

TM-1141

IJCAI-91 ICOT EXHIBITION BOOKLET

by

S. Uchida, K. Hirata & S. Taba

December, 1991

© 1991, ICOT

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

IJCAI-91

Aug. 26 ~ 29, 1991

ICOT
EXHIBITION

Logic

Knowledge



Parallel Machine



Institute for New Generation Computer Technology

=====

This booklet is edited by

Dr. Shunichi Uchida, Dr. Keiji Hirata and Mr. Susumu Taba.

Please address queries on this publication to

Dr. Shunichi Uchida, uchida@icot.or.jp,

Dr. Keiji Hirata, hirata@icot.or.jp or

International Relations Department, ICOT.

Copyright ©1991

Institute for New Generation Computer Technology, **ICOT**

Mita Kokusai Bldg. 21F

4-28 Mita 1-chome, Minatoku-ku, Tokyo 108, Japan

Phone: +81-3-3456-3195

fax: +81-3-3456-1618

telex: ICOTJ32946

Contents

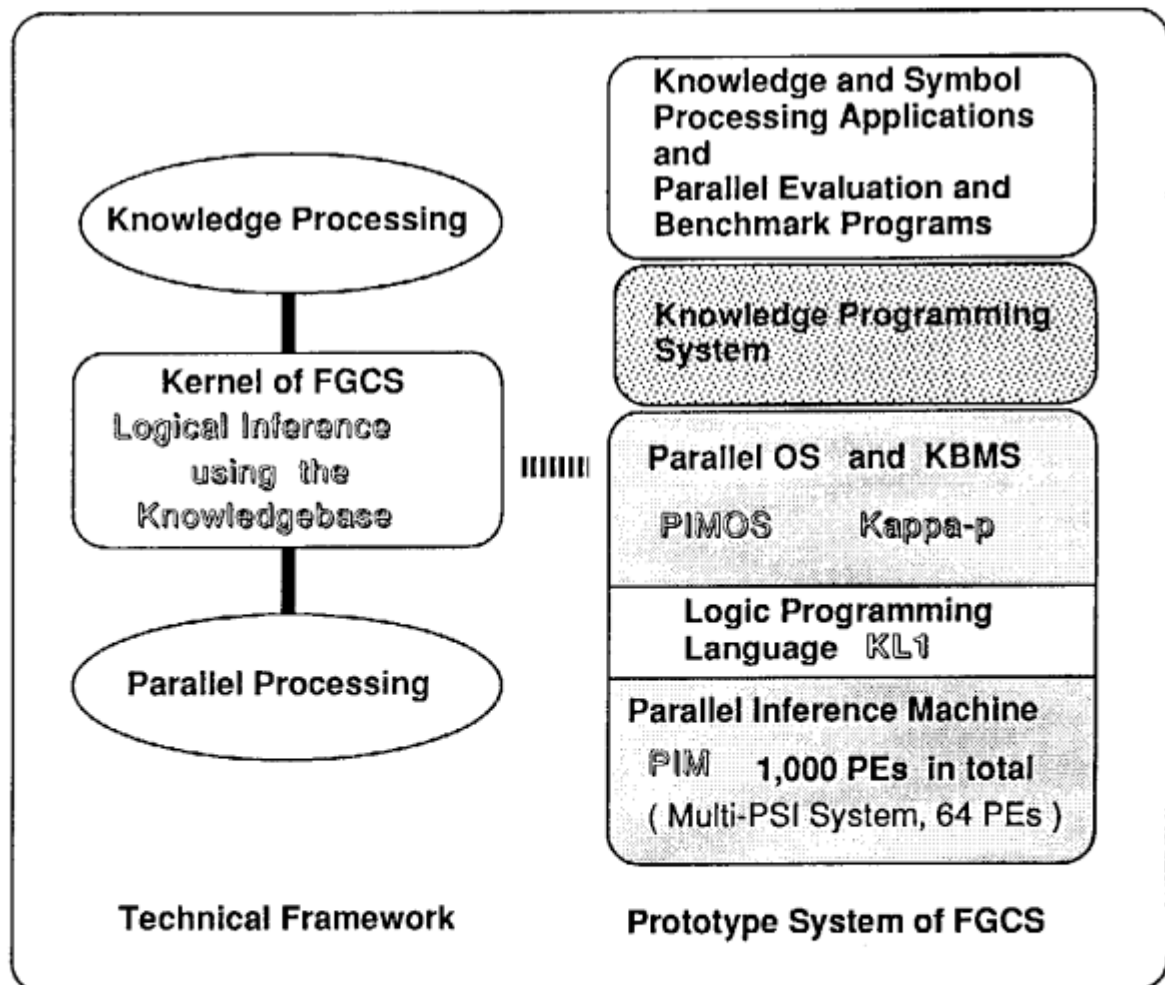
Introduction to Fifth Generation Computer Systems	1
Outline of Parallel Inference Machine: PIM	3
Outline of PSI/UX series	5
ICOT Demonstration Lineup	6
1 Logic-based Parallel VLSI-CAD System	7
2 HELIC-II (A Parallel Legal Reasoning System)	13
3 Go Generation	17
4 Genome Analysis Programs	21
5 MGTP (Parallel Model Generation Theorem Prover)	29
6 EUODHILOS (A General Reasoning System for a Variety of Logics)	33
7 CAL (Constraint Logic Programming Languages)	37
8 Kappa with Molecular Biological Data	41

Fifth Generation Computer Systems Project

A Japanese national project aimed at new computer technology for knowledge and symbol processing supercomputers.

Since 1982, the Institute for New Generation Computer Technology, ICOT, has been developing a prototype FGCS. It consists of a parallel OS, a parallel KBMS and a parallel inference machine with about 1,000 processing elements.

After 10 years of R&D, the machine is now ready for large-scale AI applications !



Developing the FGCS prototype system

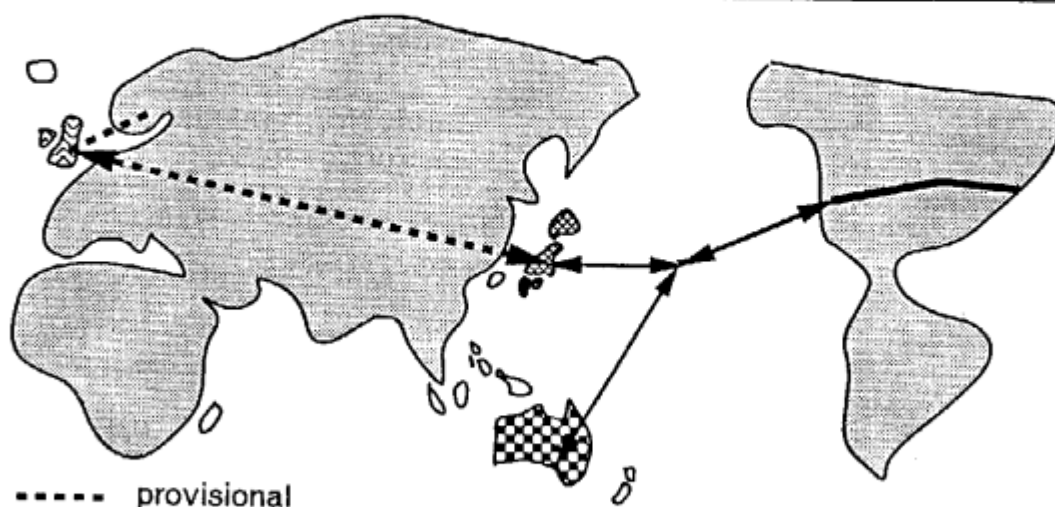
Starting with Sequential Inference Machines, and then, smoothly jumping to Parallel Inference Machines.....

Parallel Logic Programming is the key to successful development of Parallel OS and Parallel Application Software.

	Personal Sequential Inference Machine PSI and its OS SIMPOS	Parallel Inference Machine PIM and its OS PIMOS
1982	Design of Sequential Logic Programming Languages KLO and ESP	Design of Parallel Logic Programming Languages GHC and KL1
1984	PSI-1 and SIMPOS -V1 35KLIPS for KLO	
1985	PSI-2 and SIMPOS-V2 330KLIPS for KLO	Multi-PSI V2 and PIMOS-V1 64 PEs 5MLIPS for KL1
1988		
1989	PSI-3,UX and SIMPOS-V7 1.4MLIPS for KLO	Prototype of FGCS PIM and PIMOS-V2 + Kappa-P-V1 1000 PEs total 200MLIPS / 512PEs for KL1
1992		

* 1LIPS (Logical Inferences Per Second) = 30 - 50 IPS (Instructions Per Second)

The prototype system will be accessible via a worldwide computer network and will work as a powerful inference server or large -scale KB server for researchers the world over.

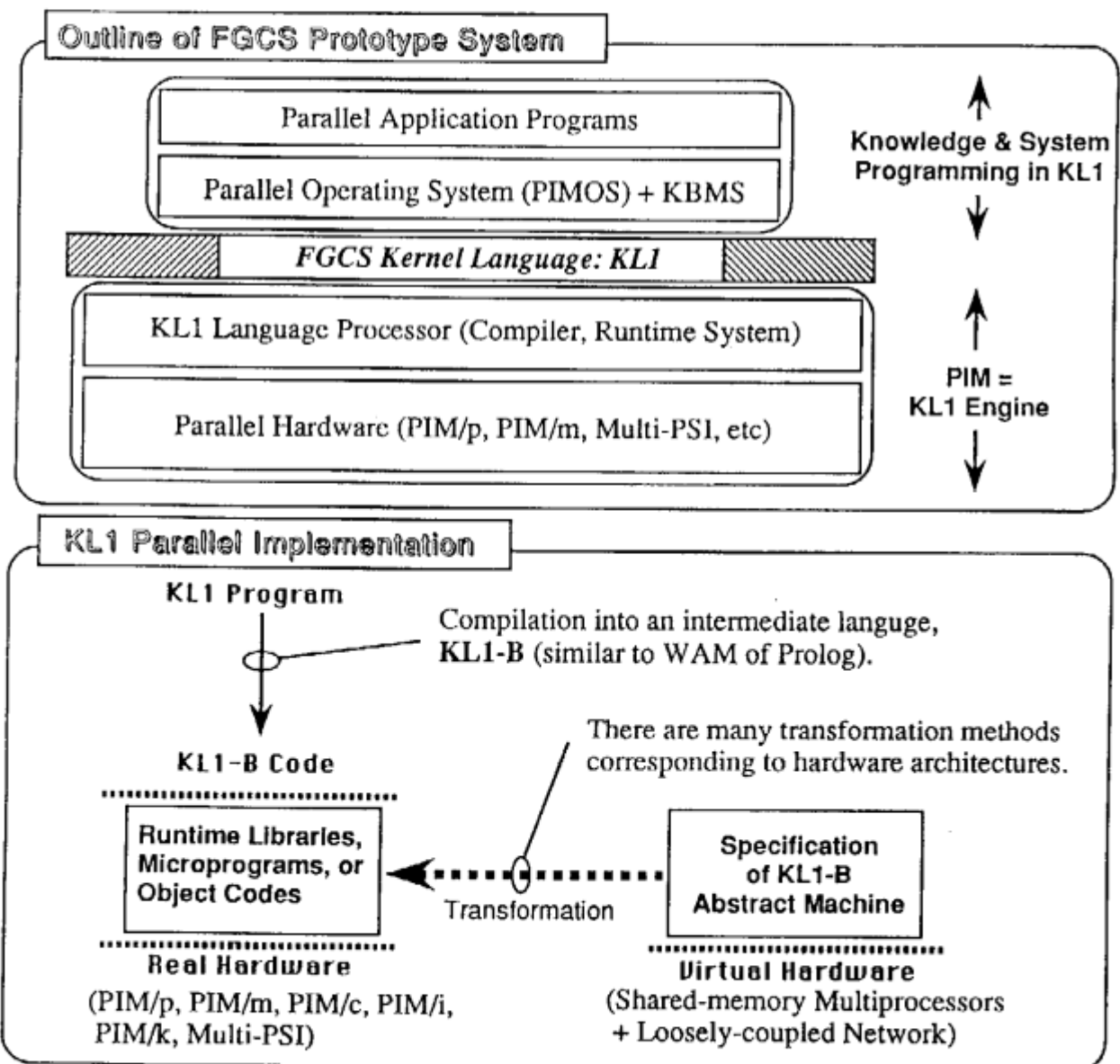


Parallel Inference Machine: PIM

A parallel inference system can realize a prototype FGCS that can provide the largest symbolic & knowledge computing capability and which, moreover, can be easily programmed by users. The FGCS kernel language *KL1* supports these remarkable features.

KL1 is a concurrent logic programming language. The advantages of KL1 are:

- * KL1 provides *capable parallel programming facilities*, e.g. almost-infinite-level priority, pragma for location where process is to be executed, distributed execution control.
- * *Implicit dataflow synchronization* with logical variables eliminates the possibility of synchronization & communication bugs.
- * *The descriptive power is high enough* to write a real parallel operating system.
- * *The flexibility* allows us to implement various programming paradigms.



PIM Hardware

PIM yields huge computing power

One processing element (PE) of PIM/p and PIM/m achieves 600 ~ 700 KRPS (RPS = Reduction Per Second). The 512-PE PIM/p system can reach 350 MRPS (= 7 GIPS approximately), and the 256-PE PIM/m 175 MRPS (= 3.5 GIPS).

PIM - a general-purpose parallel inference machine

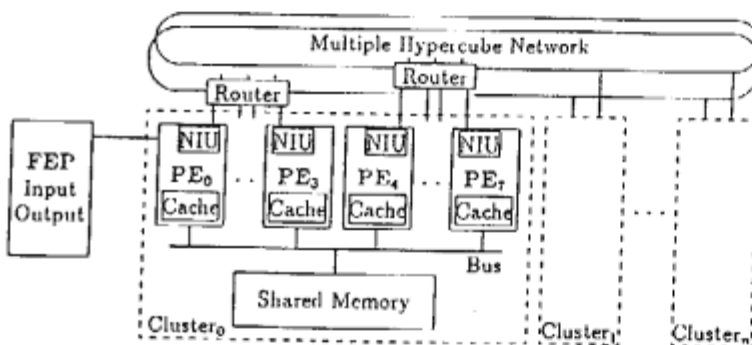
All PIM models adopt MIMD and a tagged architecture, and carry specialized hardware for efficient KL1 parallel implementation newly invented.

ICOT is developing five PIM models

This enables us to compare and evaluate the performance of several different architectures. Their names are (by manufacturer): PIM/p (Fujitsu), PIM/m (Mitsubishi), PIM/c (Hitachi), PIM/i (Oki), and PIM/k (Toshiba).

PIM/p

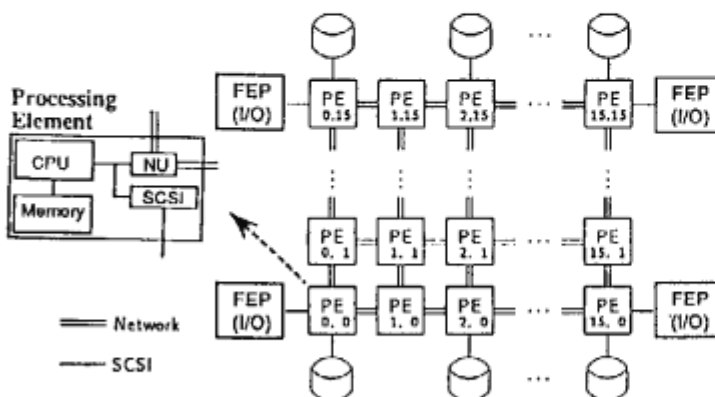
- * 6-dimensional doubled hypercube network (inter-cluster).
- * High-speed communication & synchronization by KL1-oriented coherent cache.
- * Macro-call instruction for lightweight subroutine call.



- Total PEs: 512 (64 clusters)
- 1 Cluster:
8 PEs + Shared bus
+ Shared memory (256MB)
- CPU:
RISC + Macro-call
Cycle time = 60 ns
4-stage pipeline
- Coherent cache protocol:
Invalidation, 4 states
- Inter-cluster network:
Throughput = 20MB/s

PIM/m

- * 2-dimensional mesh network (16 x 16) (inter-cluster).
- * KL1-oriented horizontal microprogram.
- * Five-stage pipelined processor controlled by microprogram.

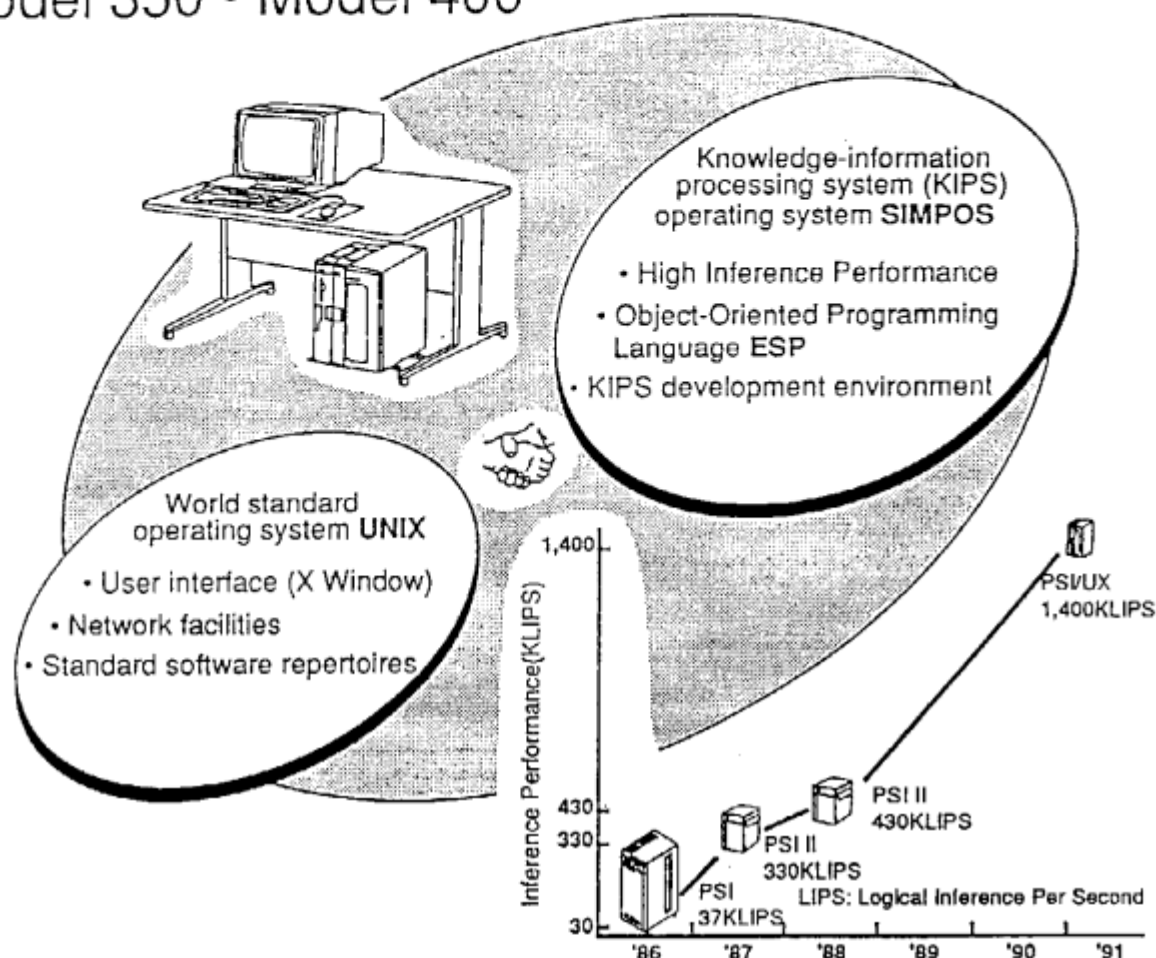


- Total PEs: 256
- 1 Cluster (Node):
1 PE + Memory (80MB)
- CPU:
Microprogram
Cycle time = 65 ns
5-stage pipeline
- Cache protocol:
(Writeback)
- Inter-cluster network:
Throughput = 4MB/s

AI Workstation

PSI/UX series

Model 350 • Model 400



The World's Highest Inference Performance

The inference unit employs VLSIs developed in the Fifth-Generation Computer System Project to achieve the world's highest inference performance, a burst rate of 1.4MLIPS (million logical inference per second).

Integration of Knowledge-Information Processing and Conventional-Information Processing

Users can operate the PSI/UX as a PSI or as a UNIX workstation, but the real value of the PSI/UX lies in its ability to integrate the two. Developers can build practical knowledge-information processing systems using a distributed approach in which the AI functions are performed on the PSI machine, conventional processing is performed under UNIX, and results integrated.

Remote Procedure Call (RPC) function provides for communication paths between the SIMPOS and UNIX application programs. The RPCs are used to call UNIX function from within a SIMPOS application program, or to call SIMPOS functions from within UNIX applications.

Highly Flexible Expert System Development Support Tools and Languages

MELPLAN, the domain specific expert system development shell for manufacturing processes, EXTKERNEL II, the general purpose expert system development tool and Acekit, the intelligent spread sheet offer many convenient functions to develop practical expert systems.

ESP (Extended Self-contained Prolog) is a sophisticated system-development language. It allows both object-oriented and predicate expressions, enabling the user to describe complex and varied system types. SIMPOS, MELCOM PSI/UX's operating system, is fully written in ESP. Conventional prolog (PSI-Prolog) is also available.

Applications

MELCOM PSI/UX allows the user to configure user-friendly high-level AI systems, such as R&D application systems, intelligent document retrieval systems, scheduling expert systems, and machine translation servers.

* MELCOM PSI, PSI II, PSI/UX are products of MITSUBISHI Electric Corp. which utilize results of the Fifth Generation Computer System project promoted by MITI.
 * UNIX Operating System is developed and licensed by UNIX System Laboratories, Inc.

ICOT Demonstration Lineup

The keywords behind our demonstrations are: **logic, parallel, and inference engine**. Demo programs (1) to (5), running on parallel inference machines (set up in Japan), are all written in KL1 logic programming language. Programs (6) to (8) are working on a sequential inference machine (here in the ICOT booth, AIE-91).

(1) Logic-based Parallel VLSI-CAD System:
Logic simulation and routing for VLSI design.
High performance achieved by parallel processing.
Benchmark for large-scale symbolic parallel computation.

(2) HELIC-II:
Experimental legal reasoning system for penal code.
Cooperative reasoning with statutory rules and past cases.
Parallel search for past cases based on similarity.

(3) Go Generation:
Go-playing system utilizing human tactics.
Modeling of complicated thinking process.
New frontiers of profound knowledge processing.

(4) Genome Analysis Programs:
High-quality multiple-sequence aligners.
Two methods examined: 3-dimensional DP-matching, and parallel simulated annealing.
PIM brings innovation to biological science.

(5) MGTP (Parallel Model Generation Theorem Prover):
One of the fastest parallel first-order theorem provers.
KL1 compiling technique for efficient implementation.
Flexible representation enables wide-ranging applications.

(6) EUODHILOS:
General reasoning system for a variety of logics.
Human-oriented proving methodology.
Visual human-computer interface for reasoning.

(7) CAL:
New programming paradigm: from "how" to "what".
The most powerful CLP language in field of algebraic constraints.

(8) Kappa with Molecular Biological Data:
High-performance knowledgebase engine.
Powerful capability of nested relational modeling.
New framework for molecular biological databases.

Organization of R&D Themes in FGCS Final Stage

Expert System

VLSI-CAD ⁽¹⁾
Legal Reasoning System ⁽²⁾
Go Playing System ⁽³⁾
Genetic Information Processing ⁽⁴⁾

Parallel Applications

Genetic DB

Natural Language Understanding

FGCS Knowledge Programming Environment

Problem Solving

Theorem Prover ⁽⁵⁾
Reasoning System ⁽⁶⁾
Constraint Logic Language ⁽⁷⁾

Knowledgebase Construction

Knowledge Representation Language
Deductive Object-oriented Database

Programming Environment

Parallel-programming Support Tools

Basic Software System

PIMOS, KBMS ⁽⁸⁾

Prototype Hardware System

Parallel Inference Machine (PIM, Multi-PSI)

TITLE

A Parallel Logic Simulation System for VLSI Design

PURPOSE

- To construct a logic simulator in a concurrent logic language KL1
- To evaluate the Time Warp mechanism

OUTLINE & FEATURES

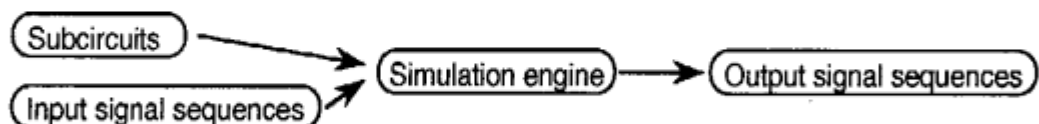
1. Event simulation
 - To simulate the gate behavior only when its input signal changes
 - Simulation is executed by passing messages between gate objects.
2. Distributed time keeping mechanism – Time Warp mechanism
 - History is recorded for each gate.
 - Rollback will happen when a message arrives out of order.
3. Load distribution – static load distribution by preprocessing
 - To attain high quality load balancing
 - To reduce inter-processor communications
 - To exploit high parallelism

SYSTEM CONFIGURATION

Preprocessing phase : static load distribution



Simulation phase : parallel execution



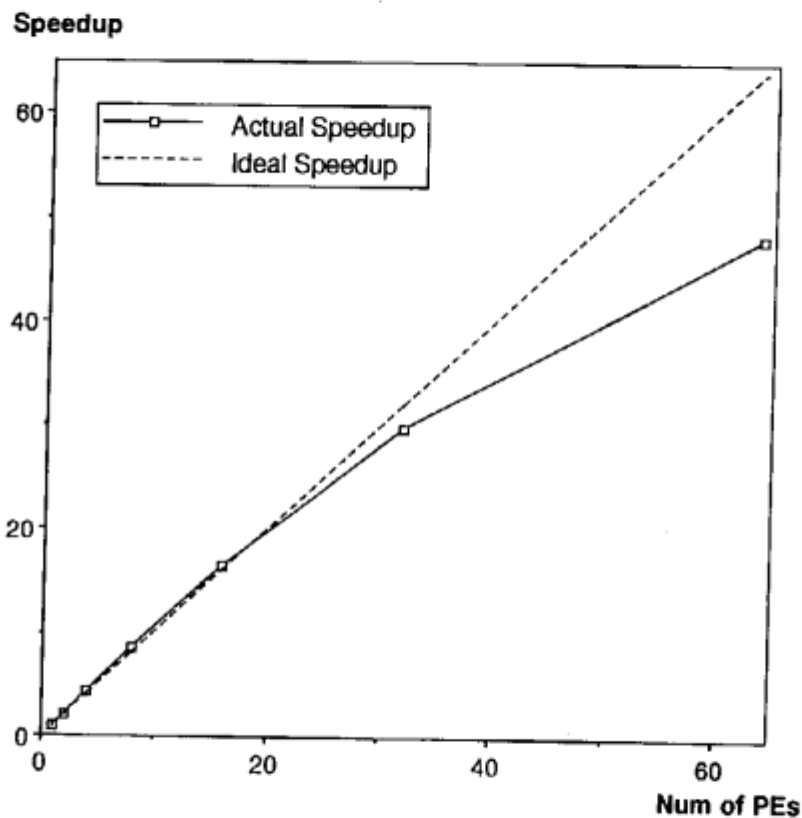
DEMONSTRATION

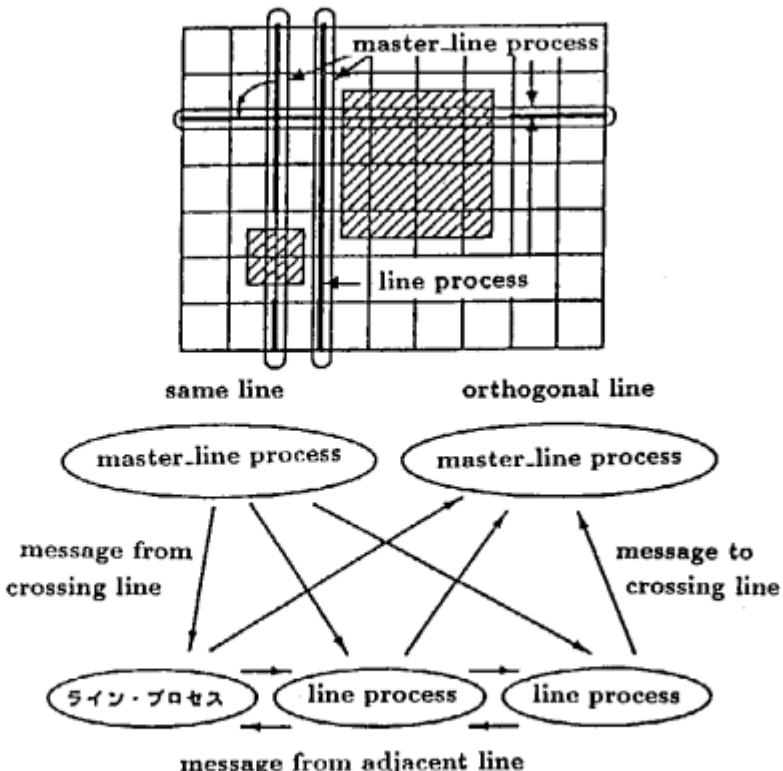
We simulate a sequential circuit which consists of about 12,000 gates. Input signal sequences are generated randomly except on clock lines.

As the result of the simulation, besides the output signal sequences, we get several data for performance evaluation. We can evaluate the execution time, and to compare the total number of messages with the number of rollback occurrences.

SPEEDUP

Good speedup of 48-fold is attained using 64 processors.



Title	Parallel LSI-CAD demonstration program LSI router
Purpose	A VLSI layout problem consists of several different problems that require massive computational power. Routing is one of those problems. Our aim is to study concurrent algorithms and load-balancing methodologies through design and development of parallel routing programs.
Outline & Features	<p>[Abstract] This program executes routing between modules on an LSI chip, after the placement of each module has been fixed. It determines the connection paths between terminals of each module.</p> <p>[Concurrent algorithm] The Basic algorithm is a sequential line search, the look-ahead line search algorithm. It is expanded for parallel execution. Major parallelism is extracted from concurrent routing between nets.</p> <p>[Implementation] As this program is based on a kind of line search algorithm, processes are assigned to each line segment on each grid line as concurrent execution primitives. Intermediate results of routing are kept as inner statuses of each line process. Routing is executed by communication between these line processes. The master line processes stand for grid lines, and manage line processes on a corresponding grid, and relay those communication messages. Line drawing and rip-up correspond to dynamic split and joint of these line processes.</p>
System Configu- ration	 <p>The diagram illustrates the system configuration of the LSI router. It is divided into two main parts: a grid-based representation and a communication flowchart.</p> <p>Grid-based Representation: A grid is shown with a shaded rectangle labeled "master_line process" and a smaller shaded rectangle labeled "line process". The grid is divided into two sections: "same line" and "orthogonal line".</p> <p>Communication Flowchart: Below the grid, a flowchart shows the communication between master and line processes. Two "master_line process" nodes (top) are connected to three "line process" nodes (bottom). Arrows indicate the flow of messages:</p> <ul style="list-style-type: none"> "message from crossing line" (from master to line) "message to crossing line" (from line to master) "message from adjacent line" (between line processes)

[Routing problem]

Routing is one of the VLSI layout problems which determines connection paths between terminals of modules on an LSI chip. Routing is executed after placement of modules has been determined in an LSI design of gate arrays, standard cells, or building blocks. There are several well known algorithms for the problem such as maze routing, line search and channel routing. We assume two routing layers, one for vertical and the other for horizontal paths. We also assume that each connection must be routed on a virtual grid on a chip surface. The block and through hole inhibition conditions are also dealt with.

[Basic algorithm]

This program is based on a kind of line search algorithm, look-ahead line search. This algorithm calculates positions that are expected to lead to a good solution before routing each line segment. Figure 1 shows this process. Start point S and a target T are given. If a line drawn downward from S turns at A, then the reachable point that is closest to point T is point a. Similarly, if the line turns at point C,D then corresponding points are c and d for each. These points (a, c, d) are called expectation points for S. Note that as the through hole is inhibited at point B, so point b cannot be an expectation point for S. Of all these expectation points, point c is the closest one to point T, so point S and point C will be connected in this search step. Similar processes are followed and thus point S and point T will be connected. In addition to the above processes, this algorithm includes two more functions. One is to get out of local optimal point in expectation points calculation. The other is backtrack for escaping from a dead-end by removing the last-connected line and returning to the point visited last. Thus this algorithm guarantees the wireability between two terminals, if connection paths exist.

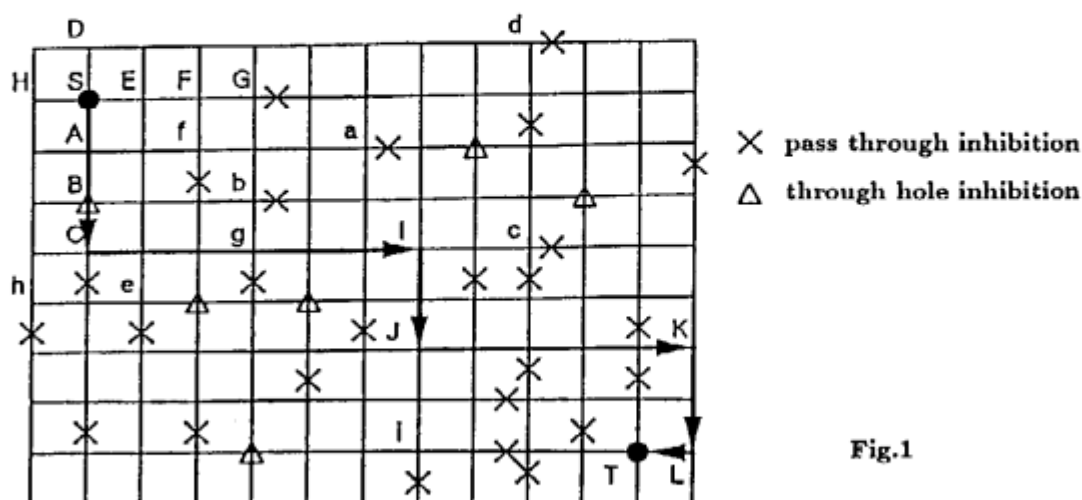


Fig.1

[Concurrent algorithm]

This program uses a parallelized version of the line search algorithm shown above. The program is designed to extract a parallelism of computation mainly from the concurrent search of multiple nets. On KL1 programming, the minimum execution unit is called process. We usually adopt an execution model in which the computation is executed by exchange of messages between these processes. This program also adopts this execution model. As our algorithm is based on the line search algorithm, so processes correspond to each lines on grid. Each line process maintains the corresponding line's status and at the same time the execution entity of search. As figure 2 shows, each process corresponds to each grid line and line segment on it. In this program, search and routing proceeds by the exchange of messages between these line processes. The routing process of one net is almost the same as that of the basic algorithm, but the computation of the expectation point, mentioned before, is parallelized. The computation of the best expectation point is executed in this program as follows. Request messages for calculation of expectation point are distributed from the line process now being searched to the line processes that cross it. Thus computation of expectation point is executed concurrently on each line process that received a calculation request message. Later, the result of each calculation will be returned from these line processes to the searching line process, then this line process aggregates these results and determines which is the best expectation point. When the best expectation point is found, the searching line is connected (fixed) to the crossing line that includes the best expectation point. (Figure 3(a)-(d))

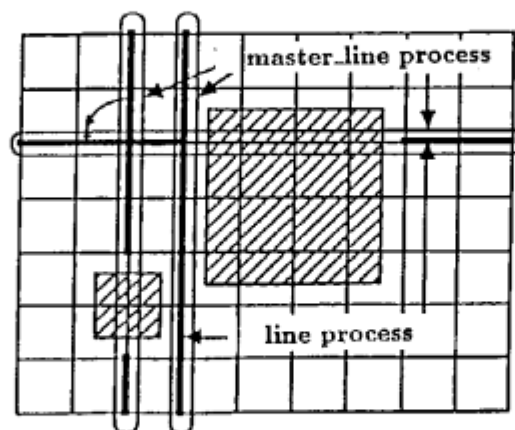


Fig.2

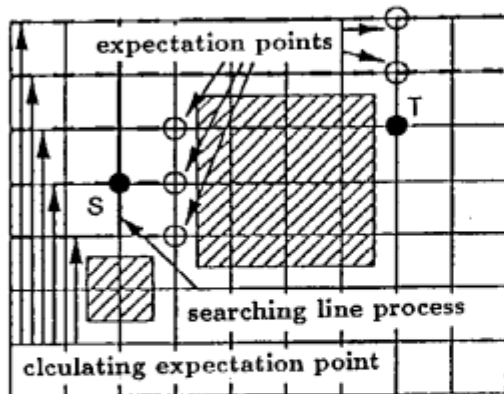


Fig.3(a)

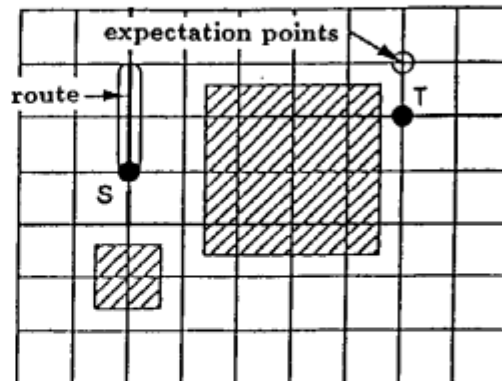


Fig.3(b)

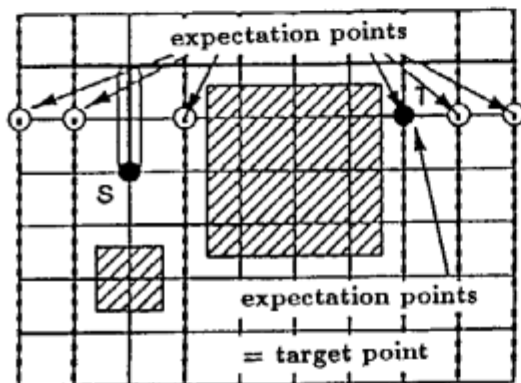


Fig.3(c)

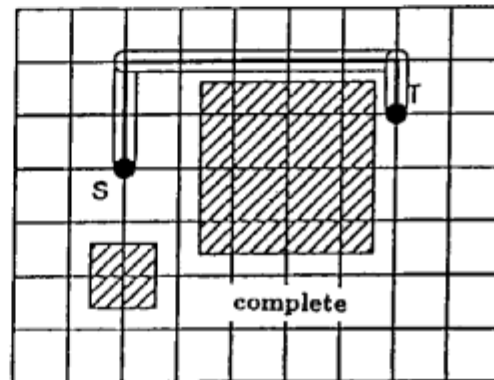


Fig.3(d)

Note again that the concurrency of computation is extracted mainly from routing of multiple nets, in other words, from parallel search for multiple nets. In this program, routing is scheduled to route nets in increasing order of their size, shortest net first and longest net last.

[demonstration]

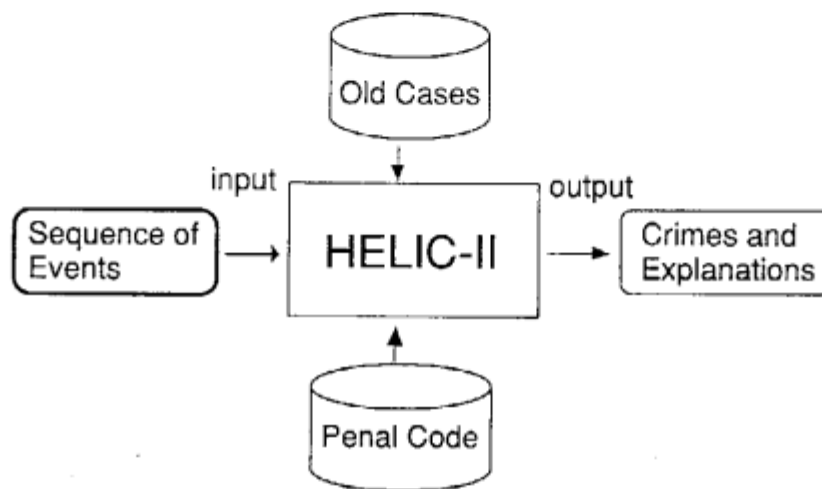
The parallel routing program, written in KL1 on the Multi-PSI, executes routing of LSI chips of a practical size. Execution results will be shown on a display in real time.

[Kitazawa,H. and Ueda,K., "A LOOK-AHEAD LINE SEARCH ALGORITHM WITH HIGH WIREABILITY FOR CUSTOM VLSI DESIGN", proc. of ISCAS 85, pp1035]

Title	HELIC-II - A Parallel Legal Reasoning System
Purpose	<ul style="list-style-type: none"> • Development of a large scale parallel intelligent system • Research into a legal reasoning model
Outline & Features	<p>[Outline] HELIC-II is an experimental parallel legal reasoning system for Penal Code. It draws legal conclusions for a given case, given statutory rules and legal precedents.</p> <p>[Features]</p> <ul style="list-style-type: none"> • HELIC-II consists of a rule-based engine and a case-based engine. • The rule-based engine logically calculates all possible crimes based on the statutory rules. • The case-based engine retrieves similar cases from a case base, and calculates legal interpretation.
System Configuration	<pre> graph TD subgraph Inference_System [Inference System] CR[(case rules)] --> CBE[case-based engine] CBE --> WM((WM)) LR[(legal rules)] --> RBE[rule-based engine] RBE --> WM end OC[old cases] -- extract --> CR SR[statutory rules] -- extract --> LR NC[new case] -- input --> WM WM --> EC[explanation constructor] EC -- output --> JR[Judgements and reasons] </pre>

1 Architecture of HELIC-II

HELIC-II is a legal reasoning system which draws legal conclusions for a given case. It consists of two inference engines - a rule-based engine and a case-based engine. The rule-based engine handles statutory rules and the case-based engine handles cases. The target domain of HELIC-II system is the Penal Code of Japan. When a new case is input as a set of facts, the system lists up all possible crimes together with their explanations by referring to both the Penal Code and precedent cases. This system is implemented on the Multi-PSI and draws conclusions in parallel.



HELIC-II

2 Reasoning based on statutory rules

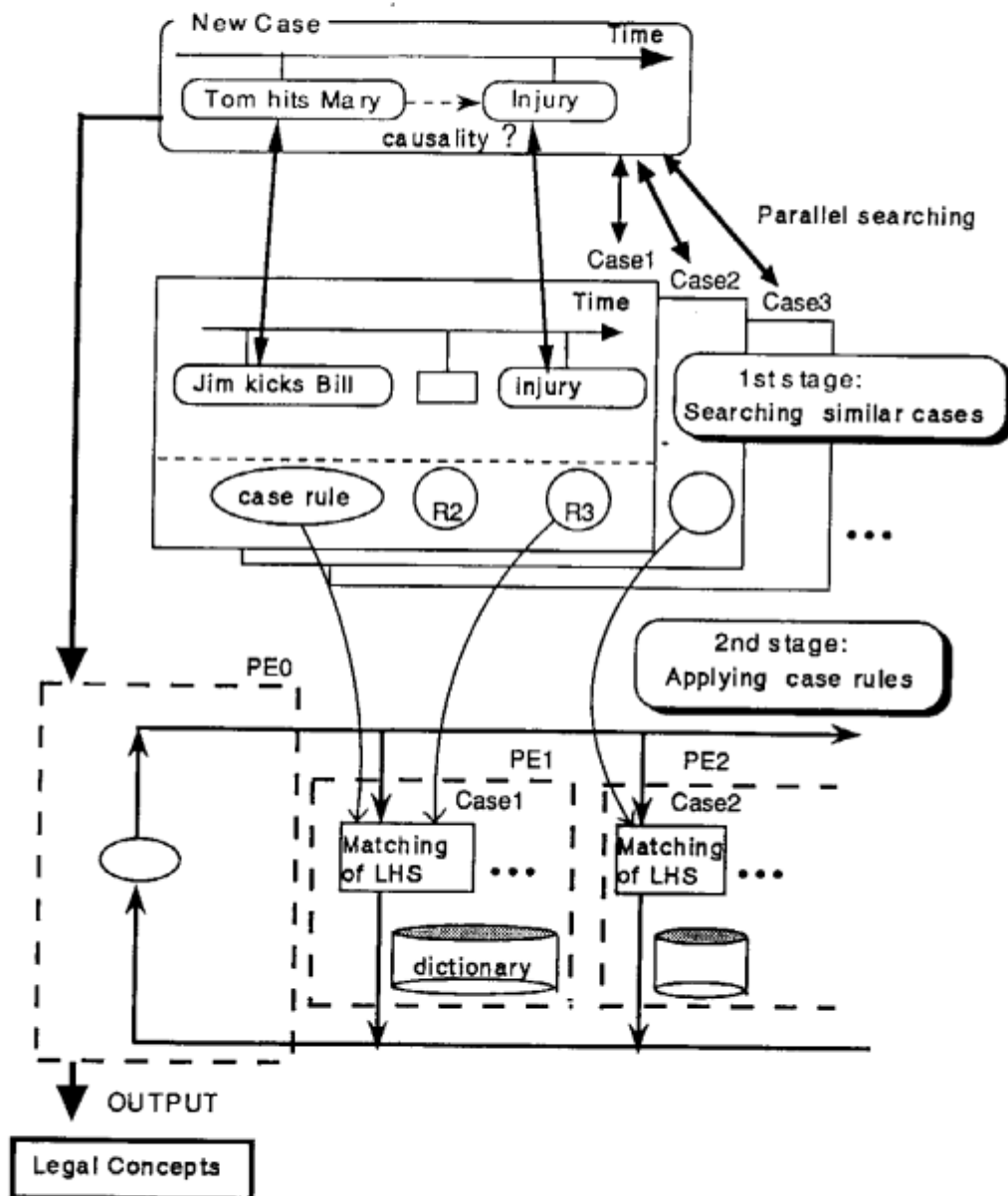
The legal rules of the Penal Code are interpreted and represented as clauses. The rule-based engine is an extension of the MGTP (Model Generation Thorem Prover). The MGTP solves range-restricted non-Horn problems by generating models. In the MGTP, each branch of this proof tree is distributed to other PEs and is executed in parallel. Therefore, this engine is useful when the given problem generates lots of branches. In HELIC-II, the different theories of interpreting legal rules generate different branches.

3 Reasoning based on similar cases

Each precedent case is represented as a *situation* and *case rules*. A situation is a sequence of actions and related information such as agents and objects. A case rule represents the arguments of both parties. The LHS (left hand side) of a case rule is part of a situation, and the RHS (right hand side) is the legal concepts insisted on by one of parties. Case rules differ from production rules because they don't contain variables and are not fired by strict matching but by similarity matching.

The reasoning of the case-based engine consists of two stages. When a new case is given as a form of case situation, as the first stage, the case-based engine searches for similar situations in a case base.

The second stage is an extension of a production system. At first, case rules of selected cases are distributed to different PEs. Then, the engine compares the situation of a new case with the LHSes of case rules in parallel. If similar LHSes are found, their RHSes are executed. The case-based engine repeats this cycle of matching and firing.



4 Demonstration

The following is an example of a problem that can be handled by HELIC-II.

On a cold winter's day, Mary abandoned her son Tom on the street, because she was very poor. Tom was just 4 months old. Bill found Tom crying on the street, and started to drive Tom to the police station. However, Bill had an accident on the way to the police station, and Tom was injured. Bill thought that Tom had died in the accident, so left Tom on the street. Then, Tom froze to death.

The problem is judging the crimes of Mary and Bill. The hard issues are the following two points.

1. Is there causality between Mary's action (abandonment) and Tom's death?
2. Is there causality between the traffic accident and Tom's death?

HELIC-II searches for similar cases, and enumerates the crimes of Mary and Bill.

5 Performance

The following shows the parallel inference performance.

number of PEs	1	2	4	8	16	32	64
time(sec)	271	268	163	127	109	97	82
speed up	1.0	1.0	1.7	2.1	2.5	2.8	3.3

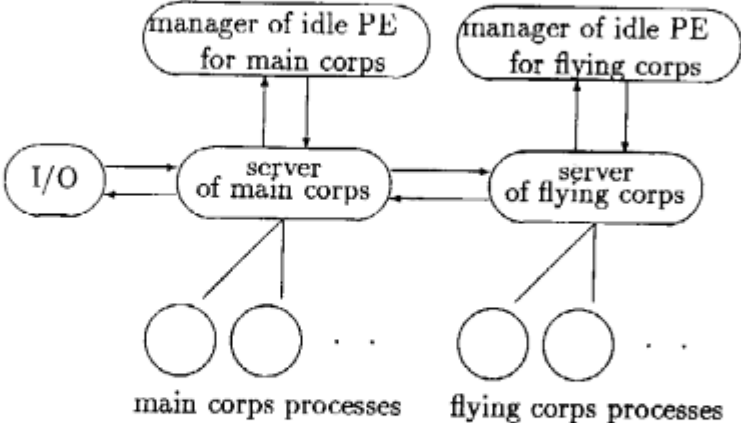
Table 1: Speed up by rule-based engine

number of PEs	1	2	4	8	16	32	64
time(sec)	-	-	-	1182	516	309	170
speed up	-	-	-	1.0	2.3	3.8	6.9

Table 2: Speed up by case-based engine (large data)

number of PEs	1	2	4	8	16	32	64
time(sec)	805	731	264	206	115	-	-
speed up	1.0	1.1	3.0	3.9	7.0	-	-

Table 3: Speed up by case-based engine (small data)

Title	Go Generation: Experimental Game-playing System for Future Knowledge Processing
Purpose	Go has been a difficult game for the computer to play. We are trying to build a strong Go program using the computer power of the parallel inference machines.
Outline & Features	<p>[Outline]</p> <ul style="list-style-type: none"> • The intermediate results of parallel Go playing system "GOG" on parallel inference machine. • This research is being jointly developed with ETL. <p>[Feature]</p> <ol style="list-style-type: none"> 1. It simulates thinking mechanism of human player. 2. The large tasks are performed in parallel using a dynamic load balancing technique. 3. We propose a new technique "frying corps" to make a game playing program stronger without losing the real-time property.
System Configuration	 <pre> graph TD IO([I/O]) <--> SMC([server of main corps]) SMC <--> SFC([server of flying corps]) SMC <--> MIPM([manager of idle PE for main corps]) SFC <--> FIPM([manager of idle PE for flying corps]) SMC --- MCPs((...)) MCPs --- MCPs_label[main corps processes] SFC --- FCPs((...)) FCPs --- FCPs_label[flying corps processes] </pre> <p>The diagram illustrates the system configuration. It features two main servers: 'server of main corps' and 'server of flying corps'. The 'server of main corps' is connected to an 'I/O' unit, a 'manager of idle PE for main corps', and multiple 'main corps processes' (represented by circles). Similarly, the 'server of flying corps' is connected to a 'manager of idle PE for flying corps' and multiple 'flying corps processes'. Bidirectional arrows indicate communication between the servers and their respective managers.</p>

Developing a computer Go playing system

Unlike checker and chess playing computer programs which have attained or are approaching the highest human skills, there have been no Go-playing programs that match average human Go-player's skills.

The difficulty of constructing a Go-playing program comes mainly from the fact that (1) the fanout of an average game tree is too large for brute force search to be feasible — the board of Go is 19×19 as compared to the chess board of 8×8 , and the player can put the next stone on almost any vacant board position, and (2) a simple and good board evaluation function does not exist — evaluation of a board configuration needs understanding of relative strengths of groups of stones, which involves pattern recognition. We have been developing a sequential computer Go playing system called "GOG" on the sequential inference machine since 1985. Currently, the system is stronger than an entry level human Go player, but considerably weaker than an average-level player.

There are a number of improvements (such as move knowledge of set moves, tactics, better board evaluation, etc) that could make the system stronger, but it would take much more processing time to incorporate them. Thus we started the development of a parallel Go-playing system which will be stronger than the sequential system but will retain the tolerable response time to make real-time play with humans possible.

It is a intermediate result of computer Go system "GOG" of parallel inference machine.

Move Making in the GOG System

The outline of the process in which the sequential GOG system determines its next moves comprises three stages.

Board Recognition

When human looks at the "Go" board, he does not see only the arrangement of stones on the board, he also sees their strategic meaning. We have abstracted five forms of stones which are thought to be recognized by human (Figure 1).

Candidate Move Generation

Based on board recognition, the system lists up plausible next moves together with heuristic value of those moves. Candidate moves include moves to enlarge a friendly territory, moves to limit an opponent territory, moves to capture an opponent string, moves to avoid capture of a friendly string, etc.

Next Move Decision

Finally, based on the proposed values of the candidate moves, the system decides on the next move, and plays it.

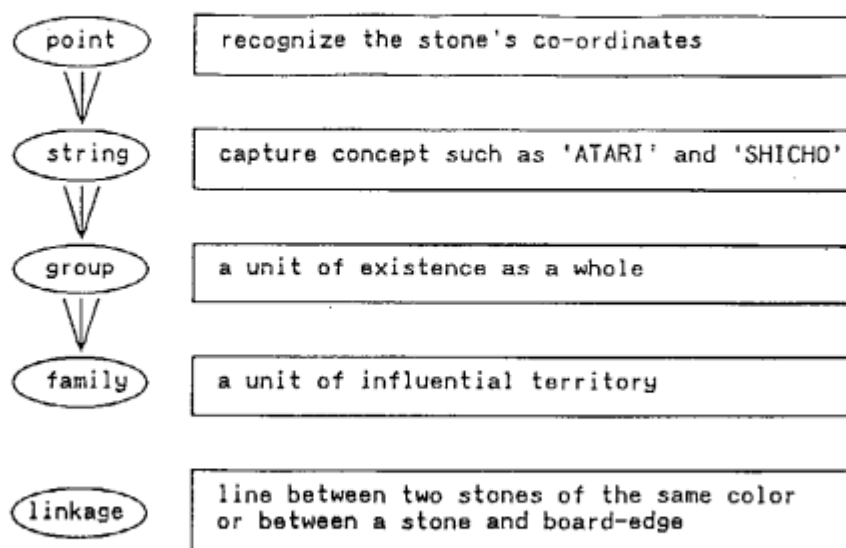


Figure 1. Data Structure

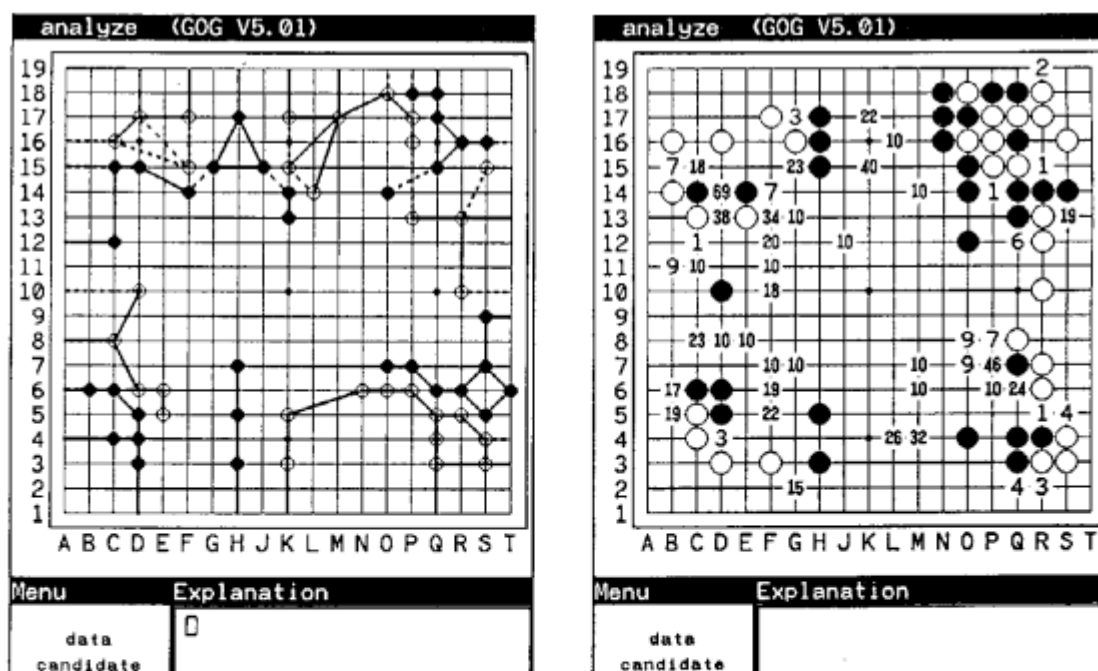


Figure 2. Linkage Data and Score Table

The Parallel GOG System

In the parallel system, one of the processors of the Multi-PSI serves as a manager processor, and the rest are worker processors.

After the system gets enemy's move, manager processor dispatches the tasks of recognition and candidate generation tasks to the worker processors. The results are sent to the master processor. Then manager processor decides the next move.

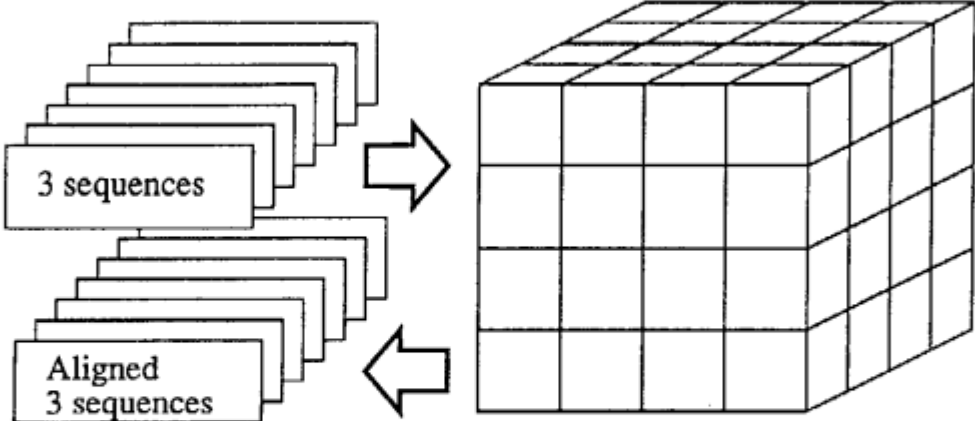
Flying Corps

To make the system considerably stronger while retaining the real-time response of the system, we proposed the concept of *flying corps*. The idea is to find out the possibilities of potentially large gain (such as capturing a large opponent group, invasion of a large opponent territory) or loss, and assign the investigation of those possibilities to *flying corps* processes.

The system which incorporates flying corps idea is consist of *main corps* processes and *flying corps* processes. Main corps treats necessary tasks to play Go and to keep strength standard level we can permit. A flying corps does the investigation independently from the immediate next move decision process, and it notifies the main corps when the investigation task is completed (that might be several moves from the initiation of the task). Note that flying corps keep on running, while the opponent is thinking of the next move. A flying corps may be aborted if it has become irrelevant or unimportant in the overall situation, or the local situation that motivated it has changed by some later move. Main corps processes have higher priority than flying corps processes. The time to decide next move depends only on main corps.

Appendix: Basics of Go

Go is a board game, and is popular in China, Korea and Japan. The board is a 19×19 grid. The two players are named black and white. The black and the white place a stone on a vacant intersection in turn (the black and the white place black and white stones, respectively). Each player tries to gain as much territory as possible (a black's (white) territory is a vacant area surrounded by black (white) stones). A group of solidly connected stones of the same color is captured when all adjacent positions are occupied by the opponent's stones. The adjacent positions that are vacant are called *dames*. When stones are captured, they are removed from the board and are added to the opponent's territory count. Thus, a player places stones as efficiently as possible to surround vacant area to maximize his/her territory. Inevitably, the black and white stones clash, which leads to compromise or fight for capturing the opponent's stones.

Title	<h1>Genetic Information Processing (1):</h1> <h2>Multiple Sequence Alignment by 3-Dimensional DP-matching</h2>
Purpose	<ul style="list-style-type: none"> • Parallel programming on a large-scale problem in KL1. • The first step to genome analysis.
Outline & Features	<p>[Outline]</p> <p>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.</p> <p>[Feature]</p> <ul style="list-style-type: none"> • Efficient DP-matching by parallel pipeline processing. • Quality improvement in three-sequence alignments.
System Configuration	 <p>3 sequences</p> <p>Aligned 3 sequences</p> <p>3D DP matcher made of KL1 process-network</p>

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-KIFSSFDCCKSGFWQVLLDQESRPLTAFT-----
---DIHPTVPNPNYLLSGLPPSHQWYTVLQDKDAFFCLRLHPTSQPLFAFEW-ROPEM
---L-FGPVQRGLPLLSALPQDWKLI-IIDIKDCFFSIPLYPRDRPRFAFTIPSLNHM
---P-FGAVQQGAPVLSALPRGWPLM-VLDLKDCCFFSIPLAEQDREAFATLPSVNNQ
---DLSSSSPGPPDL-SSLPTTLAHLQTLIDLRDAFFQIPLPKQFQPYFAFTVPQQCN
---TLTSPSPGPPDL-TSLPTALPHLQTLIDLRDAFFQIPLPKQYQPYFAFTIPQPCN
---PIPALSPGPPDL-TAIPTHPPHIICLDLKDCCFFQIPVEDRFRSYLSFTLPSGGGL
---D-FWEVQLGIPHPAGLKKKKSVT-VLDVGDAYFSYPLDEDFRKYTAFTIP SINNE
VHWPKF-AVPLQLTLANLLSTOLQWL-SLDVSAAFYHIPISPAAYPHLLVG-----
VSWPKF-AVPLQLSLTNLLSSNLSWL-SLDVSAAFYHIPLHPAAMPHLLVG-----
MRFPYR-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, ADIE 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).

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.

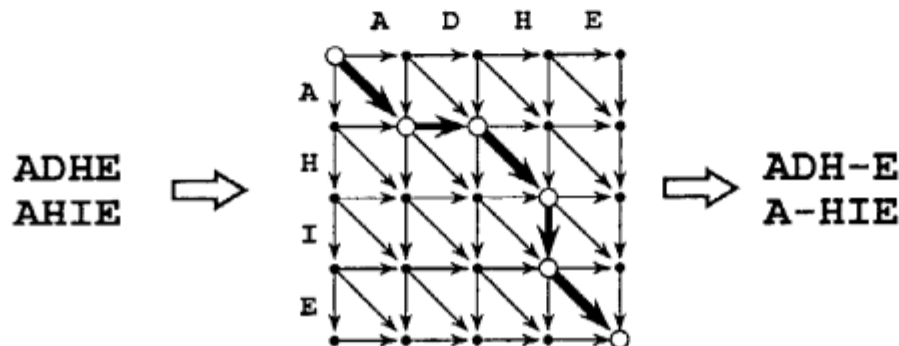


Figure 2.1 DP-matching method

	A	R	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	V	Y	B	Z	X
A	-2																				
R	2	-6																			
D	0	0	-2																		
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	-5													
H	1	-2	-2	-1	3	-3	-1	2	-6												
I	1	2	2	2	2	2	2	3	2	-5											
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	5	1	2	3	2	-2	-4	0	-6								
F	4	4	4	6	4	5	5	5	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				
V	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		
B	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	-3
X	0	1	0	1	3	1	1	1	1	1	1	1	1	2	1	0	0	4	2	1	0

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.

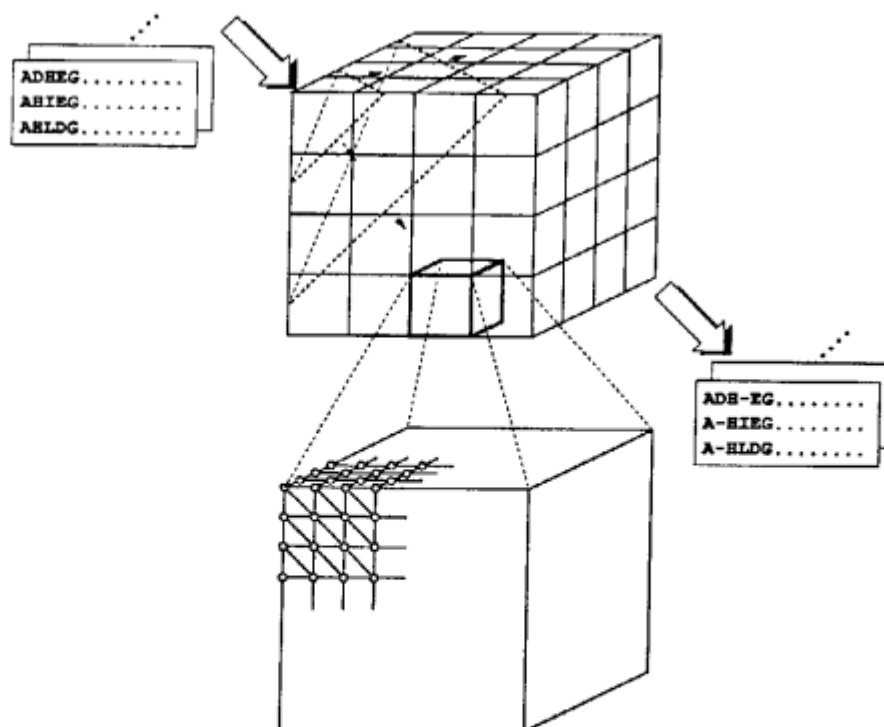
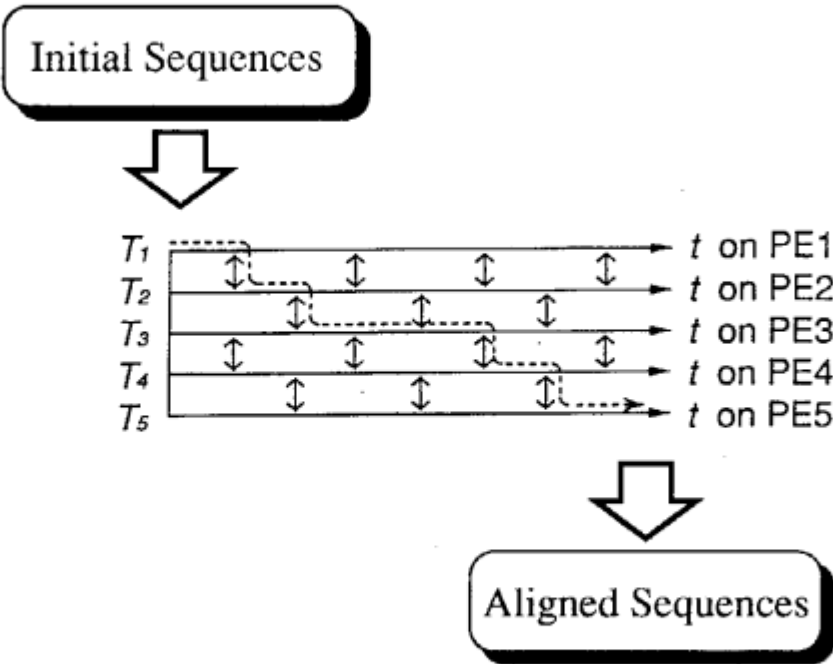


Figure 3 3-dimensional DP-matching

Title	<h1>Genetic Information Processing (2):</h1> <h2>Multiple Sequence Alignment by Parallel Simulated Annealing</h2>
Purpose	<ul style="list-style-type: none"> • Application of a parallel simulated annealing to a practical problem. • The first step to genome analysis.
Outline & Features	<p>[Outline]</p> <p>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.</p> <p>[Feature]</p> <ul style="list-style-type: none"> • 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
   $X_0 :=$  Initial solution ;
   $\{T_n\}_{n=0, \dots, N-1} :=$  Temperature (Cooling schedule);
  for  $n := 0$  to  $N-1$  do
    begin
       $X'_n :=$  Some random neighboring solution of  $X_n$ ;
       $\Delta E := E(X'_n) - E(X_n)$ ;
      if  $\Delta E < 0$  then
         $X_{n+1} := X'_n$ 
      else
        if  $\exp(-\Delta E/T_n) \geq \text{random}(0,1)$  then
           $X_{n+1} := X'_n$ 
        else
           $X_{n+1} := X_n$ 
      end;
    end;
  Output  $X_N$ ;
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.

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).

```

"-----NAPATFQRCMNDILRPLLKHCLVFSTSLD-----"
"-----LKQAPSIFQRHMDEAFRVFRKFCCVFSNNE-----"
"-----NSPTLFDEALHRDLADFRIQHPDLILLQAA-----"
"-----MANSPTICQLYVQEALEPIRKQFTSLIVIH-----"
"-----TCSPTICQLVVGQVLEPLRLKHPSLCMLHA-----"
"-----SPTLFEMQLAHILQPIRQAFPPQCTILQASP-----"

```

Figure 2.1 An initial alignment

```

"-----NAPATFQRCMNDILRPLLKHCLVFSTSLD-----"
"-----LKQAPSIFQRHMDEAFRVFRKFCCVFSNNE-----"
"-----NSPTLFDEALHRDLADFRIQHPDLILLQAA-----"
"-----MANSPTICQLYVQEALEPIRKQFTSLIVIH-----"
"-----TCSPTICQLVVGQVLEPLRLKHPSLCMLHA-----"
"-----SPTLFEMQLAHILQPIRQAFPPQCTILQASP-----"

```

Figure 2.2 An alignment after the first move

```

"-----NAPATFQ--RCM-NDIL--RPLLKHCLVFSTSLD----"
"----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.

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.

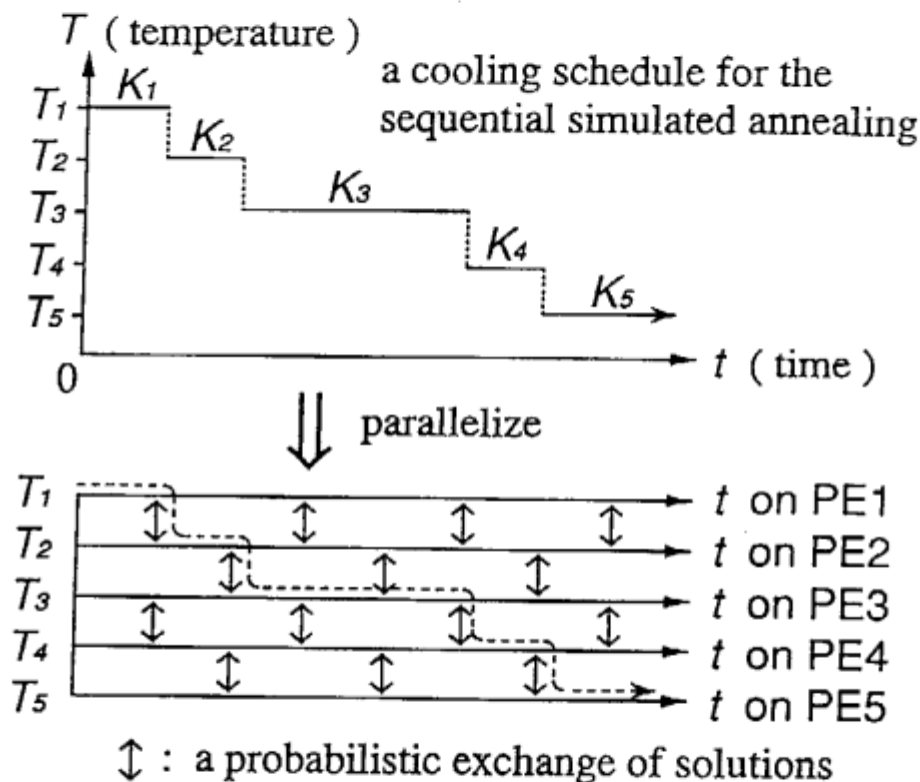


Figure 3 Scheduleless parallel simulated annealing

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.

MGTP: Model Generation Theorem Prover - Advanced Inference Engine for AI Systems -

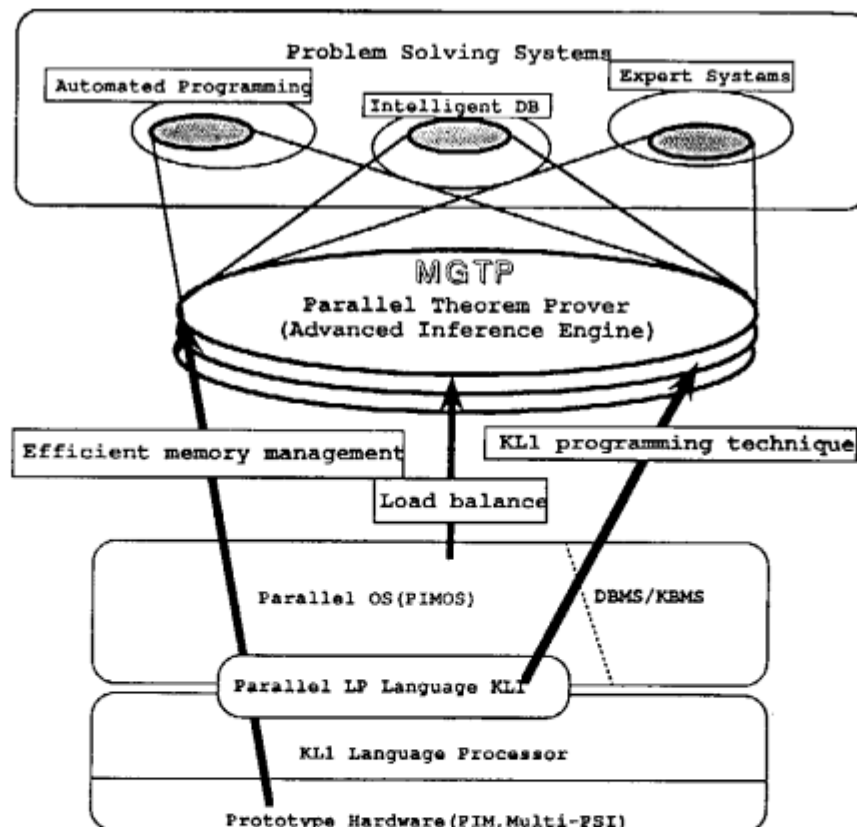
AIMS:

The goal of our research is to build a parallel automated reasoning system on a parallel inference machine, PIM, using KL1 and PIMOS technologies. The MGTP prover, currently being developed, adopts a model generation method. In developing MGTP, we aim to achieve the following:

- (a) Combine logic programming and automated reasoning to investigate an efficient first-order theorem prover.
- (b) Offer an advanced inference engine that can be applied to fields such as natural language processing, intelligent database systems, automated programming, expert systems.

APPROACH:

- Development of clause compiling techniques and meta-programming utilities to implement an efficient prover in KL1.
- Development of a support environment to make it easy to use and develop MGTP.
- Development of an automated programming system to demonstrate how programs can be derived automatically using MGTP.



MGTP Relative to Fifth Generation Computing Systems

Two Versions of MGTP

	Ground MGTP	Non-Ground MGTP
Application	Ex. Database problems	Ex. Mathematical theorems
Variable Representation	Direct use of KL1 variables	Ground term representation
Programming Techniques	<ul style="list-style-type: none"> -Translating given clauses to KL1 clauses -Efficient coding using head unification 	<ul style="list-style-type: none"> -Interpreting a given set of clauses -Using "meta-library" Ex.Unification with occurs check
Other Implementation Techniques <ul style="list-style-type: none"> - Avoiding redundancy in conjunctive matching using RAMS(Ramified Stack) and MERC(Multi-Entry Repeated Combination) - Term indexing - OR-parallelization for Non-Horn problems - AND-parallelization for Horn problems 		

Examples:

(a) Party Problem (Ground MGTP)

-Non-Horn and finite domain problem

We can always choose 3 persons who are either familiar with each other or not familiar with each other, from 6 persons who meet at a party.



```

true --> dom(1),dom(2),dom(3),dom(4),dom(5),dom(6).

dom(X),dom(Y),X>Y --> f(X,Y);nf(X,Y).

f(X1,X2),f(X2,X3),f(X3,X1) --> false.
nf(X1,X2),nf(X2,X3),nf(X3,X1) --> false.

```

(b) Overbeek's Problems (Non-Ground MGTP)

-Horn and infinite domain problem

-Group theory, ring theory, implicational logic

e.g.

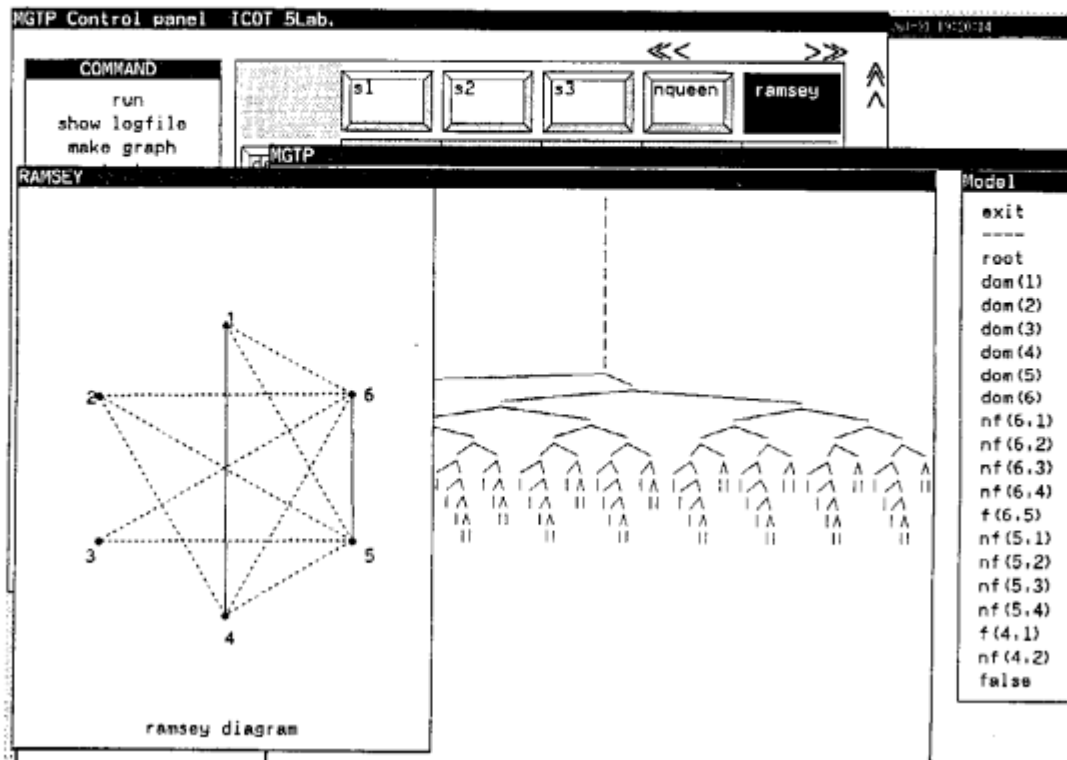
```

p(e(X,Y),pX) --> p(Y).
p(e(e(a,e(b,c)),c),e(b,a))) --> false.
true --> p(e(A,e(B,e(C,A)),e(C,B))).

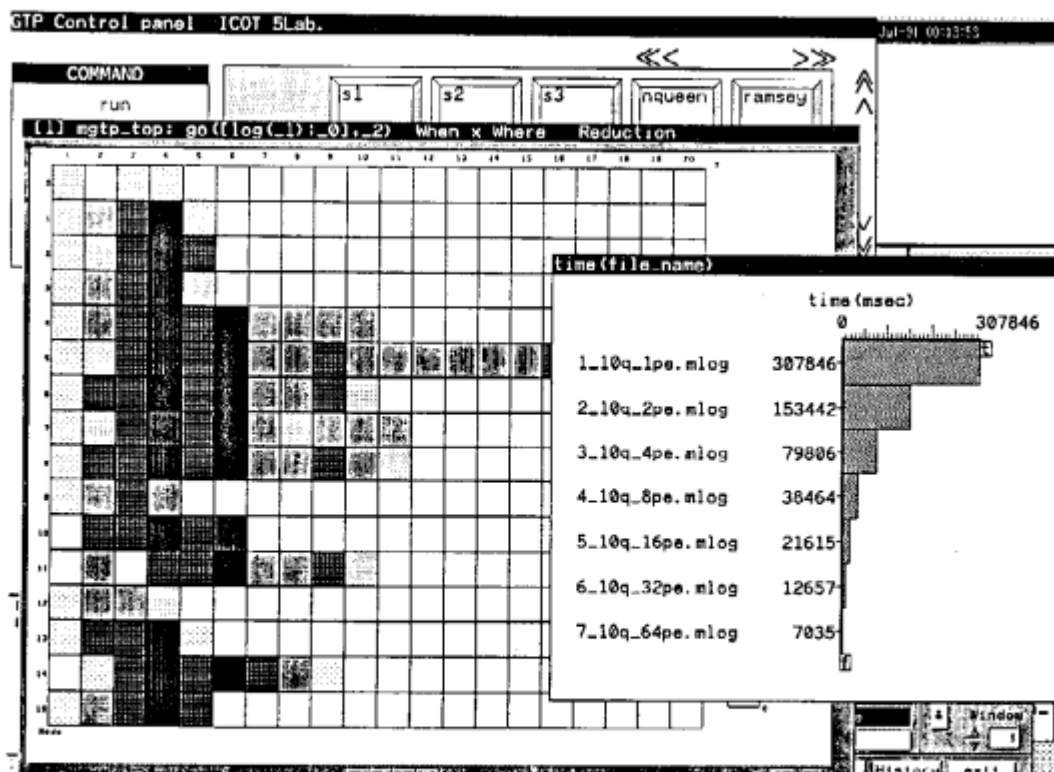
```

Demonstration (1) -MGTP and its utilities-

(a) Proof tree viewer

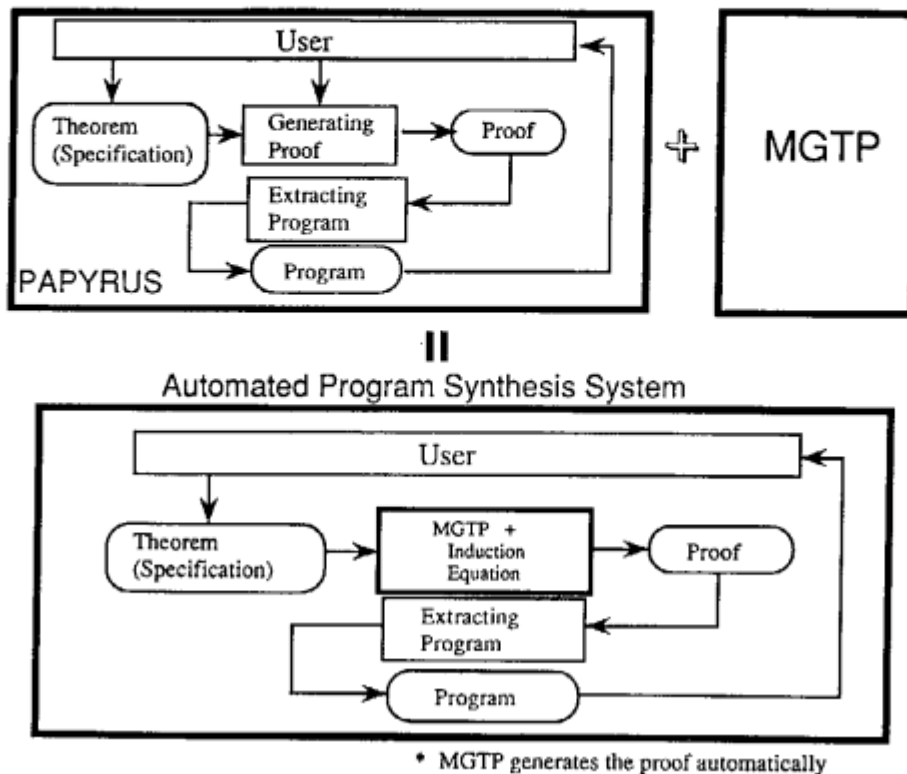


(b) Statistics analyzer for load balancing



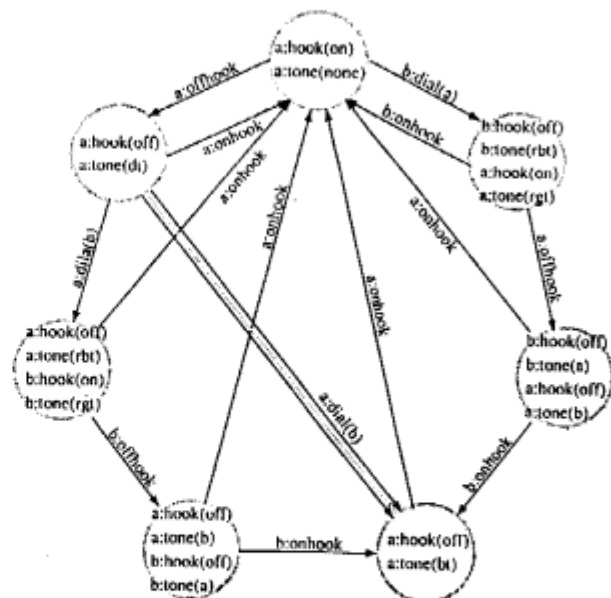
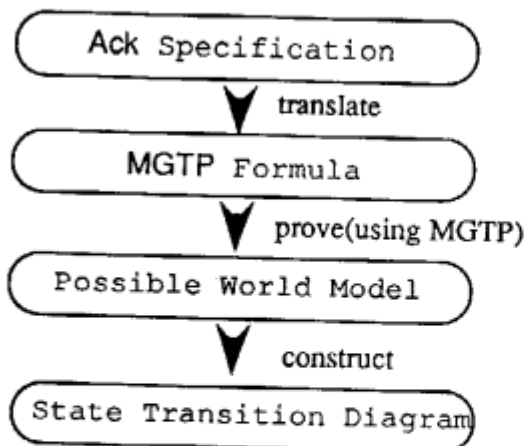
Demonstration (2) -Application Systems-

(a) PAPHYRUS + MGTP = Automated Programming System



(b) Specification Description Language for Telecommunication Protocol :Ack

Synthesis of Protocol using MGTP



EUODHILOS:

A General Reasoning System for a Variety of Logics

Purpose

EUODHILOS (Every Universe Of Discourse Has Its Logical Structure) is a general-purpose reasoning assistant system that allows users to interactively define the syntax and inference rules of a formal system and to construct proofs in the defined system.

Basic Features

(1) Formal system description language

(a) Language system (symbols, terms, formulas, etc.)

- ◆ Definite clause grammar formalism augmented with special constructs to handle variable binding, scope, etc.
- ◆ Automatic generation of a bottom-up parser, an unparser and internal structures of expressions

(b) Derivation system (axioms, inference rules, etc.)

- ◆ Axioms: a list of formulas
- ◆ Inference rules: natural deduction style presentation, and automatic method to check the side conditions
- ◆ Derived rules: definable if they are justified for validity
- ◆ Rewriting rules: definition by a pair of forms before and after rewriting, and applicable up to the given number of rewriting

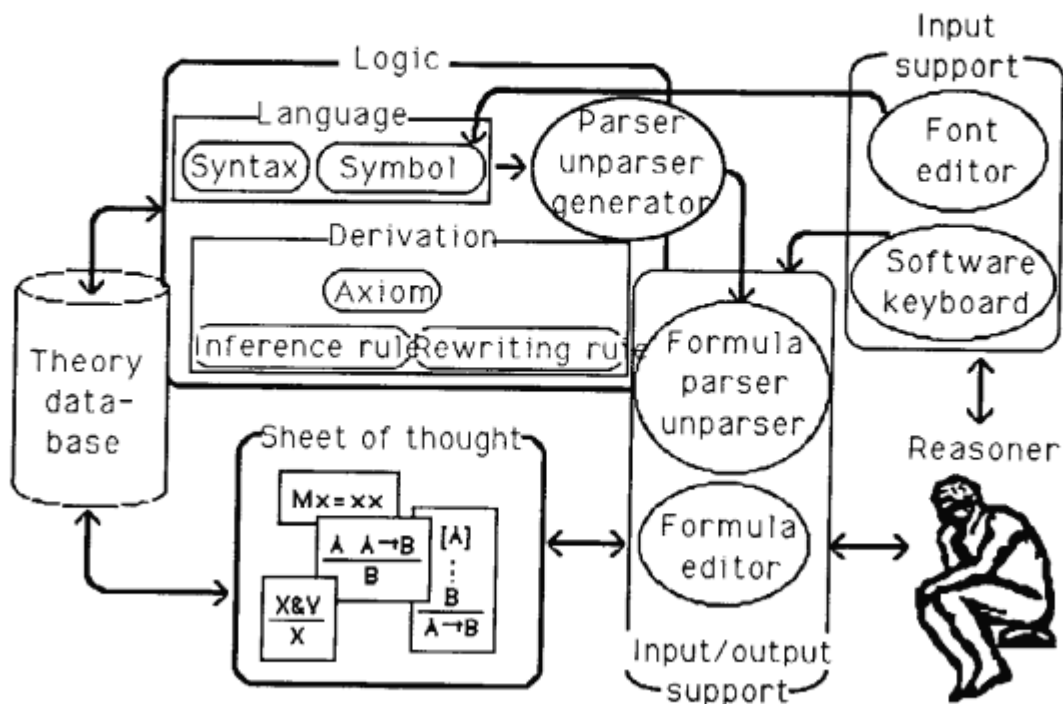
(2) Proof construction facilities

- ◆ Sheets of thought: a field of thought where we are allowed to compose a proof from its fragments, to separate a proof, etc.
- ◆ Proving methodology based on several sheets of thought: forward reasoning, backward reasoning, reasoning in a mixture of them, reasoning by lemma/derived rules, schematic proof, etc.
- ◆ Tree-form proof with justifications indicated in the right margin
- ◆ Interface with automated theorem provers/term rewriting systems

(3) Visual human-computer interface for reasoning and utilities

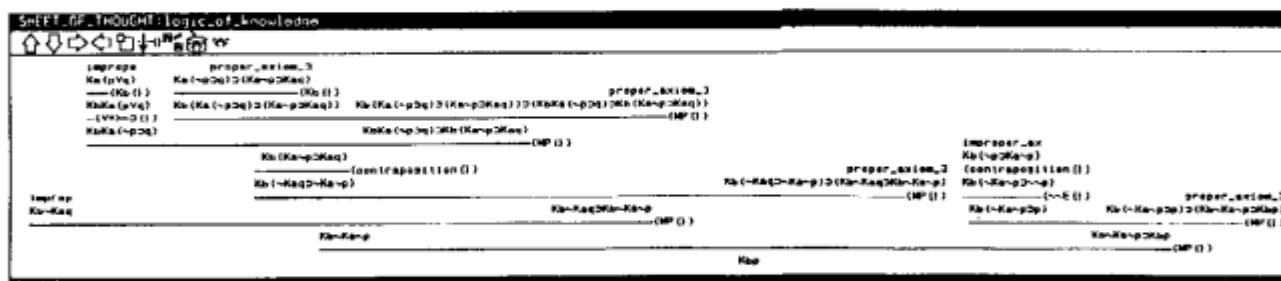
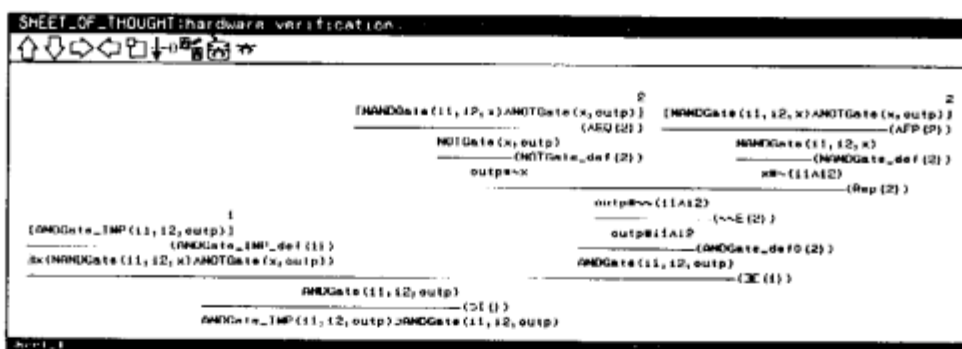
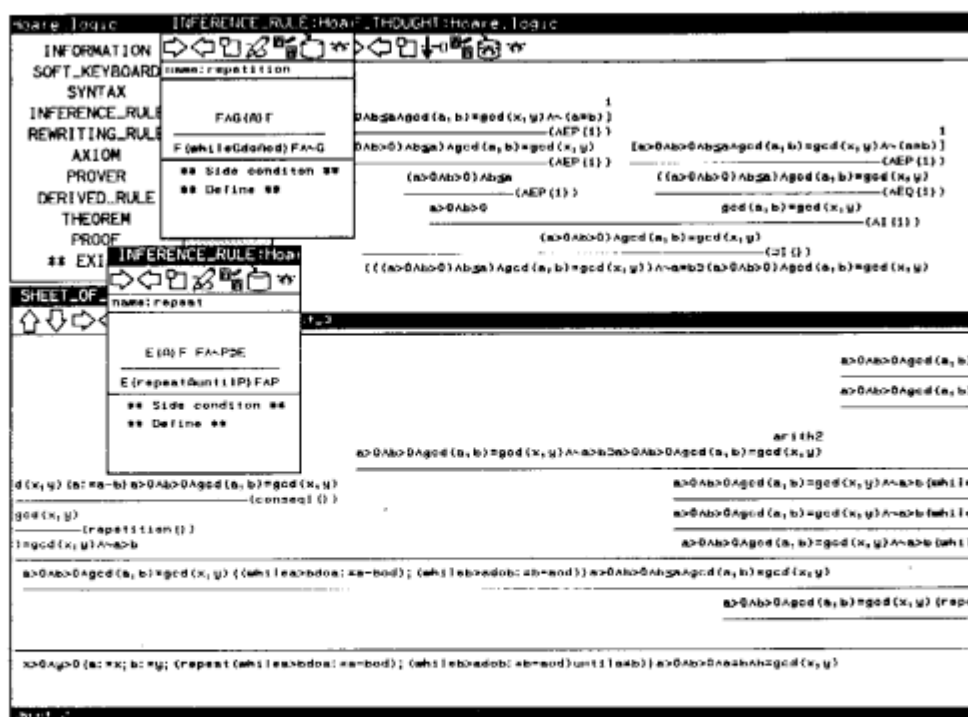
- ◆ Formula editor, font editor, software keyboard, debugger for defined syntax, stationery for reasoning

System Configuration



Logical Systems and Proof Examples Implemented in EUODHILOS

- (a) First-order logic (NK): various pure logical formulas, the unsolvability of the halting problem, an inductive proof, hardware verification (see Fig.2) and category theory
- (b) Second-order logic: the equivalence between the principle of mathematical induction and the principle of complete induction
- (c) Propositional modal logic: modal reasoning about programs
- (d) Intensional logic: the reflective proof of a metatheorem and Montague's semantics of natural language
- (e) Martin-Löf's intuitionistic type theory: constructive proofs
- (f) Hoare logic and dynamic logic: reasoning about program properties (see Fig.1)
- (g) General logic (see Fig.4),
- (h) Relevant logic
- (i) A logic of knowledge (see Fig.3).



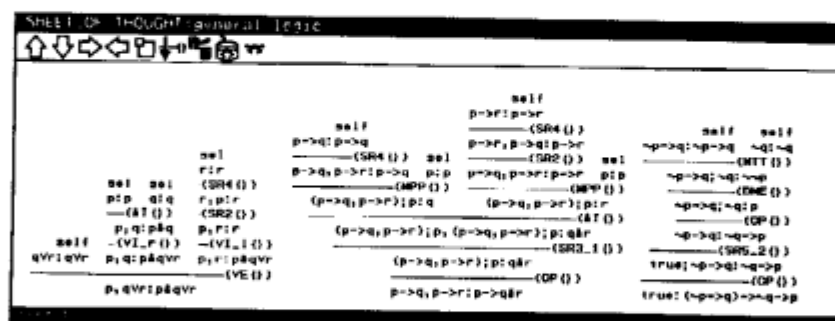
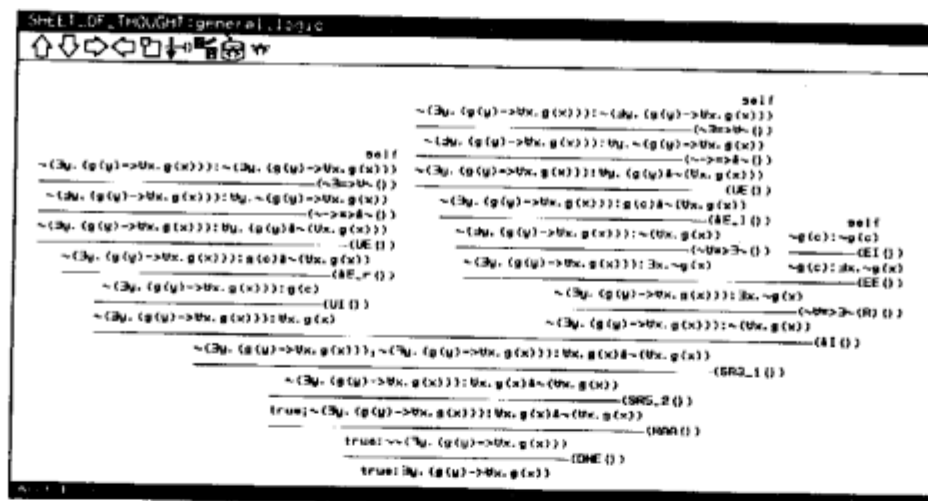
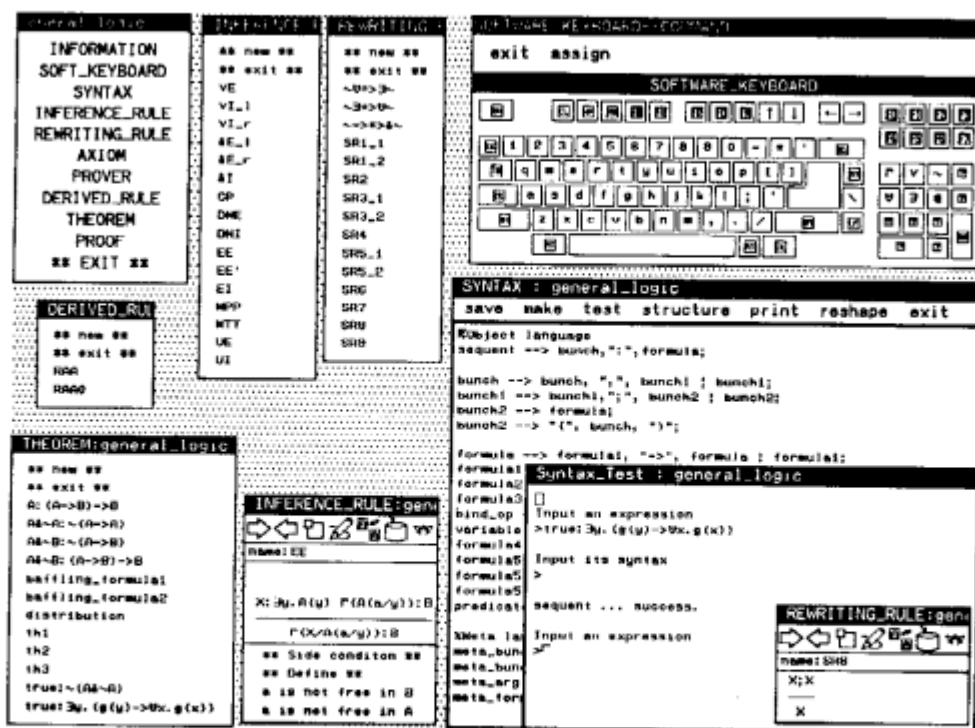


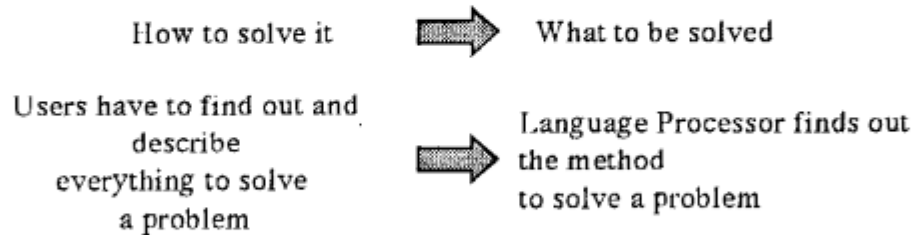
Fig. 4 Some proofs in general logic

Constraint Logic Programming Languages

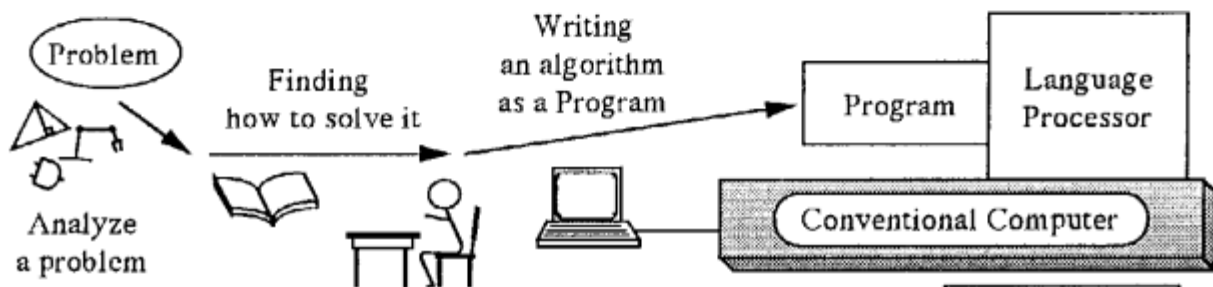
CAL to GDCC

Institute for New Generation Computer Technology

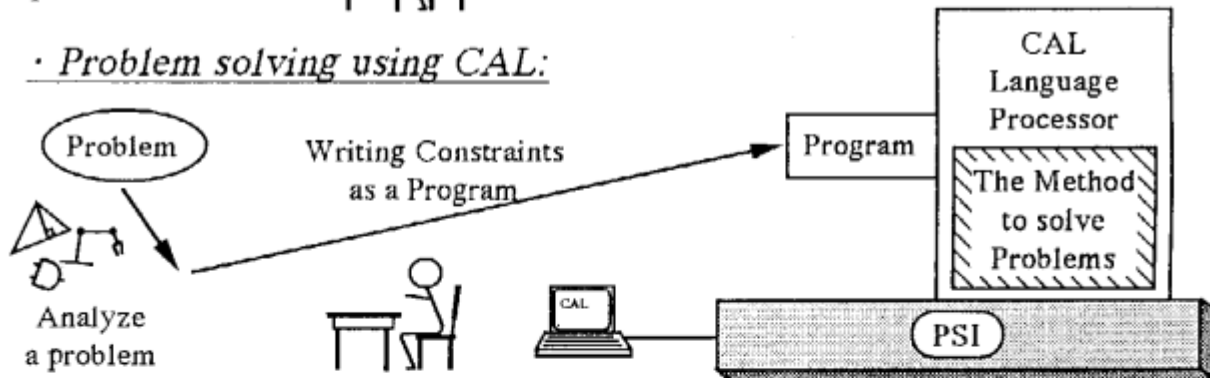
◆ Constraint Logic Programming will change Programming:



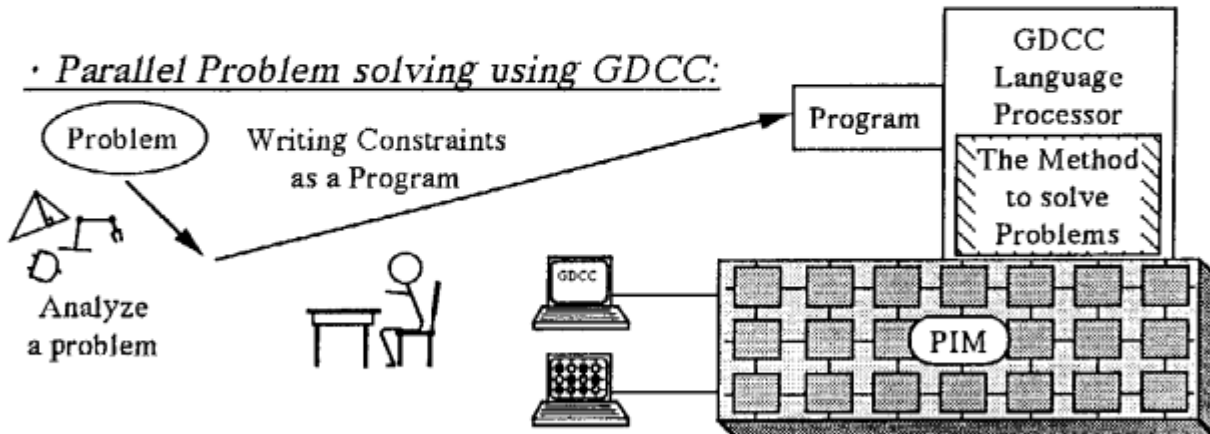
· Problem solving using conventional Programming Languages:



· Problem solving using CAL:



· Parallel Problem solving using GDCC:



(1) Heron's Formula (Algebraic Constraints-1)

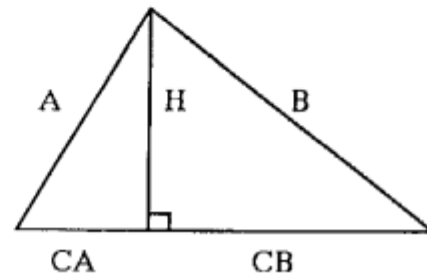
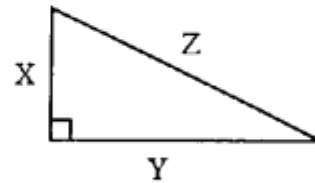
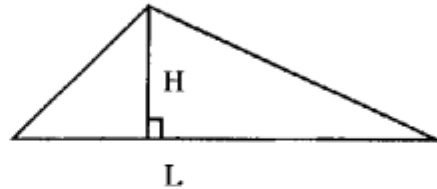
- Solving non-linear equations
- Producing a new relation from known relations
- Finding real roots of uni-variate equation

```
:- public triangle/3.
```

```
surface(H, L, S) :- alg:L * H = 2 * S.
```

```
pythagoras(X, Y, Z) :- alg:X^2 + Y^2 = Z^2.
```

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



CAL ver2.10 (solver: alg)

```
?- alg:pre(s,10), heron:triangle(3, 4, 5, s),  
    alg:get_result(eq, 1, nonlin, R), alg:find(R,Sol).
```

```
R=[s^2=36].
```

```
Sol=[s=real(-, [5121, 7921, 1030], [9184, 7986, 171])]  
    =[s=-6.000000099].
```

```
s^2 = 3.6e1
```

```
?
```

```
R=[s^2=36].
```

```
Sol=[s=real(+, [5121, 7921, 1030], [9184, 7986, 171])]  
    =[s=6.000000099].
```

```
s^2 = 3.6e1
```

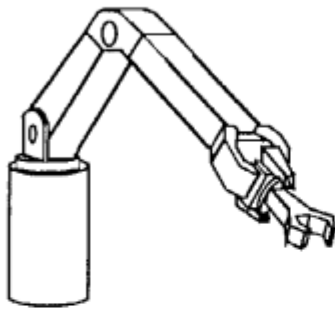
```
?
```

```
983msec
```

```
yes
```

```
?-
```

(2) Handling Robot Kinematics (Algebraic Constraints-2)



◆ Direct Kinematics:

- Rotation angle of each joint
- Length of each arm



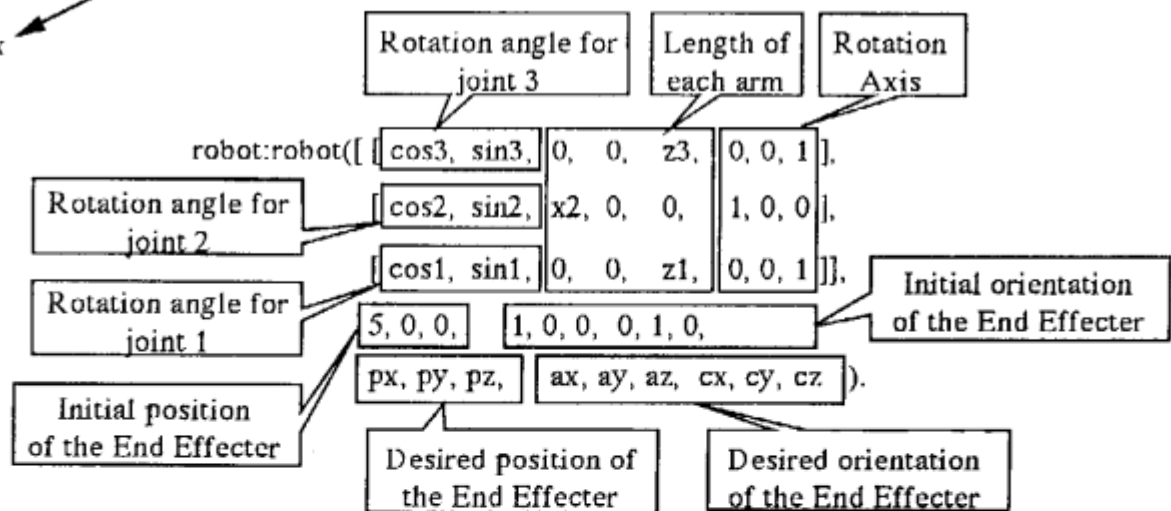
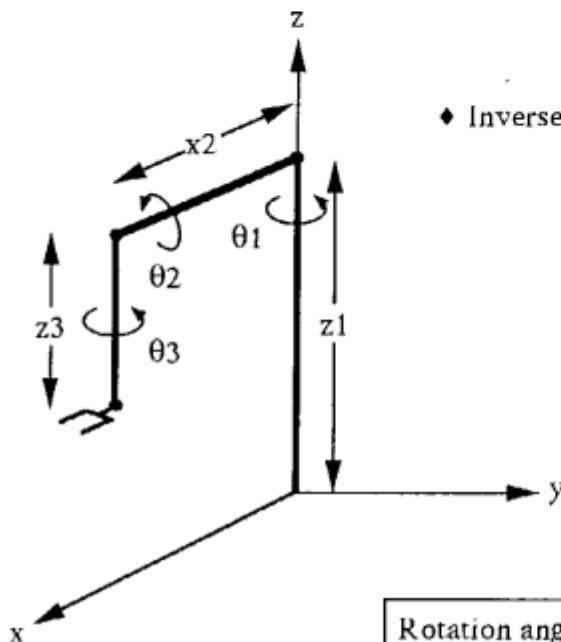
- Position and Orientation of End Effector

◆ Inverse Kinematics:

- Position and Orientation of End Effector



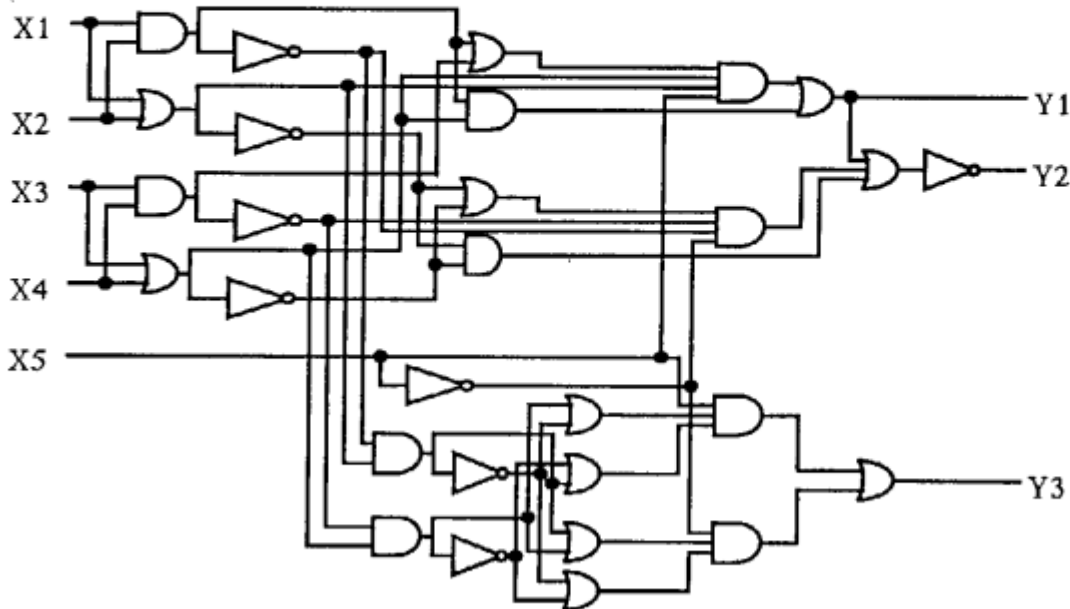
- Rotation angle of each joint
- Length of each arm



This computes necessary rotation angle of each joint and necessary length of each arm to move the end effector to the position (px, py, pz) with the orientation represented by two unit vectors (ax, ay, az), and (cx, cy, cz), which are perpendicular each other.

(3) Count one's (Boolean Constraints)

- Solving Boolean constraints
- Showing no input-output restrictions
- Producing general relationship among parameters



(4) Summary

- ◆ By using CAL/GDCC,
 - Programming will change from "How" to "What"
 - ← All users have to do is describing the problem by stating relations (constraints) which are hold among components of a problem
 - The language processor applies the method to solve the problem automatically
- ◆ We are now
 - developing Parallel "Handling Robot Design Support System"
 - developing various parallel systems to solve various constraints
 - Non-linear equations (done)
 - Boolean equations
 - Linear equations/inequalities
 - Building very high-level programming system for parallel AI-language based on Constraint Logic Programming

Kappa with Molecular Biological Data

–Nested Relational DBMS as a Knowledge Base Engine

Summary

Purpose

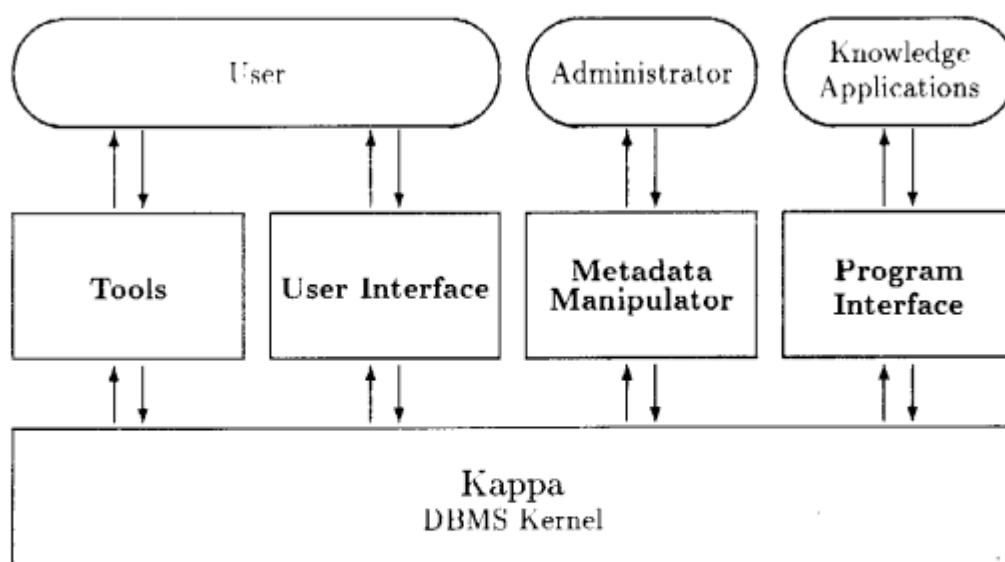
Kappa is an advanced database management system on PSI and PIM. It plays the role of an engine of our knowledge base management systems for various knowledge information systems such as genetic information process systems with molecular biological databases, legal reasoning systems with legal databases, and natural language processing systems. We show the Kappa system with a protein database: PIR and some of its tools.

Outline

Kappa is a DBMS with the following features:

- (1) Powerful modeling capability of a nested relational model
- (2) Efficient processing as a knowledge base engine
- (3) Flexible data structure: list, bag, variable length string and term
- (4) A user interface tuned for the nested relational model
- (5) Two level program interfaces: primitive commands and extended relational algebra
- (6) Customizable program interface

System Configuration



Demonstration

We implement GenBank and PIR in Kappa. We show the Kappa system with PIR and some of its tools.

(1) Metadata Manipulator

PIR database consists of three tables: `pir_gen`, `pir_ref`, and `pir_fea`. We show schema of each table with the metadata manipulator.

(2) Kappa User Interface

Kappa provide a terminal interface like a spread sheet to manipulate nested relations interactively. We show three tables: `pir_gen`, `pir_ref`, and `pir_fea`, and retrieve some records from them.

(3) Primitive Commands

Primitive commands are one kind of program interface. We retrieve records whose reference Dr. Matsubara wrote, by primitive commands.

$$\pi_{id_code}(\sigma_{authors='Matsubara*'}(pir_ref)) \bowtie pir_gen$$

(4) Feature Expression

Feature expression is one of the tools for PIR in Kappa. The tool works on X/UNIX, retrieves records from PIR in Kappa, and shows features of regions.

1 Kappa

Kappa¹ 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 processed.
- (3) A user interface tuned for the nested relational model is provided.
- (4) ESP program interface and extended relational algebra are provided.
- (5) Program interface can be customized for each application.

2 Nested Relational Model

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

$$NR \subseteq E_1 \times \dots \times E_n$$

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

where D_i is a domain, and NR is a nested relation, while a relation R is defined as:

¹Knowledge APplication-oriented Advanced database management system

$$R \subseteq D_1 \times \dots \times D_n$$

Relation:

Tour Schedule			Group	
Date	To	Group	Name	Member
6.4.90	ANL	Setting_G	Setting_G	Sugino
6.25.90	ANL	Lecture_G	Lecture_G	Ichiyoshi
			Lecture_G	Kondo
			Lecture_G	Susaki

Nested Relation:

Tour Schedule			
Date	To	Group	
		Name	Member
6.4.90	ANL	Setting_G	Sugino
6.25.90	ANL	Lecture_G	Ichiyoshi
			Kondo
			Susaki

3 Molecular Biological Databases

3.1 GenBank

GenBank² is a public database which has many fragments of nucleic acid sequences (DNA and RNA). GenBank, EMBL³ and DDBJ⁴ have been created by agreements on dividing tasks of data collection and exchanging collected data.

Each fragment (or locus) in GenBank consists of the definition, the related bibliographic information, features of the regions in it and their positions, and DNA/RNA sequence.

3.2 PIR

PIR⁵ is the representative public database of amino acid sequences of proteins.

Each entry in PIR consists of the definition, the related bibliographic information, features, and amino acid sequence.

4 Molecular Biological Databases in Kappa

We implement GenBank and PIR in Kappa. These data has complex structure, and contains large sequence data. It is difficult to implement these data in a relational database.

Advantages of Kappa are followings:

- Reducing the number of relations

Because an nested relation can expresses multivalued dependencies into itself,

²Genetic Sequence Data Bank, IntelliGenetics Inc. and Los Alamos National Laboratory, US

³Nucleotide Sequence Data Library, European Molecular Biology Laboratory, EC

⁴DNA Data Base of Japan

⁵Protein Information Resource, National Biomedical Research Foundation, US

a designer can reduce the number of relations. From a processing point of view, the number of join operation, which needs many processing power, can be reduced. From a user's point of view, it is easy to understand the overall schema because the schema is not partitioned unnaturally.

- Variable length string

It is possible to reduce storage size by suporting variable length string.

4.1 GenBank

(1) Schema

We design four tables to implement GenBank in Kappa.

gene : main table, each record of which has a locus name, the definition, accessions, keywords, identifiers to the other tables, and so on.

reference : table with references, each of which has authors, a titles, a journal in which the paper was published, and so on.

feature : table with regions of a sequence and those features.

seqdata : table with sequence data represented in string form.

(2) Storage Size

GenBank released 60.0 (6.15.89),26323 entries, 32 M bases.

Flat file (in PSI)	Kappa		
	Record	Index	Total
143M bytes	130M bytes	45M bytes (19indexes)	175M bytes

4.2 PIR

(1) Schema

We design three tables to implement PIR in Kappa.

pir-gen : main table, each record of which includes a name, a placement, sources and sequence data.

pir.ref : table with references, each of which has a author, a title, a journal.

pir_fea : table with regions of a sequence and those features.

(2) Storage Size

PIR released 25.0 (6.30.90),17731 entries, 5.0 M residues.

Flat file (in PSI)	Kappa		
	Record	Index	Total
44M bytes	34M bytes	16M bytes (34indexes)	50M bytes



International Conference on Fifth Generation Computer Systems

Date: June 1-5, 1992
Venue: Tokyo Prince Hotel, Tokyo, Japan
Secretariat: Institute for New Generation Computer
Technology (ICOT)
Mita Kokusai Bldg. 21F
4-28 Mita 1-chome, Minato-ku
Tokyo 108, Japan
phone: +81-3-3456-3195
fax: +81-3-3456-1618
e-mail: fgcs92@icot.or.jp
telex: ICOT J32964

ICOT

Institute for New Generation Computer Technology

Mita Kokusai Bldg. 21F
4-28 Mita 1-chome, Minato-ku, Tokyo 108, Japan
phone: +81-3-3456-3195
fax: +81-3-3456-1618
telex: ICOT J32964