Formal Expressions of Infinite Graphs and Their Families

HIDEHIKO OKABE

Department of Biophysics, Faculty of Science, Kyoto University, Kyoto, Japan

We discuss some methods for rigorous expressions of infinite labeled directed graphs. In these methods, every node of a graph is identified by a string of symbols from a finite alphabet, and edges are expressed as binary relations or functions among strings. We characterize these relations with some concepts in the theory of formal languages and automata, and from this characterization, we obtain various classes of graphs. Then inclusion relationships among these classes are discussed. Although our methods originate in a very naive idea, they appear to be suitable for both theoretical treatments and computer processing, so we ourselves regard this paper as a preliminary for various discussions that our formalization enables.

1. INTRODUCTION

Directed graphs which may have infinitely many nodes and edges are the most general and complicated structures in discrete mathematics. It is rather curious that the general theory of infinite graphs has not been well established, for they are so widely utilized in various contexts in pure and applied mathematics. From the following reasons, we believe that a trial must be made to fill this absence.

First of all, it is usually difficult to find rigorous and intelligible definitions of graphs, except some special graphs—trees, array structures, etc. Probably, the most plain method is to represent them by figures like Figs. 1, 2, and 3 of this paper. But this method cannot be rigorous for infinite graphs, and is not suitable for formal and algebraic treatments. Oppositely, the number theoretical description (see Section 3) is one of the most rigorous and formal methods of defining graphs. Unfortunately, even most simple graphs such as trees and array structures require rather involved expressions in this method, and such expressions are so far from our intuition that when we are given them without any supplementary explanations, it is very difficult for us to imagine the features of the graphs.

Apart from such shortcomings of each method, the situation that there is no general and standard method may cause misunderstandings and losses of efforts in mutual communication. For example, in some definitions of graphs, self-loops or parallel edges are excluded implicitly, and in some others, self-
loops are always presumed to exist for each node and therefore not expressed explicitly in the definition.

Second, formal discussions about infinite labeled graphs are required in several fields. Really, our motivation for this paper comes from our intention to generalize the theory of cellular spaces to more irregularly constructed automata. The structures of cellular spaces were generalized to group graphs by Wagner (1965) and to balanced graphs by Jump and Kirtane (1974). But more irregular constructions and more global connections of information networks are prevalent in both natural and artificial information processing systems. On one hand, we know that such generalization of structures (for example, the existence of a cell which can send signals to all other cells in the network) immensely improves the efficiency of calculations in poly-automata, and on the other hand, we wish to exclude such a nonsense trick that the answers for the inputs have already been inscribed in the structure of the poly-automaton, and the sole process that the system must do is to carry these answers to the predetermined output region. So, we want for some concepts about the complexity of the structures and initial configurations of poly-automata with general structures.

Third, it is possible that the discussions of infinite graphs give us some new approaches and new methods for the fields where we are accustomed to apply the concepts in the theory of finite graphs. Note that, while the theory of graph grammars or graph generating systems deals with infinite sets or infinite sequences of finite graphs, we are going to discuss the graphs which themselves are infinite. But we can consider an infinite graph as the limit of a growing finite graph or a sequence of finite graphs. In this sense, there is a strong relation between finite graphs and infinite graphs. Moreover, finite graphs are also defined formally by our method discussed in this paper, so our formulation may offer effective tools for finite graphs too.

Our basic idea is that, in order to define a graph formally and effectively, every node of the graph must be identified by some mathematically well-defined objects. Then the edges are defined as some binary relations among these objects. In this paper, we take a subset of some finitely generated (free) monoid as the set of objects to identify the nodes. We introduce various ways to define the edges, utilizing the familiar concepts in the theories of formal languages, automata, and semigroups. Section 2 is a preliminary for these concepts, and in Section 3, we discuss how the graphs are described and classified by these concepts.

The more general or powerful the method to define the set of nodes and edges, the richer or greater the class of graphs which are defined by it. So, we have some hierarchies of the classes of graphs after the Chomsky hierarchy in the theory of languages.

Section 4 is devoted to the results about the inclusion relationships among these classes.
In the last section, we discuss the possible applications of our description methods of infinite labeled directed graphs, and some extensions and problems of our research in this direction.

2. PRELIMINARIES

A. Graphs

A directed graph $D$ is a quadruple $(X, E, s, t)$ where $X$ is a set of nodes, $E$ is a set of (directed) edges, $s, t: E \rightarrow X$ are functions denoting the source and target of each edge.

The $r$th neighbor of a node $x \in X$, $N_D^r(x)$ is defined inductively for every integer $r$ as (i) $N_D^0(x) = \{x\}$, (ii) for $r > 0$, $N_D^r(x) = N_D^{r-1}(x) \cup \{ y \in X \mid \exists e \in E, \exists y' \in N_D^{r-1}(x), s(e) = y', t(e) = y \}$, (iii) for $r < 0$, $N_D^r(x) = N_D^{r+1}(x) \cup \{ y' \in X \mid \exists e \in E, \exists y \in N_D^{r+1}(x), s(e) = y', t(e) = y \}$.

The input boundary of a node $x$ is a set of edges $\partial^-(x) = \{ e \in E \mid t(e) = x \}$ and the output boundary of $x$ is $\partial^+(x) = \{ e \in E \mid s(e) = x \}$.

A graph $D = (X, E, s, t)$ is finite input if $\#(\partial^-(x)) < \infty$ for all $x \in X$. $D$ is bounded input if there is a natural number $k$ and $\#(\partial^-(x)) < k$ for every $x \in X$. Similar for output and $\partial^+$.

A labeling of a graph $D = (X, E, s, t)$ is a quadruple $\xi = (\Sigma_X, \Sigma_E, \xi_X, \xi_E)$, where $\Sigma_X$ and $\Sigma_E$ are finite sets of node- and edge-labels, $\xi_X: X \rightarrow \Sigma_X$ and $\xi_E: E \rightarrow \Sigma_E$ are labeling functions of nodes and edges. Then, a labeled directed graph (an LD graph) is an octuple $D = (X, E, s, t, \Sigma_X, \Sigma_E, \xi_X, \xi_E)$ and shown as the diagram following.

$$
\begin{array}{c}
\Sigma_E \xleftarrow{\xi_E} E \xrightarrow{\xi_X} X \xrightarrow{\xi_X} \Sigma_X.
\end{array}
$$

Let $D = (X, E, s, t, \Sigma_X, \Sigma_E, \xi_X, \xi_E)$ and $D' = (Y, F, u, v, \Gamma_Y, \Gamma_F, \eta_Y, \eta_F)$ be two LD graphs. $D$ and $D'$ are structurally isomorphic if there are bijections $\pi_X: X \rightarrow Y$ and $\pi_E: E \rightarrow F$, and the following diagram is commutative:

$$
\begin{array}{c}
\Sigma_E \xleftarrow{\xi_E} E \xrightarrow{s} X \xrightarrow{\xi_X} \Sigma_X \\
\pi_E \downarrow \quad \pi_X \downarrow \\
\Gamma_F \xleftarrow{\eta_F} F \xrightarrow{u} Y \xrightarrow{v} \Gamma_Y
\end{array}
$$

If $D$ and $D'$ are structurally isomorphic, they are considered as the same graph with different labelings.

Let $\xi = (\Sigma_X, \Sigma_E, \xi_X, \xi_E)$ and $\eta = (\Gamma_X, \Gamma_E, \eta_X, \eta_E)$ be two labelings of a directed graph $(X, E, s, t)$. Then $\xi$ is label morphic to $\eta$, if there are two
functions \( \rho_X : \Sigma_X \to \Gamma_X \), \( \rho_E : \Sigma_E \to \Gamma_E \) and equations \( \rho_X \cdot \xi_X = \eta_X \), \( \rho_E \cdot \xi_E = \eta_E \) hold.

A labeling \( \xi = (\Sigma_X, \Sigma_E, \xi_X, \xi_E) \) of an input bounded directed graph \((X, E, s, t)\) is standard if the following conditions hold. (i) For any pair \((e_1, e_2) \in E \times E\), \( \xi_X(e_1) \neq \xi_X(e_2) \) if \( t(e_1) = t(e_2) \), (ii) for any pair \((x_1, x_2) \in X \times X\), if \( \xi_X(x_1) = \xi_X(x_2) \) then \( \xi_X(\partial^-(x_1)) = \xi_X(\partial^-(x_2)) \), (iii) if \( \xi_X(x_1) \neq \xi_X(x_2) \) then \( \xi_X(\partial^-(x_1)) \cap \xi_X(\partial^-(x_2)) = \emptyset \). (\( \xi_X(\partial^-(x)) = \{ \sigma \in \Sigma_E : \sigma = \xi_E(e), e \in \partial^-(x) \}. \)

Such generalizations of functions will be done without any comment, in this paper.

**Proposition 1.1.** If \( D = (X, E, s, t, \Gamma_X, \eta_X, \eta_E) \) is an input bounded labeled directed graph, there is a standard labeling \( \xi = (\Sigma_X, \Sigma_E, \xi_X, \xi_E) \) of \( D \), which is label morphic to \( \eta = (\Gamma_X, \Gamma_E, \eta_X, \eta_E) \).

**Proof.** (i) For every node \( x \in X \), we define a function \( \tilde{\xi}_x : \Gamma_E \to \mathbb{N} \) by \( \tilde{\xi}_x(y) := \#(\{ e \in \partial^-(x) \mid \eta_E(e) = y \}) \). (\( \mathbb{N} \) is the set of all nonnegative integers.) As \( \#(\partial^-(x)) \) is bounded, the set \( \{ \tilde{\xi}_x(x) \mid x \in X \} \) is finite. Then let \( \Sigma_X = \{ (\eta_X(x), \tilde{\xi}_x) \mid x \in X \} \) and \( \xi_x(x) = (\eta_X(x), \tilde{\xi}_x) \) for all \( x \in X \).

(ii) Let \( \#(\Gamma_E) = l \) and \( \Gamma_E = \{ \gamma_1, \gamma_2, \ldots, \gamma_l \} \). The edges of \( \partial^-(x) \) are numbered by the following procedure, for every node \( x \in X \).

**Step 1.** \( k := 0, i := 1. \)

**Step 2.** The \( \tilde{\xi}_x(\gamma_i) \) edges of \( \partial^-(x) \) with label \( \gamma_i \) get the numbers from \( k + 1 \) to \( k + \tilde{\xi}_x(\gamma_i) \) in an arbitrary order.

**Step 3.** \( k := k + \tilde{\xi}_x(\gamma_i), i := i + 1. \) If \( i \leq n \) then go to Step 2, else stop. (\( n = \max\{ \#(\partial^-(x)) \mid x \in X \}. \)

By this procedure, the edges of \( \partial^-(x) \) are associated to mutually different numbers \( 1, 2, \ldots, \#(\partial^-(x)) \). We denote this number of an edge \( e \in E \) by \( \tilde{\xi}_x(e) \).

(iii) Now, let \( \Sigma_E = \{ (\eta_X(x), \tilde{\xi}_x, \tilde{\xi}_E(e)) \mid e \in E, x := t(e) \} \) and \( \xi_E(e) = (\eta_X(t(e)), \tilde{\xi}_x(t(e)), \tilde{\xi}_E(e)) \).

Then, it would be easily verified that \( \xi \) is a standard labeling of \( D \), and a pair of functions \( (\rho_X, \rho_E) \) defined by \( \rho_X(\eta_X(x), \tilde{\xi}_x, \tilde{\xi}_E(e)) = \eta_X(x) \) and \( \rho_E(\eta_X(t(e)), \tilde{\xi}_x(t(e)), \tilde{\xi}_E(e)) = \eta_E(e) \) is a label morphism from \( \xi \) to \( \eta \).

For a standardly labeled directed graph \( D = (X, E, s, t, \Sigma_X, \Sigma_E, \xi_X, \xi_E) \) (which, of course, is an input bounded graph), we introduce another description as follows.

(i) \( \Sigma_X = \{ \sigma_1, \sigma_2, \ldots, \sigma_m \} \) is replaced by a set of natural numbers \( M = \{ 1, 2, \ldots, m \} \) and \( X_i \) is defined by \( X_i = \{ x \in X \mid \xi_X(x) = \sigma_i \} \) for \( i = 1, 2, \ldots, m \). Then, the sequence \((X_1, X_2, \ldots, X_m)\) conserves all the information contained in \( X \) and \( \xi_X \).

643/44/2-5
(ii) As the labeling of $D$ is standard, $\Sigma_E$ can be divided into mutually
disjoint sets $\Sigma_{E_1}, \Sigma_{E_2}, \ldots, \Sigma_{E_m}$, where $\Sigma_{E_i} = \xi_E(\xi^{-1}(x))$ for $\xi_E(x) = \sigma_i$. If we
number each symbol of $\Sigma_{E_i}$ in some order, a pair of numbers $(i, j)$ indicates
just one symbol of $\Sigma_{E_i}$, which is the $j$th symbol of the set $\Sigma_{E_i}$. Then the edge
label alphabet $\Sigma_E$ can be replaced by a set of pairs of numbers $N = \{(i, j) | i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n_i\}$ where $n_i = \text{card}(\Sigma_{E_i})$.

(iii) As $D$ is standardly labeled, it is clear that an edge $e \in E$ can be
identified solely by $t(e)$ and $\xi_E(e)$ or $(i, j)$. All the information that an edge $e$
carries in the graph $D$ are (a) its existence, (b) its source $s(e)$ and its target $t(e)$, and
(c) its label $\xi_E(e)$. But the same information can be expressed by a set of functions $\{f_{ij} | f_{ij} : X_i \rightarrow X, (i, j) \in N, f_{ij}(x) = y \text{ if and only if there is an edge } e \in E, s(e) = y, t(e) = x, \xi_E(e) = (i, j)\}$.

Note that, for every node $x \in X_i$, there is always a unique edge $e$ such that
t(e) = x and $\xi_E(e) = (i, j)$.

So, summing up the discussion above, we can construct a canonical description
system $(X_1 : f_{11} \cdot f_{12} \cdots \cdot f_{1n_1} ; X_2 : f_{21} \cdot f_{22} \cdots \cdot f_{2n_2} \cdots : X_m : f_{m1} \cdot f_{m2} \cdots \cdot f_{mn_m})$ for
any standardly labeled directed graph $D$. Conversely, whenever such a system
which satisfies the following conditions is given, we can regard it as a definition
of a standardly labeled directed graph. (i) $X_i \cap X_j = \emptyset$ for every $i \neq j$.
(ii) $X = \bigcup_{i=1}^{m} X_i$, then for every function $f_{ij}$, $\text{Dom } f_{ij} = X_i$ and $\text{Im } f_{ij} \subseteq X$.

By the grace of Proposition 1.1, the canonical description is also possible
for every input bounded labeled directed graph.

B. Languages, Automata, and Monoids

The languages appearing in this paper are of the families of languages which
are recursively enumerable, recursive, context-sensitive, context-free, regular, and
finite, and they are abbreviated as RE, RC, CS, CF, RG, and FN, respectively.
The functions are classified as recursively enumerable functions, recursive func-
tions, generalized sequential machine mappings, and homomorphisms, and their
abbreviations are RE-functions, RC-functions, GSM-functions, and HMF-
functions, respectively.

For the definitions and results concerning with these languages and functions,
please see, for example, "Formal Languages" (Salomaa, 1973). Here we only
supplement the definitions of RE- and RC-functions.

A partial function $f : \Sigma^* \rightarrow \Sigma^*$ is an RE-function if there is a deterministic
Turing machine with a tape which is infinite only to the right, and this machine
calculates $f(w)$ for every $w \in \Sigma^*$ in the following manner. (i) Initially, $w$
is written from the left end of the tape and the head of the Turing machine is
situated at the left end of the tape in its predetermined initial state. Then the
machine begins its calculation. (ii) If the machine falls into one of the predeter-
mined halting states, the string written on the tape from the left end to the
present position of the head of the machine is considered as the function value
FAMILIES OF INFINITE GRAPHS

A partial function \( g: \Sigma^* \to \Sigma^* \) is an \( RC \)-function if and only if there is a total \( RE \)-function \( g: \Sigma^* \to \Sigma^* \cup \{ \text{"UNDETERMINED"} \} \) such that \( g(w) = g(w) \) for all \( w \in \text{Dom} \) and \( g(w) = \text{"UNDETERMINED"} \) for all \( w \in \Sigma^* - \text{Dom} g \). \( (\Sigma^* \cap \{ \text{"UNDETERMINED"} \}) = \varnothing \).

A finite generating system of a monoid is a pair \( (A, P) \), where \( A = \{ a_1, a_2, \ldots, a_n \} \) is a finite set of generators, and \( P = \{ \alpha_1 = \beta_1, \alpha_2 = \beta_2, \ldots, \alpha_k = \beta_k \} \) is a finite set of fundamental relations of the monoid, where \( \alpha_i, \beta_i \in A^* \) for \( i = 1, 2, \ldots, k \).

For every \( \gamma, \delta \in A^* \), \( \gamma \alpha_i \delta = \gamma \beta_i \delta \) is a fundamental equivalence, if and only if \( \alpha_i = \beta_i \in P \). The equivalence relation \( \sim \) in \( A^* \) is the transitive and reflexive closure of the fundamental equivalences. Then the monoid \( M = \langle G, \cdot \rangle \) : \( \langle A, P \rangle_m \) is defined as follows. (i) The set of elements of \( M \), i.e., \( G \), is the set of all equivalence classes, \( A^*/\sim \). (ii) Let the equivalence class to which \( w \in A^* \) belongs be denoted as \( C(w) \), then the binary operation \( \cdot \) of \( M \) is defined by \( C(w) \cdot C(w') = C(ww') \). (\( ww' \) is the concatenation of \( w \) and \( w' \)).

Two generating systems \( (A, P) \) and \( (A', P') \) are isomorphic if there is a bijective homomorphism \( h: A^* \to A'^* \), and \( P' = \{ h(\alpha_i) = h(\beta_i) \mid \alpha_i = \beta_i \in P \} \). It must be noted that the isomorphism of two finitely generated monoids \( M = \langle A, P \rangle_m \cong M' = \langle A', P' \rangle_m \), do not necessarily imply the isomorphism of their generating systems \( (A, P) \) and \( (A', P') \). For example, let \( A = \{ a_1, a_2, \ldots, a_k \} \), \( P = \{ a_ia_j = a_ja_i \mid 1 \leq i, j \leq 8 \} \cup \{ a_2 = a_3a_2, a_4 = a_4a_5, a_6 = a_6a_7, a_8 = a_2a_1 \} \cup \{ a_ia_j = \epsilon \mid i = j \pmod{4} \} \), where \( \epsilon \) is the unit element\} and \( A' = \{ b_1, b_2, b_3, b_4 \} \), \( P' = \{ b_i b_j = b_j b_i \mid 1 \leq i, j \leq 4 \} \cup \{ b_1 b_3 = \epsilon, b_2 b_4 = \epsilon \} \). Both \( \langle A, P \rangle_m \) and \( \langle A', P' \rangle_m \) are isomorphic to the two-dimensional free Abelian group \( \mathbb{Z}^2 \), but at any rate \( (A, P) \) is not isomorphic to \( (A', P') \). The reader will see that \( (A, P) \) and \( (A', P') \) correspond to the Moor-neighborhood and Neumann-neighborhood of two-dimensional cellular space, respectively.

3. Several Methods for Defining Graphs

In the previous section, graphs were given in a very abstract or vague manner, because the (possibly) infinite sets of nodes and edges were not rigorously characterized by specified or formalized descriptions. The main purpose of this section is to establish the formal methods which can define infinite graphs by finite descriptions. Although, this is not our final aim, because, if we only wish to have finite descriptions of infinite graphs, we already have the method which may be called the number theoretical description. By this method, we first identify every node of a graph by a natural number, and we formally (for example, using Turing machines), define several recursive functions which calculate the label, the neighbors, and the label of each edge going
to a neighboring node, for every natural number which identifies a node. For the natural numbers which do not identify any node, we assume the value "undetermined" or "0" for these functions. This is, in the sense of the theory of languages, one of the most general or powerful method of defining graphs. But, unfortunately, this method often leads us to extremely artificial and complicated descriptions even for most (intuitively) simple graphs, for example, \( n \)-dim array structures of \( \mathbb{Z}^n \). So we are going to look for more flexible methods which can supply the description having the characters as follows. (i) Intuitively simple graphs have simple descriptions. (ii) The construction of a real image of a graph from its formal description is easy for both human intuitions and computers. (iii) Various styles of description of graphs (which are sometimes not so rigorously stated), must be included or easily reformulated in our method of description. For example, strings of the symbols 0 and 1 seem to be the most adequate identifiers of the nodes of binary trees, and on the other hand, the nodes of \( n \)-dim array structures are usually identified by \( n \)-tuples of integers. These different kind of expressions will be both included in our formal descriptions of graphs and will be discussed in the same contexts. We regard the last condition as the most important one, because, when it is fulfilled, we can utilize the original ideas and skillful techniques of many authors who bring forth new types and new concepts of graphs, and then conditions (i) and (ii) will naturally be satisfied.

From now on, our objects are restricted to the graphs which are \textit{weakly connected} and \textit{standardly labeled}, and these conditions are always presumed for every graph appearing in Sections 3 and 4.

Our reason for this restriction will be discussed in the last section.

\textbf{Definition 3.1.} A regular concatenative expression \( R \) of a (weakly connected standardly labeled) graph \( D \) is a system \((\Sigma; Y_1; R_{11}, R_{12}, \ldots, R_{1n_1}; \ldots; Y_m; R_{m1}, R_{m2}, \ldots, R_{mn_m}; \omega)\), where \( \Sigma \) is a finite alphabet, \( Y_i \subseteq \Sigma^* \) is a regular set (for \( 1 \leq i \leq m \)), \( R_{ij} \subseteq \Sigma^x \Sigma^* \) is a regular set on \( \Sigma \cup \{\epsilon\} \) (for \( 1 \leq i \leq m \), \( 1 \leq j \leq n_i \)), and \( \omega \in Y = \bigcup_{i=1}^{m} Y_i \). Furthermore, conditions (i) \( Y_i \cap Y_k = \emptyset \) for \( \forall i, k, i \neq k \), and (ii) \( \text{card}(R_{ij} \cap (\Sigma^x \epsilon \Sigma)) = 1 \) for every \( ij \), must be satisfied.

The interpretation of this expression is as follows. We define inductively the relation \( \sim \) on \( Y \) by (i) \( y \sim y \) for every \( y \in Y \), (ii) if \( yey' \in R_{ij} \) for some \( ij \), then \( y \sim y' \), (iii) if \( y \sim y' \) then \( y' \sim y \), and (iv) if \( y \sim y' \) and \( y' \sim y'' \) then \( y \sim y'' \). As \( \sim \) is an equivalence relation, we divide \( Y \) by \( \sim \), and let the equivalence class containing \( \omega \) be denoted as \( X \). Then the canonical description system of \( D =: (X_1; f_{i1}, \ldots, f_{in_1}; \ldots; X_m; f_{m1}, \ldots, f_{mn_m}) \) is given by (i) \( X_i = X \cap Y_i \), (ii) \( f_{ij}(x) = T_c(R_{ij} \cap (\Sigma^x \epsilon \Sigma)) \) for every \( x \in X_i \), where \( T_c: \Sigma^x \epsilon \Sigma^* \rightarrow \Sigma^* \) is defined by \( T_c(y \epsilon x) = y \). (Similarly, we define \( S_c: \Sigma^x \epsilon \Sigma^* \rightarrow \Sigma^x \) by \( S_c(y \epsilon x) = x \).)

In other words, \( Y_i \)'s are the sets of nodes (identifiers of nodes) with label \( i \), and \( R_{ij} \)'s are the sets of edges with label \( ij \) on them, but we only pick up the
weakly connected component to which \( \omega \) belongs. Note that this last operation extremely strengthens the defining ability of our description method, contrasted with the case that we would define the set of the nodes of the graph, \( X \), to be exactly equal with \( Y \).

If the words “regular” in Definition 3.1 are all replaced by “context-free,” “context-sensitive,” or “recursive,” then we get the definitions of context-free-, context-sensitive-, and recursive-concatenative expressions, respectively. We denote the class of the graphs which can be defined or expressed by a regular concatenative expression by \( G(R, C, RG) \), where “\( R' \)” means that the edges of the graph are expressed as “relations” in \( \Sigma^* \), and “\( C \)” means that these relations are expressed as the set of the concatenations of the identifiers of the two nodes which are connected by an edge (which is symbolized as a letter “\( c \)” in the expression). In the same way, we define \( G(R, C, CF) \), \( G(R, C, CS) \), and \( G(R, C, RC) \). We can also define \( G(R, C, RE) \) which is the class of graphs whose node sets \( Y_i \)'s and edge sets \( R_{ij} \)'s are recursively enumerable sets. But then, because of the condition (ii) of Definition 3.1, every \( R_{ij} \cap \Sigma^*cY \) must actually be a recursive set, and the sole procedure which possibly does not halt is the procedure to determine whether a string \( x \in \Sigma^* \) is a member of \( Y \).

For example, let all Turing machines be enumerated by some order, and \( M(i) \) be the 1) Turing machine. We consider a graph \( D_1 = (\Sigma : X_1 ; R_{11} : X_2 ; R_{21} : \omega) \) where \( \Sigma = \{a, b\}, X_1 = a^+, R_{11} = \{a^{i-1}ca^i \mid i \in \mathbb{N}\}, X_2 = \{b^i \mid M(i) \) is a Turing machine which halts (for a initially blank tape), \( R_{21} = \{a^i cb^i \mid i \in \mathbb{N}\}, \) and \( \omega = a \). Then, it would be clear that \( D_1 \in G(R, C, RE) \) -- \( G(R, C, RC) \).

While a very similar graph \( D_2 = (\Sigma : X_1 ; R_{11} : X_2 ; R_{21} ; R_{22} : \omega) \) is not a member of \( G(R, C, RE) \). \( (\Sigma, X_1, R_{11}, , \omega, a, b) \) are the same as that of \( D_1, \) and \( X_2 = a^+, R_{21} = \{a^i cb^i \mid M(i) \) halts\} \( \cup \{b^i cb^i \mid M(i) \) does not halt\} \) and \( R_{22} = \{b^i cb^i \mid i \in \mathbb{N}\}. \)

In the definition above, we wrote the names or the identifiers of two nodes which are connected, in the “concatenative” form. But if we take up the recognition processes which read the two names in parallel, then we get another kind of descriptions of graphs.

**Definition 3.2.** Let \( \Sigma = (\Sigma \cup \{e\}) \times (\Sigma \cup \{e\}) \rightarrow (e, e) \), where \( e \in \Sigma \), and two homomorphisms \( S_p \) and \( T_p : (\Sigma^2)^* \rightarrow \Sigma^* \) be defined by \( S_p(a, b) = a \) for \( a \in \Sigma, S_p(e, b) = e \), and \( T_p(a, b) = b \) for \( b \in \Sigma, \) \( P_p(e, a) = e \). (\( e \) is the unit element of the monoid \( \Sigma^* \)). Then, a regular direct productive expression \( R \) of a graph \( D \) is a system \( (\Sigma, X_1, R_{11}, ..., R_{1n_1}, ..., Y_1, ..., Y_m, R_{m1}, ..., R_{mn_m}, \omega) \). The sole distinction from the regular concatenative expression is that, every \( R_{ij} \) is a regular set on \( \Sigma^2 \), so the condition (ii) of Definition 3.1 must be modified as “\( \text{card}(R_{ij} \cap T_p^{-1}(y)) \leq 1 \) for \( y \in Y, \) for every \( ij \).” A word \( w \) in \( R_{ij} \subseteq (\Sigma^2)^* \) is interpreted as an edge from the node \( S_p(w) \) to \( T_p(w) \), with label \( ij \) on it.

The class of graphs which can be expressed by some regular direct productive expression is denoted by \( G(R, D, RG) \), where “\( D \)” is obviously the initial
letter of "direct product." $G(R, D, CF)$, $G(R, D, CS)$, $G(R, D, RC)$, and $G(R, D, RE)$ are defined in the same way.

Different from the concatenative expression, the direct productive expression of an edge is ambiguous, in the sense that two or more different words $w_1, w_2, ..., w_r$ can occur in the same $R_{ij}$ expressing the same edge, i.e., $S_p(w_1) = ... = S_p(w_r)$ and $T_p(w_1) = ... = T_p(w_r)$. As $(e, e)$ is not the member of $\Sigma^2$, the inequality $s \leq |w_i| \leq s \cdot t$ holds, and the exact upper bound of $r$ is given by $\sum_{k=0}^{r} s \cdot k \cdot C_k \cdot s \cdot C_{s+k-t}$ where $s = \max(|S_p(w_i)|, T_p(w_i))$ and $t = \min(|S_p(w_i)|, T_p(w_i))$. To avoid this ambiguity of expression, we can restrict every $R_{ij}$ to a regular set which satisfies the inclusion, $R_{ij} \subseteq (\Sigma \times \Sigma)^* \cdot \{(\Sigma \times \{e\})^* \cdot (\{e\} \times \Sigma)^* \cdot \{(e, a) : a \in \Sigma\}\}$.

Such a restriction causes another family of the classes of graphs, $G(R, D', RG)$, $G(R, D', CF)$, etc.

Now, we return to the canonical description of graphs, where the edges are expressed by functions $f_{ij}$'s. If we keep identifying nodes by members of a finitely generated free monoid, we can apply the theory of automata for the characterization of $f_{ij}$'s.

**Definition 3.3.** A homomorphism expression $H$ of a graph $D$ is defined by a system $(\Sigma : Y_1 : h_{11}, h_{12}, ..., h_{i1} : ... : Y_m : h_{m1}, h_{m2}, ..., h_{mn} : \omega)$ where $h_{ij} : Y_i \rightarrow Y$ ($Y := \bigcup_{i=1}^{m} Y_i \subseteq \Sigma^*$) is a homomorphism of $\Sigma^*$ for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n_i$, and every $Y_i$ is a mutually disjoint regular subset of $\Sigma^*$. $\omega \in \Sigma^*$ is the origin node which specifies the weakly connected component by the same manner that we described in Definition 3.1.

We define some classes with more weak constraints on the sets $Y_i$'s and the functions $h_{ij}$'s. It is reasonable to claim that these constraints are of the same degree by means of the theory of languages and automata. So, we define four classes as follows, where the constraints on $Y_i$'s and $h_{ij}$'s seem to be balanced or matched. "$F$" is the abbreviation of "function," because the edges of the graph are regarded as actions of functions, in this expression.

$G(F, HM)$: every $Y_i$ is a regular set and every $h_{ij}$ is an $HM$-function,
$G(F, GSM)$: every $Y_i$ is a regular set and every $h_{ij}$ is a $GSM$-function,
$G(F, RC)$: every $Y_i$ is a recursive set and every $h_{ij}$ is an $RC$-function,
$G(F, RE)$: every $Y_i$ is an $RE$-set and every $h_{ij}$ is an $RE$-function.

Note that, in the expressions utilizing languages—concatenative or direct productive expressions—the operations to determine $N_{D^1}(x)$ and to determine $N_{D^3}(x)$ for its arbitrary node $x$ were not essentially different. Both processes were accomplished by intersection of $R_{ij}$'s with a very simple regular language determined only by $x$ and the style of expression. For example, in concatenative expression, $x \Sigma^*$ for $N_{D^1}(x)$ and $\Sigma^* cx$ for $N_{D^3}(x)$. As we have taken the language classes $RG$, $CF$, $CS$, $RC$, and $RE$, each of which is closed under intersection
with regular languages, the tracing of graphs in both directions does not cause any increasing difficulty. However, in this functional expression, the essential difficulty of calculation is usually very different for the determination of $N_P'(x)$ and $N_P^{-1}(x)$. But, in our choice of the classes of functions shown above, the difficulty seems quite even in the tracing of both directions—$h_i(x)$ and $h_i^{-1}(x)$.

The class of group graphs introduced by Wagner (1965) is a useful and important concept in the theory of graphs, especially in the domain relating with the theory of automata, for this class is characterized by the highest symmetry—uniform structure. So, we take it as the third method for the description of graphs, while, as we were aiming at more general structures or the structures with more degraded symmetry, we generalize it to the monoid graphs.

**Definition 3.4.** The graph $G$ of a finite generating system of a monoid $(A, P)$, which we denote by $\langle A, P \rangle$, is defined as follows.

(i) The nodes of $G$ are the all elements of $M = \langle A, P \rangle$ .

(ii) There is an edge labeled with $j$, from $a$ to $b$, if and only if the equation $\alpha = \beta a_j$ holds in $M. (a_j \in A.)$

Obviously from the definition, every monoid graph (more precisely, a graph of a finite generating system of a monoid) is a graph with unique node label.

Note that the correspondence of $(A, P)$ and $\langle A, P \rangle$ is not bijective. However, different from the case of $(A, P)_m = \langle A, P \rangle$ if and only if $A = A'$ and $\langle A, P \rangle_m = \langle A', P \rangle_m$.

The class of monoid graphs is denoted as $G(G, MON)$, where “$G$” is the abbreviation of “generating system.” We also abbreviate the words “group” and “Abelian group” as “GRP” and “AB,” respectively. Then the following inclusions are obvious, and it would also be clear that they are proper

$$G(G, MON) \supseteq G(G, GRP) \supsetneq G(G, AB).$$

Until now, graphs are strictly distinguished from their expressions. But, from now on, we ignore this distinction, and the lines such as $D = (\Sigma : Y_1 ; R_{11}, ..., R_{1w_1} : \cdots : Y_m ; R_{m1}, ..., R_{mm} : \omega)$ will be written, unless they cause any confusions.

Here, we show some examples of our methods of the description of graphs, which will be utilized in the proofs of the theorems of the next section.

**Example 3.1.** The graph $D_1$ shown in Fig. 1 can be expressed as a member of $G(R, C, CF), G(R, D', RG), G(F, HM)$, and $G(G, MON)$. In fact,

(i) $D_1 = (\Sigma : X ; R_1, R_2 : \omega), \Sigma = \{a, b\}, X = (a^*b^* \cup (\Sigma^2)^* \cup (b^*a^* \cup \Sigma \cdot \Sigma^2)^*), R_1 = \{sacx^\alpha | x \in (\Sigma^2)^* \cup \{b\alpha c^\alpha \cup \alpha \in \Sigma \cdot (\Sigma^2)^*\}, R_2 = \{b\alpha c^\alpha \cup \alpha \in \Sigma \cdot (\Sigma^2)^*\}.
(\Sigma^2)^* \cup \{aaca^e \mid a \in \Sigma \cdot (\Sigma^2)^*\}, and \omega = e, where e is the unit element of the free monoid \Sigma^*, and a^e is the reversed word or mirror image of the word a.

(ii) \(D_1 = ([a, b] : a^*b^*; R_1', R_2 : \epsilon),\) where \(R_1' = \{x \cdot (a, e) \mid x \in (a, a)^* \cdot (a, b) \cdot (b, b)^*\} \cup \{x \cdot (a, e) \cdot a \in (a, a)^*\},\) and \(R_2' = \{x \cdot (b, e) \cdot a \in (a, a)^* \cdot (b, b)^*\}.\)

(iii) \(D_1 = ([\#, @, a, b] : #a^*b^*@; h_1, h_2 : @[#]),\) where \(h_1\) and \(h_2\) are homomorphisms defined by \(h_1(a) = a, h_1(b) = b, h_1(\#) = @, h_1(@) = @;\) and \(h_2(a) = a, h_2(b) = b, h_2(\#) = @, h_2(@) = @.\)

(iv) \(D_1 = \langle A, P \rangle \), where \(A = \{a, b\},\) and \(P = \{ab = ba\}.\)

**Example 3.2.** The graph in Fig. 2 is a member of \(G(R, C, CF), G(R, D', RG), G(F, GSM)\) and \(G(G, AB).\)

(i) \(D_2 = ([a, b] : a^* \cup b^*; R_1, R_2 : \epsilon),\) where \(R_1 = \{aaca^e \mid a \in a^*\} \cup \{xcbx, a \in b^*\}\) and \(R_2 = \{ab^2cx \mid a \in b^*\} \cup \{axca^2a \mid a \in a^*\} \cup \{bca\}.\)

(ii) \(D_2 = ([a, b] : a^* \cup b^*; R_1', R_2 : \epsilon),\) where \(R_1' = \{x \cdot (e, a) \mid x \in (a, a)^* \cup \{x \cdot (a, e) \cdot a \in (a, b)^*\}\) and \(R_2' = \{x \cdot (b, e) \cdot a \in (b, b)^* \cup \{x \cdot (b, e) \mid a \in (b, b)^*\} \cup \{(a, b)\}.\)

(iii) \(D_2 = ([a, b, \#, @]; #b^*@ \cup a^*@; f_1, f_2 : #@[@]),\) where \(f_1\) and \(f_2\) are GSM-functions defined by the transition tables, Table I and II, respectively. (Initial state is always the first column.)

(iv) \(D_2 = \langle [a, b], \{ab = ba, aab = 0\} \rangle,\) where 0 is the unit element of the group of this graph (which is isomorphic to the module of \(\mathbb{Z}\)).
TABLE I
The Transition (Next State/Output) Table of the GSM for $f_1$ of Example 3.2(iii)

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>$s_0$</th>
<th>$s_1$</th>
<th>$s_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>$s_0/#$</td>
<td>$s_2/##$</td>
<td>$s_1/##$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$s_1/e$</td>
<td>$s_1/a$</td>
<td>$s_1/##$</td>
</tr>
<tr>
<td></td>
<td>$b$</td>
<td>$s_2/bb$</td>
<td>$s_2/##$</td>
<td>$s_2/b$</td>
</tr>
<tr>
<td></td>
<td>$@$</td>
<td>$s_1/b@$</td>
<td>$s_1/@$</td>
<td>$s_1/@$</td>
</tr>
</tbody>
</table>

* We always put the initial state at the first column.

TABLE II
The Transition (Next State/Output) Table of the GSM for $f_2$ of Example 3.2(iii)

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>$s_0$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>$s_0/#$</td>
<td>$s_2/##$</td>
<td>$s_1/##$</td>
<td>$s_1/##$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$s_1/aa$</td>
<td>$s_1/a$</td>
<td>$s_1/##$</td>
<td>$s_1/##$</td>
</tr>
<tr>
<td></td>
<td>$b$</td>
<td>$s_2/e$</td>
<td>$s_2/##$</td>
<td>$s_2/e$</td>
<td>$s_1/b$</td>
</tr>
<tr>
<td></td>
<td>$@$</td>
<td>$s_1/aa@$</td>
<td>$s_1/@$</td>
<td>$s_1/a@$</td>
<td>$s_1/@$</td>
</tr>
</tbody>
</table>

4. Inclusion Relations

In Section 3, we defined several hierarchical classifications of graphs, $G(R, \delta, \gamma)$, $G(F, \eta)$, and $G(G, \xi)$, where $\delta \in \{C, D, D\}'$, $\gamma \in \{RG, CF, CS, RC, RE\}$, $\eta \in \{HM, GSM, RC, RE\}$, and $\xi \in \{MON, GRP, AB\}$. In this section, we investigate the inclusion relations among these classes. The result can be seen in Fig. 4.

It should be noted that the proper inclusions in the Chomsky hierarchy, automata hierarchy, and that of monoids do not imply that the corresponding inclusion relations are proper ones. It is because of the freedom of the naming of nodes, which gives rise to the possibility that the graph can be expressed more easily under some other namings. For example, in the expression $\{(a, b) : a^*b^*; R_1 = \{axc\alpha, \alpha \in \Sigma^*\}, R_2 = \{xbc\alpha, \alpha \in \Sigma^*\} : \epsilon\}$, $R_1$ and $R_2$ are proper
members of the class of context-sensitive languages with respect to the context-
free languages, although this expresses the same graph as Example 3.1 where
we have given a context-free concatenative expression of the graph.

The same can be said for the inclusions \( G(\mathcal{R}, \mathcal{D}, \gamma) \supseteq G(\mathcal{R}, \mathcal{D}', \gamma) \) and
\( G(\mathcal{R}, \mathcal{D}, \gamma) \supseteq G(\mathcal{R}, \mathcal{C}, \gamma) \) (\( \gamma \in \{ \mathcal{R}G, \mathcal{C}F, \mathcal{C}S, \mathcal{R}C, \mathcal{R}E \} \)). The first inclusion is
obvious, and the latter is easily verified using the coding \( \varphi: \Sigma^* \Sigma^* \rightarrow (\Sigma^2)^* \),
where \( \varphi(a_1 \cdots a_k c b_1 \cdots b_i) = (a_1, e)(a_2, e) \cdots (a_k, e)(e, b_i) \cdots (e, b_i) \).

**Theorem 4.1.**

\[
G(\mathcal{R}, \mathcal{C}, \mathcal{R}C) = G(\mathcal{R}, \mathcal{D}, \mathcal{R}C) = G(\mathcal{R}, \mathcal{D}', \mathcal{R}C) = G(\mathcal{F}, \mathcal{R}C),
\]
and

\[
G(\mathcal{R}, \mathcal{C}, \mathcal{R}E) = G(\mathcal{R}, \mathcal{D}, \mathcal{R}E) = G(\mathcal{R}, \mathcal{D}', \mathcal{R}E) = G(\mathcal{F}, \mathcal{R}E).
\]

**Proof.** (i) It would be clear that, in the process of the recognition of
the set \( R_{ij} \) by a Turing machine, little modification is needed from the
differences in the styles of the description of edges, "\( \mathcal{C} \)," "\( \mathcal{D} \)," and "\( \mathcal{D}' \)." So
\( G(\mathcal{R}, \mathcal{C}, \mathcal{R}C) = G(\mathcal{R}, \mathcal{D}, \mathcal{R}C) = G(\mathcal{R}, \mathcal{D}', \mathcal{R}C) \).

(ii) Let \( \mathcal{C}y \) be calculated from \( \mathcal{C}y \) by a Turing machine \( M_{ij} \) in the \( \mathcal{G}(\mathcal{F}, \mathcal{R}C) \)
style. Then we can construct a procedure or a Turing machine \( P_{ij} \) which
recognizes \( w = \mathcal{C}y \in \Sigma^* \Sigma^* \) in the following way.

**Step 1.** \( P_{ij} \) calculates \( \mathcal{C}y \) from \( \mathcal{C}y \) by imitating the movements of \( M_{ij} \) utilizing
only the tape space after "\( \mathcal{C} \)" of the input word \( \mathcal{C}y \). And the result will be the
tape written \( \mathcal{C}x \).

**Step 2.** \( P_{ij} \) compares the subwords before and after "\( \mathcal{C} \)," and if \( \mathcal{C} = \mathcal{C} \),
then accepts the input string \( \mathcal{C}y \).

If \( \mathcal{C}y \) is not an edge defined by \( M_{ij} \) then step 1 or 2 will halt and reject
the word \( \mathcal{C}y \). So, \( G(\mathcal{R}, \mathcal{C}, \mathcal{R}C) \supseteq G(\mathcal{F}, \mathcal{R}C) \).

(iii) Let \( \mathcal{C}y \) be in a recursive set \( R_{ij} \). Then we can construct a Turing
machine \( M_{ij} \) which calculates \( \mathcal{C} \) from \( \mathcal{C}y \) in the following steps.

**Step 1.** \( M_{ij} \) marks a distinguished boundary marker in the next tape
space of the input \( \mathcal{C}y \).

**Step 2.** \( M_{ij} \) generates the words of \( R_{ij} \) in some order. This is possible
because the set \( R_{ij} \) is recursively enumerable.

**Step 3.** Whenever a word \( \mathcal{C}x_k \mathcal{C}y_k \) of \( R_{ij} \) is generated, \( M_{ij} \) compares
the subword after \( \mathcal{C} (= \mathcal{C}y_k) \) with \( \mathcal{C}y \). If \( \mathcal{C} \mathcal{C}y_k \) then \( M_{ij} \) erases all tape symbols except
the subword between the marker and \( \mathcal{C} (= \mathcal{C}y_k) \), and then shifts the word \( \mathcal{C}x_k \)
to the left end of the tape. Else \( M_{ij} \) erases \( \mathcal{C}x_k \mathcal{C}y_k \) and returns to step 2, to generate
the next word of \( R_{ij} \), \( \mathcal{x}_{k+1} \mathcal{C}y_{k+1} \).
It is certain that \( M_{ij} \) halts for every word \( y \in X_i \) because of the property of \( R_{ij} \) assumed in Definition 3.1, condition (ii). Thus, \( G(R, C, RC) \subseteq G(F, RC) \).

A very similar discussion must be repeated to establish the relations \( G(R, C, RE) = G(R, D, RE) = G(R, D', RE) = G(F, RE) \). So we omit the proof for these relations.

**Theorem 4.2.** \( G(R, C, CS) = G(R, D, CS) = G(R, D', CS) \).

**Proof.** Let there exist an edge from \( x \) to \( y \). Then the edge is expressed by a string with length \( l = |x| + 1 \cdot y \) in the concatenative expression, with length \( k \) such that \( \max\{x', y\} \leq k \leq |x' + y| \) in the direct productive expression, and with length \( m = \max\{x', y'\} \) in the restricted direct productive expression “\( D' \).”

It is known that the recognition capability of a linear bounded automaton, which accepts a context-sensitive language, can be simulated by another L.B.A, even if the input strings are coded into strings whose length is proportional to the originals, provided that the coding process is also done within the ability of some L.B.A.

So, the theorem holds from the inequalities \( k < l \leq 3k \) and \( m \leq k \leq 2m \).

**Theorem 4.3.** \( G(F, GSM) \supseteq G(F, HM) \).

**Proof.** The graph \( D_2 \) of Example 2 is a member of \( G(F, GSM) - G(F, HM) \). We assume that \( D_2 \in (\Sigma : X; h_1, h_2 : \omega) \in G(F, HM) \).

Now, as we have mentioned in the example, the graph is isomorphic to the module \( \mathbb{Z} \), and the homomorphism \( h_1 \) is equivalent to the operator “\( \cdot+1 \)” in \( \mathbb{Z} \). So every node in \( D_2 \) is identified by an integer, as \( x_i (i \in \mathbb{Z}) \), and \( h_1(x_i) = x_{i-1} \). As \( x_i \neq x_j \) for \( i \neq j \), \( \lim \sup_{i \to \infty} |x_i| \) and \( \lim \sup_{i \to \infty} |x_i| \) must be infinite. Therefore, for infinitely many \( i \in \mathbb{Z} \), \( |x_i| \leq i \cdot h_1(x_i) \).

So, for some symbol \( a \in \Sigma \), \( h_1(a) = \varepsilon \). Let \( \Sigma_1 = \{a \in \Sigma : 3k > 0, h_1(a) = \varepsilon\} \), and \( \Sigma_2 = \Sigma - \Sigma_1 \). Note that \( k \) cannot be greater than \( n = \text{card}(\Sigma) \). Let \( x \) be the number of occurrences of symbols of \( \Sigma_1 \) in the string \( x \), and \( x_{i2} \) be that of \( \Sigma_2 \), respectively. Then, it is impossible that \( \lim \sup_{i \to \infty} \ |x_i| \) is a nondecreasing function of \( i \). So we assume \( \lim \sup_{i \to \infty} \ |x_i| = M \) for some natural number \( M \). Let \( N = \max_{a \in \Sigma} h_1(a) \), then for any \( x_i \) such that \( |x_i| \leq M \), \( |x_{i+1}| \to |h_1(x_i)| \leq M \cdot N^n \). So, for infinitely many \( x_i \)'s, \( |x_i| \leq M \cdot N^n \). This implies that there exist integers \( i \) and \( j \), such that \( x_i = x_j \). This is a contradiction.

**Lemma 4.4.** Let \( D \in G(R, C, RG) \). Then there are only a finite number of nodes \( x \)'s such that \( \partial^+(x) \neq \emptyset \).

**Proof.** Let \( (\Sigma : X_1 ; R_{11} \ldots ; X_m ; R_{n1} \ldots ; R_{mn} ; \omega) \) be a regular
concatenative expression of $D$. Then, it suffices to prove that for every $R_{ij}$, only a finite number of strings in $\Sigma^*$ fulfill the condition $x \in \Sigma^* \cap R_{ij} \neq \emptyset$.

Now, considering the condition that the connector symbol "c" occurs exactly once in every word of $R_{ij}$, it must be in the regular expression $(T_1 c T'_1) \cup (T_2 c T'_2) \cup \cdots \cup (T_k c T'_k)$ where $l$ is a natural number and every $T_k$ and $T'_k$ are regular sets in $\Sigma^*$. Now, every $T_k$ must be a singleton $\{x_k\}$, for the condition (ii) of Definition 3.1—$\text{card}(R_{ij} \cap \Sigma^c) = 1$. So, only the nodes $x_1, x_2, \ldots, x_k, \ldots, x_l$ satisfy the condition $x \in \Sigma^* \cap R_{ij} \neq \emptyset$.

The graphs with such character can be illustrated schematically as Fig. 3. In the figure, part A is the set of nodes $x$'s where $\partial^+(x) \neq \emptyset$, and part B is the set of the other nodes.

The converse of Lemma 4.4 is also true. Namely,

**Lemma 4.5.** A graph $D$, such that for only a finite number of its nodes $x$'s, $\partial^+(x) \neq \emptyset$, can be expressed by a regular concatenative expression.

**Proof.** Our proof is an explanation of Fig. 3. First, we name every node in part A with mutually distinct symbols $a_1, a_2, \ldots, a_m$. Then we classify the nodes in part B by $N^{-1}(x) (\subset \Sigma_A = \{a_1, a_2, \ldots, a_n\})$ and by a vector $\xi_E(\partial^-(x))$, where $\xi_E$ is the labeling function of edges. Then, we assign the symbols $b_1, b_2, \ldots, b_n$ to each nonempty class of the node part B. If the cardinality of the class assigned $b_p$ is infinite, then we name every node of this class by every element of $b_p \cdot d^r$. If the cardinality is $r < \infty$, then the nodes of this class are $b_p \cdot d, b_p \cdot d^2, b_p \cdot d^3, \ldots, b_p \cdot d^r$.

Now, referring to the figure, it would be clear that every $R_{ij}$ is regular under this naming of nodes.

It is clear that every graph of this type is included in $G(F, HM)$ under this naming. On the other hand, the graph $D_1$ of Example 3.1, that we defined by the expressions in $G(R, D', RG)$ and in $G(F, HM)$, contains infinitely many nodes that $\partial^+(x) \neq \emptyset$. So, the following theorem holds.
THEOREM 4.6. \( G(R, D', RG) \supseteq G(R, C, RG) \), \( G(F, HM) \supseteq G(R, C, RG) \) and \( G(R, C, RG) \supseteq G(FN) \), where \( G(FN) \) is the class of all finite graphs.

THEOREM 4.7. \( G(R, C, CF) \supseteq G(R, D, RG) \).

Proof. Let \( R = (\Sigma : X_1 ; R_{11} \ldots , R_{1n_1} : \ldots ; X_m ; R_{m1} \ldots , R_{mn_m} : \omega) \) be a regular direct productive expression of a graph \( D \) in \( G(R, D, RG) \). We construct a context-free concatenative expression \( R' = (\Sigma' : X_1' ; R'_{11} \ldots , R'_{1n_1} : \ldots ; X_m' ; R'_m \ldots , R'_{mn_m} : \omega') \) of \( D \) as follows.

(i) \( \Sigma' : \Sigma \cup \{\#\} \), where \( \# \notin \Sigma, \# \notin \{e, c\} \). (Recall that \( e \) and \( c \) are also extra symbols with \( \Sigma \).)

(ii) \( X_i' = \{x \# x^n \mid x \in X_i\} \). Observing that \( X_i \) is a regular set, it is easy to imagine the DPDA which recognizes \( X_i' \). So every \( X_i' \) is a context free language.

(iii) \( \omega' : \omega \# \omega^o \).

(iv) \( R_{ij}' = \{x \# x^o c y \# y^o \mid x = S_p(w), y = T_p(w), w \in R_{ij}\} \).

It suffices to prove that every \( R_{ij}' \) is recognized by a nondeterministic pushdown automaton \( M_{ij} \).

Let \( R_{ij} \) be recognized by a complete deterministic finite automaton \( (S, \delta, s_0, S_f) \) where \( S \) is the state set, \( \delta: S \times \Sigma \rightarrow S \) is the transition function, \( s_0 \) is the initial state, and \( S_f \) is the final state set of the machine. As \( R_{ij}' \) is also a regular set, we assume that \( R_{ij}' \) is recognized by a complete deterministic finite automaton \( (S', \delta', s'_0, S'_f) \).

We define a pushdown automaton by a system \( (I, \Xi, T, \varphi, t_0, \xi_0, T_f) \) where \( I \) and \( \Xi \) are the input and pushdown alphabets, \( T \) is the set of internal states, \( \varphi: \Xi \times T \times (I \cup \{e\}) \rightarrow \Xi^* \times T \) is the finite relation which defines the (nondeterministic) behavior of the \( NPDA \), \( t_0 \) \((eT)\) and \( \xi_0 \) \((e\Xi)\) are the initial state and the start symbol (or the “bottom marker” of the stack), and \( T_f \) is the final state set. Now, we define \( M_{ij} \) as follows:

(a) \( I = \Sigma \cup \{e\} \cup \{\#\} \) and \( \Xi = \Sigma \cup \{\#\} \cup \{\xi_0\} \) \((\xi_0 \notin \Sigma \cup \{\#\})\),

(b) \( T = S \cup S' \cup \{t_0, t_f\} \) where the union is disjoint, and

(c) \( T_f = S'_f \).

Let \( \Sigma_1 = \Sigma \times \{e\} \), \( \Sigma_2 = \{e\} \times \Sigma \), and \( \Sigma_3 = \Sigma \times \Sigma \). Then \( \Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \), where the union is disjoint. We define \( \varphi \) by

\( (d_1) \quad \varphi(\xi, t_0, \gamma) = (\xi \gamma, t_0) \) \quad \text{for} \quad \gamma \in \Sigma \cup \{\#\}, \quad \xi \in \Xi, \)
\( = (\xi, s_0) \) \quad \text{for} \quad \gamma = e, \quad \xi \in \Xi, \)

\( (d_2) \quad \varphi(\xi, s, \gamma) = (e, \delta((\gamma, \xi), s)) \) \quad \text{for} \quad (\gamma, \xi) \in \Sigma_3, \quad s \in S, \)
\( = (\xi, \delta((\gamma, e), s)) \) \quad \text{for} \quad (\gamma, e) \in \Sigma_1, \quad s \in S, \)
\[(d_3)\] \ \varphi(\xi, s, e) = (e, \delta((e, \xi), s)) \quad \text{for} \quad (e, \xi) \in \Sigma^*, \ s \in S,

\[(d_4)\] \ \varphi(\#, s, \#) := (e, st) \quad \text{for} \quad s \in S_r,

\quad = (e, t_r) \quad \text{for} \quad s \in S_f,

\[(d_5)\] \ \varphi(\xi, s, \gamma) = (e, \delta'((\gamma, \xi), s)) \quad \text{for} \quad (\gamma, \xi) \in \Sigma^*, \ s \in S',

\quad = (\xi, \delta'((\gamma, e), s)) \quad \text{for} \quad (\gamma, e) \in \Sigma^1, \ s \in S',

\[(d_6)\] \ \varphi(\xi, s, e) = (e, \delta'((e, \xi), s)) \quad \text{for} \quad (e, \xi) \in \Sigma^2, \ s \in S', \ \text{and}

\[(d_7)\] \ \varphi(\xi, t_r, \gamma) := (\xi, t_r).

By the rules of \((d_4)\), \(M_{ij}\) stacks \(x\#x^e\), then it behaves as a two-input-tape finite automaton, where one input is the rest of input tape and the other is the input from its stack. Though the latter is in the reversed order, the machine recognizes correctly the pair \(x\) and \(y \) (or \(x^e\) and \(y^o\)) because \((x\#x^e)^o = x\#x^e\). We skip the precise but tedious discussions to prove the equation \(R_{ij} = L(M_{ij})\).

Now the injection \(\pi_x : \Sigma^* \rightarrow \Sigma^* \# \Sigma^*\), \(\pi_x(x) := x\#x^o\) assures that \(R'\) and \(R\) describe the same graph \(D\).

**Theorem 4.8.** \(G(R, D, RG) \supseteq G(F, GSM)\).

**Proof.** The class of \(R_{ij}\)'s in our regular direct productive expressions is equal with the class of functional rational relations between free monoids. And it is equivalent to the class of the results of generalized sequential bimachines (Eilenberg, 1974, Vol. A, Chaps. IX and XI.7). The theorem holds, as the class of \(GSM\) is a subclass of the class of generalized sequential bimachines.

**Proposition 4.9.** \(G(F, GSM) \supseteq G(G, AB), G(R, D', RG) \supseteq G(G, AB)\).

**Outline of proof.** The coding method that we utilized in Example 3.2 can easily be generalized to the general finitely generated Abelian group graphs.

It is obvious that the inclusions are proper.

**Proposition 4.10.** \(\{G(F, GSM) \setminus G(F, HM)\} \supseteq \{G(G, AB) \setminus G(FN)\}\).

This is a corollary of Theorem 4.3, because every infinite Abelian group has, as its subgroup, an infinite circular group generated by one of its generators.

**Proposition 4.11.** \(G(R, C, RE) \supseteq G(R, C, RC)\).

The proof is present in the discussion after Definition 3.1.

**Proposition 4.12.** \(G(G, MON) \supseteq G(G, GRP) \supseteq G(G, AB)\).
This has already been discussed in the phrases following Definition 3.4. We can define a semigroup graph by specifying some nodes $H$ of a monoid graph, provided that $H$ is closed under the operation of the monoid. Then we have $G(G, SMG) \supsetneq G(G, MON)$, where $SMG$ is the abbreviation of "semigroup."

In ending this section, we introduce a quite strange result, deduced from the famous theorem that the word problem in groups is unsolvable (Novikov, 1955; Britton, 1963).

**Theorem 4.13.** There exists a graph $D$ in $G(G, GRP)$ such that no effectively calculated isomorphism is existent to some graph $D'$ in $G(R, C, RE)$.

**Proof.** Let $D$ be a group graph whose word problem is unsolvable, and $D'$ is isomorphic to $D$ where the isomorphism from $D$ to $D'$ is calculated by an effective procedure. It is clear that, for every graph $D'$ in $G(R, C, RE)$, we have a general algorithm that decides whether there is a closed loop from $x$ to $x$ with length $k$, where $x$ is an arbitrary node and $k$ is an arbitrary fixed natural number.

\[
\begin{align*}
G(R, Y, RE) &= G(F, RE) \\
G(R, Y, RC) &= G(F, RC) \\
G(R, Y, CS) &= G(F, CS) \\
G(R, D, CF) &= G(F, CF) \\
G(R, D', CF) &= G(F, CF) \\
G(R, D, RG) &= G(F, GSBM) \\
G(R, D', RG) &= G(F, GSM) \\
G(R, C, RG) &= G(F, HM) \\
G(R, C, RG) &= G(F, AB) \\
G(FN) &= \{D, D', C\}
\end{align*}
\]

**Fig. 4.** Inclusion relations among the classes discussed in Sections 3 and 4.
As $D$ is isomorphic to $D'$, this gives an algorithm which solves the same problem on $D$. The word problem is the special case of this problem, where $x$ is the node expressing the unit element of the group. So, this is a contradiction.

This result can be interpreted that we do not have an effective procedure to draw successively the picture of $N_{D'}^k(x)$, for $k = -1, -2, ...$. For $k = 1, 2, ...$, such impossibility can take place even for a graph in $G(R, C, CS)$, because the infiniteness and the emptiness are not decidable for $CS$-languages.

Undecidable problems are existent also in the classes $G(R, D', RG)$ and $G(F, GSM)$. We can construct a description system of a graph such that no effective procedure can determine whether there is a path from the origin node $\omega$ to an arbitrary node $x$, i.e., whether $\exists k \in \mathbb{N}$, $\omega \in N_{D'}^{-k}(x)$, or not. This is because of the fact that a single step of the calculation of a Turing machine can be regarded as a $GSM$ mapping of the configuration of the machine. The configuration change is very local, so the relation of configurations before and after a single-step calculation can also be defined by our $(D', RG)$-expression.

Now, we summarize the results of this section as Fig. 4, where $G(F, GSBM)$ denotes the class of graphs which can be expressed by generalized sequential bimachines, that we discussed in the proof of Theorem 4.8. The solid lines in the figure mean that the inclusion relations are proper, and the broken lines are the relations that we do not know whether they are proper or not.

However, our conjecture is that all of them are proper.

5. Discussion

A. Generalizations of the Description Methods

The restrictions that graphs are input bounded, standard, and weakly connected, are mainly due to our initial motivation which lies in the theory of cellular spaces. These restrictions can be easily removed as follows.

(i) Instead of the axiom $\omega$, we put a (possibly infinite) set of axioms $\Omega$. Then we can describe a graph with (infinitely) many weakly connected components. We denote a class of graphs with this generalization as $G'(\cdots)$, for example, $G'(R, D', RG)$, $G'(F, RC)$, etc.

(ii) We remove condition (ii) from Definition 3.1. Then there may be (if you like, infinitely) many edges with a same label $ij$, connected to a node $x \in X_i$ as its inputs. In order to obtain the same effect for the description systems utilizing functions, i.e., $G(F, HM)$, $G(F, GSM)$, etc., we admit nondeterministic behaviors of machines. For example, a nondeterministic homomorphism $\lambda: \Sigma^* \rightarrow 2\Sigma^*$ is nothing but a $0L$-scheme $(\Sigma, \lambda)$ (Herman and Rozenberg, 1975).
For this second generalization, we put "N" before the same of the class of languages or functions. For example, $G(R, C, NRG)$, $G(F, NHM)$, $G'(F, NGSBM)$, etc.

We did not discuss these generalizations, mainly because of the reason that simply connected input bounded graphs seem to cover a very large part of the applications of infinite graphs. And we also thought that our idea would be expressed most clearly in the simple situation.

B. Applications

Now, we list some examples of the applications of infinite graphs.

(i) Infinite networks of information processing elements, such as parallel processors, integrated logical circuits, nervous systems of animals, etc. can be regarded as poly-automata with highly complicated structures. (We often idealize them as infinite poly-automata.) Infinite graphs are utilized not only to express their spatial structures (i.e., connections among the elements), but also to describe the spatiotemporal structures of the behaviors of systems. For example, the behaviors of a one-dimensional cellular space can be expanded as patterns on the "skeleton graph," shown in Fig. 5. This expansion is a general method to visualize the behaviors of systems. But, if we proceed to handle more complicated networks, it would be clear that we cannot do anything without some knowledge about this kind of infinite graph. The merit of such expansions will be more evident in case that the spatial structure of the system itself is a function of time or a function of the preceding configuration.

![Fig. 5. The "expansion skeleton" of a one-dimensional cellular space along the time axis.](image)

(ii) The space of the finite configurations of a finite or infinite system is a directed graph with countable nodes. When the action of the system is deterministic, the graph is input bounded, provided that an edge $x' \rightarrow x$ corresponds to the lapse of unit time $x \rightarrow x'$ in the configuration space.
(iii) Generative grammars and $L$-systems can be expressed as infinite graphs in two ways. First, we regard a derivation tree of a grammar as an infinite graph. Second, $\Sigma^*$ is considered as a configuration space, and single-step derivations are the edges of the configuration space. The structures of various kinds of rewriting systems can be studied by the latter method. For example, the configuration space of an arbitrary deterministic Turing machine can be expressed by a graph in $G'(R, D', RG) \cap G'(F, GSM)$. Every DTOL-scheme can be regarded as a member of a special subclass of $G'(F, HM)$, and if we adopt the generalization that we have described in part A of this section, the graphs which express TOL-schemes, form a subclass of $G'(F, NIH)$. Then the geometric properties of these graphs, such as dimension, connectedness, path multiplicity will give us some information of the behavior of these systems, such as growth order, membership, accessibility, ambiguity.

(iv) Although it is not well shown in this paper, $G(F, GSM)$ is a very rich class so that any conceivable data structure will find its description in $G(F, GSM)$. On the other hand, GSM-functions are most easy and inexpensive operations done in a computer. It is also possible to have some special circuits for special GSM-functions. Combined with good hashing algorithms, such system can realize very universal data structures with high efficiency and low cost.

(v) Recursive functions can be expanded as infinite directed graphs, where an edge corresponds to a "call" of a function.

For example, the recursive definitions of $n!$ and the Fibonacci series $F(n)$ are expanded as Figs. 6 and 7, respectively.

![Fig. 6. The recursive call diagram of $n! = n \times (n - 1)!$](image)

![Fig. 7. The recursive call diagram of the Fibonacci series, $F(n) = F(n - 1) + F(n - 2)$](image)

Both graphs belong to $G(F, GSM)$ and $G(R, D', RG)$. If we invert the direction of all edges, then the resulting graphs are the members of $G(F, HM)$. Such structures can be regarded as a very special case of associative data
structures. In general, if "associations" are rigorously calculated by some algorithms, then our description methods are applicable. So, our description methods may have some role in the theory of associative memories, and more widely, in the semantical fields in logic, computer sciences, and linguistics.

(vi) As a matter of fact, the utilization of infinite graphs is a most natural and usual method in each of these fields.

Therefore, it would be of some merit that we have common and general concepts and formalizations of problems appearing in various fields of mathematics.

C. Further Discussions

Although the framework of the treatment of infinite directed graphs is constructed in this paper, our discussions will have to be enriched by further investigations about various properties of graphs. We are preparing of a paper concerning some geometrical traits of graphs, such as dimension, connectedness, and some graph operations, such as direct product, blocking, substitution, graph morphism, concatenation. (We must beg the reader's pardon for irresponsible use of such terms without any definition or explanation.)

Our discussions of Sections 3 and 4 must also be extended to the general cases that we mentioned in part A of this section.

We may find some interesting results by discussing the classes of languages or automata that we did not mention in Section 3, (for example, indexed grammars, linear grammars, and languages generated by various kinds of $L$-systems). We believe that such research will reveal some new and essential characters of languages and automata. For example, $G(R, D', RG)$ and $G(F, GSM)$ are two distinct restrictions of the class $G(R, D, RG)$. $G(R, D, RG)$ is the class of graphs which can be defined by functional rational relations (Eilenberg, 1974) and the former two classes may be characterized by the concepts almost sequential and initial segment preserving, respectively. Our conjecture is that $G(F, GSM) \supseteq G(R, D', RG)$. If this inclusion is proved to be true, it will show us the essential dominance of the complexity of the $G(F, GSM)$ against $G(R, D', RG)$. Such a notion will contribute to our better understanding of the natures of rational relations.

Theorem 4.7 is rather meaningful, because it shows that the difference in the generative powers of grammars or machines is preserved against the unfavorable coding. Combined with the other results in Section 4, this theorem tells us how essential the Chomsky hierarchy was. We suppose that more faint differences in generative powers do not conserve against the differences of codings. Of course, such differences, i.e., the differences among "$D$," "$D'$," and "$C$," will be effective for grammars with weak generating abilities. Such discussions can contribute to the measurement and comparison of the efficiency
of different kinds of parallel processings of computers. "C" corresponds to serial processing, and "D" and "D'" correspond to parallel processing streams without stream control and with mutual stream control, respectively.

Received: October 6, 1978; revised: September 12, 1979

References