5
M  1  0  2  3  6  1  2  3  2 -2 -4  0 -6
F  4  4  4  6  4  5  6  6  2 -1 -2  5  0 -9
P -1  0  1  1  3  0  1  1  0  2  3  1  2  5 -6
S -1  0 -1  0  0  1  0 -1  1  1  3  0  2  3 -1 -2
T -1  1  0  0  2  1  0  0  1  0  2  0  1  3  0 -1 -3
W  6 -2  4  7  8  5  7  7  3  5  2  3  4  0  6  2  5-17
Y  3  4  2  4  0  4  4  5  0  1  1  4  2 -7  5  3  3  0-10
V  0  2  2  2  2  2  2  1  2 -4 -2  2 -2  1  1  1  0  6  2 -4
B  0  1 -2 -3  4 -1 -2  0 -1  2  3 -1  2  5  1  0  0  5  3  2 -2
Z  0  0 -1 -3  5 -3 -3  1 -2  2  3  0  2  5  0  0  1  6  4  2 -2 -3
X  0  1  0  1  3  1  1  1  1  1  1  1  1  2  1  0  0  4  2  1  0  1  1
```

Figure 2.2

Dayhoff's odds matrix

3 Parallel pipeline processing of 3-dimensional DP

If 3-dimensional DP can be executed rapidly, it is useful for partial matching because it tolerates noise better than 2-dimensional DP does. We have implemented 3-dimensional DP on the parallel machine, Multi-PSI, and improved the speed of three-sequence matching.

Our system constructs a 3-dimensional prism network with KL1 processes (Figure 3). The prism network is divided into 64 subprisms of equal volume and is mapped to 64 process elements (PEs). The KL1 is suitable for constructing such mesh-like process networks and the network can be used as data-flow pipeline easily.

If many different combinations of three-sequence alignments are available, we expect to merge whole sequences adequately for multiple alignment. This system provides optimal three-sequence alignments by parallel pipeline processing.

4 Demonstration

The demonstration system solves three-sequence alignment problems continuously by parallel pipeline processing. After several initial alignment data are fed to PE0, their optimal alignments come out from PE63 and are displayed at short intervals. During processing, the performance meter window shows that several wavefronts pack and propagate from PE0 to PE63 clearly.

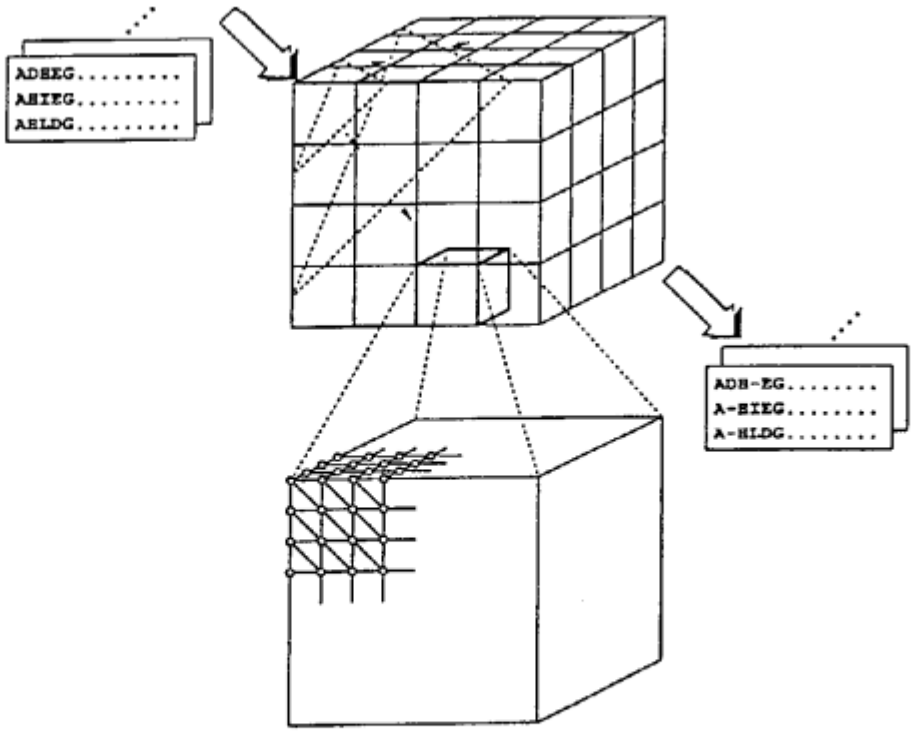
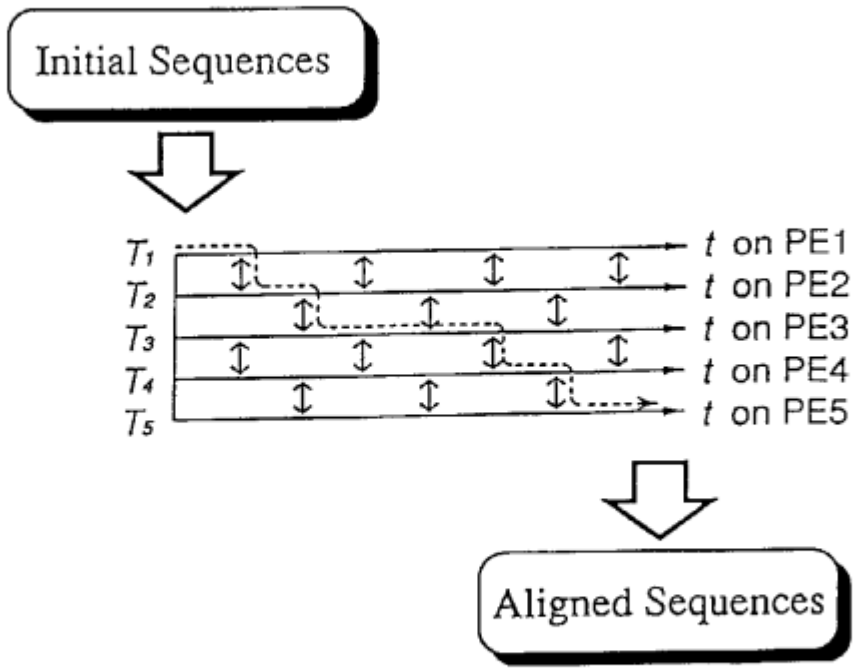


Figure 3 3-dimensional DP-matching

Title	Genome Analysis Program (2): Multiple Sequence Alignment by Parallel Simulated Annealing
Purpose	• Application of a parallel simulated annealing to a practical problem. • The first step to genome analysis.
Outline & Features	[Outline] The system solves a multiple sequence alignment problem by scheduleless parallel simulated annealing. Each PE has a constant temperature and exchanges solutions with neighbor PEs in some probabilistic way. [Feature] • Simulated annealing without designing a cooling schedule. • Generating various alignments in different local minima.
System Configuration	

1 Simulated annealing algorithm

In many important practical problems, a solution is an arrangement of a set of discrete objects according to a given set of constraints. Such problems are typically known as combinatorial problems. The set of all solutions is referred to as the solution space and an energy function is defined for all solutions. To solve a combinatorial problem is to find a minimum-energy spot in the solution space.

A general strategy to search in the space is the method of 'iterative improvement'. The method requires a set of moves that can be used to modify a solution. One starts with an initial solution and examines its moves until a neighboring solution with a lower energy is discovered. The neighbor becomes the new solution and the process is continued to examine the neighbors of the new solution. This iteration terminates when it arrives at a spot that has locally minimum energy.

Simulated annealing algorithm is an extension of the method of iterative improvement based on an analogy between a combinatorial problem and the problem of determining the ground state of a physical system. To bring a fluid to a highly ordered state like a single crystal, a process called 'annealing' can be employed. We first melt the system by heating it to a high temperature, then cool it slowly, spending a long time at temperatures in the vicinity of the freezing point. Kirkpatrick et al suggested that better results to combinatorial problems can be obtained by simulating the annealing process of physical systems (Figure 1).

```
begin
  X0 := Initial solution ;
  {Tn}n=0,...,N-1 := Temperature (Cooling schedule);
  for n := 0 to N-1 do
  begin
    X'n := Some random neighboring solution of Xn;
    ΔE := E(X'n) - E(Xn);
    if ΔE < 0 then
      Xn+1 := X'n
    else
      if exp(-ΔE/Tn) ≥ random(0,1) then
        Xn+1 := X'n
      else
        Xn+1 := Xn
  end;
  Output XN;
end;
```

Figure 1 Simulated annealing algorithm

2 Multiple alignment as a combinatorial problem

There may be some ways to formulate multiple sequence alignment as a combinatorial problem. Kanehisa, a professor at Kyoto university, developed an ingenious formulation in order to solve multiple alignment problems by simulated annealing algorithm. We adopt his formulation.

D e t a i l s (2/3)

Kanehisa's idea is as follows. First, we make an initial alignment by adding a number of gaps to both head and tail of each sequence (Figure 2.1). To modify the alignment, we focus on one sequence in the alignment and select a gap and an amino acid randomly in that sequence. Moving the gap to the other side of the selected amino acid gives the modified alignment (Figure 2.2).

The energy of an alignment is calculated by summing up each correlation value of pairs of characters located in the same column. The correlation value comes from Dayhoff's odds matrix. If the energy of the modified alignment is lower than that of the previous one, the modified alignment is always regarded as a new alignment. If not, whether the modified one is regarded as a new alignment or not depends on the probability derived by temperature. The temperature is decided according to a cooling schedule. This annealing operation often brings good alignment (Figure 2.3).

```
"-------NAPATFQRCMNDILRPLLNKHCLVFSTSLD-------"
"-------LKQAPSIFQRHMDEAFRVFRKFCCVFSNNE-------"
"-------NSPTLFDEALHRDLADFRIQHPDLILLQAA-------"
"------@MANSPTICQLY@QEALEPIRKQFTSLIVIH-------"
"-------TCSPTICQLVVGQVLEPLRLKHPSLCMLHA-------"
"-------SPTLFEMQLAHILQPIRQAFPQCTILQASP-------"
```

Figure 2.1 An initial alignment

```
"-------NAPATFQRCMNDILRPLLNKHCLVFSTSLD-------"
"-------LKQAPSIFQRHMDEAFRVFRKFCCVFSNNE-------"
"-------NSPTLFDEALHRDLADFRIQHPDLILLQAA-------"
"------MANSPTICQLYV-QEALEPIRKQFTSLIVIH-------"
"-------TCSPTICQLVVGQVLEPLRLKHPSLCMLHA-------"
"-------SPTLFEMQLAHILQPIRQAFPQCTILQASP-------"
```

Figure 2.2 An alignment after the first move

```
"------NAPATFQ--RCM-NDIL--RPLLNKHCLVFSTSLD---"
"----LKQAPSIFQ--RHM--DEA-FRVF-RKFCCVFSNNE----"
"------NSPTLFDEALH-R-DLADFRIQH-PDLILLQAA-----"
"----MANSPTICQLYV-QEA-LEPIR-KQFTSLIVIH-------"
"-----TCSPTICQLVVGQ-V-LEPLRLKH-PSLCMLHA------"
"-------SPTLF-EMQLAHI-LQPIRQA-FPQCTILQASP----"
```

Figure 2.3 A good alignment

3 Scheduleless parallel simulated annealing

Designing a cooling schedule is troublesome because the optimal cooling schedule depends on the type and the scale of combinatorial problems. Without careful temperature reduction, a solution is trapped in a local minimum which has relatively high energy. Kimura, a member of ICOT, developed the method of parallel simulated annealing that makes it possible to avoid designing the cooling schedule.

D e t a i l s (3/3)

In Kimura's method, each process element (PE) maintains one solution and performs the annealing operation concurrently under a constant temperature that differs from PE to PE. The solutions obtained by the PEs are occasionally exchanged between PEs that hold neighbor temperatures (Figure 3). This exchange of solutions is controlled in some probabilistic way. Kimura proposed a scheme of the probabilistic exchange, and justified it from the viewpoint of the probability theory. He applied his method to a graph-partitioning problem, one of the representative combinatorial problem. That proved his method to be efficient.

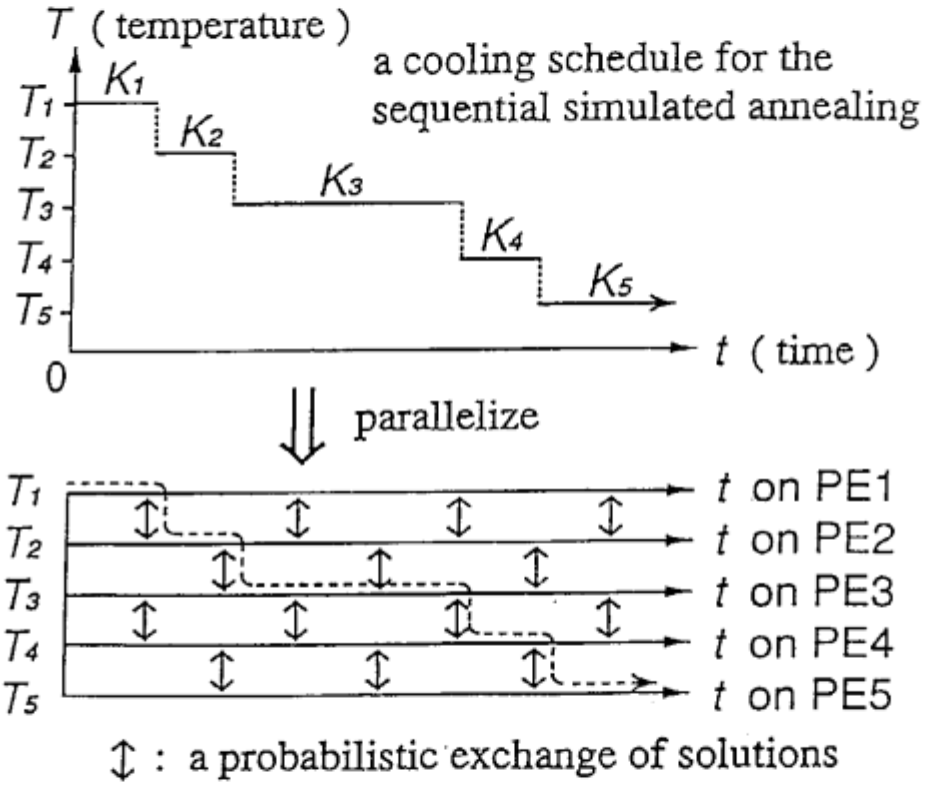
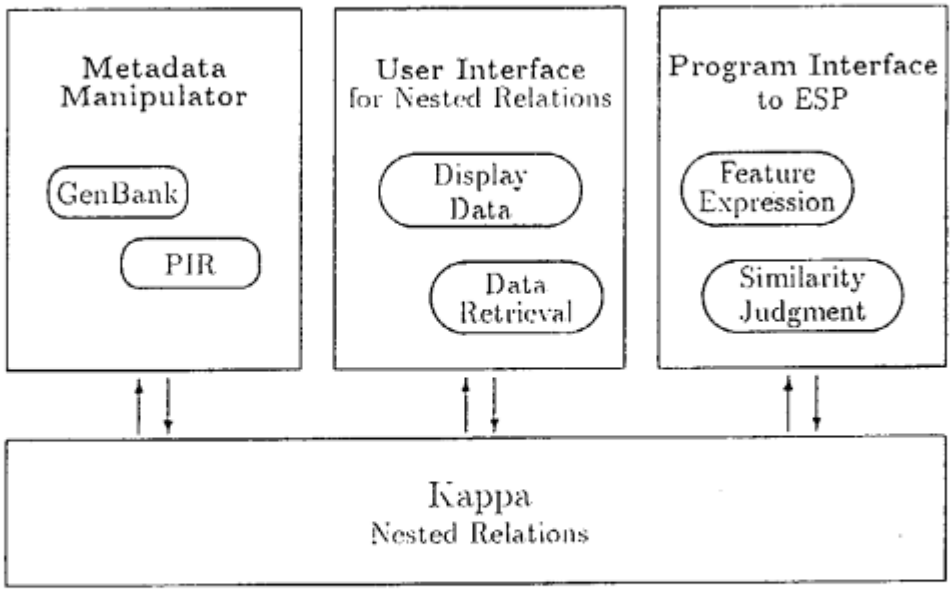


Figure 3 Scheduleless parallel simulated annealing

4 Demonstration

The demonstration system solves multiple sequence alignment problems by the parallel simulated annealing method. The multiple alignment problem is formulated as a combinatorial problem by Kanehisa's idea, and the simulated annealing operation is processed by Kimura's method.

Generally, it takes hundreds of hours for optimization by simulated annealing. The demonstration is a brief version of multiple alignment. It shows you gradual improvement of the alignment of some small protein sequences.

Title	Molecular Biological Database in Kappa
Purpose	We are providing a new-concept DBMS, together with several tools, for the effective use of molecular biological data. We are planning to integrate molecular biological databases on our database management system. First we stored GenBank and PIR; now we are researching the best way to use them together.
Outline & Features	Demonstration Outline: (1) Nested relational schemas for molecular biological database - of GenBank - of PIR (2) Functions and performance of the user interface of Kappa - Display Data - Data Retrieval (3) Examples of tools, their simplicity and performances (Functions of the program interface of Kappa) - Feature Expression - Similarity Judgment
System Configuration	

D e t a i l s (1/6)

1 Preliminary

1.1 Kappa

Kappa (Knowledge APPlication-oriented Advanced database management system) is one of the ICOT KBMS projects, and aims to provide DBMS for Knowledge Information Processing Systems (KIPS). Kappa is the DBMS on PSI-II/SIMPOS, while Kappa-P, which we are now developing, is the parallel version of Kappa, on PIM/PIMOS.

Kappa is a DBMS with the following features:

(1) Nested relational model is employed.

(2) Large amounts of data are effectively accessed.

(3) An user interface tuned for nested relational model is provided.

(4) ESP program interface and extended relational algebra are provided.

(5) Program interface can be customized for each application.

1.2 Nested Relational Model

The definition of the nested relational model (which is employed by Kappa) is intuitively as follows:

$NR \subseteq E_1 \times \ldots \times E_n$

$E_i ::= D \mid 2^{NR}$

Compared to the relational model:

$R \subseteq D_1 \times \ldots \times D_n$

where D_i is a domain, R is a relation and NR is a nested relation.

Relational:

Tour Schedule

Date	To	Group
90.6.4	ANL	Setting_G
90.6.25	ANL	Lecture_G

Group

Name	Member
Setting_G	Sugino
Lecture_G	Ichiyoshi
Lecture_G	Kondo
Lecture_G	Susaki

Nested Relational:

Tour Schedule

Date	To	Group	
		Name	Member
90.6.4	ANL	Setting_G	Sugino
90.6.25	ANL	Lecture_G	Ichiyoshi
			Kondo
			Susaki

1.3 Molecular Biology

DNA sequence: It can be considered as a string, whose length can be more than several hundred thousand. It consists of 4 characters, namely A,T,G,C.

amino acid sequence: Protein may also be considered as a string. It consists of 20 characters, which represent amino acids.

protein coding region: It is part of a **DNA sequence**. It is translated into **amino acid sequences** by a certain rule.

exon and intron: **Protein coding regions** of some organisms include sequences which are not translated into **amino acid sequences**. We call them **introns**, while **exons** are the translated parts.

genome: The 'full set' of **DNA sequences** of an organism. Notice that each human possesses two **genomes** in his chromosomes.

restriction enzyme: We use it to cut a **DNA sequence** to a modest length, to read DNA. The position it cuts is called **restriction site**.

1.4 Molecular Biological Databases

1.4.1 GenBank

There are two major databases of DNA sequences: GenBank[1] and EMBL[2]. They and DDBJ[3] have agreements on dividing tasks of data collection and exchanging data collected. The distribution forms of the data are flat files, in MT, FD, CD-ROM, or through networks. Some portion of distribution will remain in the flat file for the time being, while it will soon be something hierarchical, and exchanged between DBMS. GenBank began to manage their data on a relational database in 1989.

1.4.2 PIR

PIR[4] is the representative database of protein sequence. It contains a DNA sequence database in a similar format. It also uses flat file distribution. It is distributed with access methods on VAX/VMS, called NAQ and PSQ.

1.4.3 Other Databases

Protein Structure PDB[5] is the database of atomic coordinates of amino acids, namely 3-dimensional structures of proteins.

Maps of DNA We have maps of restriction sites, which are used as physical maps, and of relational distances between genes, which are used as genetic maps.

[1] Genetic Sequence Data Bank, IntelliGenetics Inc. and Los Alamos National Laboratory, US
[2] Nucleotide Sequence Data Library, European Molecular Biology Laboratory, EC
[3] DNA Data Base of Japan
[4] Protein Information Resource, National Biomedical Research Foundation, US
[5] Protein Data Bank

2 Specifications

2.1 Schema of GenBank in Kappa

The schema based on the nested relational model for GenBank is shown in Fig. 1 in detail.

gene : main table which has locus name, definition, accession, keywords, identifiers to the other tables, and so on.

reference : table which has authors, titles, journals in which the paper was published, and so on.

feature : consists of a region of the sequence and its feature.

seqdata : sequence data represented in string form.

2.2 Schema of PIR in Kappa

pir_gen : main table which includes names, placements, sources and sequence data.

pir_ref : table which has authors, titles, journals and so on.

pir_fea : consists of a region of the sequence and its feature.

Schema for PIR is shown in Fig. 2.

2.3 Stored Data

GenBank	Release 60.0 (89.6.15) 26323 entries, 32 M bases
PIR	Release 21.0 (89.6.30) 6158 entries, 1.7 M residues

3 Demonstration

3.1 Display Tables : Kappa User Interface

3.2 Retrieve Data

Example: make a gene table which consists of entries (records) whose reference Dr. Woese wrote.

$$\pi_{\text{ref\_id}}\left(\sigma_{\text{authors='Woese*'}}(\text{reference})\right) \bowtie \text{gene}$$

3.3 Feature Expression

3.4 Similarity Judgment

Flowchart is shown in Fig. 3.

Translation We select a DNA sequence in the feature table to translate into an amino acid sequence. The sequence is translated according to the table shown in Fig. 4.

DP matching We compare the *'translated'* sequence with another (selected and translated) sequence. The sequences are compared according to the table selected by the user shown in Fig. 5, for example.

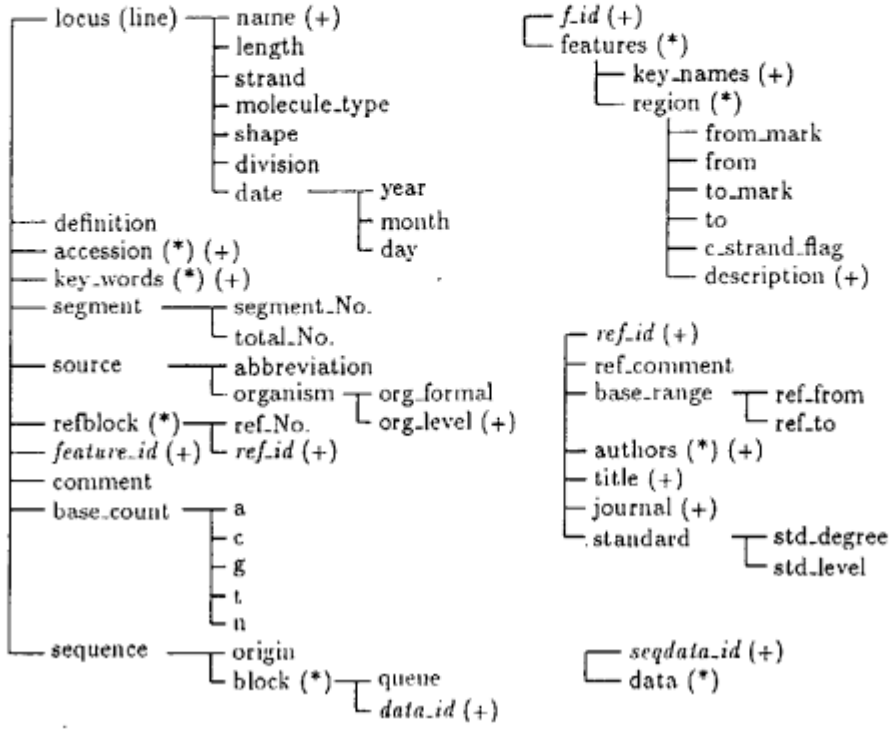


Fig. 1 *Schema of GenBank in Kappa*

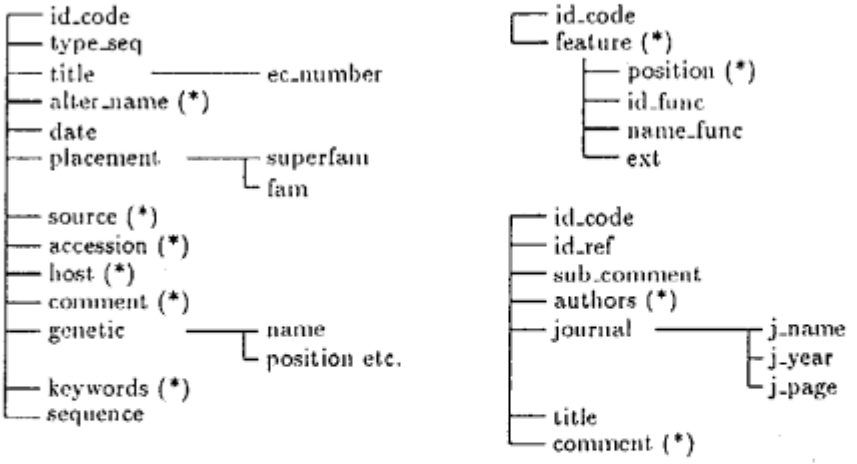


Fig. 2 *Schema of PIR in Kappa (excerpt)*

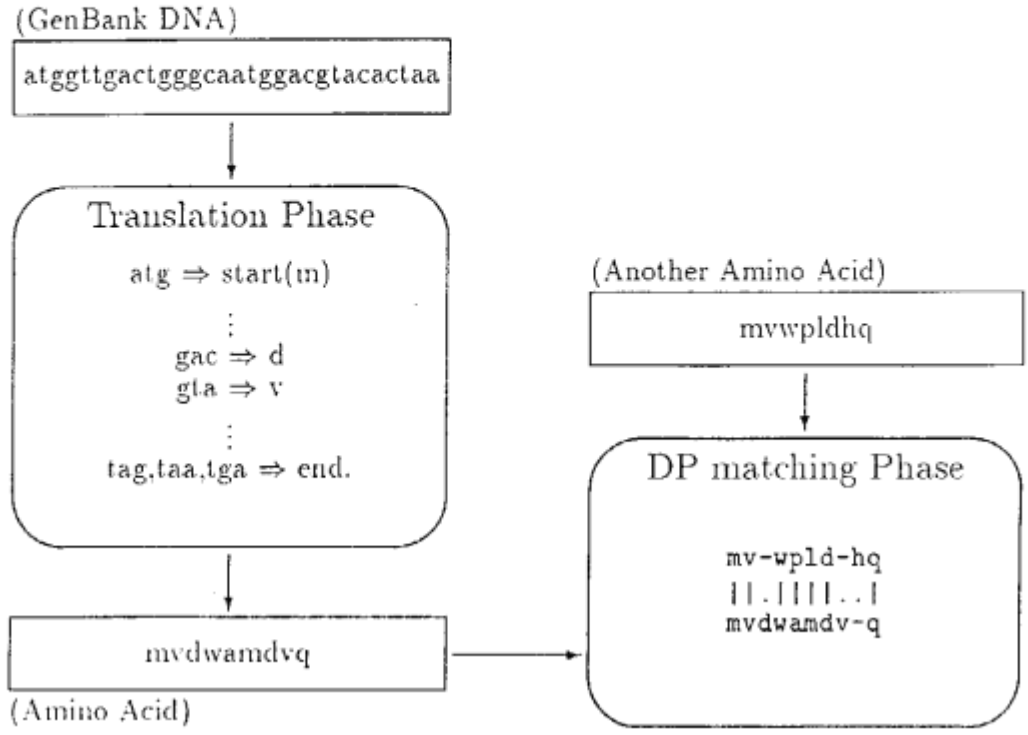


Fig. 3 *Similarity Judgment between DNA's*

Fig. 4 *Translation Table*

1st (5'end)	2nd T	2nd C	2nd A	2nd G	3rd (3'end)
T	f	s	y	c	T
					C
	l		*	*	A
			*	w	G
C	l	p	h	r	T
					C
			q		A
					G
A	i	t	n	s	T
					C
			k	r	A
	m				G
G	v	a	d	g	T
					C
			e		A
					G

(* terminate codon)

Fig. 5 *Mutation Table (Feng)*

Mutability	c	s	t	p	a	g	n	d	e
c: cysteine	6								
s: serine	4	6							
t: threonine	2	5	6						
p: proline	2	4	4	6					
a: alanine	2	5	5	5	6				
g: glycine	3	5	2	3	5	6			
n: asparagine	2	5	4	2	3	3	6		
d: aspartic acid	1	3	2	2	4	4	5	6	
e: glutamic acid	0	3	3	3	4	4	3	5	6
q: glutamine	1	3	3	3	3	2	3	4	4
h: histidine	2	3	2	3	2	1	4	3	2
r: arginine	2	3	3	3	2	3	2	2	2
k: lysine	0	3	4	2	3	2	4	3	4
m: methionine	2	1	3	2	2	1	1	0	1
i: isoleucine	2	2	3	2	2	2	2	1	1
l: leucine	2	2	2	3	2	2	1	1	1
v: varine	2	2	3	3	5	4	2	3	4
f: phenylalanine	3	3	1	2	2	1	1	1	0
y: tyrosine	3	3	2	2	2	2	3	2	1
w: tryptophan	3	2	1	2	2	3	0	0	1

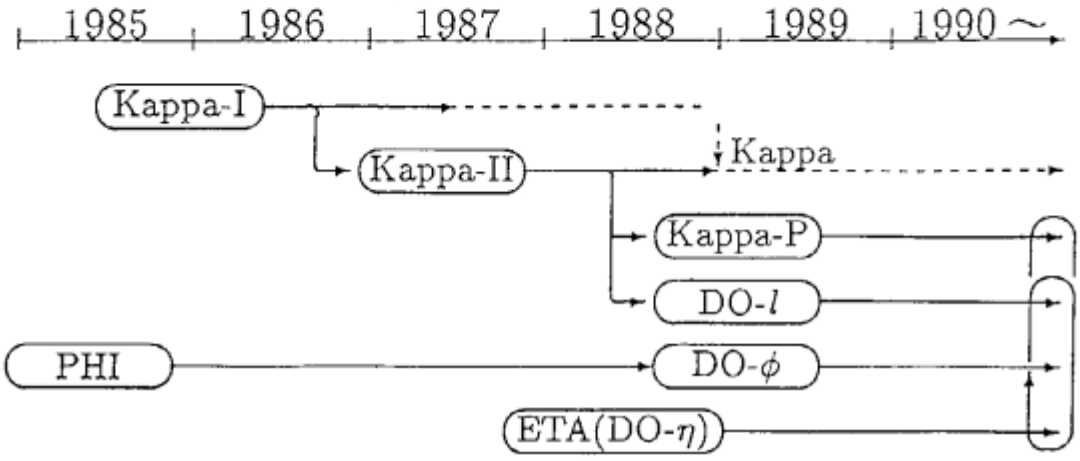
知識ベース管理システム Kappa と遺伝子情報検索への応用実験

第3研究室　横田 一正
田中 秀俊
第7研究室　新田 克己

- 知識ベース管理システムの枠組 / 計画
- 遺伝子情報とは何か?
- なぜ遺伝子情報処理か?
- 遺伝子情報処理の計画
- Kappa の特徴
- 遺伝子配列情報データベース GenBank
- デモ: GenBank in Kappa
- 遺伝子情報に関する応用
- デモ: 配列情報の翻訳と類似検索

知識ベース管理システムの枠組 / 計画

- これまでの研究開発経緯



DOOD
⊕
並列知識処理システム
(自然言語, 証明支援, CAD, 遺伝子情報, ···)
⊕
PSI/Multi-PSI/PIM, SIMPOS/PIMOS, ESP/KL1 環境

- 並列 DBMS、知識ベース / 知識表現モデル、応用の 3 層

	KBMS
	応用 (分子 & 生物 DB)
知識ベース	(DOOD, 生物 DB?, newCIL, DP-cu)
	データベース (並列 DBMS)

遺伝子情報とは何か?

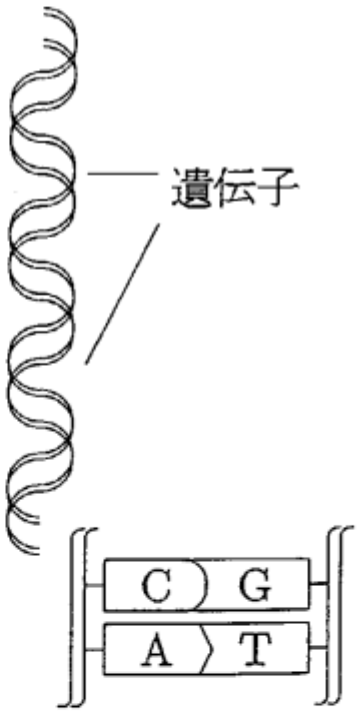
DNA: 核酸 (A,T,C,G) の 2 重らせん — 30 億個 (2m)

ゲノム: DNA 全体

遺伝子: ゲノムの部分配列 — 10 万種類

mRNA: 遺伝子をコピー

アミノ酸配列: 核酸 3 個
⇒ アミノ酸 1 個の翻訳



タンパク質: アミノ酸配列の構造化 (折り畳み)

なぜ遺伝子情報処理か?

- 応用の重要性

  - タンパク質 (主として医薬品) の合成
    目標となるタンパク質を生成する遺伝子部分を探索
    (⇒ その遺伝子を適当な方法で増やしタンパク質を生成)

    例：インシュリン (糖の消化酵素) の開発、
    AIDS ワクチンの開発

  - 遺伝病・癌の解明

    例: 鎌型赤血球貧血病の解明
    ⇒ (答) タンパク質 1 箇所の生成ミス
    正常なヘモグロビン：グルタミン酸
    鎌状のヘモグロビン：バリン

  - 生物の進化過程の解明
    例: チンパンジー (オランウータン科 ⇒ ヒト科)

- 情報量の爆発
  例: 遺伝子の配列情報 ⇒ 年間倍増のペース

- 情報処理としてはまだ未開拓分野

遺伝子情報処理の計画

- 統合的な分子生物データベース
  現状: 多くのデータベースにモデルのない状態

  - 遺伝子データベースのモデリング

  - 種々の実体 (系統発生、種、クローン、染色体、遺伝子、RNA、タンパク質、分子、など) を含む知識ベース / データベースの研究開発

  - Kappa 上に統合的な知識ベース / データベースを試作

- タンパク質の機能予測

  現状: 機能や構造のわかっているタンパク質は
  全体の数パーセントに過ぎない！

  - タンパク質の (進化、機能、構造ごとの) 分類

  - 保存パターンの抽出 (アミノ酸配列や、立体構造などで共通するパターンを抽出する)

  - 保存パターンと構造、機能との関係抽出 (抽出されたパターンと蛋白質の構造や機能との関係を求める)

  - 上記の保存パターンの抽出と機能予測の (PIM 上の) システムを研究試作

Kappa の特徴

- 非正規関係モデルを採用
  - 複雑な知識 / データを素直に表現
  - 結合演算の減少により効率的な処理
- PSI/Multi-PSI/PIM 上で効率的な動作
- 大量データ用に 2 次記憶と主記憶の効率的な使用
- 応用向けにユーザ・インタフェースを定義可能
- ESP/KL1 と親和性のあるプログラム・インタフェース
- 非正規関係に適したの端末インタフェース (⇒ デモ)

遺伝子配列情報データベース GenBank

核酸配列、記載文献、特徴記述などのデータ
27,000 遺伝子を格納 (120,000,000 文字)
年間倍増のペース

階層的なデータ構造をとる大量なデータ
属性 1 つに複数のデータ (著者、文献など)
. . . 関係 DB への格納には不向き

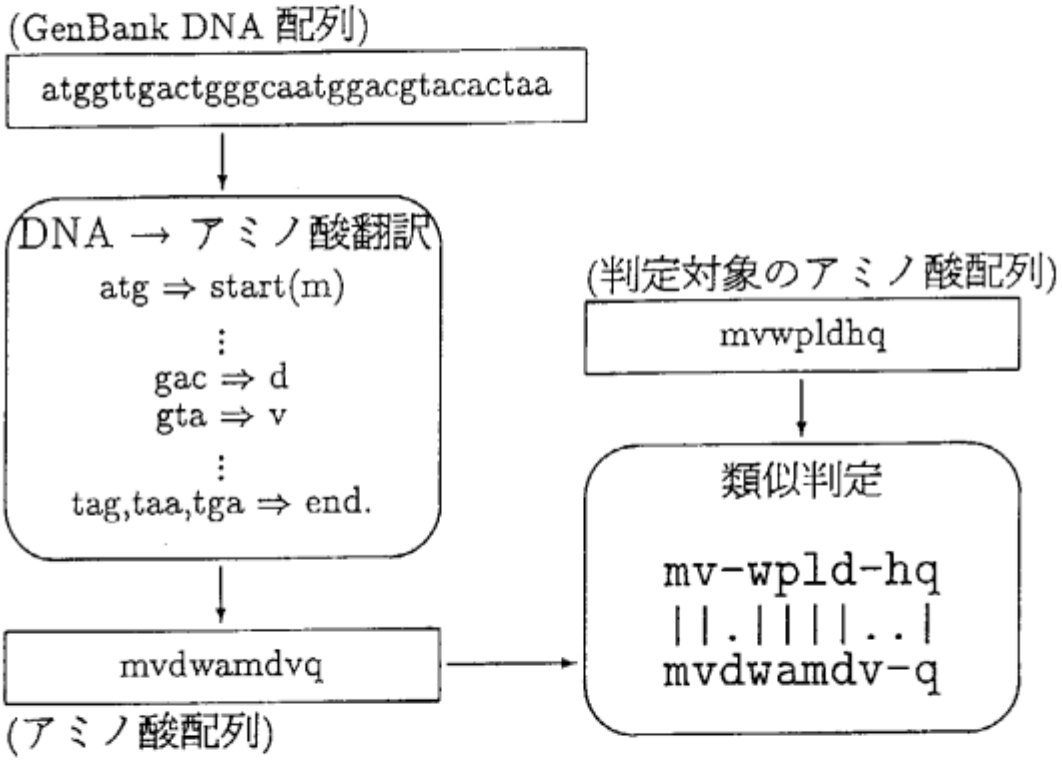
3ヶ月毎に版が変わる。
データ構成に変更が多い。
. . . 関係 DB では変更への対応が困難

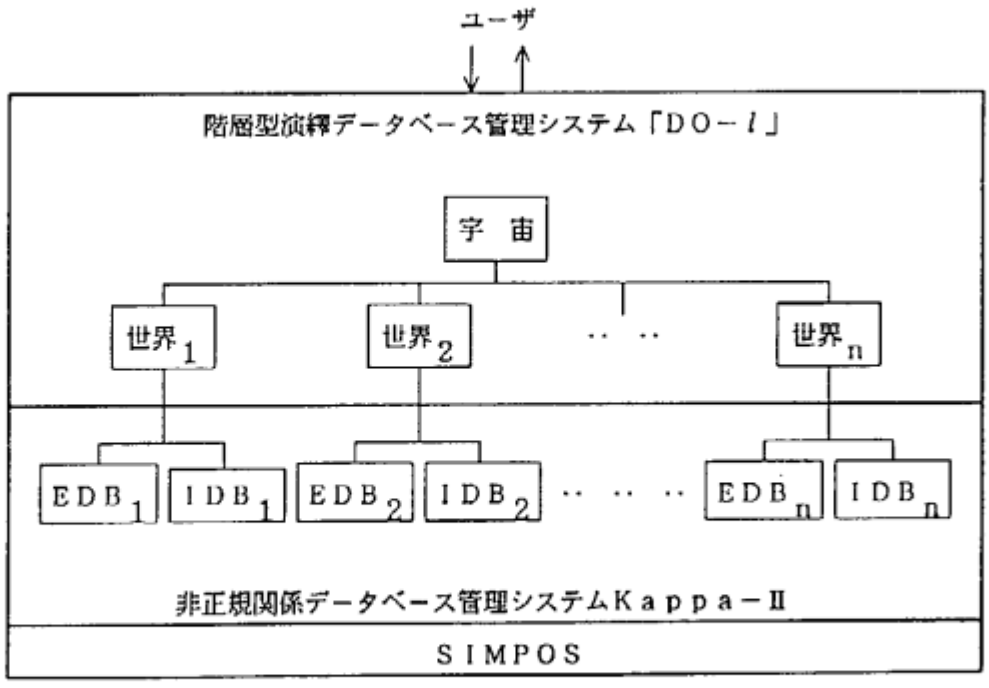
Kappa に格納した GenBank

GenBank の約 1/4 (7285 遺伝子 = 2200 万文字)
→ 3000 万文字分の記憶領域に格納
(インデクスは 10 個付加)

PSI-II の 2 次記憶に格納

4 つのテーブルに分けて格納。
(= 関係 DB では 38 テーブル必要)
遺伝子、文献、特徴、核酸配列の 4 つ。


(GenBank DNA 配列)
atggttgactgggcaatggacgtacactaa
DNA → アミノ酸翻訳
atg ⇒ start(m)
gac ⇒ d
gta ⇒ v
tag,taa,tga ⇒ end.
mvdwamdvq
(アミノ酸配列)
(判定対象のアミノ酸配列)
mvwpldhq
類似判定
mv-wpld-hq
||.||||..|
mvdwamdv-q

題　名	階層型演繹データベース実験システム 「ＤＯ－ｌ」
目　的	Ｋａｐｐａ上の知識ベースシステムの実験システムであり、構造化された事実やルールを知識として表現し、推論することを目的とし、今後の演繹・オブジェクト指向データベース（ＤＯＯＤ）システムの基盤技術の一つとなることを目指している。
概要及び特徴	本システムは、以下の特徴を持っている。 1．非正規関係に対応した知識表現言語ＣＲＬで事実とルールを表現している。 2．演繹データベースは、モジュール化され、複数の世界として管理される。 3．各世界間には半順序関係があり、それに従って事実とルールが動的に継承される。 4．各世界の知識、階層関係及びそれらの更新を管理するために『宇宙』が介在する。 5．各世界と宇宙はプロセスに対応しており、問合せはプロセス間通信によって、協調的に処理される。 問合せ処理の概要は、以下の通りである。 1．問合せに対し必要なルールが各世界で抽出され、問合せの束縛情報がそれらに伝播され、抽出されたルールは問合せ処理の効率化のためにマジック集合法で変換される。 2．問合せは、述語（ＣＲＬ項）間の依存グラフに従って、世界間で協調しながら順次セミナイーブ評価法で評価される。
構　成	

デ　　モ　　内　　容　（1／3）

本システムでは、知識を分類しモジュール化し階層型演繹データベースとすることによって、以下のことが行える。

・特定のモジュール化された知識だけを使って問合せを評価する。

・新しい知識を既存の知識に影響を与えることなく付加する。

・各知識モジュールをプロセスとし、それらのプロセス間通信によって問合せを協調的に処理する。

本システムは図１にあるように、世界間の階層関係を木構造のメニューとして表示するユーザインタフェース（ＴＲＥＥウインドウ）を持つ。このウインドウ上で特定の世界をマウスでクリックすると、その世界に関してのＩＤＢやＥＤＢの情報の表示、ルールの追加、継承関係の変更などができる。

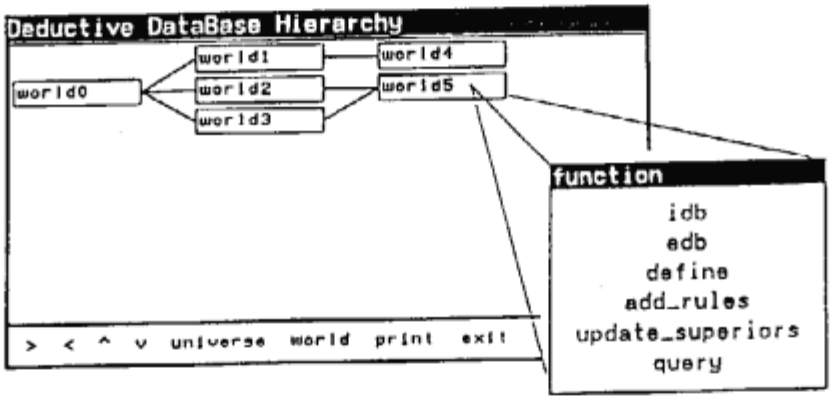


図１　世界間の階層関係を示すＴＲＥＥウインドウ

ＴＲＥＥウインドウのｕｎｉｖｅｒｓｅを選択し、問合せを与えると、図２のようなモニタウインドウが表示される。このウインドウは、世界間の階層関係に対応して自動的に区切られる（最上位のウインドウは宇宙プロセスに対応している）。区切られた各ウインドウは、対応する世界プロセスの処理の様子と他のプロセスとの通信内容などを表示する。

特定の問合せを宇宙に与えると、答えることのできる可能性をもつ世界が自動的に選択され、処理が順次行われる。例えば、問合せに対して図１のｗｏｒｌｄ４とｗｏｒｌｄ５が選択されると、それぞれの知識を使って問合せ処理が行われ、２種類の（必ずしも同じでない）答えを得ることができる。問合せとしてはこの他に、対象の世界を特定することもできる。

デ　　モ　　内　　容　（2／3）

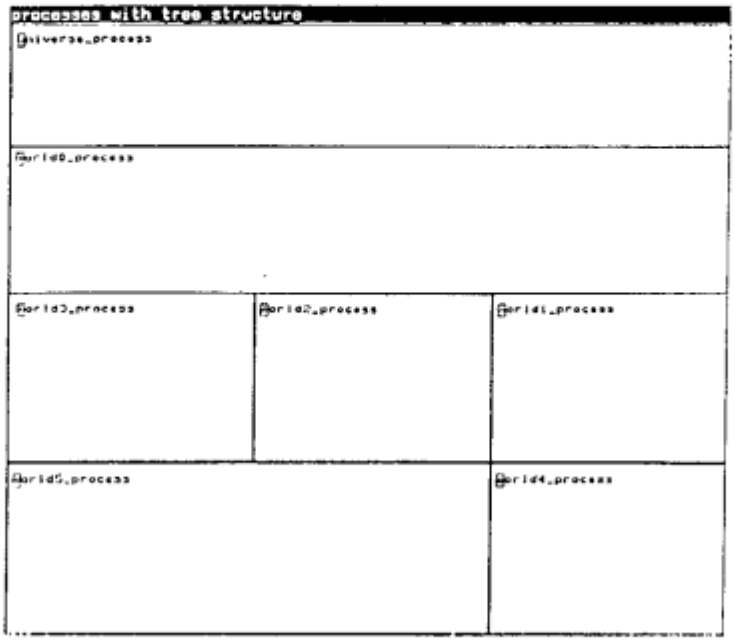


図2　各プロセスの処理を示すモニタウインドウ

ルールの追加や世界間の継承の変更も、ＴＲＥＥウインドウから容易に行うことができる（図3）。但し、ここでの追加、変更は、データベースの更新ではなく、問合せ（トランザクション）ごとに宇宙を動的に再構成することによって、問合せ処理の拡張として位置付けている。この拡張された問合せ処理によって、不確定情報を組み込んだり相反する規則をそれぞれ“仮説”として追加し、それに基づいた推論（問合せ処理）を行うこともできる。

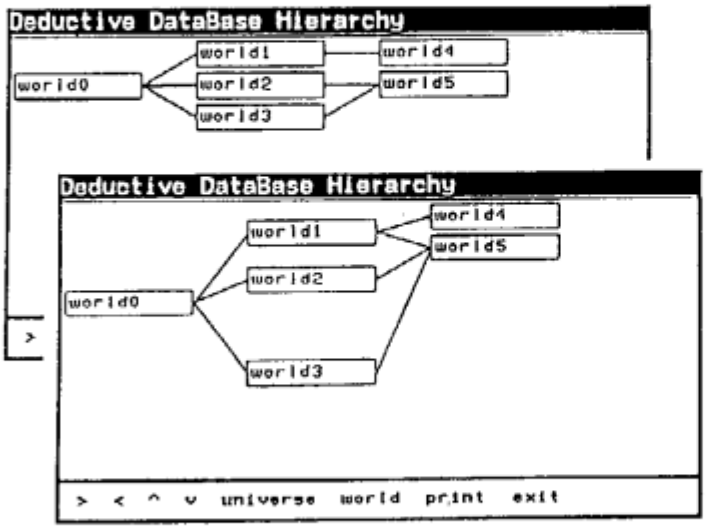


図3　世界の継承関係の変更

また、再帰問合せにおいて、答えの経路を求めることができる。巡回データの場合の経路は、ボトムアップ評価では一般的に問合せ処理の停止性が保証されない。そこで本システムでは、経路情報をＣＲＬ項のパラメータとして持ち、その情報をトップダウンに評価することによって経路情報の問合せ処理を実現している。この機構に基づいて、さらに一般的に、リスト処理や算術演算などに基づく集約関数を実現することができる。

デ　　モ　　内　　容　（3／3）

図4は、各プロセスの処理やプロセス間通信の内容を示すモニタウインドウである。

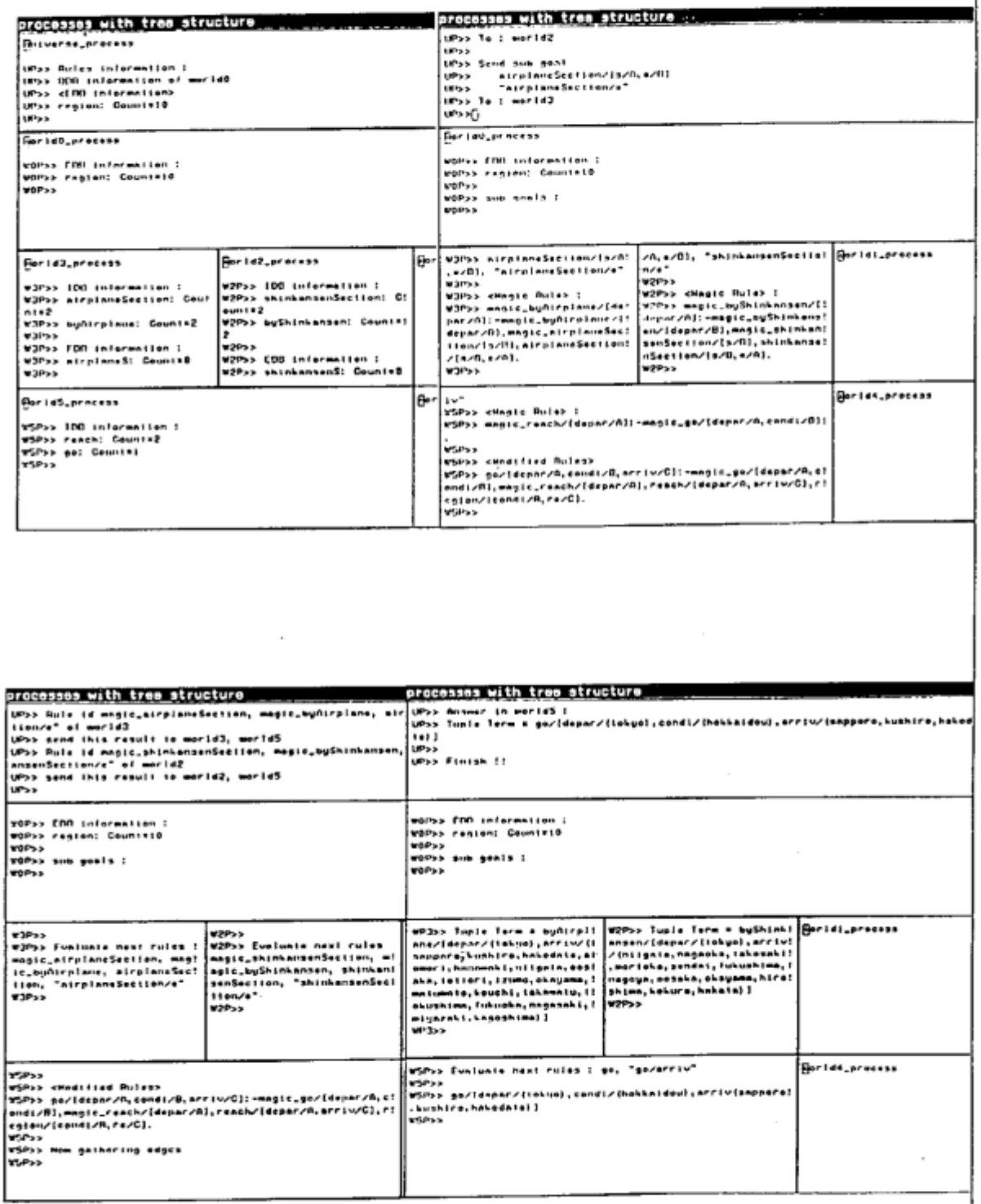
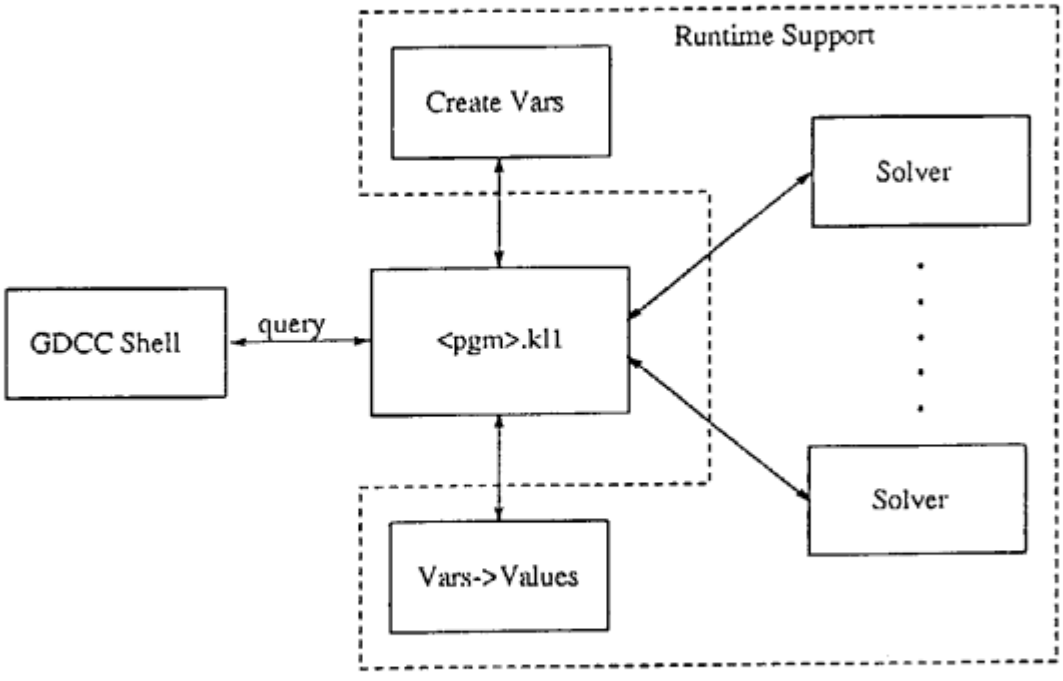


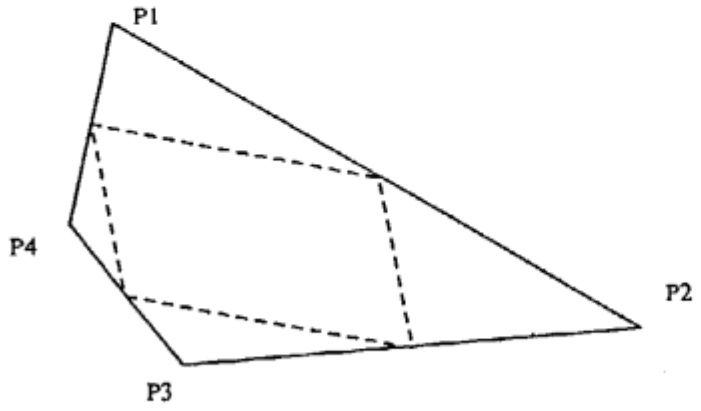
図4　各プロセスの処理やプロセス間通信の内容を示すモニタウインドウ

Title	Guarded Definite Clause with Constraints Experimental System
Purpose	1. Research on Parallel Constraint Solving 2. Commited Choice Approach to Control in Parallelism 3. Test Bed for Parallel CAL
Outline & Features	1. Amalgamation of Concurrent Logic Programming and Constraint Paradigm 2. Provision For Multiple Constraint Solvers 3. Solution of non-linear Algebraic Equations
System Configuration	

(2) Quadrilateral Mid-Points Problem

```
GDCC Listener

?- para:start(p(X1,Y1),p(X2,Y2),
               p(X3,Y3),p(X4,Y4)),
   alg(X1,X2,X3,X4,Y1,Y2,Y3,Y4).
 X1 = atom$8389319
 X2 = atom$8389320
 X3 = atom$8389321
 X4 = atom$8389322
 Y1 = atom$8389323
 Y2 = atom$8389324
 Y3 = atom$8389325
 Y4 = atom$8389326
 yes.

?- □
```

```
:- module para.
:- domains alg(=).
:- public start/4.
start(P1,P2,P3,P4) :- true |
    create_points([A,B,C,D]),
    mid_points([P1,P2,P3,P4,P1],[A,B,C,D]),
    check_para(A,B,C,D).

{input  alg X1,Y1,X2,Y2,X3,Y3,X4,Y4
}check_para(p(X1,Y1),p(X2,Y2),p(X3,Y3),p(X4,Y4)) :-
        alg:(X1-X2)*(Y3-Y4)=(Y1-Y2)*(X3-X4),
        alg:(X1-X4)*(Y2-Y3)=(Y1-Y4)*(X2-X3) |
    true.

{input alg X1,Y1,X2,Y2,X3,Y3}
mid(p(X1,Y1),p(X2,Y2),p(X3,Y3)) :- true |
    alg:+2*X3=X1+X2,
    alg:+2*Y3=Y1+Y2.

{create alg X,Y}
create_point(P) :- true | P=p(X,Y).

create_points([]).
create_points([I|Is]) :- true |
    create_point(I),
    create_points(Is).

mid_points([_],[]).
mid_points([A,B|Is],[M|Ms]) :- true |
    mid(A,B,M),
    mid_points([B|Is],Ms).
```



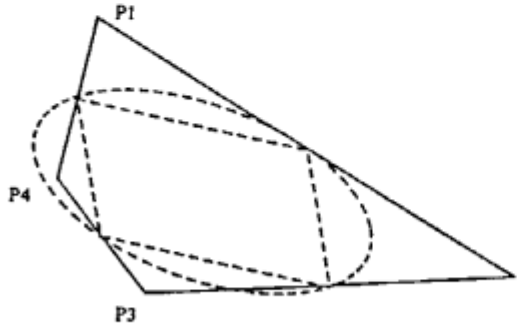
(3) Ellipse Problem

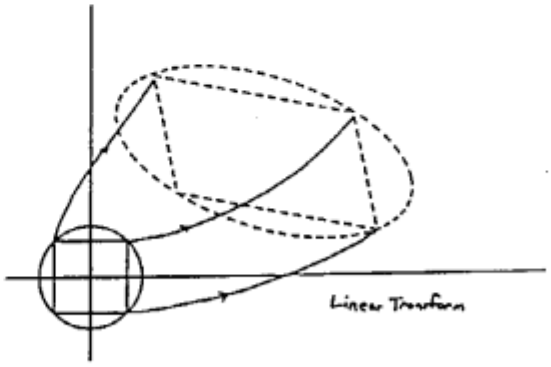
GDCC Listener

```
?- alg(Y1, X, Y), P1=p(-1, Y1), P2=p(-1, 1),
   P3=p(1, 1), P4=p(1, -1),
   ellipse:start(P1, P2, P3, P4, p(X, Y)).
 P1 = p(-1, atom$8389295)
 P2 = p(-1, 1)
 P3 = p(1, 1)
 P4 = p(1, -1)
rule(X*X*Y1 = -2*atom$8389312 *Y+
 -2*atom$8389312*X+3*X*X+1*X*Y1+
  2*atom$8389312+2*Y+ -1*X+ -2* 1)
rule(atom$8389312*Y1 = 2*X*Y1+
  3*atom$8389312+4*Y+ -2*X+ -1*Y1+ -1*1)
rule(atom$8389312*atom$8389312 =
  2*atom$8389312 *X+ -2*X*X+ 1* 1)
 yes.

?- □
```

```
:- module ellipse.
:- domains alg(=).
:- public start/5.
start(P1,P2,P3,P4,P) :- true |
  create_points([A,B,C,D]),
  mid_points([P1,P2,P3,P4,P1],[A,B,C,D]),
  calc_ellipse(A,B,C,D,P).

{input  alg X1,Y1,X2,Y2,X3,Y3,X4,Y4
}calc_ellipse(P1,P2,P3,P4,P) :-
    P1=p(X1,Y1),P2=p(X2,Y2),
    P3=p(X3,Y3),P4=p(X4,Y4),
    alg:(X1-X2)*(Y3-Y4)=(Y1-Y2)*(X3-X4),
    alg:(X1-X4)*(Y2-Y3)=(Y1-Y4)*(X2-X3) |
  transform_matrix(T),
  calc_transform(T,P1,P2,P3),
  create_point(Pp),
  transform(Pp,T,P),
  circle_2(Pp).

{input alg X,Y}
circle_2(p(X,Y)) :- true |
  alg:X^+2 + Y^+2 = +2.

{create alg A1,B1,C1,A2,B2,C2}
transform_matrix(T) :- true |
  T=[[A1,B1,C1],[A2,B2,C2]].
```

```
calc_transform(T,P1,P2,P3) :- true |
  square_2(LL,UL,UR,LR),
  transform(LL,T,P1),
  transform(UL,T,P2),
  transform(UR,T,P3).

calc_transform1(T,P1,P2,P3,P4) :-
    true |
  square_2(LL,UL,UR,LR),
  transform(LL,T,P1),
  transform(UL,T,P2),
  transform(UR,T,P3),
  transform(LR,T,P4).

square_2(LL,UL,UR,LR) :- true |
  UL=p(-1,+1), UR=p(+1,+1),
  LL=p(-1,-1), LR=p(+1,-1).

{input alg Xp,Yp}
transform(P,[L1,L2],Pp) :- true |
  Pp = p(Xp,Yp),
  transform1(P,L1,Xp),
  transform1(P,L2,Yp).

{input alg X,Y,A,B,C,P}
transform1(p(X,Y),[A,B,C],P) :- true |
  alg:P=A*X+B*Y+C.
```





制約ロジック・プログラミング実験システム CAL

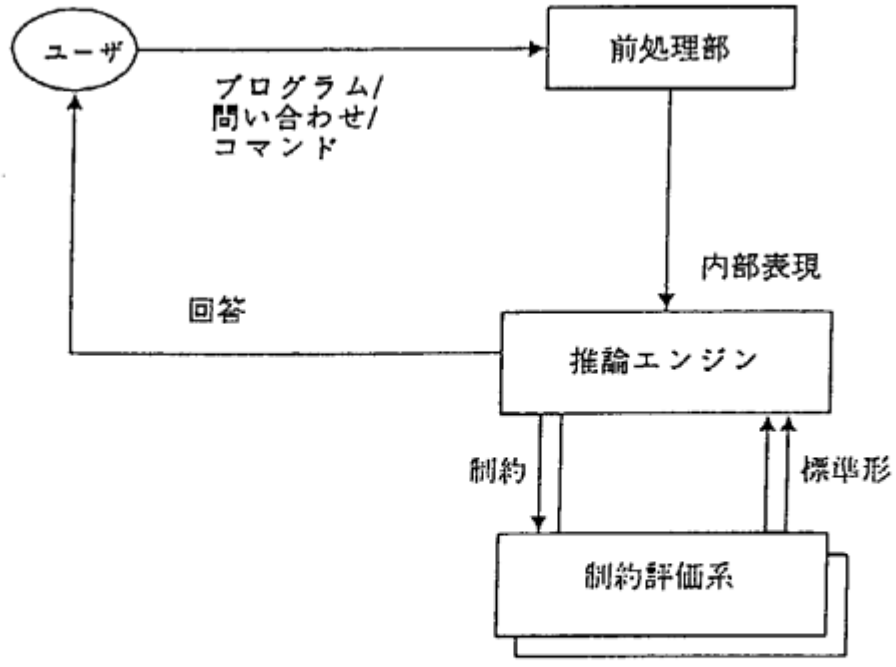
- 目的:
  1) より書きやすく、読みやすいプログラムを目指す
  2) より抽象度の高いプログラミングを目指す
  3) より効率の良い問題解決技法の研究を目指す

$CAL =$ *Contrainte* — 制約プログラミング

$+$

Logique — ロジック・プログラミング

- 非常に高水準の宣言的プログラミング言語である
- 強力な問題記述能力と問題解決能力を持っている
- ロジック・プログラミング部分の並列化により、並列化も可能である



★ CALの例：鶴亀算

- 「鶴の数」、「亀の数」、「頭の数」、「足の数」の間の関係からなる問題
- これらの間には、次の2つの関係がある

亀の数 + 鶴の数 = 頭の数

4×亀の数 + 2×鶴の数 = 足の数

```
trkm(Tsuru, Kame, Atama, Ashi) :-
        Kame + Tsuru = Atama,
        4*Kame + 2*Tsuru = Ashi.
```

このプログラムに対して、次のような問い合わせを行う事ができる：

```
亀の数 = 2          =>    頭の数 = 5
鶴の数 = 3                足の数 = 14

頭の数 = 6          =>    亀の数 = 3
足の数 = 18               鶴の数 = 3
   :                         :
```

もし、同じ問題をPrologを使って解くと.....

```
trkm(Tsuru, Kame, Atama, Ashi) :– int(Kame), int(Tsuru), !,
        Atama is Kame + Tsuru,
        Ashi is 4*Kame + 2*Tsuru.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Kame), int(Atama), !,
        Tsuru is Atama – Kame,
        Ashi is 3*Kame + Atama.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Kame), int(Ashi), !,
        Tsuru is Ashi/2 – 2*Kame,
        Atama is Ashi/2 – Kame.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Tsuru), int(Atama), !,
        Kame is Atama –Tsuru,
        Ashi is 4*Atama – 2*Tsuru.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Tsuru), int(Ashi), !,
        Kame is Ashi/4 – Tsuru/2,
        Atama is Ashi/4 + Tsuru/2.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Atama), int(Ashi), !,
        Kame is Atama/3 –Ashi/6,
        Tsuru is 2*Atama + Ashi/6.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Kame), !,
        Tsuru = Atama – Kame,
        Ashi = 2*Kame + 2*Atama.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Tsuru), !,
        Kame = Atama – Tsuru,
        Ashi = 4*Atama – 2*Tsuru.
trkm(Tsuru, Kame, Atama, Ashi) :– int(Atama), !,
        Kame = Atama – Tsuru,
        Ashi = 4*Atama – 2*Tsuru.

            :
            :
            :
```

Loan

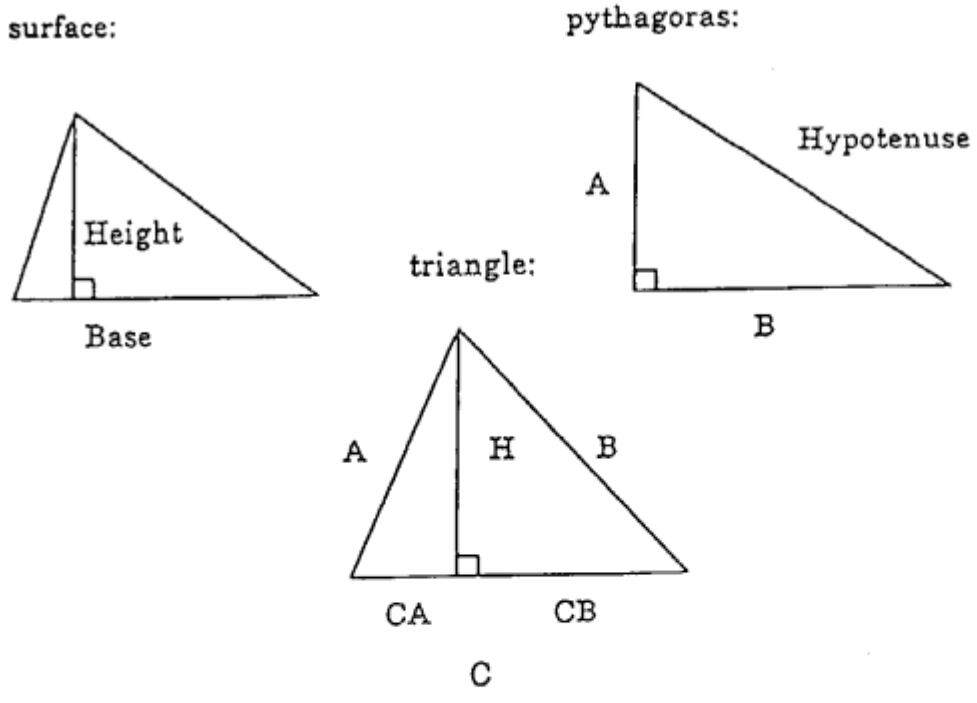
```
loan(R, P, T, SB, EB) :-
        T=0, SB=EB.
loan(R, P, T, SB, EB) :-
        non_zero(T), Interest=R*SB,
        NB=SB+Interest-P, NT=T-1,
        loan(R, P, NT, NB, EB).
non_zero(A) :- A*Alpha=1.
```

Now we can ask some questions:

$$\left.\begin{array}{c} Rate = 0.01/month \\ Time = 15 months \\ StartBalance = 10000 \end{array}\right\} \Rightarrow \text{Payment} \approx 725$$

★ CALの例：ヘロンの公式(非線形制約)

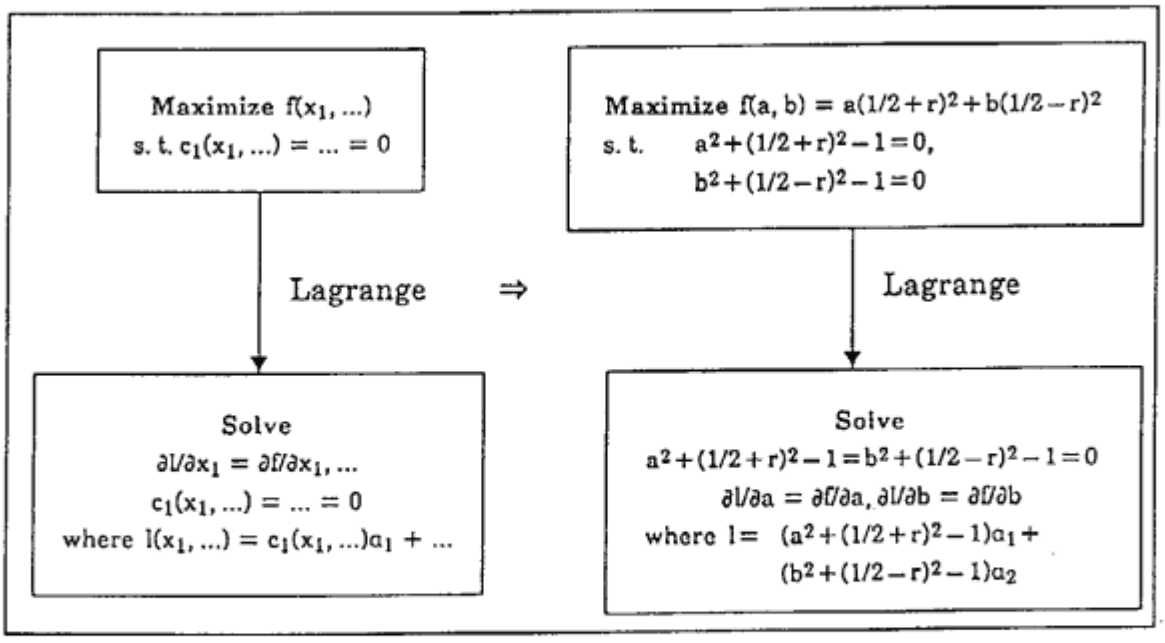
```
surface(Height, Base, Area) :-
        alg:Base*Height = 2*Area.
pythagoras(A, B, Hypotenuse) :-
        A^2+B^2 = Hypotenuse^2.

triangle(A, B, C, S) :-
        alg:C = CA+CB,
        pythagoras(CA, H, A),
        pythagoras(CB, H, B),
        surface(H, C, S).
```

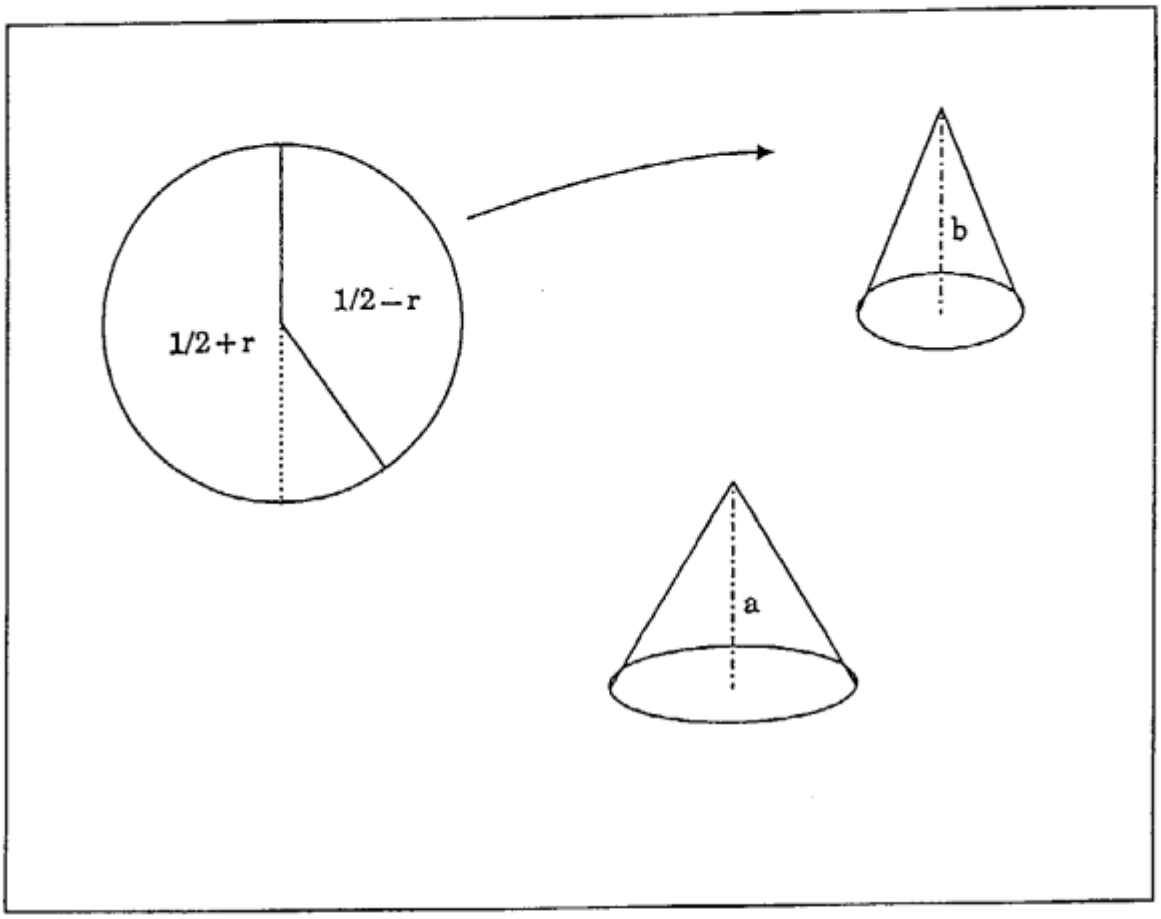


★CALの例：円すいの体積の和の最大化(動的に得られる制約)

プログラム、問い合わせ、回答

```
:- public lagrange/3.
lagrange(F, Constraints, Vars) :-
        construct_l(Constraints, L),
        partials(Vars, F, L).
partials([ ], _, _) :- !.
partials([Var|Vars], F, L) :-
        dif(F, Var) = dif(L, Var):alg,
        partials(Vars, F, L).
construct_l([ ], 0) :- !.
construct_l([C|Cs], C*Alpha + L) :-
        C = 0:alg,
        construct_l(Cs, L), !.

?-lagrange((1/2+r)^2*a+(1/2-r)^2*b,
                [a^2+(1/2+r)^2 = 1, b^2+(1/2-r)^2 = 1],
                [a, b, r]).

r^7=(29/12)*r^5+(-17/48)*r^3+(5/576)*r
```

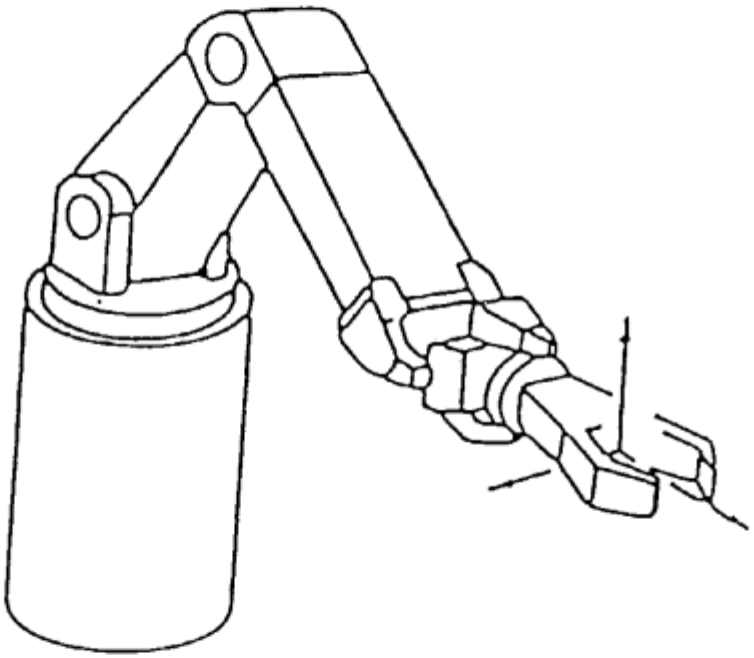




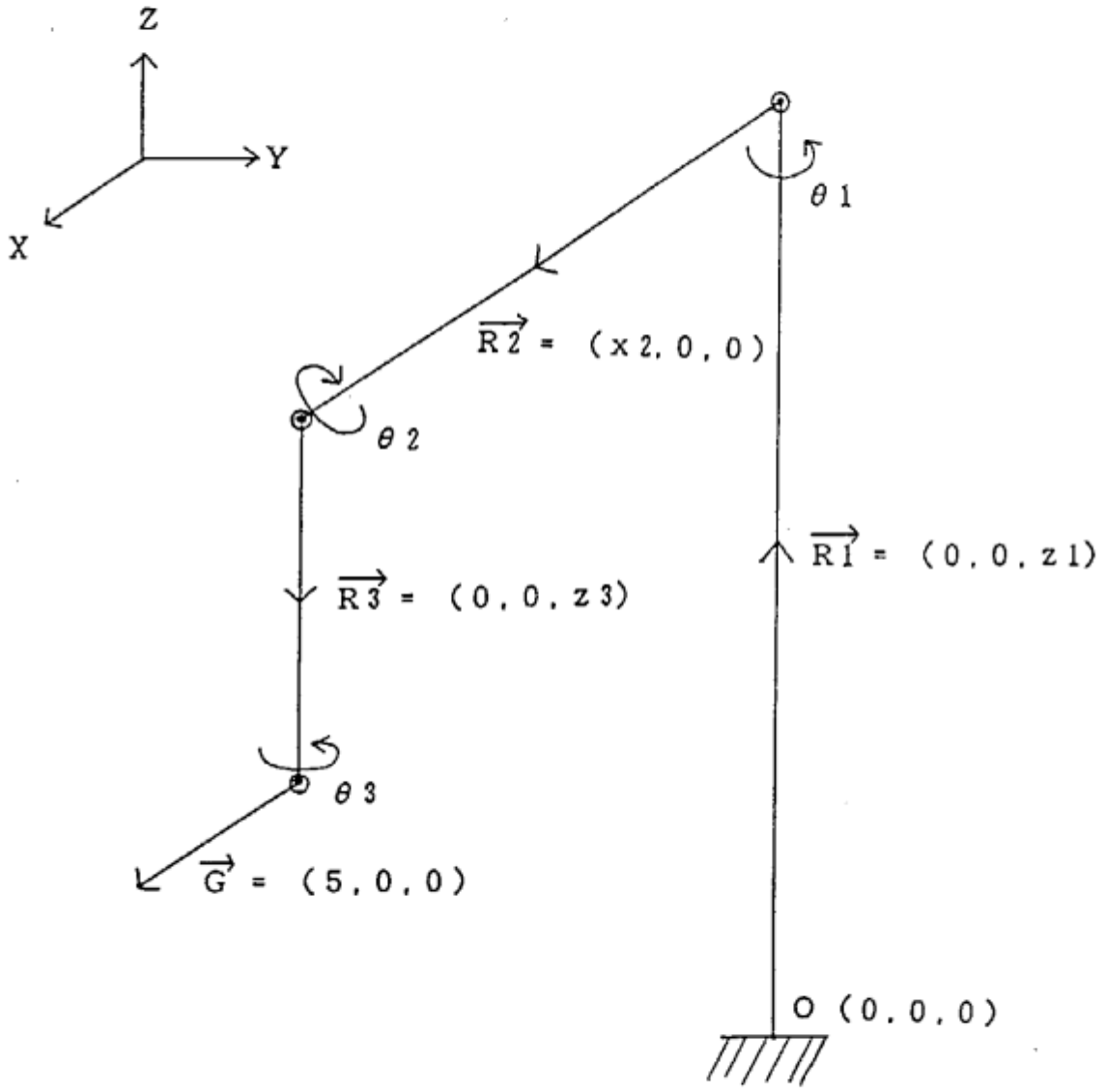
問い合わせ

```
?−lagrange((1/2+r)^2*a+(1/2−r)^2*b,
           [a^2+(1/2+r)^2 = 1, b^2+(1/2−r)^2 = 1],
           [a, b, r]).
```

Handling Robot Kinematics



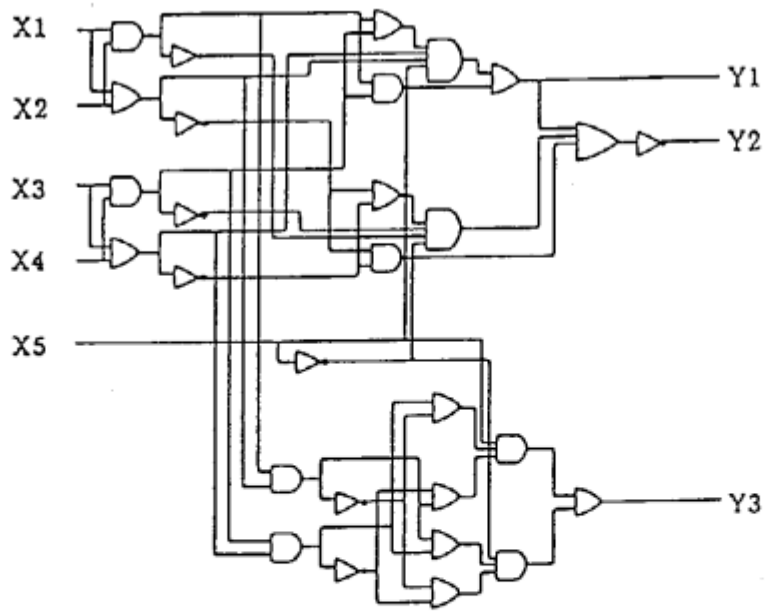
ex. Vector Sketch of a Robot
(3 Joints and 6 freedoms)



```
 yes
?-  robot3:
 robot([[cos3,sin3,0,0,z3,0,0,1],
        [cos2,sin2,x2,0,0,1,0,0],
        [cos1,sin1,0,0,z1,0,0,1]],
        5,0,0,1,0,0,0,1,0,
        40,-30,20,1/3,2/3,-2/3,-2/3,2/3,1/3)
sin1^2 = 1/5 .
cos2 = -5/3*sin1 .
sin2 = 2/3 .
cos1 = -2*sin1 .
z1 = 6 .
cos3 = 2*sin1 .
sin3 = -1*sin1 .
z3 = 26 .
x2 = 105*sin1 .

 yes
?-
```

★ブールCALの例：1の数を数える回路

```
circuit(X1, X2, X3, X4, X5, Y1, Y2, Y3) :-
        bool:I1 = X1&X2,  bool:I2 = X1VX2,  bool:I3 = X3&X4,
        bool:I4 = X3VX4,  bool:I5 = ~I1,  bool:I6 = ~I2,
        bool:I7 = ~I3,  bool:I8 = ~I4,  bool:I9 = I1VI3,
        bool:I10 = I1&I3,  bool:I11 = I6VI8,  bool:I12 = I6&I8,
        bool:I13 = ~X5,  bool:I14 = I5&I2,  bool:I15 = I7&I4,
        bool:I16 = ~I14,  bool:I17 = ~I15,  bool:I18 = I15VI16,
        bool:I19 = I14VI17,  bool:I20 = I14VI15,  bool:I21 = I16VI17,
        bool:I22 = I9&I4&I2&X5,  bool:I23 = I11&I7&I5&I13,
        bool:I24 = X5&I8&I9,  bool:I25 = I13&I20&I21,
        bool:I26 = I22VI10,  bool:I27 = I26VI23VI2,
        bool:Y1 = I26,  bool:Y2 = ~I27,  bool:Y3 = I24VI25.
```



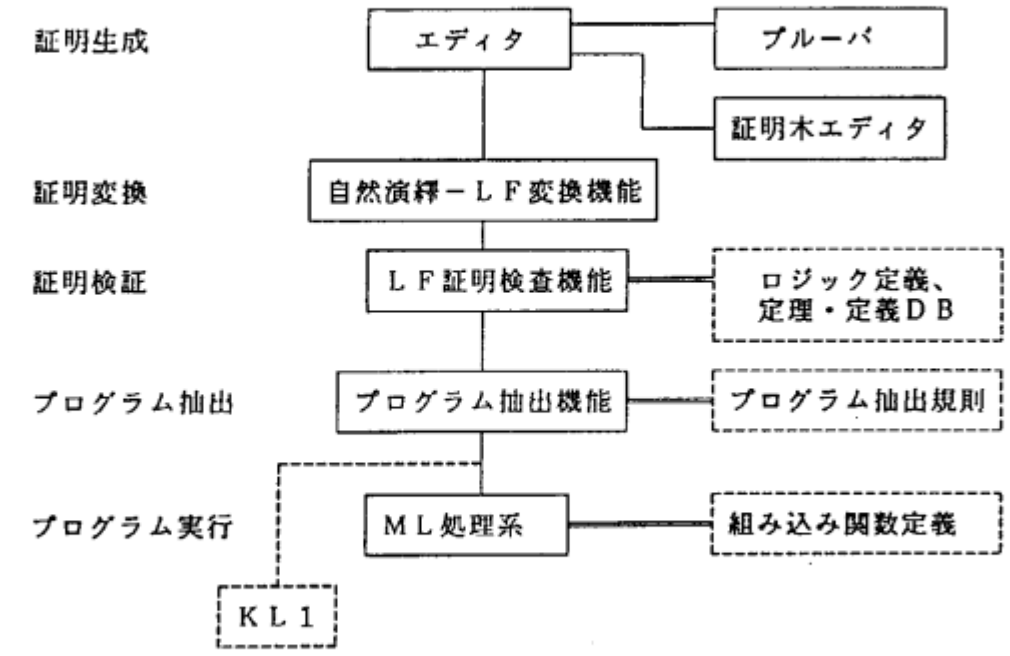
★まとめ

- 制約ロジック・プログラミングを用いると、問題を簡単に、かつ素直に記述することができる。

- 記述された問題の解法は、システムが見つける。

- プログラマは、問題の解法を書かなくとも済む。

- 制約ロジック・プログラミングを用いることによって、プログラマの生産性の向上が見こまれる。

題　名	# ＣＡＰ　ソートプログラムの抽出
目　的	高品質プログラムの生成と構成的プログラミング技術の開発、および自動証明技術を応用した自動プログラミング技術の研究開発を目的とする。
概要及び特徴	(1) 理論の積み重ねを利用したプログラムの部品化・抽象化を可能とする自然演繹による証明記述 (2) 型検査による誤り位置、内容の明確化 (3) メタプログラミングを可能とする高階理論の利用 (4) 推論規則・公理に対応したプログラム抽出規則、組み込み関数の設定
構　成	システム構成 

デ　　モ　　内　　容　（1／3）

本デモンストレーションでは，ソートアルゴリズムを例として，構成的論理に基づいた証明からのプログラム抽出実験を行う。以下で抽出原理，証明とソートアルゴリズムの関係，デモ手順の説明を行う。

プログラム抽出

証明とプログラムは，下表のように対応させることができる．この対応に従ってプログラムを抽出する．

プログラム	証明	
・繰り返し（recursion）	Induction	数学的帰納法などの帰納的な証明は繰り返しに対応
・分岐（if-then-else）	∨-elimination	場合分けの証明は，if then else などの分岐に対応
・関数（function）	∀,∃ -introduction	すべてのxに対して，yが存在することの証明は，y=f(x)に対応
・関数適用	∀ -elimination	すべてのxに対して成り立つ定理の適用は関数適用に対応

（例）

・繰り返し（recursion）

$sort([\,],[\,]).$
$sort([X|Y],Z) :- sort(Y), insert(X,Y,Z).$
$\Rightarrow$

$$\dfrac{\begin{array}{cc}\exists y : L.sorted(nil,y) & \exists y : L.[sorted(tail(x),y)] \\ \vdots & \vdots \\ \exists y : L.sorted(nil,y) & \exists y : L.sorted(x,y)\end{array}}{\forall x : L, \exists y : L.sorted(x,y)}\ \text{(String-Induction)}$$

・分岐

$sort(L,S) :- (L = [\,], !, S = [\,]; L = [X|L1], sort(L1,LS), insert(X,LS,S)).$
$\Rightarrow$

$$\dfrac{\begin{array}{ccc} & [x = nil] & [x \neq nil] \\ & \vdots & \vdots \\ x = nil \vee x \neq nil & \dfrac{sorted(x,nil)\quad nil : L}{\exists y : L.sorted(x,y)} & \dfrac{sorted(x,insert(head(x),t))}{\exists y : L.sorted(x,y)}\end{array}}{\exists y : L.sorted(x,y)}\ \text{(∨ -elimination)}$$

・関数

$addone(X, succ(X))$
$\Rightarrow$

$$\dfrac{\dfrac{\begin{array}{c}[x : Nat] \\ \vdots \\ succ(x) = succ(x)\end{array}}{\exists y : Nat.y = succ(x)}\ \text{(∃ -Intro)}}{\forall x : Nat.\exists y : Nat.y = succ(x)}\ \text{(∀ -Intro)}$$

・関数適用

$F(X) :- addone(zero, X).$
$\Rightarrow$

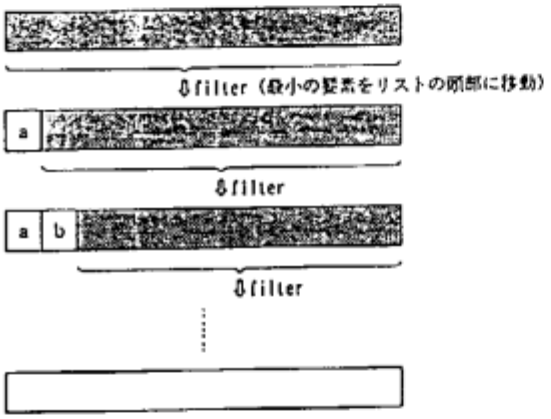
$$\dfrac{\forall x : Nat, \exists y : Nat.y = succ(x) \quad zero : Nat}{\exists y : Nat.y = succ(zero)}\ \text{(∀ -elimination)}$$

ソートアルゴリズムと証明

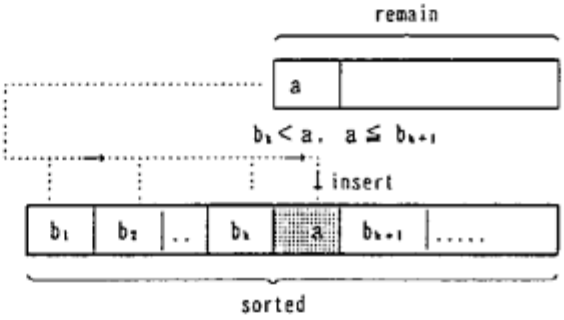
『任意のリストに対して，必ず要素を昇順（または降順）に並び変えたリストが存在する．』という命題の証明から，ソートのプログラムが抽出される．同じ命題でも以下に示すように証明の方針によって異なったソートアルゴリズムが得られる．

- リストのソート
  - 要素＋残りのリストに分割（数学的帰納法）
    - ①極小元とその他に分割 ..... Bubble Sort
    - ②先頭とその他に分割 .. Insert Sort
  - 複数リストに分割（累積帰納法）
    - ③appendによる結合 .... Quick Sort
    - ④マージによる結合 .... Merge Sort

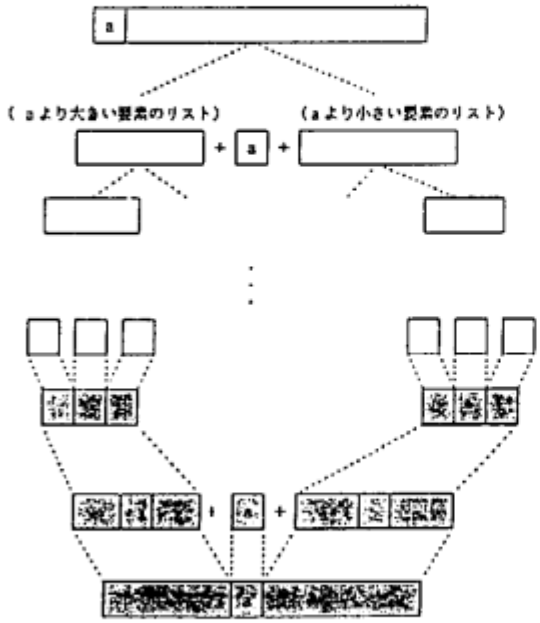
① Bubble Sort



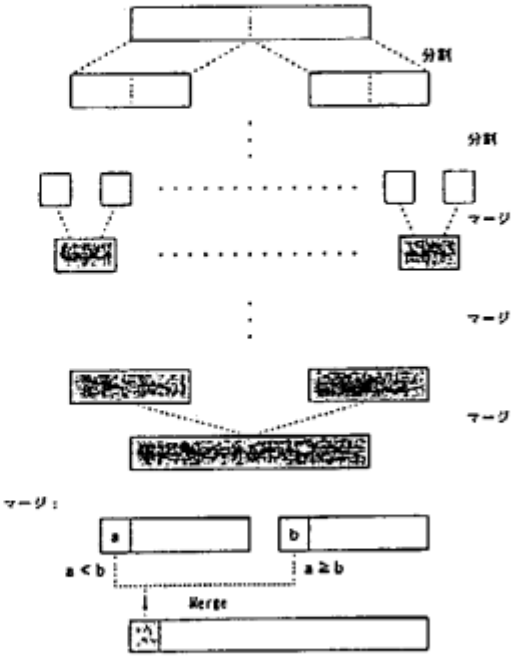
② Insert Sort



③ Quick Sort



④ Merge Sort



デモ手順

① ベースとなる推論規則や公理を Logical Framework によって記述する．
② 証明木表示ウインドウやエディタウインドウを用いて，証明を表示し，生成・編集を行なう．プルーバーを使用した自動生成も行なう．
③ 証明を型付きλ式に変換し，ＬＦの型チェック機能を応用して，証明のチェックを行なう．
④ 証明を表わす型付きλ式から，仕様（定理）を満足し，バグのないプログラムを抽出する．
⑤ 抽出されたプログラム（λ式）を適当な言語に変換して実行する．

① 推論規則・公理の記述

（ＮＫ証明木）

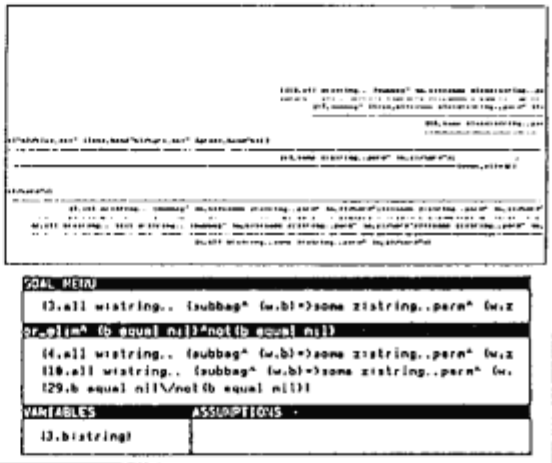
[a:t]
:
p(a)
———————all_intro
∀x:t. p(x)

（ＬＦによる定義）

all_intro :
∀t:type. ∀p:t→prop.
(Π x:t. true(p(x)))
→true(∀x:t.p(x))

```
PHACS_52
Define all_intro :
    (all t:type .. all p:t->prop ..
        ((pi x:t .. true(p^x))
            -> true(all x:t .. p^x))).
PHACS(esp)(51,5) #52/1%    --Top--
```

② 証明の生成



```
GOAL MENU
(3.all w:string.. (subbag^ (w,b)->some z:string..perm^ (w,z
or_elim^ (b equal nil)^not(b equal nil)
(4.all w:string.. (subbag^ (w,b)->some z:string..perm^ (w,z
(10.all w:string.. (subbag^ (w,b)->some z:string..perm^ (w,
(29.b equal nil\/not(b equal nil))
VARIABLES          ASSUMPTIONS
(3.b:string)
```

③ 証明の検査

● Judgement as type 原理による証明検査

証明 ———— 型付きλ式
証明検査＝型チェック
命題 ———— 型

```
LF WINDOW
Line 1   ..... ### Checking Success ###

Line 2
Matching error occured :
string does not match string*string

Type check failed
```

④ プログラムの抽出

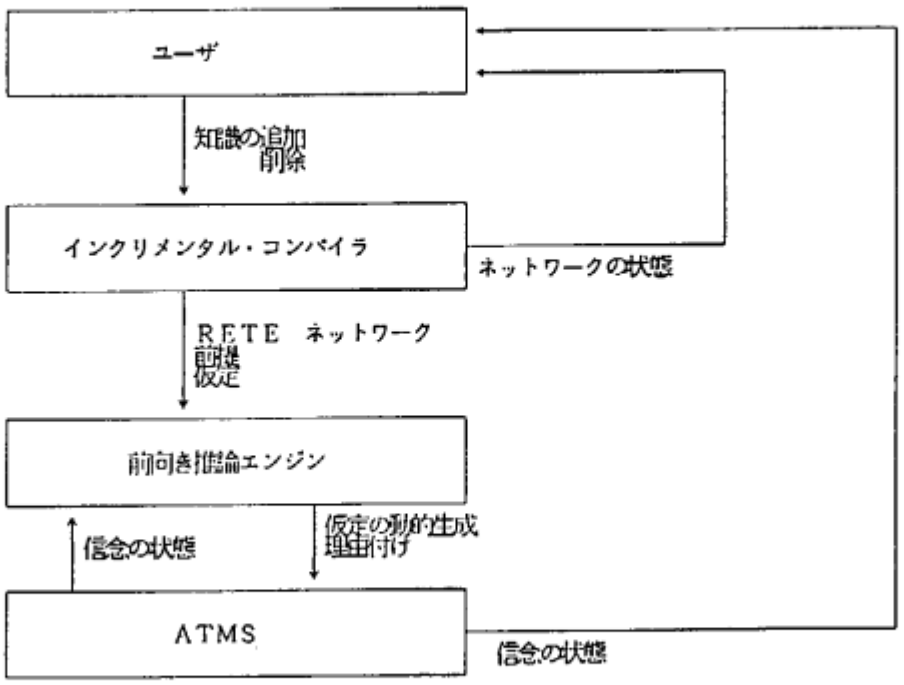
● Curry Haward Isomorphism

プログラム ———— 証明（型付きλ式）
仕様 ———— 命題

```
Command Window
Extracting Program .....
(val (quick_sort= (l:a) val rec y=fun A..(if (A=nil) then (nil.empty.l
empty) else (append^ ((first^ (z^ ((first^ (odiv^ (head^A)^ (tail^A)))
)).cons^ (head^A)^ ((first^ (z^ ((first^ (second^ (odiv^ (head^A)^ (t
ail^A)))))).empty.empty)inzend)))
[]
```

⑤ プログラムの実行

```
Command Window

ML>>quick_sort^[5,4,3,2,1].
Value  quick_sort^[5,4,3,2,1] ::* list(int)*f(1)*f(2)
[1,2,3,4,5].empty.empty

ML>>
```

題　名	仮説推論実験システム：　APRICOT/0
目　的	仮説に基づく効率的な問題解決機構を実現し それを設計問題へ応用し評価する。
概要及び特徴	[概要] （1）節とデフォルト・ルールをRETEネットワークに変換するコンパイラ、RETEベース推論エンジンおよびATMSとによる仮説推論システム （2）論理回路合成問題への応用および評価 [特徴] 推論エンジンが中間的な理由付けをATMSに送り中間的なデータのラベルをRETEネットワークの2入力ノードに保存することによる効率的な仮説推論機構の実現
構　成	

デ　　モ　　内　　容　（1／3）

１．仮説推論システム

仮説推論は、常に正しいかどうかわからない知識（不完全な知識）を矛盾が導かれない限り正しいと仮定して推論を行う推論形態であり、不完全な知識ベースに基づく問題解決機構を実現するために用いることができる。ここでは、論理回路設計問題を例として、ＡＰＲＩＣＯＴ／０が効率良く仮説推論を実行することを示す。

１－１　知識ベースの例

対象とする仮説推論システムの入力は、ホーン節の集合（完全な知識）と正規デフォルト・ルールの集合（不完全な知識）であるものとする。

図１に知識ベースの例を示す。ルールｒは、”回路に減算器（記号：ｓ）が必要（記号：ｎ）で、Ｘが加算器（記号：ａ）で、Ｙが１の補数回路（記号：ｃ）ならば、［Ｘ，Ｙ］は減算器である”ことを表現しているものとする。ここに、ａｓｓｕｍｅは、”制約に違反しない”という仮定に基づくことを意味する（ルールｒは正規デフォルト・ルールである）。また、減算器が必要、ｘは加算器、ｙ１は１の補数回路、ｙ２は１の補数回路という仮説が知識ベースに含まれているものとしている。

```
r : : n (s), a (X), c (Y) -> assume (s ([X, Y])).
assume (n (s)).        assume (a (x)).
assume (c (y1)).       assume (c (y2)).
```

図１．知識ベースの例

１－２　仮説推論システムの従来例

不完全な知識ベースに基づき推論を実行する仮説推論システムを、前向き推論エンジンと仮定に基づく真偽値管理機構（Ａｓｓｕｍｐｔｉｏｎ－ｂａｓｅｄ　Ｔｒｕｔｈ　Ｍａｉｎｔｅｎａｎｃｅ　Ｓｙｓｔｅｍ：　ＡＴＭＳ）とにより構成することができる。

前向き推論エンジンはボトム・アップ探索により、新しいデータを導出すると共に理由付け（記号⇒で示す含意）をＡＴＭＳに送る。ＡＴＭＳは、データの真偽値（現時点で信じられているか、あるいは信じられていないという意味で信念の状態と呼ばれる）を管理しており、推論エンジンに、あるデータの信念の状態を返す。ＡＴＭＳが管理するデータは、与えられた理由付けの集合の基で、そのデータを成り立たせる仮定（記号Γで表す）の集合の極小集合によってラベル付けされる。このデータとラベルの組をＡＴＭＳノード（γ：<データ，ラベル>）と呼ぶ。

図２は、前向き推論エンジンとＡＴＭＳとを単純に結合した従来の仮説推論システムの構成と動作を示すものである。前向き推論エンジンは対象問題に関する知識を与えることにより、その対象問題の問題解決器となる。

まず、ＡＴＭＳから問題解決器へｎ（ｓ），ａ（ｘ），ｃ（ｙ１），ｃ（ｙ２）のノード $\gamma n(s)$, $\gamma a(x)$, $\gamma c(y1)$, $\gamma c(y2)$ が送られているとする。｛｛Γ_1｝｝はｎ（ｓ）のラベル、｛｛Γ_2｝｝はａ（ｘ）のラベル、｛｛Γ_3｝｝はｃ（ｙ１）のラベル、｛｛Γ_4｝｝はｃ（ｙ２）のラベルである。ルールｒが実行されるとｓ（［ｘ，ｙ１］），ｓ（［ｘ，ｙ２］）に対する理由付けＪ１，Ｊ２がＡＴＭＳに送られ、｛Γ_1｝∪｛Γ_2｝∪｛Γ_3｝∪｛Γ_5｝と　｛Γ_1｝∪｛Γ_2｝∪｛Γ_4｝∪｛Γ_6｝が計算される。

しかしながら、問題解決器を一階述語論理ベースの前向き推論エンジンとすれば、このような構成・動作に基づくシステムは、ラベル計算量が多く、必ずしも効率的ではない。

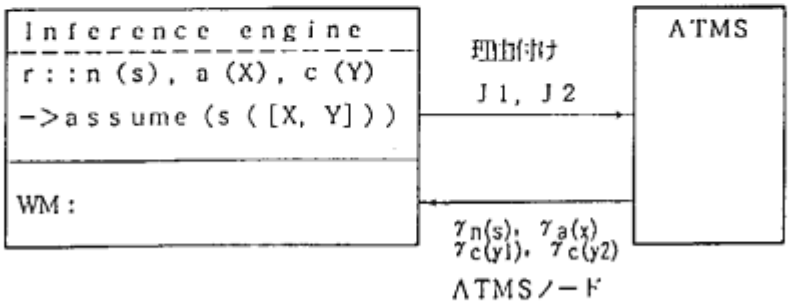


図2．従来の仮説推論システムの構成と動作（その1）

J1：　<n (s)，｛{Γ1}｝>，　<a (x)，｛{Γ2}｝>，　<c (y1)，｛{Γ3}｝>，
　　<Γ5，｛{Γ5}｝> ⇒ <s ([x, y1])，｛{Γ1，Γ2，Γ3，Γ5}｝>.

J2：　<n (s)，｛{Γ1}｝>，　<a (x)，｛{Γ2}｝>，　<c (y2)，｛{Γ4}｝>，
<Γ6，｛{Γ6}｝> ⇒ <s ([x, y2])，｛{Γ1，Γ2，Γ4，，Γ6}｝>.

図2．従来の仮説推論システムの構成と動作（その2）

1 - 3　APRICOT／0

図3は、APRICOT／0の構成と動作を示すものである。その特徴は、知識ベースをRETEネットワークにコンパイルしておき、その2入力ノードにおいて推論エンジンが中間的な理由付け（J3，J4，J5）をATMSに送り、その結果得られたATMSノードをそのワーキング・メモリ（WM）に保存しておくことにより、ATMSのラベル総計算量を削減できることである。

例においては、集合操作∪が、従来例では6回（　{Γ1} ∪ {Γ2} ∪ {Γ3} ∪ {Γ5}，{Γ1} ∪ {Γ2} ∪ {Γ4} ∪ {Γ6}）、APRICOT／0では5回（{Γ1} ∪ {Γ2}，{Γ1，Γ2} ∪ {Γ3}，{Γ1，Γ2} ∪ {Γ4}，{Γ1，Γ2，Γ3} ∪ {Γ5}，{Γ1，Γ2，Γ4} ∪ {Γ6}）となり、ラベル総計算量がAPRICOT／0の方が少なく効率的である。

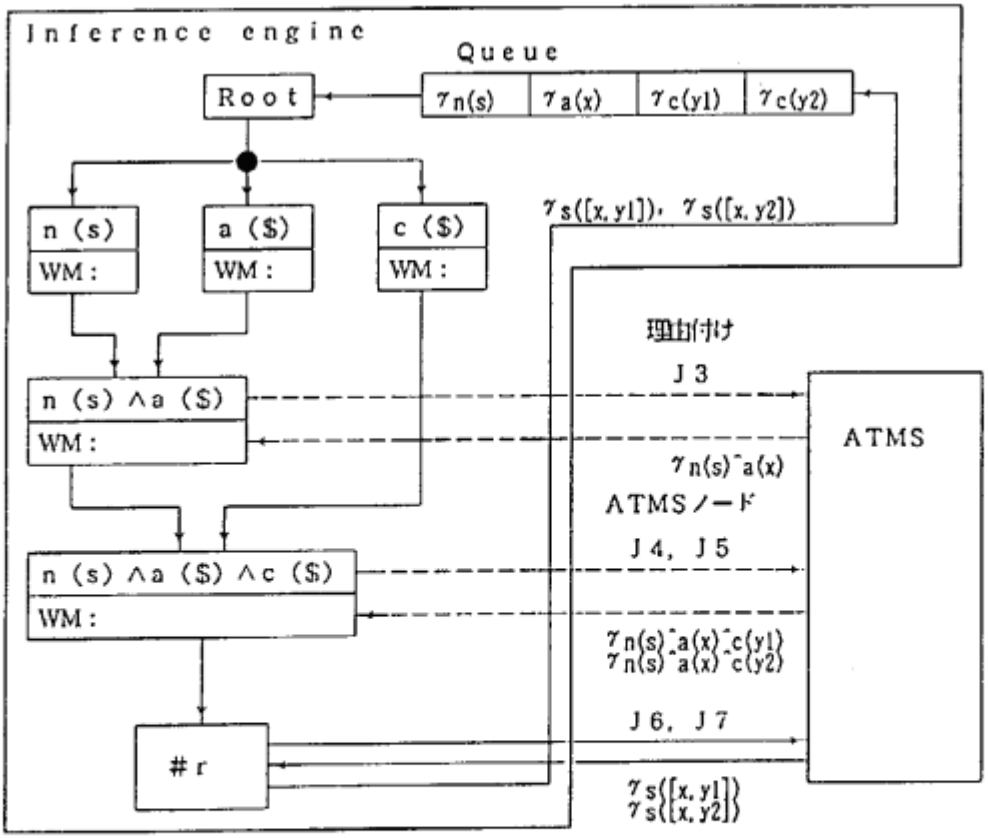


図3．APRICOT／0の構成と動作（その1）

デ　　モ　　内　　容　（3／3）

J3： $<n(s), \{\{\Gamma_1\}\}>, \quad <a(x), \{\{\Gamma_2\}\}> \Rightarrow$
$<n(s)\hat{}a(x), \{\{\Gamma_1, \Gamma_2\}\}>.$

J4： $<n(s)\hat{}a(x), \{\{\Gamma_1, \Gamma_2\}\}>, \quad <c(y1), \{\{\Gamma_3\}\}> \Rightarrow$
$<n(s)\hat{}a(x)\hat{}c(y1), \{\{\Gamma_1, \Gamma_2, \Gamma_3\}\}>.$

J5： $<n(s)\hat{}a(x), \{\{\Gamma_1, \Gamma_2\}\}>, \quad <c(y2), \{\{\Gamma_4\}\}> \Rightarrow$
$<n(s)\hat{}a(x)\hat{}c(y2), \{\{\Gamma_1, \Gamma_2, \Gamma_4\}\}>.$

J6： $<n(s)\hat{}a(x)\hat{}c(y1), \{\{\Gamma_1, \Gamma_2, \Gamma_3\}\}>, \quad <\Gamma_5, \{\{\Gamma_5\}\}> \Rightarrow$
$<s([x, y1]), \{\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_5\}\}>.$

J7： $<n(s)\hat{}a(x)\hat{}c(y2), \{\{\Gamma_1, \Gamma_2, \Gamma_4\}\}>, \quad <\Gamma_6, \{\{\Gamma_6\}\}> \Rightarrow$
$<s([x, y2]), \{\{\Gamma_1, \Gamma_2, \Gamma_4, \Gamma_6\}\}>.$

図3．APRICOT／0の構成と動作（その2）

２．　例題題

論理回路等の設計問題は、設計過程の各階層で極めて多くの選択枝が存在する。これらの選択枝を段階的に生成される仮説とみなすと仮説推論の枠組により、問題解決を行うことができる。

２・１　　問題題

下記の機能と制約を満たす回路を設計せよ。

・機能：２つの整数（８ビット）の最大公約数を計算する。
・制約：チップ面積（基本セル数がある値以下）と性能（遅延時間がある値以下）

２・２　　入力

仮説推論システムへの入力は、データパス設計、コンポーネント設計、テクノロジ・マッピング（ＣＭＯＳ）および制約知識を表現したホーン節やデフォルト・ルールである。本実験システムの知識ベースは、ホーン節とデフォルト・ルールとをあわせて約１００なる規模である。

２・３　　出力

基本セル数が４００以下、遅延時間が６０［ｎｓ］以下なる制約を与えた場合の全解を図４に示す。基本セル数と遅延時間の平面上にマッピングされている点は、ＣＭＯＳ標準セルにより具体化された一つの回路であり、仮説推論システムが出力する結論の一つを表現している。

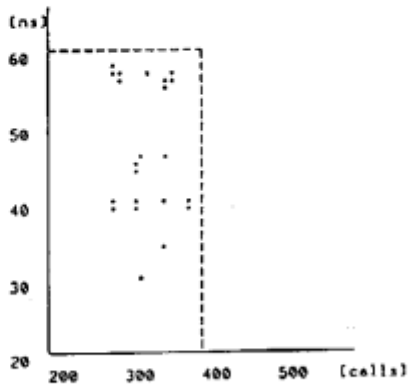


図4．基本セル数と遅延時間の平面上に解をマッピングした例

談話理解実験システム DUALS

平成元年９月５日

財団法人 新世代コンピュータ技術開発機構 (ICOT)

談話理解とは

人間が話すことばをコンピュータによって処理することを自然言語処理といい,現在,これに関してはさまざまなシステムが華々しく発表されている. この自然言語処理を高度化し,もっと精度の高い,知的に応答するシステムの構築を目指すためには,自然言語で表現された文や談話の内容をコンピュータに理解させる必要がある.

ICOT では,このような談話理解のための基礎的な研究を行っている. 談話理解というのは,単に一つ一つの文の意味を理解するにとどまらず,段落全体さらには文章全体の意味を理解したり,人間との対話において,話題の焦点や話者の意図を理解したりすることであり,これは高度な自然言語処理を実現するために乗り越えなくてはならない重要なテーマである.

ここでデモを行う DUALS は,談話理解を実現するための個々の要素技術を実証し,その上で談話にまつわるさまざまな研究テーマに関する実験を行うためのシステムである. 今回は,小学校６年生の国語の教科書から選んだテキストを用いて,システムが文章を読んで理解し,その内容に関する質問に答えるという形でデモを進める.

要素技術

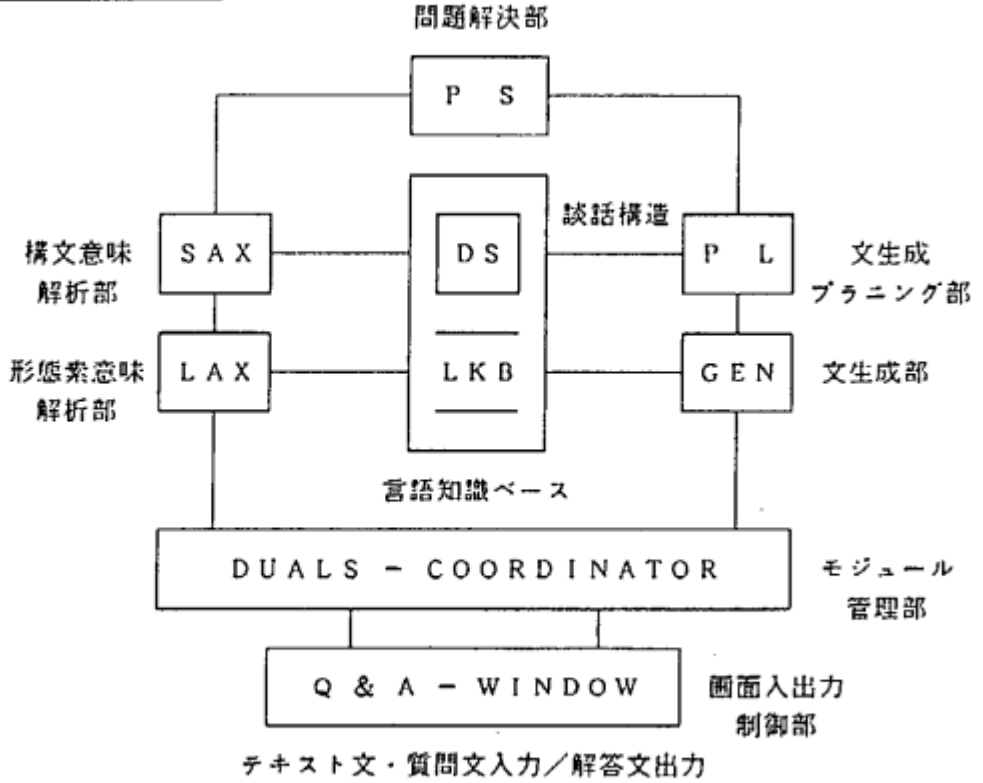
- 形態素解析: 文中の語の接続関係を解析して文節の構造を調べる
- 構文解析: 文節相互間の係り受け関係を解析して文の構造を調べる
- 文生成: コンピュータ内部の意味表現から,それを表現する日本語の文を作る

談話に関するテーマ

- 問題解決: テキストに関する談話構造を作り上げ,それを参照して質問に答える
- 文生成プランニング: 質問者の意図に沿って答えるためにプランを立てる

知識の扱い

- 言語知識ベース: 語彙に関わる言語的な知識を辞書の形で蓄積している
- 談話構造: 文章を読みながら,各文の内容を関係付け,構造化して蓄積する

題　名	談話理解実験システム　ＤＵＡＬＳ
目　的	高度な自然言語処理に必要な要素技術の確立と 知識表現形式や談話構造、辞書の意味記述などの定式化を行い 実験システムの開発を通じて実証的に研究する。
概要及び特徴	内　容：小学校６年生の国語のテキストを理解し、その内容に関する質問に対して答えを推論し、解答する。 処理系：システムは汎用日本語処理系ＬＴＢの上に構築。 モジュール機能： 形態素意味解析部ＬＡＸ：入力文の形態素を認識し文節に区切る。文節の構文情報と意味情報を形態素の意味記述から構成。 構文意味解析部ＳＡＸ：係り受け文法により、文節間の依存関係を解析する。文中の物と物との間の関係を意味構造として談話構造ＤＳに格納する。 問題解決部ＰＳ：談話構造ＤＳに蓄えられたテキスト情報と質問文の意味構造から、質問の答えを推論する。 文生成プラニングＰＬ：問題解決部から得られた答えと質問文の意味構造から、解答文の構文構造を構成する。語の省略等も行う。 文生成部ＧＥＮ：質問文の構文構造を基に、実際の解答文を生成する。活用語尾の語尾変化など形態的な処理を行う。 談話構造ＤＳ：テキストと質問文の意味構造が蓄積される。 言語知識ベースＬＫＢ：動詞の格情報などの共通的言語知識のデータベース。
構　成	

デ　　モ　　内　　容　（1／3）

□ デモ手順

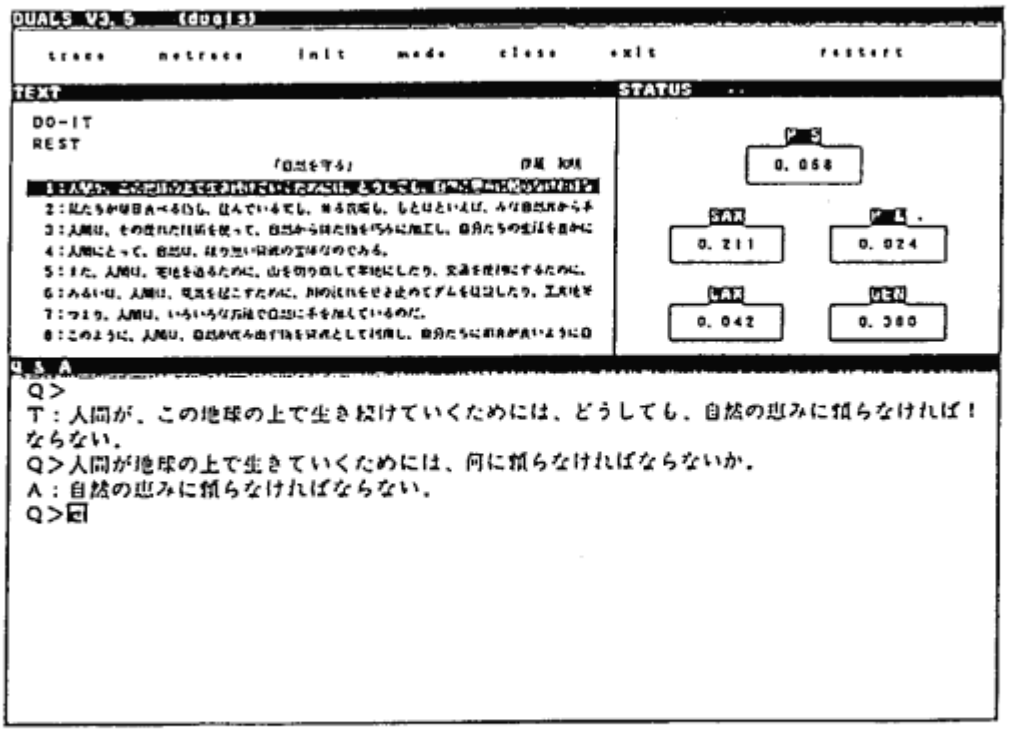
(1) ＬＡＸ＆ＳＡＸをトレース・オン
(2) テキストを入力 ... ＬＡＸ＆ＳＡＸの処理過程を説明
(3) ＰＳ＆ＰＬ＆ＧＥＮをトレース・オン
(4) 質問文を入力 ... ＰＳ＆ＰＬ＆ＧＥＮの処理過程を説明
(5) 解答文出力
(6) テキスト文入力
(7) 質問文入力＆解答文出力（以下(6)-(7)の繰り返し）

※ 各モジュールのデモは、それぞれ独自のウインドウにより行う。

□ デモ・ウインドウ説明

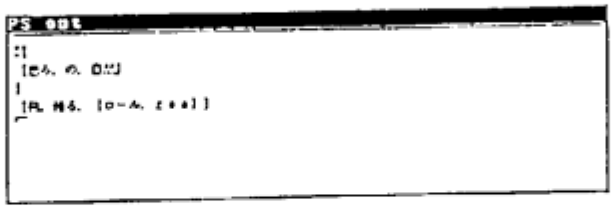
＜モジュール名＞
ＬＡＸ： 形態素意味解析
ＳＡＸ： 構文意味解析
Ｐ　Ｓ： 問題解決
Ｐ　Ｌ： 文生成プラニング
ＧＥＮ： 文生成

＜ウインドウ説明＞
上 ： 各種制御用メニュー，
中左： テキスト選択メニュー
中右： 各モジュール状態表示
下 ： テキストのエコーバック，
質問文入力，解答文出力

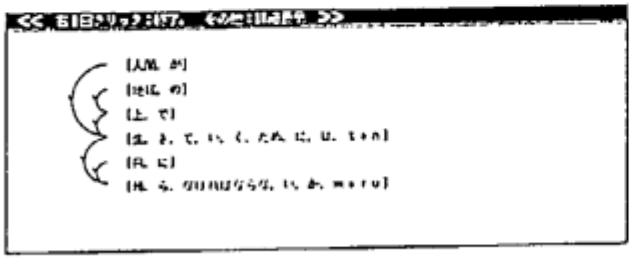


デ　モ　内　容　(2／3)

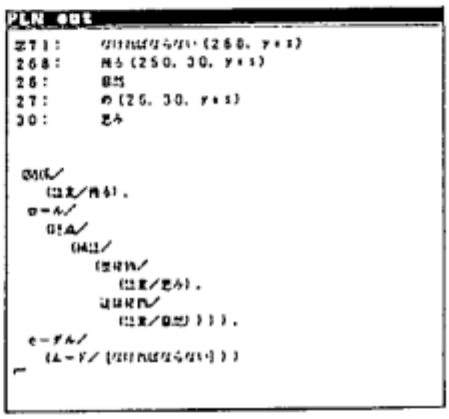
□ トレース画面



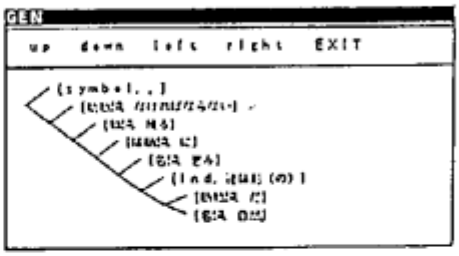
↑ 問題解決部



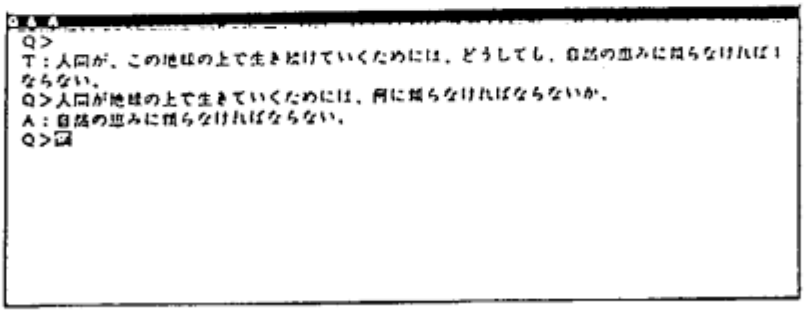
← 構文意味解析部



↑ 形態素意味解析部



↑ 文生成部



← 質問応答

デ　モ　内　容（3／3）

□ デモ・テキスト（全文）

「自然を守る」　　伊藤 和明

人間が、この地球の上で生き続けていくためには、どうしても、自然の恵みに頼らなければならない。私たちが毎日食べる物も、住んでいる家も、着る衣服も、もとはといえば、みな自然界から手に入れた物である。人間は、その優れた技術を使って、自然から得た物を巧みに加工し、自分たちの生活を豊かにしている。人間にとって、自然は、限り無い資源の宝庫なのである。

また、人間は、宅地を造るために、山を切り崩して平地にしたり、交通を便利にするために、森を切り開いて道路を造ったりしている。あるいは、人間は、電気を起こすために、川の流れをせき止めてダムを建設したり、工業地帯にするために、海を埋め立てて陸地に変えたりしている。つまり、人間は、いろいろな方法で自然に手を加えているのである。このように、人間は、自然が産み出す物を資源として利用し、自分たちに都合が良いように自然の姿を変えて生活している。人間だけが、自然の資源を思う存分利用したり、自然を改造したりする知恵と力とを備えた生物なのである。しかし、それだからといって、人間が思いのままに自然の姿を変え、その資源を手当たり次第に自分たちの物にしてしまって良いのだろうか。

（光村図書：小学６年生国語教科書「創造」より）

□ 標準質問文（一部）

－ 第一文 －

人間が地球の上で生きていくためには、何に頼らなければならないか。
人間は、何のために自然の恵みに頼らなければならないか。
誰が自然の恵みに頼らなければならないか。

－ 第二文 －

人間は何を着るか。
私たちが住んでいる家は何であるか。
何が、私たちが自然界から手に入れた物であるか。

－ 第三文 －

人間は何を使って自然から得た物を加工するのか。
人間はどのようにして自分たちの生活を豊かにしているのか。
人間は、その優れた技術を使って、何を加工するか。

－ 第四文 －

人間にとって自然は、どのような資源の宝庫であるか。
人間にとって自然は、何であるか。
人間にとって何が、資源の宝庫であるか。

－ 第五文 －

人間は、なぜ、山を切り崩して平地にするか。
人間は、なぜ、森を切り開いて道路を造るのか。
人間は、交通を便利にするために、何を切り開いて道路を造るのか。
人間は、森を切り開いて何を造るのか。

自然を守る
The Preservation of Nature

伊藤 和明
Kazuaki Ito

1. 人間が、この地球の上で生き続けていくためには、どうしても、自然の恵みに頼らなければならない。
   In order to continue to live on the earth, human being necessarily must depend on the blessing of nature.

   Q: 人間が地球の上で生き続けていくためには、何に頼らなければならないか。
   What must human being depend on to continue to live on the earth?
   A: 自然の恵みに頼らなければならない。
   (He) must depend on the blessing of nature.

   Q: 人間は、何のために自然の恵みに頼らなければならないか。
   Why must human being depend on the blessing of nature?
   A: この地球の上で生き続けていくために頼らなければならない。
   In order to continue to live on the earth, (human being) must depend on (the blessing of nature).

   Q: 私たちは何の恵みに頼らなければならないのか。
   What kind of blessing must we depend on?
   A: 自然の恵み。
   The blessing of nature.

   Q: 人間はどこで生きていくために、自然の恵みに頼るのか。
   Where does human being continue to live depending on the blessing of nature?
   A: この地球の上で生きていくために頼る。
   (He) depends on (the blessing of nature) to live on the earth.

   Q: 誰が自然の恵みに頼らなければならないか。
   Who must depend on the blessing of nature?
   A: 人間が頼らなければならない。
   Human being must depend on (the blessing of nature).

2. 私たちが毎日食べる物も、住んでいる家も、着る衣服も、もとはといえば、みな自然界から手に入れた物である。
The things we eat every day,the houses we live in, or the clothes we wear are all originally what we obtain from nature.

Q: 私たちは何を食べるか。
What do we eat?

A: 物を食べる。
(We) eat things.

Q: 人間は何を着るか。
What does human being wear?

A: 衣服を着る。
(He) wears clothes.

Q: 人間はどこに住んでいるか。
Where does human being live?

A: 家に住んでいる。
(He) lives in house.

Q: 私たちが住んでいる家は何であるか。
What is the houses we live in?

A: 自然界から手に入れる物だ。
(Those are) what we get from nature.

Q: 私たちが食べる物は、どこから手に入れた物であるか。
Where do we get the things we eat from?

A: 自然界から手に入れる物。
(We get) the things from nature.

Q: 何が、私たちが自然界から手に入れた物であるか。
What we get from nature?

A: 毎日食べる物と住む家と着る衣服がだ。
(We get) the things we eat every day, the houses we live in, and the clothes we wear.

3. 人間は、宅地を造るために、山を切り崩して平地にしたり、交通を便利にするために、森を切り開いて道路を造ったりしている。
To make residential areas, human being levels mountains and smooths the terrain, and for convenient transportation, cuts down forests and constructs roads.

Q: 人間は、なぜ、山を切り崩して平地にするか。
Why does human being level mountains and smooth the terrain?

A: 宅地を造るために切り崩して平地にする。
(He) levels (mountains) and smooths terrain in order to make residential areas.

Q: 人間は、宅地を造るために、何を切り崩すか。
What does human being level in order to make residential areas?

A: 山を切り崩す。
(He) levels mountains.

Q: 人間は山を切り崩して何にするか。
What does human being make by means of leveling mountains?

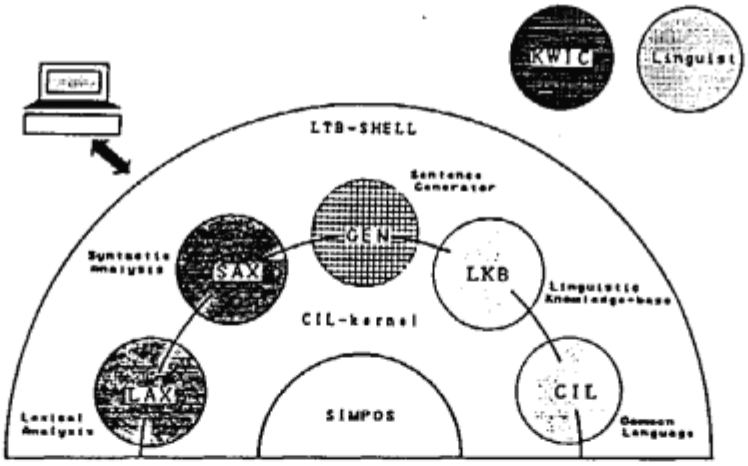
A: 平地にする。
(He) makes flat grounds.

Q: 人間は、なぜ、森を切り開いて道路を造るのか。
Why does human being cuts down forests and constructs roads?

A: 交通を便利にするために切り開いて造る。
(He) cuts down (forests) and constructs (roads) because of convenient transportation.

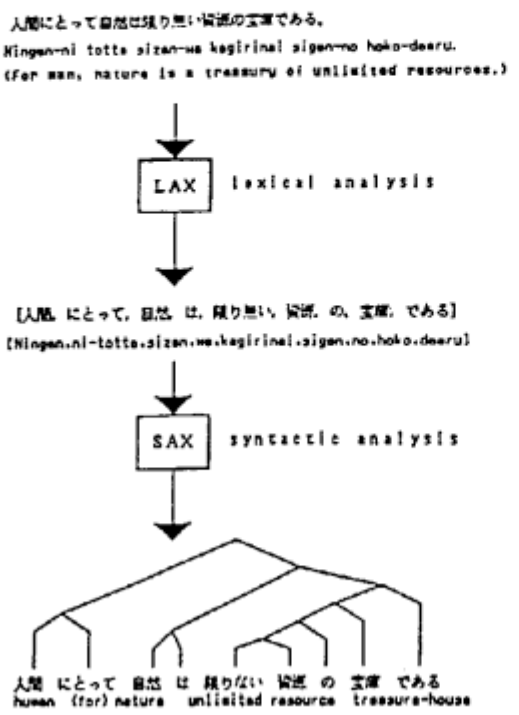
Q: 人間は、森を切り開いて何を造るのか。
What does human being construct by means of cutting down forests?

A: 道路を造る。
(He) makes roads.

Title	Language Tool Box (LTB): A Software Library for Japanese Language Processing
Features	LTB provides a general and flexible development environment for the Japanese based natural language processing system. LTB is developed by integrating such component techniques for NLP that were established through the development of Discourse Processing System (DPS) in ICOT. Currently, LTB consists of lexical analyzer (LAX), syntactic analyzer (SAX), Japanese sentence generator(GEN), knowledge representation language (CIL), and so forth. All the modules in LTB are implemented in terms of ESP language and run on the PSI machine.
Outline of Demo	The demonstration consists of two parts. First, analysis and generation process of a Japanese sentence are displayed through the LTB-shell. LTB-shell enables us to activate each sub-module and control the flow of data structures.After that, major debugging tools including tracer and inspector for each sub-modules (LAX, SAX, GEN) are demonstrated briefly.
System Configuration	 • LAX(Lexical Analyzer): constructs a sequence of phrases with their meanings from a Japanese sentence using extended regular grammar. • SAX(Syntactic Analyzer): constructs a parse tree or a semantic structure of a sentence using extended DCG grammar. • CIL(KR-Language): supplies the basic data structure(PST:partially specified term) necessary for semantic processing. • GEN(Japanese sentence generator): generates a Japanese sentence from a sort of syntactic structure. • KWIC(Key Word In Context): Database for linguistic research. • Linguist(Grammar Workbench): DCG grammar prototyping system.

1/6

Process of Lexical and Syntactic Analysis



input sentence (Japanese)

result of lexical analysis

result of syntactic analysis

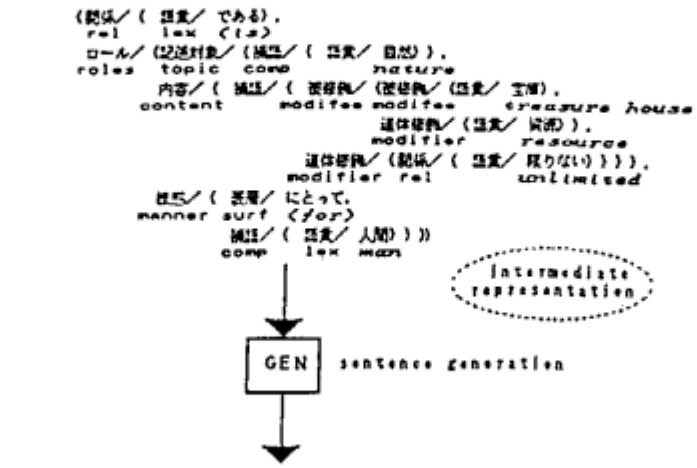
result of morphological analysis

a sequence of phrases with meanings

semantic structures of the input sentence

```
scps:244::>sys>user>ltb>ltbshell
LTB> geal $lax exec -s -p "人間にとって自然は限り無い資源の宝庫である。"
 人間
 にとって
 自然
 は
 限り無い
 資源
 の
 宝庫
 である
 。

LTB> geal $lax exec "人間にとって自然は限り無い資源の宝庫である。"
 (1. [ ( 人間 (人間, 名詞, end, 文節) ], noun (人間), 3, 1) ], A. [ ( (にとっ!
 て (にとって, 格助詞, end, 文節) ], p (にとって), 7, 2) ], B. C. D. [ ( (自然 (自!
 然, 名詞, end, 文節) ], noun (自然), 9, 3) ], E. [ ( (は (は, 格助詞, end, 文!
 節) ], p (は), 10, 4) ], [ ( (限り無い・限り無い, 形容詞, end, 文節) ], adj (限!
 り無い), 14, 5) ], F, G, H. [ ( (資源 (資源, 名詞, end, 文節) ], noun (資源) |
 , 16, 6) ], I, [ ( (の (の, 連体助詞, end, 文節) ], ne (の), 17, 7) ], [ ( (!
 宝庫 (宝庫, 名詞, end, 文節) ], noun (宝庫), 19, 8) ], J. [ ( (である (である, |
 基準助詞, end, 文節) ], aux (である), 22, 9) ], K. L. [ ( (L C, 特記号, e!
 nd, 文章) ], kigou (。), M, 10) ])

LTB> geal $lax exec "人間にとって自然は限り無い資源の宝庫である。" | g!
eal $sax exec -s
 [ ( (be ( (ferC ( $$atom$FA8A' ), ob j1 ( $$atom$FA8B' ), !
 ob j2 ( $$atom$FA8D' ) ) ), 人間 ( $$atom$FA8A' ), 自然 ( $$a!
 tom$FA8B' ), 限り無い ( (ob j1 ( $$atom$FA8D' ) ) ), の ( (ob j1 ( !
 $$atom$FA8C' ), ob j2 ( $$atom$FA8D' ) ) ), 資源 ( $$atom$!
 FA8C' ), 宝庫 ( $$atom$FA8D' ) ) ], [ (be ( (ferC ( $$atom$F!
 A8A' ), ob j1 ( $$atom$FA8D' ), ob j2 ( $$atom$FA8D' ) ) ) !
 , 人間 ( $$atom$FA8A' ), 自然 ( $$atom$FA8B' ), の ( (ob j1 ( $!
 $atom$FA8C' ), ob j2 ( $$atom$FA8D' ) ) ), 限り無い ( (ob j1 ( $!
 $atom$FA8C' ) ) ), 資源 ( $$atom$FA8C' ), 宝庫 ( $$atom$FA8!
 D' ) ] ] ].
LTB> □
```

Process of Japanese Sentence Generation



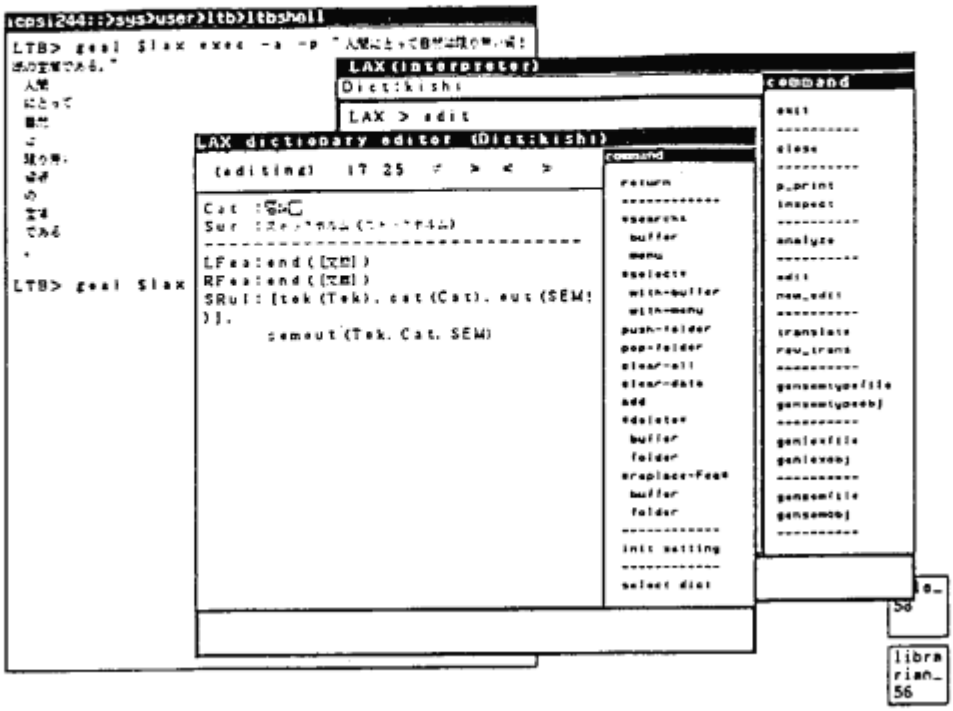
result of syntactic analysis

write the result to "out.sax"

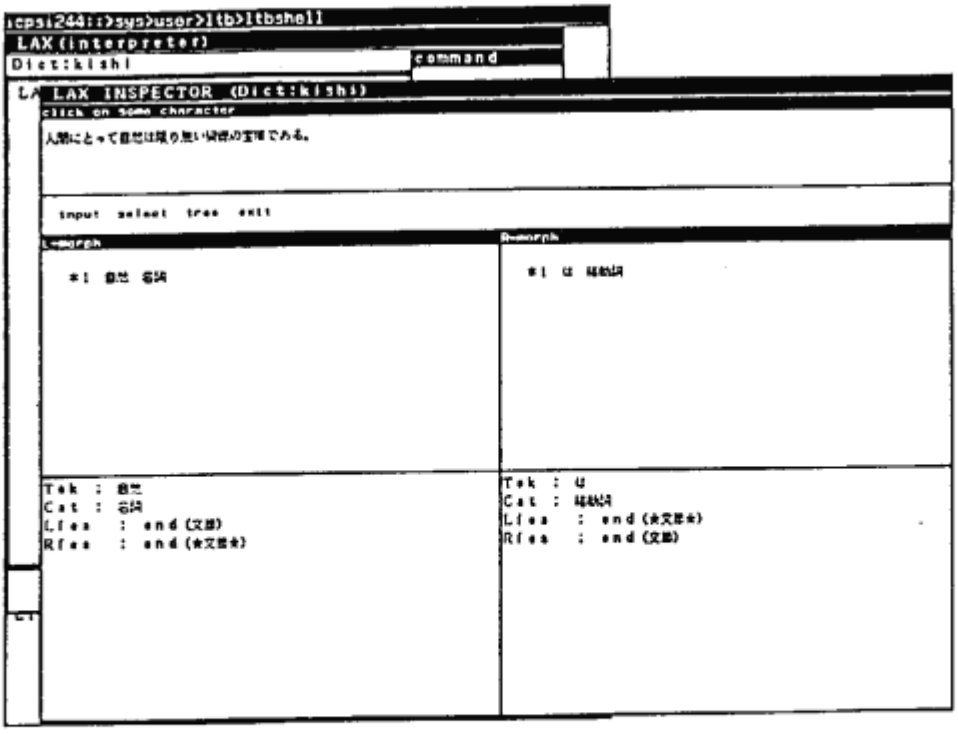
convert it to the input form of GEN

result of sentence generation

```
LTB> geal $lax exec "人間にとって自然は限りない資源の宝庫である。" | g!
eal $sax exec -s
[[[be ([forC ( $$atom$FA8A' ), obj1 ( $$atom$FA8B' ), !
obj2 ( $$atom$FA8D' )]), 人間 ( $$atom$FA8A' ), 自然 ( $$a!
tom$FA8B' ), 限りない ([obj ( $$atom$FA8D' )]), の ([obj1 ( !
$$atom$FA8C' ), obj2 ( $$atom$FA8D' )]), 資源 ( $$atom$!
FA8C' ), 宝庫 ( $$atom$FA8D' )]), [[be ([forC ( $$atom$F!
A8A' ), obj1 ( $$atom$FA8B' ), obj2 ( $$atom$FA8D' )])!
, 人間 ( $$atom$FA8A' ), 自然 ( $$atom$FA8B' ), の ([obj1 ( $!
$atom$FA8C' ), obj2 ( $$atom$FA8D' )]), 限りない ([obj ( $!
$atom$FA8C' )]), 資源 ( $$atom$FA8C' ), 宝庫 ( $$atom$FA8!
D' )]]].

LTB> geal $lax exec "人間にとって自然は限りない資源の宝庫である。" | g!
eal $sax exec -s ) out.sax

LTB> geal $cil exec "utter" ( out.sax
(関係/ (言葉/である), ロール/ (記述対象/ (属性/ (言葉/自然) ), 内容/ (属性/ (被修飾!
/ (被修飾/ (言葉/宝庫), 連体修飾/ (言葉/資源) ), 連体修飾/ (関係/ (言葉/限りない) ) !
)), 様態/ (属性/ (言葉/人間), 表現/にとって) ),
(関係/ (言葉/である), ロール/ (記述対象/ (属性/ (言葉/自然) ), 内容/ (属性/ (被修飾!
/ (言葉/宝庫), 連体修飾/ (被修飾/ (言葉/資源), 連体修飾/ (関係/ (言葉/限りない) ) !
)), 様態/ (属性/ (言葉/人間), 表現/にとって) ).
   yes

LTB> geal $cil exec "utter" ( out.saxl geal $ge!
n exec

"自然は人間にとって限りない資源の宝庫である。"

LTB>
```

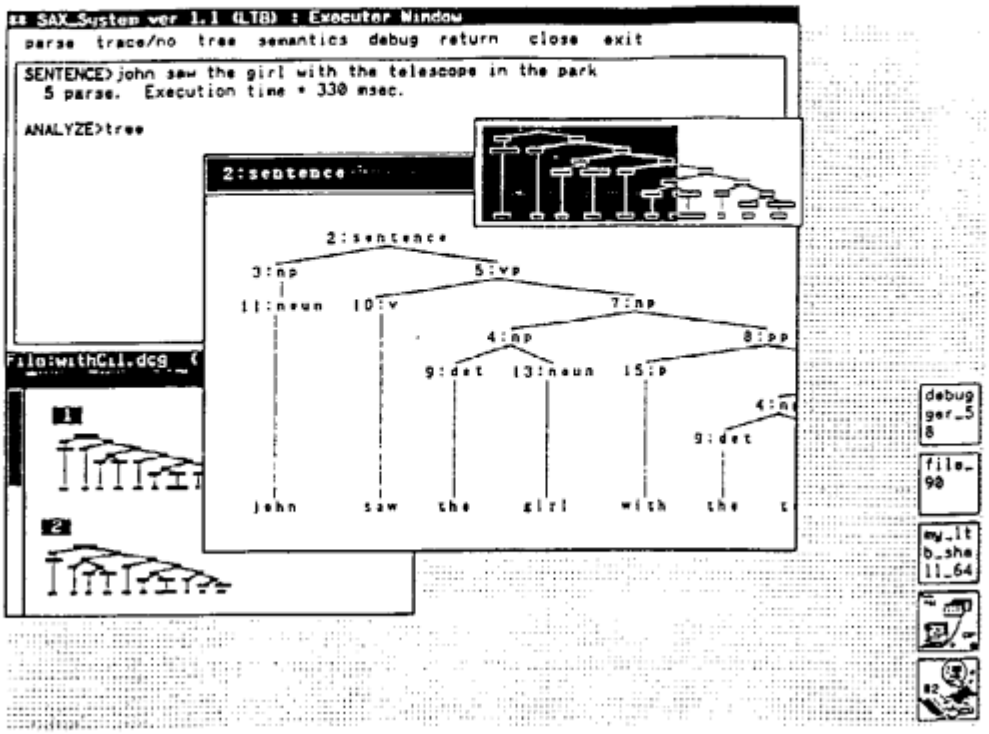


Dictionary Editor

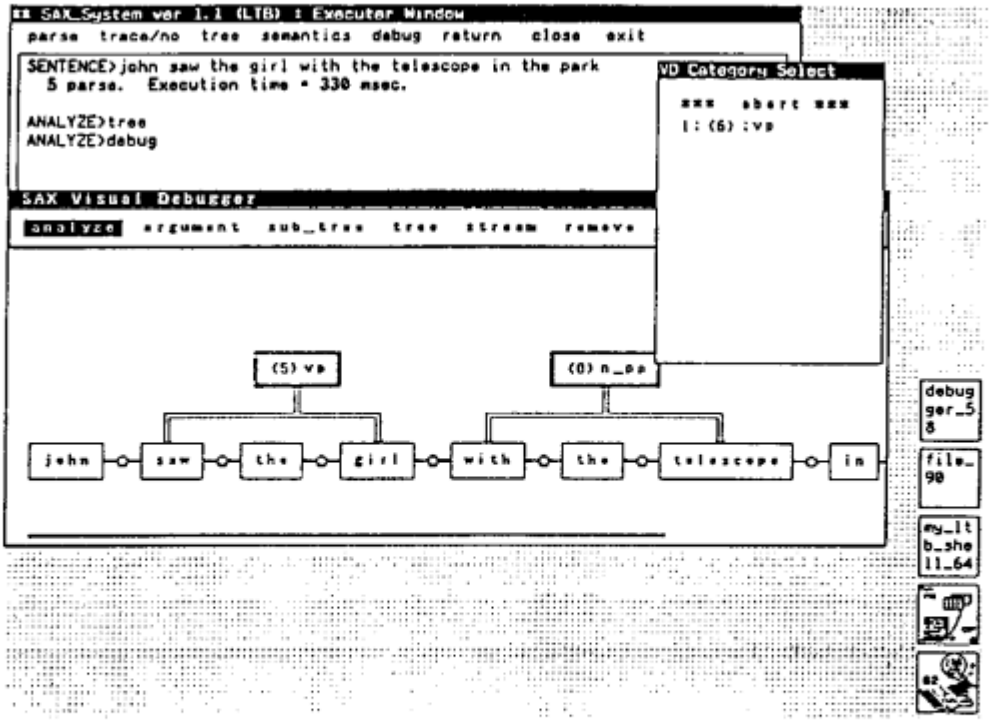


Inspector

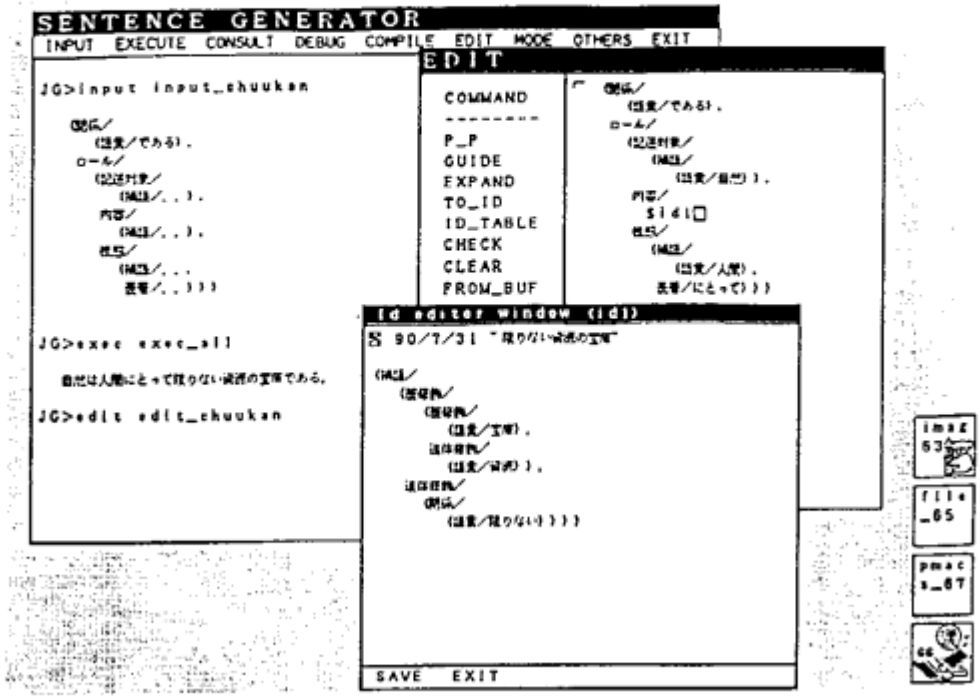
Debugging Tools for Syntactic Analyzer SAX



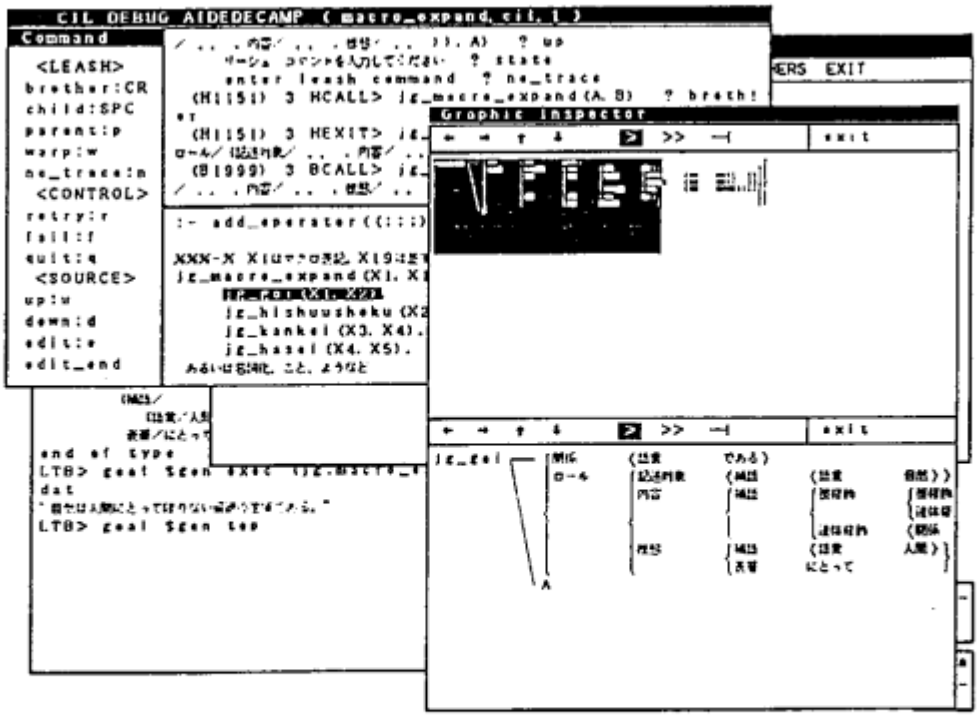
Tree Browser



Visual Debugger



Intermediate Representation Editor



Source Level Debugger(CIL)

文法開発支援環境 LINGUIST

5.1 はじめに

自然言語処理は、知識情報処理分野において一つの重要な位置を占めています。言語情報処理技術の進展は、ワードプロセッサや機械翻訳など、多くの有用な成果をもたらしました。現在、より高度な処理枠組の構築を目指し、談話処理をはじめとする意味処理技術の研究が果敢に進められています。このような状況下、多くの言語情報処理に携わる研究者のために、共通の枠組と平明な理解によって、誰もが使用することが可能であり、しかも広範な言語現象をカバーしている文法の重要性は益々高くなっています。

このような要請の中、そうした機械適合性のある文法を提供すべく、その記述の枠組を提案しました。さらに、機械を通じて処理が行えることの要請に答えるため、文法開発支援環境(以下、LINGUIST)を構築しました。

5.2 特徴

LINGUISTは文法の効率的な開発を第一義とするものであり、環境整備も文法開発の効率向上を主たる目的としていますが、同時に言語現象の研究と実証をも可能とする機能を提供しています。以下に本システムの特徴を列挙します。

1. 位置付け
   - ロジックプログラミングの枠組において構築された自然言語処理実験支援環境である
2. 役割り
   - 解析、生成両面において、諸々の言語現象を分析し処理する
   - 論理文法を基本とし、これまでに得られた日本語研究の成果を反映していく
3. 処理結果
   - 分析で得られる結果は、適用領域や応用分野に依存しない解析構造を設定した
   - 出力構造(構文-意味構造)は、意味処理に必要なデータを含む
   - 構文-意味構造は、曖昧性を減らす際に有効である
4. 動作環境
   -ICOTで開発された逐次型推論マシンPSI-II上で動作
   - 豊富な入出力デバイス、マウスを中心とするダイレクト操作を実現した
5. 拡張性
   - 文脈処理への適応を試みるための拡張性についても考慮した

5.3 文法の枠組

機械処理の観点から、改めて日本語文法を研究調査し、機械適合性を重視した文法の枠組を提案しました。この枠組では説明的な記述を重視した結果、扱える言語現象の正確な類別が可能となっています。

詳細は別資料を参照して下さい。

5.4 システム構成

図5.1はLINGUISTシステムの全体構成の概略図を示しています。LINGUISTはロジックプログラミング環境において対話処理を行うユーザインタフェース、文法の編集を行うエディタ、文法情報の管理を行うデータベース管理機構、解析過程を視覚的にユーザに表示するビジュアルデバッガ、DCG記述の文法をESP形式に変換するトランスレータなど一連のツール群から構成されており、それらのツールは機能ごとに集約されています。各ツールの処理の流れはできるかぎり視覚的にウインドウ上に表示し、文法開発者がその作業においてリアルタイムに開発、評価、検証ができるような枠組となっています。

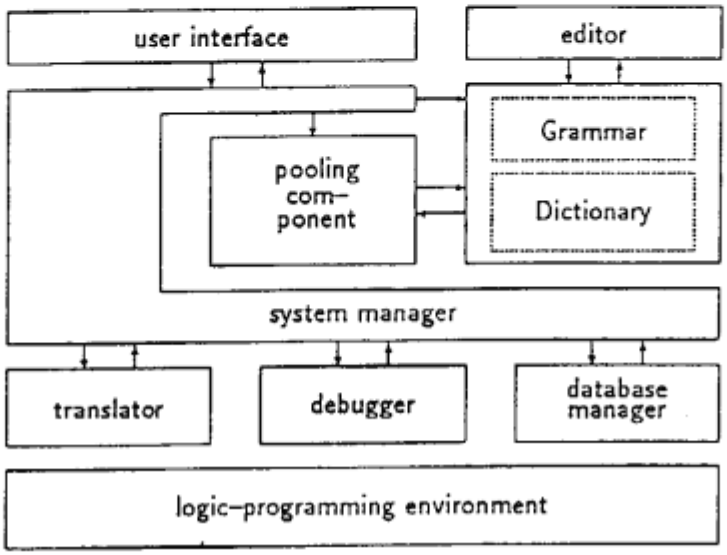


図 5.1: LINGUIST システム (全体構成)

5.4.1 トップレベル

LINGUISTが起動されると図5.2で示すウインドウを表示します。本システムは処理項目をマウスで選択するマウスオペレーションによるメニュー選択方式を基本としています。

5.4.2 ツール

以下に各ツールに関する概要を示します。

ルール管理

ルール管理は図5.3に示すように、文法のシンタックスチェックや使用する文法ファイルの指定などの処理を行います。

VIEWERはテキスト解析、構文生成といった一連の処理の流れを視覚的にユーザに見せる処理系です。本プロセスは以下のツールを提供しています。図5.4はVIEWERにおける処理画面を示すものです。

ビジュアルデバッガ

解析する過程を視覚に訴える形で表示します。どの文法が適用されたかはルール番号とも対応づけられており、解析の流れを詳細にわたってトレースすることができます。図5.5はその処

理画面を示すものです。

ストリーム

LINGUIST とその外部環境との情報のやりとりはストリームを用いたデータ入出力方式によって実現されています。

メモリ上にデータを保持する入出力ストリームは、ファイルシステム、ウインドウシステム及びキーボードなど様々な媒体からのデータのアクセスを可能としています。そして、それらの入出力媒体はユーザがリアルタイムに指定変更することができ、入力媒体からデータが読み込まれると、データが原文テキストの場合は解析用のストリームへ、構成素構造であれば生成用のストリームへそれぞれ蓄積されます。解析 / 生成結果は同様にして、ユーザ定義の出力媒体へ出力されます。

プール

データプールは実験データを記憶する役割を果たすものであり、プール管理機構が管理しています。解析、生成後にそれらのデータはそれぞれのプールへ蓄積され、必要な際に随時任意に取り出して再利用することができます。また、プール機能では解析された結果である構成素構造の細部のチェックを行ったり、構成素構造の内容を変更したりすることができるインスペクタを備えています。図 5.6では解析結果を蓄積する構造プールと構成素構造の中身を表示したインスペクタを示します。

インスペクタ

インスペクタは、入力文の解析結果である構成素構造を視覚的に表示し、個々の構成素を変更するためのツールです。これは次のような機能を持っています。(図 5.7参照)

- 階層表示機能
- 編集機能
- 変数の管理機能

インスペクタのほとんどの操作は、メニューの項目をマウスクリックすることによって簡単にできます。

辞書エディタ

辞書エディタは、辞書データの作成や変更、辞書属性の参照を行うためのツールです。このエディタは、辞書属性を変更すると該当する辞書エントリの属性を一括して自動変更する機能を備えています。

なお、操作はコマンドを与えることによって行います。

5.4.3 バッチ処理

LINGUIST には解析及び生成のバッチ処理があります。このバッチ処理は多数のテキスト文を一括して解析したり生成したりするのに便利な機能です。またこの機能にはバックトラックによる解の探索を選択することができます。

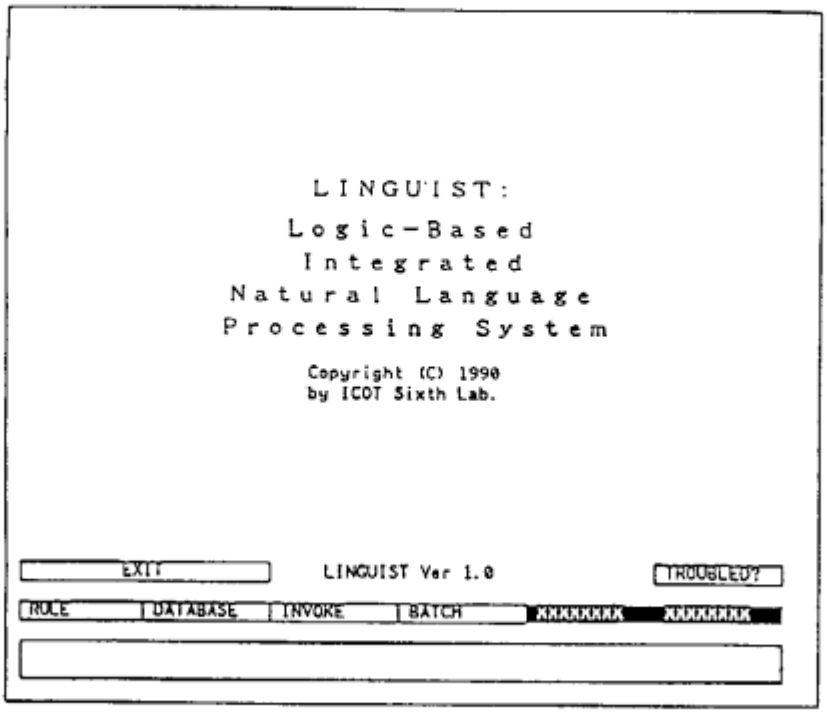


図 5.2: LINGUIST トップウインドウ

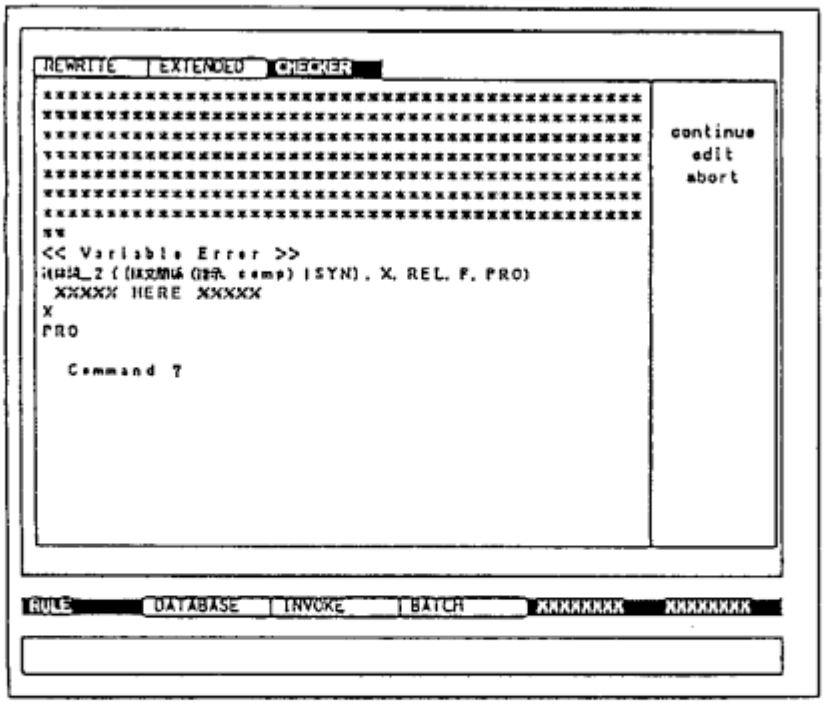


図 5.3: ルール管理 (シンタックスチェック)

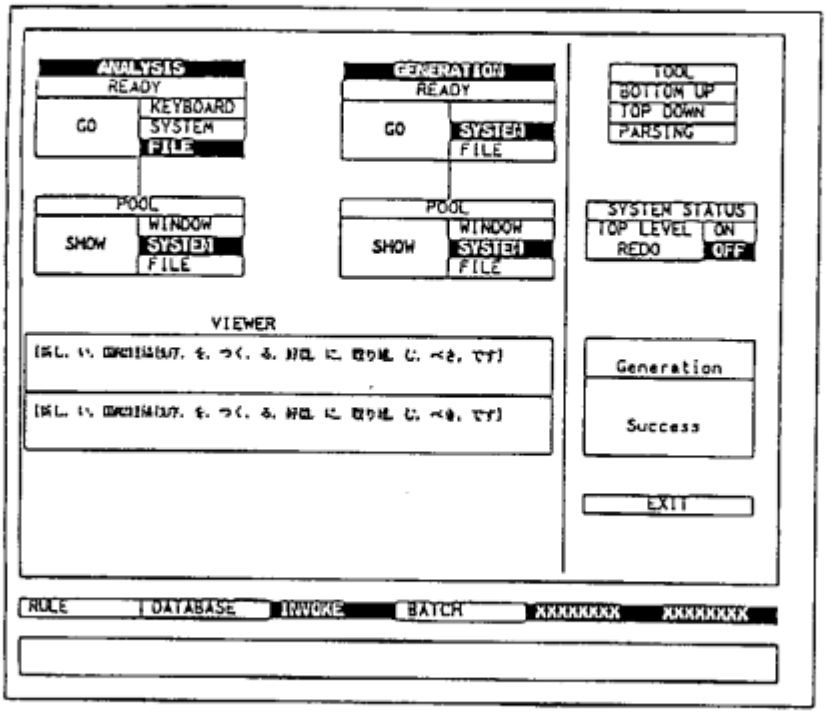


図 5.4: VIEWER ウインドウ

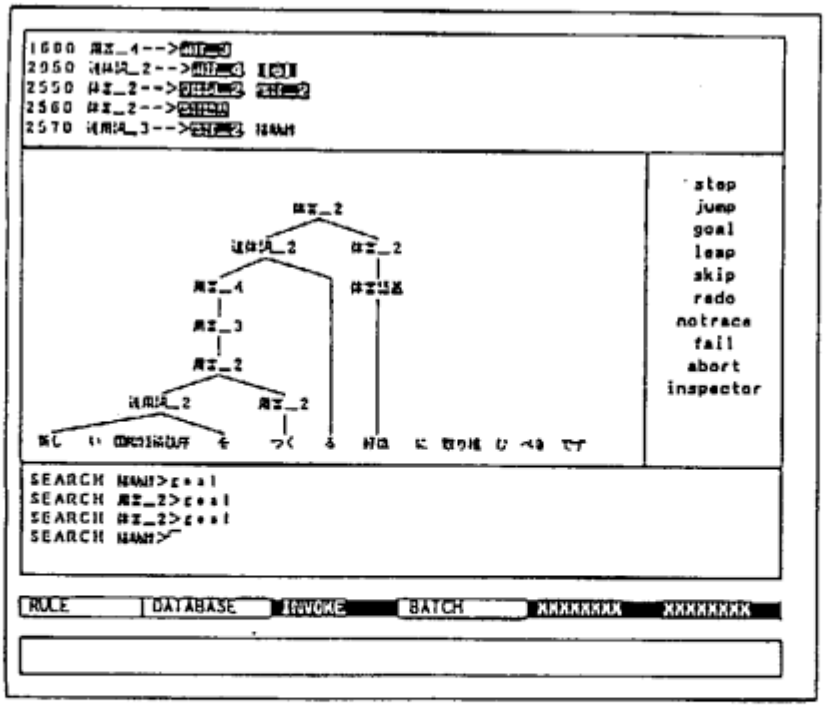


図 5.5: ビジュアルデバッガ

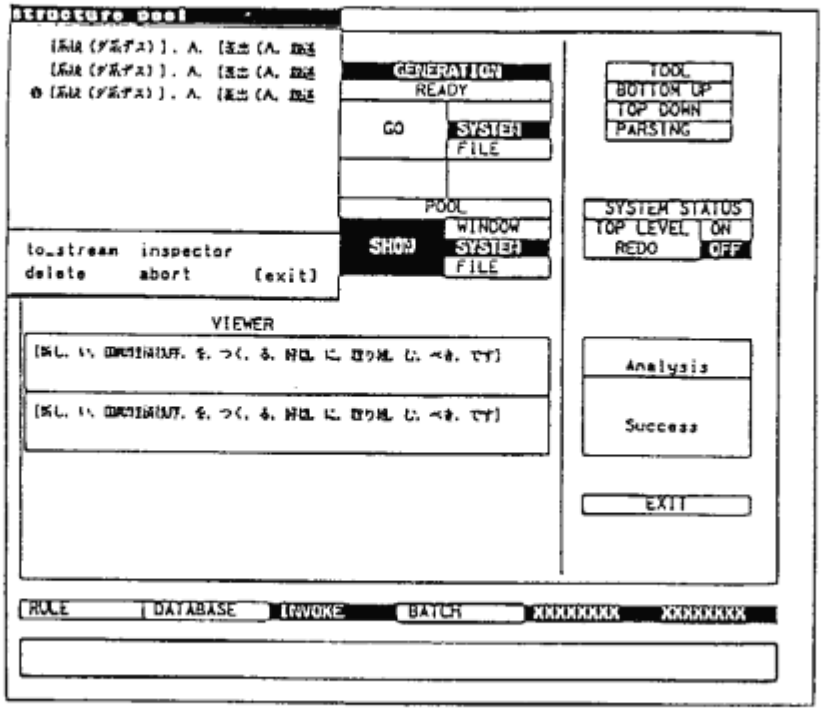


図 5.6: プール

```
inspector( top menu )
**  exit  **
**  quit  **
構文構造          系統（ダ系デス）
言語構造          喪出（A，叙述）
                  ムード（A，未完了）
                  認め方（A，肯定）
                  アスペクト（A，開放）
                  〈*，*，*〉
                  nil
                  ソート（nil，だ）
                  引数（nil，[B，K]）
言語関係（特定）  nil
        （未特定）  （が，comp，K）
参照              nil
```

図 5.7: インスペクタ