Liquid Brain: Kinetic Model of Structureless Parallelism

Alexander N. Gorban, Katya O. Gorbunova
Institute of Computational Modelling SB RAS
660036, Akademgorodok, Krasnoyarsk-36, Russia
E-mail: gorban@cc.krascience.rssi.ru

Abstract

A new formal model of parallel computations – the Kirdin kinetic machine – is suggested. It is expected that this model will play the role for parallel computations similar to Markov normal algorithms, Kolmogorov and Turing machine or Post schemes for sequential computations. The basic ways in which computations are realized are described; correctness of the elementary programs for the Kirdin kinetic machine is investigated. It is proved that the determined Kirdin kinetic machine is an effective calculator. A simple application of the Kirdin kinetic machine – heap encoding – is suggested. Subprograms similar to usual programming enlarge the Kirdin kinetic machine.

Keywords: parallel computations, fine-grained parallelism, distributed computing

1 Introduction

The problem of effective programming with fine-grained parallelism is far from being solved. It seems that, despite numerous efforts, we have not yet understood parallel computations, considering them mainly as result of usual algorithm parallelization.

Long ago sank into oblivion the naive idea: let take more processors, and the efficiency of computer will increase proportionally. There is a well known "Minsky hypothesis" (fig. 1): efficiency of a parallel system increases (approximately) proportionally to logarithm of the number of processors; at least, it is a convex upward (i.e. concave) function.

Nowadays more and more often as a way to overcome this limitation the following approach is used: for different classes of problems maximum parallel algorithms of solution are constructed which use some abstract architecture (paradigm) of fine grain parallelism, and for concrete parallel computers means are developed for realization of parallel processes of a given abstract architecture (fig. 2). As a result, an efficient technique of production of parallel programs is created.

There are some promising approaches based on models of computing environments constructed from large number of elementary calculators of the same type (neural networks, cellular automata etc.). If it is possible to implement a problem in such environment (for example, by methods of neural networks training [7]), further

realization with parallel computers can be easily constructed within the framework of the ideas "similar tasks for different elements".

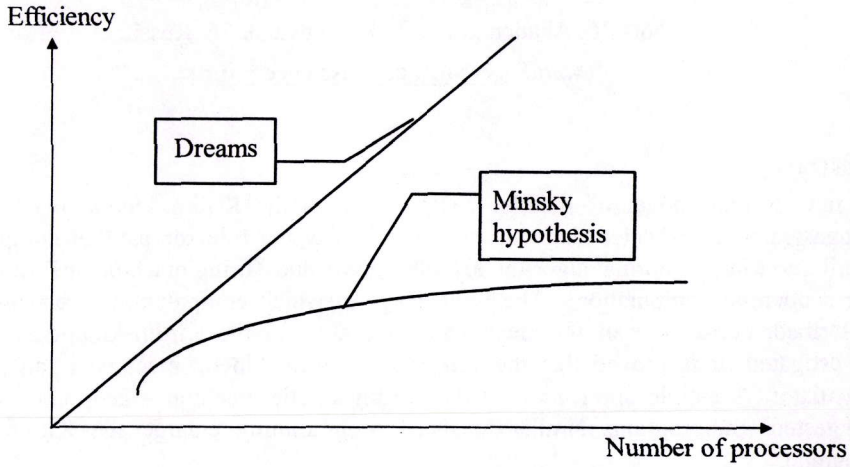

Fig. 1: Minsky hypothesis.

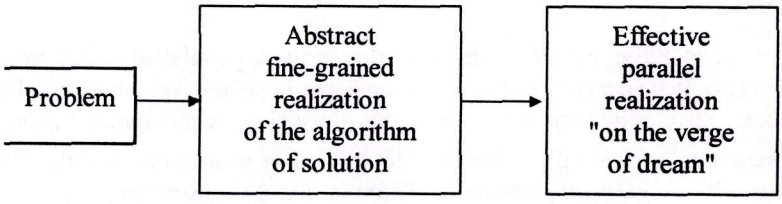

Fig. 2: Way to effective parallelism.

There are other perspective ideas and approaches to construction of models of fine-grained parallelism besides neural networks.

Parallel Substitution Algorithms (PSA) [9] conceptually go back to von Neumann cellular automata, but have more powerful expressive capabilities. PSA are capable of processing multidimensional data arrays that are represented as a set of cells. Based on PSA concepts a theory has been developed which comprises the correctness conditions, equivalent transformations and a number of methods for algorithm and architecture synthesis. A computer simulation system allows constructing cellular algorithms and observing computation processes in dynamics.

The chemical computer (SCAM – *Statistic Cellular Automata Machine*) is offered in [8] in development of the cellular automata theory. SCAM is based on imitation simulation by Monte-Carlo methods of a class of heterogeneous chemical reactions occurring in a thin layer of molecules adsorbed on the surface of a crystal catalyst. Algorithmic universality for SCAM has been proved.

Artificial Immune Systems [10] are highly distributed systems based on the principles of natural system. This is a new and rapidly growing field offering powerful and robust information processing capabilities for solving complex problems. Like artificial neural networks, artificial immune systems can learn new information, recall previously learned information, and perform pattern recognition in a highly decentralized fashion.

A new abstract model of computations – *the Kirdin kinetic machine* – is investigated in this paper. This model is expected to play the same role for parallel computations as the Turing machine and other abstract algorithmic calculators for sequential computations. The Kirdin kinetic machine is based on chemical reactions in liquids or gases. Our optimistic expectations go back to the theorem of M.D.Korzuhin [12] on chemical reactions ability to imitate any dynamic system for finite times and to the theorem of A.N.Gorban [13] on chemical systems approximating any dynamic systems.

2 The Kirdin Kinetic Machine

A processed unit for the Kirdin kinetic machine is *an ensemble* of strings M from the alphabet L, which is identified with a function F_M taking non-negative integer values: $F_M: L^* \to N \cup \{0\}$. The value of $F_M(s)$ is interpreted as a number of copies of a string s in the ensemble M.

The processing consists of an aggregate of *elementary events*, which occur non-deterministically and in parallel. An elementary event $S: M \mapsto M'$ means that from the ensemble M an ensemble K^- is removed and an ensemble K^+ is added. The ensembles K^- and K^+ are unambiguously set by *rules* or *commands*, which are combined in *a program*. The commands can be of only three kinds (u, w – arbitrary, v, f, g, k, q, s are fixed):

Disintegration $uvw \to uf + gw$ (1)

Synthesis $uk + qw \to usw$ (2)

Replacement $uvw \to usw$ (3)

The program $\boldsymbol{P}$ *is applicable* to an ensemble M, if any command of $\boldsymbol{P}$ is applicable to M. Elementary event S is unambiguously determined by a rule p from the list of commands of the program $\boldsymbol{P}$, and by an ensemble K^- determined by this rule and such that $F_{K^-}(s) \le F_M(s)$ for any s. An elementary event S *is allowable* for an ensemble M and a program $\boldsymbol{P}$ if there is a rule p in the list of commands of the program $\boldsymbol{P}$ and the values of function F_M for strings in the ieft part of this rule are positive.

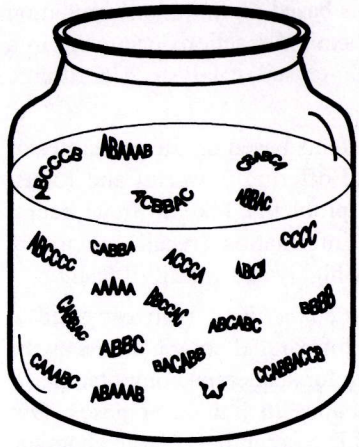

Fig. 3: A jar with strings.

Let's say that N of allowable events *are compatible*, if $F_M - \sum_{i=1}^{n} F_i^- \geq 0$, where F_i^- is a removed ensemble for i-th event.

Ensemble M is called *a final ensemble* if no command of the program $\boldsymbol{P}$ is applicable to it. $\boldsymbol{P}$ is referred to as *a finite program* if application of commands of the program to the initial ensemble always leads to a final ensemble. If all final ensembles coincide, the program $\boldsymbol{P}$ is named *deterministic* for the ensemble M.

The Kirdin kinetic machine can be informally described as a jar with strings (see figure 3). We add rules-catalysts to this jar, some of them, colliding with the strings, promote their disintegration, others, meeting a pair of suitable strings, promote their synthesis, and the third replace some subchains in the strings.

The basic ways of realization of calculations are described; *correctness* for the elementary programs for the Kirdin kinetic machine is investigated.

3 Statistical Method of Implementation

The "statistical" method of implementation of the Kirdin kinetic machine is of particular interest. In the limit of large ensembles it gives kinetic behavior, similar to the behavior of a complex system of chemical reactions (wherefrom, in fact, the name "the Kirdin kinetic machine" comes from). Some non-negative number r_i – "constant of rate" is compared with each i-th command in statistical realization. The realization can be presented as follows: for small Δt with the probability $r_i \Delta t$ one of commands-reactions is chosen, and with probability $1-\Sigma_i r_i \Delta t$ empty command is chosen (nothing occurs); the probability of choice of two or more commands is infinitesimal ($o(\Delta t)$). If i-th command-reaction is chosen, then from the ensemble strings are chosen randomly and with equal probability, in the amount necessary for fulfillment of the command. If the command is applicable to this set of strings, it is executed and we pass to the following Δt, if it is inapplicable, the ensemble does not change and we pass to the following Δt all the same. In the limit $\Delta t \to 0$ we obtain random process – statistical model of the Kirdin kinetic machine.

4 Correctness of the Programs Consisting of one Command

The commands of disintegration and replacement are undetermined, if conditions 1&2 or 1&3 of the following list are fulfilled.

1. The chain v being replaced can be decomposed as *aba*, i.e. its beginning and end coincide.
2. In strings, to which these commands are applicable, there are the chains of the kind *ababa*, i.e. the chain v can be chosen in two ways.
3. In the strings obtained after application of these commands there are chains of the kind *ababa*, i.e. the chain v can be chosen in two ways.

For the programs consisting of any number of commands of disintegration and replacement, these criteria are easily generalized.

A program consisting of commands of synthesis is always finite, and in general cases undetermined.

5 Algorithmic Universality of the Kirdin Kinetic Machine

We assume that the Kirdin kinetic machine possesses a universal character. Thus a question arises: how the Kirdin kinetic machine correlates with consecutive standard algorithmic formal models.

Theorem. *The determined Kirdin kinetic machine is equivalent to any consecutive standard algorithmic formal model, such as the Turing machine or Markov normal algorithms* [5]. *Hence, it is an effective calculator.*

The halting problem for the Turing machine is unsolvable in the general case [6], *hence finiteness the Kirdin kinetic machine for the programs consisting of commands of replacement is also unsolvable.*

The Kirdin kinetic machine is undetermined in the general case. It is natural that it cannot be completely equivalent to the determined calculator. Nevertheless, we have seen that it completely includes all determined universal calculators. What can be said about the Kirdin kinetic machine in the undetermined case? Let's distinguish another class of the Kirdin kinetic machine.

We will call the Kirdin kinetic machine *partially determined*, if its program consists of determined commands of replacement and disintegration and any commands of synthesis.

A partially determined Kirdin kinetic machine can be modeled by a specially arranged system of algorithmic calculators, for instance, let them be the Turing machines.

Consider a partially determined Kirdin kinetic machine, $|M|$ is the number of strings in its ensemble. From the beginning $|M|$ Turing machines are initialized, each of them processes its own string. The program for each machine consists of commands of replacement and disintegration. The application of a command of disintegration means initiation of a new Turing machine, and the configuration of the first of them corresponds to the string uf, and the configuration of the second – to the string gw.

At the same time, an *over-calculator* is functioning, with a program consisting of all commands of synthesis of the initial program. It compares configurations of Turing machines with strings *uk* and *qw*. If both are present in the configurations, it "switches off" one of these machines, and the configuration of the another turns into *usw*.

6 Unstructured Memory

Unstructured memory is an organic elementary application of the Kirdin kinetic machine. Its basic idea is to store the information about a long text by means of a special dictionary, consisting of strings which length is much shorter than the length of the initial text. The list of all strings of length q, included in the given text, referred to as q-carrier of the given text. Strings starting from any place in the text are considered. For a text of the length N there are N-q+1 of such strings. If each string of the q-carrier is put into correspondence to the frequency of its occurrence in the text, we obtain the frequency dictionary of length q.

Transition from the text to its frequency dictionary is a useful technique, which allows comparing texts of different lengths and performing their information analysis, which was successfully made for genetic texts in [11]. Besides, the frequency dictionary fixes the information about the text in a set of small objects – strings with their frequencies, which can be stored separately, "in a heap". There exist probabilistic estimations of the length of the dictionary, sufficient for the text unambiguous reconstruction.

If a dictionary of the length d contains strings, which occur uniquely, then for the dictionary of the length d+1 and larger the text is restored unambiguously. This very case will be considered. The following program for the Kirdin kinetic machine constructs the dictionary of the length k from an initial (long) text. This program is finite and determined. If the final ensemble consist of an unique string – !, then the obtained dictionary does not contain complete information about the initial text. It means, that such length of the dictionary is insufficient for the unambiguous reconstruction of the initial text. Hence, the given procedure should be started anew for the initial text, but with k increased by 1. And so on, until we obtain a dictionary of length k as final ensemble, and now it is necessary to start the program for the last time, to construct the dictionary of length k+1, from which the initial text can be reconstructed unambiguously.

Let us introduce a new designation: v^k in the left part of a rule denotes an arbitrary string of the length k in the initial alphabet. All entries of v^k in one rule denote the same string. Entries of the symbol v^k in different rules are not connected.

Program 6.1:

$uv^1v^{k-1}v_1^1w \rightarrow uv^1v^{k-1} + v^{k-1}v_1^1w$

$v^k + v^k \rightarrow !$

$!+v^k \rightarrow !$

Now, storing and, probably, transferring the initial text through communication channels as the dictionary, we can always unambiguously reconstruct it from this dictionary. The following program of the Kirdin kinetic machine is intended for this purpose.

Program 6.2:

$uv^k + v^k w \rightarrow uv^k w$

7 Structural Programming in Structureless Parallelism

A natural necessity arose from programming for the Kirdin kinetic machine to create an analogue of subprograms in usual programming. The subprograms can be useful for conveyer programming and for encapsulation of a piece of an ensemble with its own program.

A processed unit for an enlarged Kirdin kinetic machine is at this time an ensemble of strings from a certain alphabet L. At the moment of initialization a number of employ cells for subprograms appears. For every cell an identificator is created. It consists of symbols from the alphabet I (with $L \cap I = \varnothing$) and positive integer number – i.e. this cell arity. For example, Cell(3) – a cell with the Cell identificator and 3 arity. Arity corresponds to the number of position for input string – parameter of a cell.

The processing consists of an aggregate of elementary events, which occur non-deterministically, in parallel and are regulated by rules or commands. The commands of disintegration, synthesis and replacement for the basic program effect the ensemble without cells. To exchange strings between a subprogram (cell) and the ensemble there are special commands in a program. They are the commands of conditional adsorption and desorption.

Adsorption: $Cell()_i + uvw \rightarrow Cell(uvw)_i$

$Cell()_i$ is the i-th vacant position in the *Cell*. Expression $Cell(uvw)_i$ means the i-th position is taken by the string uvw.

When the program starts working the strings of the $Cell()_i$ type are initiated for all input strings of all subprogram cells. One of commands of the type $Cell()_i + uvw \rightarrow Cell(uvw)_i$ having been applied, the i-th input position of the *Cell* is taken until it is released from the subprogram *Cell* as demonstrated below. Then the string $Cell()_i$ again appears in the ensemble.

Every cell is associated with a subprogram consisting of disintegration, synthesis and replacement commands. The subprogram effects only the strings occurring in this cell. Input cell parameters can be used with the release of an input parameter position or without it. For the use of an input parameter without a position release a special command appears similar to the synthesis command of the type

$In(uk)_i + qw \rightarrow usw + In(uk)_i$ or $uk + In(qw)_i \rightarrow usw + In(qw)_i$.

For the use of an input parameter with a position release the command $In(uvw)_i \rightarrow uvw + In()_i$ is applied. Application of the command formally means the appearance of the string $Cell()_i$ in the basic ensemble of the Kirdin kinetic machine.

Desorption is initiated out of a cell. The right part of any subprogram command can carry *Out(usw)* string for synthesis command or *Out(uf)* or *Out(gw)* strings for disintegration commands. The application of such a command means that the string *s* from the *Out(s)* ia taken from a cell and goes to the basic ensemble of the Kirdin kinetic machine.

The following types of information processing in cell are possible:

1.Sequential processing:

- A necessary set of strings goes into a cell
- Processing takes place inside the cell
- The modified set of strings goes out of the cell
- The cell is ready to accept a new set of strings

2.Parallel processing:

- A cell is constantly ready to accept new strings which are included into the process occuring inside the cell
- From time to time the cell discharges strings-results of its perfomance into the basic ensemble.

3.Conveyer processing:

- New strings constantly enter a cell
- Strings are processed without any interaction between themselves
- After processing the strings goes out of the cell independently

Example of a program with subprograms

Problem: *an ensemble consists of a set of patterns-strings from the alphabet L. A detected string is added to it and indicated with the symbol * in the beginning (with $*\notin L$). The problem is to find out if the string coincides with a pattern.*

Program:

Main: $Reverse_1()+*w \to Reverse_1(*w)$

$Compare_1()+u* \to Compare_1(u*)$

$Compare_2()+u^{(L)} \to Compare_2(u^{(L)})$

Reverse(1): $In_1(*w) \to In_1(*w)+*w$

$u*v^1w \to uv^1x*w$

$uv_1{}^1v^1xw \to uv^1xv_1{}^1w$

$v^1xw \to v^1w$

$v^{(L)}* \to Out(v^{(L)}*)$

Compare(2): $In_2(u^{(L)}) \to In_2()+u^{(L)}$

$In_1(u*)+v^{(L)} \to u*v^{(L)}+ In_1(u*)$

$uv^1*v^1w \to u*w$

$* \to Out(*)$

$uv_1{}^1*v_2{}^1w \to uxw$

$uv^1*w \to uxw$

$u*v^1w \to uxw$

$x+w \to w$

This program is finite and deterministic. Subprogram *Reverse(1)* realize sequential processing. Subprogram *Compare(1)* realize conveyer processing. The answer to the problem is "yes" if final ensemble consists of one or several * symbols, "no" if final ensemble is empty.

8 Conclusions

The Kirdin kinetic machine is based on two paradigms:

- fine-grained parallelism
- structureless parallelism

These seem to be the most perspective directions of the development of computer science.

We have seen that the Kirdin kinetic machine is a universal calculator. The ways of solution of the problem of program execution correctness for the Kirdin kinetic machine are offered. Determination of finiteness for the Kirdin kinetic machine is very complicated and, in the general case, unsolvable. But the same is known about the Turing machine.

Determinacy means definiteness of the result. Most likely, for some range of problems we will not be interested in strict determinacy, but in near determinacy or even simply probabilistic distributions of the final ensemble.

According to the well-known "Minsky hypothesis" the efficiency of a parallel system increase proportionally to logarithm of processor number. To overcome this restriction the following approach often applied. Extremely parallel algorithms of solutions are built for different types of problems. The algorithms use some abstract paradigm of fine-grained parallelism, for example, structureless parallelism. For particular parallel computers means of parallel processes realization with a given abstract architecture are created. As a result an effective tool for parallel program production appears.

References

1. Kirdin A.N. (1997), Ideal ensemble model of parallel computations, *Abstracts of V All-Russian Workshop "Neural informatics and its applications"*, Gorban A.N. (ed.), Krasnoyarsk: Krasnoyarsk State Technical University Press, p. 101. (in Russian).
2. Gorbunova E.O. (1998), The analysis of elementary programs for ideal ensemble model of parallel computations, *Abstracts of INPRIM-98*, Lavrent'ev M.M. (ed.), Novosibirsk: Institute of Mathematics SB RAS, 1998, p. 77. (in Russian).
3. Gorbunova E.O. (1998), Finiteness and determinacy of simple programs for the Kirdin kinetic machine, *Methods of Neuroinformatics* (Collection of papers), Gorban A.N. (ed.), Krasnoyarsk: Krasnoyarsk State Technical University Press, pp. 23-40 (in Russian).

4. Gorbunova E.O (1998), To the question of algorithmic universality of the Kirdin kinetic machine, *Abstracts of VI all-Russian Workshop "Neuroinformatics and its applications"*, Gorban A.N. (ed.), Krasnoyarsk: Krasnoyarsk State Technical University Press, pp. 47-48 (in Russian).
5. Markov A.A., Nagorny N.M. (1984), The theory of algorithms, Moscow: Nauka (in Russian).
6. Uspensky V.A., Semenov A.L. (1997), The theory of algorithms: basic discoveries and applications, Moscow: Nauka (in Russian).
7. Gorban A.N., Rossiev D.A. (1996), Neural networks on personal computer, Novosibirsk: Nauka (in Russian).
8. LATKIN E.I. (1995), SCAM chemical computer, The theory of computations and languages of specifications, *Computing Systems*, V. 152, Novosibirsk: Computing Center SB RAS, pp. 140-151 (in Russian).
9. Achasova S., Bandman O., Markova V. and Piskunov S. (1994), Parallel substitution algorithm. Theory and Application, WORD SCIENTIFIC.
10. Dasgupta D. (Ed.) (1998), Artificial immune systems and their applications, Collection of papers, SPRINGER, XIV.
11. Bugaenko N.N., Gorban A.N. and Sadovsky M.G. (1998), Maximum entropy method in analysis of genetic text and measurement of its information content, *Open Sys. And Information Dyn.*, 5, pp.265-278.
12. Jabotinskiy A.M. (1974), Concentration Selfooscillations, Moscow: Nauka (in Russian).
13. Gorban A.N., Bykov V.I., Yablonsky G.S. (1986), Essays about chemical relaxation, Novosibirsk: Nauka (in Russian).