Bull. Nov. Comp. Center, Comp.Science, 17 (2002), 39–50

Parallel computations on graph structures

I.V. Kudrin

The Graph Automaton intended for modeling parallel fine-grained transformations of graph structures is presented. It works in mode of simultaneous application of a set of substitution commands to a graph image. Substitution commands are presented in graph notation, too. The example of algorithm implementation on the Graph Automaton is given.

Introduction

The Graph Automaton is an extension of Parallel Substitution Algorithm [1] to the field of cellular spaces with irregular structure. Unlike PSA, a working field structure is not determined by external rules, but defined by a state of working field itself. Moreover, the automaton can change not only state of cells of working field, but also its structure, during parallel program execution.

1. Graph Automaton definition

Let A be a finite alphabet and M be a set of names (at most countable).

A pair $(a, m) \in A \times M$ is called *a vertex* where a is *the state* and m is *the name* of a vertex.

Let M^* be a set of all subsets of M which contain one or two elements. A triple (a, L, D), where $a \in A$, $L \in M^*$, $D \subseteq L$, is called *an edge*. In the triple, a is *the state*, L is *the link* and D is *the direction* of an edge.

If the link component L of an edge contains two names, then the edge connects two vertices. Otherwise, when the link component L contains one element the edge describes a self-loop. The direction component D is a subset of link component L and describes a set of the edge directions. A triple $(a, L, \emptyset)$ expresses an undirected edge. The one-element direction component D designates a directed edge; the two-element one describes a two-directed edge.

When the link component of an edge e contains the name m, it is said that the edge e *refers to* a vertex with the name m.

Definition 1. A finite set G of vertices and edges, with no pair of vertices having identical names and no pair of edges having identical link components is called *a partial graph image*.

Denote $M_G = \{m \mid (\cdot, m) \in G\}$.

Definition 2. A partial graph image G is called *a complete graph image* if the following condition is met: $\forall(\cdot, L, \cdot) \in G \quad L \subseteq M_G$.

Remark. In a graph image, there is no more than one edge for any pair of vertices, and no more than one self-loop for each vertex.

In a complete graph image, any edge refers to vertices existing in the image, while in a partial graph image the link component L (and direction component D) of the edge may contain names outside the image. Such edge $(a, L, D) \in G$, $L \cap M_G \neq L$ is called *a hanging* edge.

Example 1. Let $A = \{a, b\}$, $M = \mathbb{N}$, $\mathbb{N} = \{1, 2, \ldots\}$, then

$$G_1 = \{(a, 1), (a, 3), (a, \{1, 2\}, \emptyset), (b, \{1, 3\}, \{1\})\}$$

is a partial graph image with hanging edge $(a, \{1, 2\}, \emptyset)$, and

$$G_2 = \{(a, 1), (b, 2), (a, 3), (a, \{1, 2\}, \emptyset), (b, \{1, 3\}, \{1\})\}$$

is a complete graph image.

Definition 3. Two complete graph images G_1 and G_2 are called *isomorphic* if their alphabets coincide and there is a one-to-one correspondence $f : M_{G_1} \to M_{G_2}$ between their name sets, such that

1. $(a, m) \in G_1 \Leftrightarrow (a, f(m)) \in G_2$,
2. $(a, L, D) \in G_1 \Leftrightarrow (a, f(L), f(D)) \in G_2$, where $f(X) = \{f(x) \mid x \in X\}$.

The set of all complete graph images, where no pair of isomorphic graph images exists, and vertices and edges have states from A and names from M is denoted by $K(A, M)$.

Example 2. Graph images

$$G_1 = \{(a, 1), (b, 2), (a, 3), (a, \{1, 2\}, \emptyset), (b, \{2, 3\}, \{2\})\}$$

and

$$G_2 = \{(a, 9), (b, 4), (a, 5), (a, \{4, 9\}, \emptyset), (b, \{5, 4\}, \{4\})\}$$

are isomorphic ones.

Definition 4. An expression of the form

$$\Theta : G_1 * G_2 \to G_3,$$

where G_1, G_2 and G_3 are partial graph images, $G_1 \cup G_2$ and $G_2 \cup G_3$ are complete graph images, $G_1 \cap G_2 = \emptyset$, $G_2 \cap G_3 = \emptyset$, is called *a substitution command*. $G_1 * G_2$ is *the left-hand side* of a substitution command, G_1 is *the base*, G_2 is *the context*, and G_3 is *the right-hand side* of a substitution command.

Denote $M_{\Theta L} = \{m \mid (\cdot, m) \in G_1 \cup G_2\}$, $M_{\Theta R} = \{m \mid (\cdot, m) \in G_3\} \setminus M_{\Theta L}$, $M_\Theta = M_{\Theta L} \cup M_{\Theta R}$.

Remark. In a substitution command, the hanging edges from the base and from the right-hand side must refer to the vertices of the context. The hanging edges from the context must refer to the vertices of the base and of the right-hand side simultaneously. In this case, the unions of base and context, and also of context and right-hand side, are complete graph images.

Definition 5. Let G be a complete graph image. A function $\varphi_{\Theta G} : M_\Theta \to M$ is called *a covering function* $\Theta \to G$ if the following conditions are met:

1. $\begin{cases} \varphi_{\Theta G}(m_\Theta) \in M_G, & m_\Theta \in M_{\Theta L}, \\ \varphi_{\Theta G}(m_\Theta) \in M \setminus M_G, & m_\Theta \in M_{\Theta R}, \end{cases}$
2. $m_1, m_2 \in M_\Theta, m_1 \neq m_2 \Rightarrow \varphi_{\Theta G}(m_1) \neq \varphi_{\Theta G}(m_2)$,
3. $\forall (a, m) \in G_1 \cup G_2 \quad \exists (a, \varphi_{\Theta G}(m)) \in G$,
4. $\forall (a, L, D) \in G_1 \cup G_2 \quad \exists (a, \varphi_{\Theta G}(L), \varphi_{\Theta G}(D)) \in G$,
   where $\varphi(X) = \{\varphi(x) \mid x \in X\}$.

Remark. The covering function $\Theta \to G$ establishes a correspondence between the left-hand side of a substitution command Θ and a subgraph of a graph image G. The selected subgraph is isomorphic to the graph image of the left-hand side of a substitution command. For all names of the right-hand side of a substitution command, which are absent in the left-hand side (the set $M_{\Theta R}$), new names absent in a source graph image are generated.

Definition 6. A set $\Phi_{\Theta G} = \{\varphi_{\Theta G}\}$ of all possible covering functions $\Theta \to G$, where

1. $\forall \varphi_1, \varphi_2 \in \Phi_{\Theta G}, \varphi_1 \neq \varphi_2 \Rightarrow \exists m \in M_{\Theta L}, \varphi_1(m) \neq \varphi_2(m)$
2. $\forall \varphi_1, \varphi_2 \in \Phi_{\Theta G}, \varphi_1 \neq \varphi_2, m_1, m_2 \in M_{\Theta R} \Rightarrow \varphi_1(m_1) \neq \varphi_2(m_2)$

is called *a covering set* $\Theta \to G$.

Remark. Each covering function of a covering set $\Theta \to G$ selects a unique subgraph from the source graph image G (according to rule 1). If a set $M_{\Theta R}$ of a substitution command Θ is not empty, then each covering function of a covering set $\Theta \to G$ generates unique set of names absent in the source graph image G (according to rule 2).

Example 3. Let

$$\Theta : \{(a,1),(c,\{1,2\},\emptyset)\} * \{(a,2)\} \to \{(b,1),(a,3),(c,\{1,3\},\emptyset)\}$$

be a substitution command defined for the complete graph image

$$G = \{(a,1),(a,2),(c,\{1,2\},\emptyset)\},$$

where $A = \{a,b,c\}$, $M = \mathbb{N}$. So, the covering set $\Theta \to G$ is

$$\begin{aligned} \Phi_{\Theta G} &= \{\varphi_{\Theta G_1}, \varphi_{\Theta G_2}\}, \\ \varphi_{\Theta G_1} &= \{1 \to 1, 2 \to 2, 3 \to 4\}, \\ \varphi_{\Theta G_2} &= \{1 \to 2, 2 \to 1, 3 \to 5\}. \end{aligned}$$

There are two various ways of covering the left-hand side of the command Θ to the graph image G, so the covering set $\Phi_{\Theta G}$ contains two covering functions. Since the set $M_{\Theta R} = \{3\}$ for this command is not empty, the new names are generated by each covering function.

Definition 7. A *covering separation function* is defined as follows:

$$\psi(\widetilde{G}, \varphi) = \{(a, \varphi(m)) \mid (a,m) \in \widetilde{G}\} \cup \{(a, \varphi(L), \varphi(D)) \mid (a,L,D) \in \widetilde{G}\},$$

where $\widetilde{G}$ is any component (G_1, G_2, or G_3) of a command $\Theta : G_1 * G_2 \to G_3$ and φ is a covering function $\Theta \to G$.

Remark. The covering separation function ψ translates a partial graph image $\widetilde{G}$ of a command component to the name space of the source graph image G according to the covering function φ.

Example 4. With conditions of Example 3:

$$\begin{aligned} &\psi(\{(a,1),(c,\{1,2\},\emptyset)\}, \varphi_{\Theta G_2}) = \{(a,2),(c,\{1,2\},\emptyset)\}, \\ &\psi(\{(b,1),(a,3),(c,\{1,3\},\emptyset)\}, \varphi_{\Theta G_2}) = \{(b,2),(a,5),(c,\{2,5\},\emptyset)\}. \end{aligned}$$

Definition 8. An expression like $\Theta_{G\varphi} : G'_1 * G'_2 \to G'_3$, where $G'_i = \psi(G_i, \varphi)$, is called *a substitution.*

Remark. The substitution $\Theta_{G\varphi}$ is a translation of substitution command Θ to the name space of the source image G, according to the covering function φ. The substitution exists only when a covering function exists, i.e., a subgraph that is isomorphic to a graph image of the left-hand side of a substitution command exists in the source graph image.

Definition 9. If the substitution $\Theta_{G\varphi} : G'_1 * G'_2 \to G'_3$ exists, then the substitution command Θ is said to be *applicable* to the graph image G, and the substitution $\Theta_{G\varphi}$ can be *executed* in the graph image G, resulting in:

$$\Theta_{G\varphi}(G) = F\left((G \setminus G'_1) \cup G'_3\right),$$

where F is the "lost" edges elimination function:

$$F(G) = \{(a, m) \mid (a, m) \in G\} \cup \{(a, L, D) \mid (a, L, D) \in G, L \subseteq M_G\}$$

Remark. The function F removes all hanging edges from a partial graph image G, so the result of its execution is a complete graph image.

In compliance with the substitution command definition, there are four variants of substitution execution behavior, respecting to an element e of a command:

1. $e \in G_1$, $e \notin G_2 \cup G_3$: the element will be deleted;
2. $e \in G_3$, $e \notin G_1 \cup G_2$: the element will be added;
3. $e \in G_1$, $e \in G_3$,$e \notin G_2$: the element will be changed;
4. $e \in G_2$, $e \notin G_1$,$e \notin G_3$: the element is used for context search only.

Definition 10. An execution of all possible substitutions, which are produced by the covering command $\Theta : G_1 * G_2 \to G_3$ to the complete graph image G, is called *an execution* of the command Θ in the graph image G:

$$\Theta(G) \;=\; F\left(\left(G \setminus \Theta^{-}(G)\right) \cup \Theta^{+}(G)\right),$$

where $\Theta^{-}(G)$ is a set of elements to be removed:

$$\Theta^{-}(G) \;=\; \bigcup_{\varphi \in \Phi_{\Theta\, G}} \psi(G_1, \varphi),$$

and $\Theta^{+}(G)$ is a set of elements to be added:

$$\Theta^{+}(G) \;=\; \bigcup_{\varphi \in \Phi_{\Theta\, G}} \psi(G_3, \varphi).$$

Example 5. With conditions of Example 3, the execution of the substitution command Θ in the graph image G results in the graph image

$$\Theta(G) = \{(b,1),(b,2),(a,4),(a,5),(c,\{1,4\},\emptyset),(c,\{2,5\},\emptyset)\}.$$

The set of elements to be removed is

$$\begin{aligned}\Theta^-(G) &= \{(a,1),(c,\{1,2\},\emptyset)\} \cup \{(a,2),(c,\{2,1\},\emptyset)\} \\ &= \{(a,1),(a,2),(c,\{1,2\},\emptyset)\},\end{aligned}$$

and the set of elements to be added is

$$\begin{aligned}\Theta^+(G) &= \{(b,1),(a,4),(c,\{1,4\},\emptyset)\} \cup \{(b,2),(a,5),(c,\{2,5\},\emptyset)\} \\ &= \{(b,1),(b,2),(a,4),(a,5),(c,\{1,4\},\emptyset),(c,\{2,5\},\emptyset)\}.\end{aligned}$$

So, during the execution of the substitution command Θ in the source graph image G:

1. The state of vertices with names "1" and "2" are changed from "a" to "b",
2. The edge $(c,\{1,2\},\emptyset)$ are removed,
3. The vertices $(a,4)$ and $(a,5)$ and the edges $(c,\{1,4\},\emptyset)$ and $(c,\{2,5\},\emptyset)$ are added.

Definition 11. A finite set of substitution commands $\Xi = \{\Theta_1,\ldots,\Theta_v\}$ is called *a Parallel Substitution System* (PSS). A PSS is said to be *applicable* to a complete graph image G if there exists a non-empty subset of commands $\Xi' \subseteq \Xi$ applicable to G. The execution of the PSS Ξ in G results in

$$\Xi(G) = F\left(\left(G \setminus \left(\bigcup_{\Theta\in\Xi'} \Theta^-(G)\right)\right) \cup \left(\bigcup_{\Theta\in\Xi'} \Theta^+(G)\right)\right).$$

A PSS Ξ is referred to as *non-contradictory* with respect to a subset $K^* \subseteq K(A,M)$ if its execution in any $G \in K^*$ results in a complete graph image. If the PSS is non-contradictory with respect to any $G \in K(A,M)$, it is called *non-contradictory*.

Definition 12. Let $\Xi = \{\Theta_1,\ldots,\Theta_v\}$ be a non-contradictory PSS. The following iterative procedure is called *a synchronous mode of the PSS* Ξ *execution* in G. Let G^i be the result of its ith step, then

1. If no substitution command $\Theta \in \Xi$ is applicable to G^i, then G^i is the result of computation;
2. If there exists a non-empty subset of substitution commands $\Xi' \subseteq \Xi$ applicable to G^i, then G^{i+1} is substituted for G^i, i.e., $G^{i+1} = \Xi'(G^i)$.

Definition 13. A PSS together with the synchronous mode of execution of a PSS, which is non-contradictory with respect to $K^* \subseteq K(A, M)$, is called *a Graph Automaton* (GA) and is denoted by $\Pi = \langle \Xi, K^* \rangle$. The result of the execution of Π in $G \in K^*$ is referred to $\Pi(G)$. A complete graph image, in which GA is executed, is called *a working field* of the Graph Automaton.

2. Special cases and extensions of the GA

Functional substitution. Improvement of expressive capabilities of the GA can be achieved by extension of the states of vertices and edges alphabet A by additional variables and functions whose domain and range are in A. To the alphabet A the set of variable symbols Z and functional symbols F are added, so that the resulting alphabet is $A' = A \cup Z \cup F$.

The new alphabet is used in substitution commands definition. During the application of a command to the working field, variable symbols are associated with the states of working field cells. Function values are evaluated on the basis of associated variables. In result, the substitution is formed with calculated values. The special symbol "$-$" is used for a designation of variables with don't care values.

Example 6. The command

$$\Theta : \{(-, 1)\} * \{(x, 2), (-, \{1, 2\}, \{1\}, 3)\} \to \{(x + 1, 1)\}$$

defines an operation of increment and transfer of a vertex state by the edge specified direction.

Multiple vertices and edges allow to define commands which being applied to the working field select variable size sets of vertices and edges. With functional substitutions, there is a capability to use special aggregate functions, such as count(), sum(), min(), etc. The mark * in a name is used for multiple vertex designation. An edge incident to multiple vertex is also multiple.

Example 7. The command

$$\Theta : \{(-, 1)\} * \{(x, 2^*), (-, \{1, 2^*\}, \{1\})\} \to \{(\max(x), 1)\}$$

sets the state of the vertex to the maximal of its neighborhood vertices values.

Graph image in geometrical space. Each item of a working field can be attributed with certain coordinates. Thus, it is possible to set the graph layout in a coordinate space. Substitution commands are supplemented

with opportunities to read and set spatial coordinates of vertices. Hence, it is possible to combine logic and spatial modeling on graph image. This opportunity allows the usage of Graph Automaton for simulating physical processes.

3. Graphic representation of the GA

One of the key advantages of Graph Automaton is the possibility of its visual representation. This is an easy-to-interpret GA's form and it is achieved by creation of the equivalent graphic representing of GA's components.

Work field representation. The vertices of a working field are represented as circles, and edges are represented as the lines connecting these circles. The state of the vertex is written inside a circle, and the state of the edge is written next to a line representing it. The direction of the edge is indicated by an arrow. The example of graphic representation of a working field is shown in Figure 1. One may note, that the names of a graph image vertices disappeared in a graphic representation, but the references between elements are reflected in obvious way.

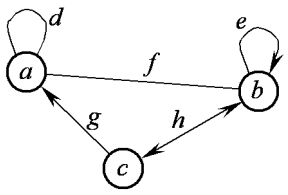

Figure 1. Graphic representation of graph image $G = \{(a,1),\ (b,2),\ (c,3),\ (d,\{1\},\emptyset),\ (e,\{2\},\{2\}),\ (f,\{1,2\},\emptyset),\ (g,\{1,3\},\{1\}),\ (h,\{2,3\},\{2,3\})\}$

Substitution command representation. The substitution command consists of three partial graph images. Therefore in graphic representation three identical size rectangular areas bounding graphic representation of each command part are used. The relative geometrical position inside bounding rectangle is interpreted as a name of command's vertex. This allows indicating references of elements from different command parts. The example of graphic representation of a command is shown in Figure 2.

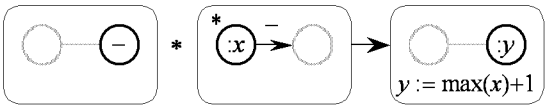

Figure 2. Graphic representation of substitution command $\Theta : \{(-,1)\} * \{(x,2^*),(-,\{1,2^*\},\{1\})\} \to \{(\max(x)+1,1)\}$

4. Grata the system of modeling

The system of simulation modeling Grata is intended for creation models of graph algorithms on the computer. The system works in Windows environment and uses the standard user interface. The program environment provides a visual designing of graph images of working field and substitution commands. The executive block performs an interpretation of the algorithm constructed within the system framework, its debugging and representation of work's results (Figure 3).

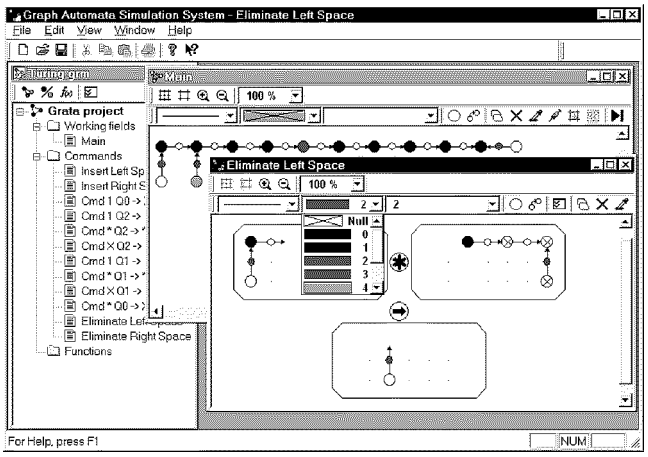

Figure 3. Appearance of system Grata

All components of a graph algorithm are united in a project within the framework of the system. A project determines a common settings for an algorithm, and groups a components by their functions. Two main components of a project are a representation of a working field and a set of substitution commands. In the Grata, a representation of a working field and substitution commands is made on the basis of graphic representation of the GA. It provides convenient visual editing of the Graph Automaton components.

In the system, an integer and a real numbers and a strings are used as states of graph elements. The numerical data can be displayed using colors of selectable pallets.

The Grata system allows to work both with base variant of Graph Automaton, and with the extensions described above. The C-like programming language is built into the system, realizing functional substitutions. For work with a spatial arrangement, three coordinates (x, y, z) are attributed to each vertex of working field. The system functions set is extended by an ability to read and to set spatial coordinates.

The GA, as well as its software implementation, has enough of opportunities for modeling such tasks, as equivalent transformations of automata

networks by the certain criteria, functional modeling based on the hierarchical networks and so on. The system can be used for research of algorithms for various graph problems, such as topological sort or shortest path search.

5. Example of model on Graph Automaton

As an example of the Graph Automaton work, the implementation of algorithm of parallel list prefix computation ([2]) will be demonstrated.

Let $\otimes$ be an associative binary operation. The problem of prefix computation consists in the following: given a sequence $\langle x_1, x_2, \ldots, x_n \rangle$, a sequence $\langle y_1, y_2, \ldots, y_n \rangle$, for which $y_1 = x_1$ and

$$y_k = y_{k-1} \otimes x_k = x_1 \otimes x_2 \otimes \ldots \otimes x_k$$

for $k = 2, 3, \ldots, n$, should be constructed.

Let us consider, that the elements $x_1, \ldots, x_n$ are connected in the singly linked list. Let there be a set of N processors, each of which keeps its own value of y, and a pointer to the next element $next$.

The description of algorithm under these agreements is given in [2]:

```
LIST-PREFIX(L)
1  for each processor i
2      do y[i] ← x[i]
3  while there is an object i, for which next[i] ≠ NIL
4        do for each processor i
5              do if next[i] ≠ NIL
6                    then y[next[i]] ← y[i] ⊗ y[next[i]]
7                         next[i] ← next[next[i]]
```

At Graph Automaton implementation, the vertices of a working field will represent processors. The state of vertex corresponds to a state of a field y of the appropriate processor. The pointer $next$ is represented by the directed edge. Thus, as the field $next$ can store both valid pointer, and NIL, then, for uniformity of substitution commands execution, the additional vertex with a state "0" is created on the working field. The valid pointer is represented as an edge with a state "0", and the empty one as an edge with a state "1". The following substitution command corresponds to lines 3–7 of the given algorithm (initial state of a working field after initialization appropriate to lines 1–2 is considered):

$$\Theta : \{(i,1),(0,\{1,2\},\{1\})\} * \{(j,2),(-,3),(n,\{1,3\},\{3\})\} \rightarrow \{(i \otimes j,1),(n,\{2,3\},\{3\})\}$$

In Figure 4, the graphic representation of this command is shown.

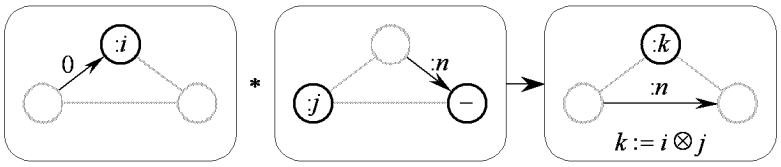

Figure 4. The substitution command for LIST-PREFIX algorithm

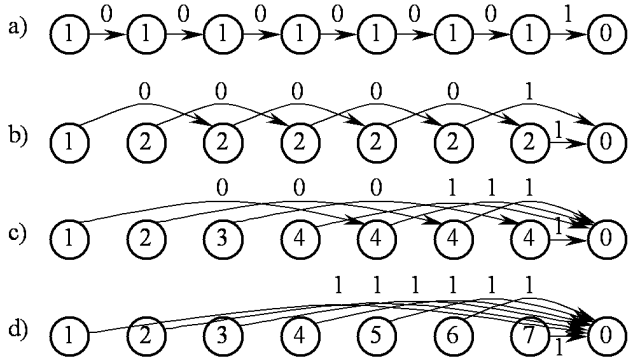

Figure 5. The sequence of changes of working field states for LIST-PREFIX algorithm: a) initial state, b), c) intermediate states, d) final state

Let the operation $\otimes$ be an addition. Then, when filling initial cells as $x_i = 1$, $i = 1 \ldots n$, the special case of a considered problem, the list ranking problem is occurred. The sequence of graphic representation of working field states for this case is shown in Figure 5.

Conclusion

The Graph Automaton being a toolset of modeling of parallel fine-grained transformations of graph structures is submitted in this article. The formal description of the GA base is given and some its extensions and additions are designated. The way of graphic representation of a working field and substitution commands is given, that allows providing convenient and evident work with the Graph Automaton components. Based on the graphic representation, the system of simulation modeling Grata is created. System is capable to create, to edit and to investigate models of parallel graph algorithms. The example of algorithm of list prefix parallel processing implementation is given.

The presented system is capable to simulate an extensive class of systems: from physical models up to functional chart of devices. The system gives common basis for constructing a wide class of functional models and parallel algorithms.

References

[1] Achasova S.M., Bandman O.L., Markova V.P., Piskunov S.V. Parallel Substitution Algorithm. Theory and Application. – Singapore: World Scientific, 1994.

[2] Cormen Thomas H., Leisersor Charles E., Rivest Ronald L. Introduction to Algorithms. – Cambridge, Mass.: MIT Press, 1990.