数理解析研究所講究録
943巻 1996年 88-96

restricted RNLC グラフ言語の学習

谷 聖一 (Sei'ichi Tani)　山崎 浩一 (Koichi Yamazaki)

東海大学理学部数学科　電気通信大学情報工学科
神奈川県平塚市北金目 1117　東京都調布市調布ヵ丘 1-5-1
sei-ichi@sm.u-tokai.ac.jp　yamazaki@cs.uec.ac.jp

Abstract

本稿では、あるグラフの集合族が多項式時間で学習可能であることを示す。RNLC (regular node controlled) グラフ文法 G が 制限つき RNLC グラフ文法であるとは、G が次の性質を持つときをいう:
(1) G は対称性を持つ。すなわち、$(a,b) \in conn$ ならば $(b,a) \in conn$ かつそのときに限る、ただし、$conn$ は接続関係である。
(2) 任意の非終端ラベル A と任意の終端ラベル a に対して、$(a,A) \in conn$ である。
(3) 開始グラフは、単一の頂点からなる。
定理 t を固定された非負整数とする。制限つき RNLC グラフ文法で、その Parikh 写像が高々 t 周期となる文法で記述されるグラフの集合族は、制限つき subset query と制限つき superset query から多項式時間限定で同定できる。

RNLC グラフ文法、グラフの学習、質問による学習

1 Introduction

In this paper, we consider the learning problem of graph languages (sets of graphs). In computational learning theory, many researchers have investigated learnability or non-learnability for many objects such as sets of strings, semilinear sets, and formulas. There exists few reports of learning theory for sets of graphs [7]. There are several ways to define sets of graphs by finite devices. The main ones are graph grammars [9, 10, 13, 14], graph rewriting systems[5, 12], graph automata[6], obstruction sets[8], and so on. Using graph grammars as devices to define sets of graphs, there are merits in applications for practical problems. In this paper, we use a type of graph grammars as a representation language in learning of graph languages.

One of the well-known graph grammar models is the node-label-controlled (NLC) graph grammars. The class of regular NLC (RNLC) graph languages is a natural subclass of NLC graph languages. In this paper, we consider the learning problem of the graph languages generated by restricted RNLC graph grammars. A graph grammar G is restricted RNLC graph grammar, if G has the following properties:
(1) G is a symmetric RNLC graph grammar, namely $(a,b) \in conn$ iff $(b,a) \in conn$, where $conn$ is the connection relation,
(2) for all nonterminal label A and all terminal label a, $(a,A) \in conn$,
(3) the start graph (axiom) consists of a node.
For example, the set of complete k-partite graphs is generated by a restricted RNLC graph grammar.

In computational learning theory, learning problems are investigated in several learning models. D. Angluin introduced the notion of exact-learning via queries[2, 3]. We consider exact-learning of restricted RNLC graph languages via queries. In this paper, we present an algorithm to construct a restricted RNLC graph grammar which generates a given unknown restricted RNLC graph language, using restricted superset and restricted subset queries. Furthermore, we show that for a fixed

nonnegative integer t, this algorithm halts in polynomial time when the Parikh image of the given unknown restricted RNLC graph languages has at most t-periods.

In section 2, we investigate the relationship between the restricted RNLC graph languages and those Parikh image in order to show the correctness of our learning algorithm. Let L be a restricted RNLC graph language and $\Psi(L)$ be Parikh image of L. We show that given $\Psi(L)$, one can construct a restricted RNLC graph grammar G such that $L(G) = L$ in polynomial time. In section 3, we present the learning algorithm for restricted RNLC graph languages.

2 Restricted RNLC graph grammars and Parikh mapping

2.1 Restricted RNLC graph grammars

We consider *finite undirected node labeled graphs* without *loops* and without *multiple* edges. The set of all graphs over Σ is denoted by $\mathcal{G}_\Sigma$.

Definition 2.1. For a set of *labels* Σ, a *graph* X (*over* Σ) is specified by V_X, E_X, and φ_X, where V_X is a finite nonempty set of *nodes*, E_X is a subset of $\{\{x,y\} \mid x,y \in V_X, x \neq y\}$, called set of *edges*, and φ_X is a function from V_X into Σ, called the *labeling function.*

Definition 2.2 A *node-label-controlled* (NLC) *graph grammar* is a system $G = (\Sigma, \Delta, P, conn, Z_{ax})$, where Σ is a finite nonempty set of *labels*, Δ is a nonempty subset of Σ (the set of *terminals*), P is a finite set of pairs (d,Y) where $d \in \Sigma - \Delta$ and $Y \in \mathcal{G}_\Sigma$ (the set of *productions*), *conn* is a relation between Σ and Σ (the *connection relation*), and $Z_{ax} \in \mathcal{G}_\Sigma$ (the *axiom*).

The set $\Sigma - \Delta$ is referred to as the set of *nonterminals.* A node x is a *terminal* (*nonterminal* respectively) *node*, if x is labeled by elements of Δ ($\Sigma - \Delta$ respectively).

Now, we explain the method of derivation of a graph. Let $G = (\Sigma, \Delta, P, conn, Z_{ax})$ be an NLC graph grammar, $p = (d, W)$ be a production in P, X and Y be graphs over Σ such that $V_X \cap V_Y = \emptyset$ and Y be isomorphic to W, and x be a node labeled by d in X. Then, a graph Z is derived from the graph X by the production (d, W) in the following way:

Step 1: Delete the node x (and the edges which are incident with x) from X, (Notice that x is labeled by d.)

Step 2: Add to Y instead of x, (Notice that Y is a copy of W.)

Step 3: Connect every node y in Y to the neighbor x' of x by an edge iff $(\varphi_Y(y), \varphi_X(x')) \in conn$ holds.

Consequently, graph Z is obtained, where

$$V_Z = V_{X-x} \cup V_Y,$$
$$E_Z = E_{X-x} \cup E_Y \cup \{\{x', y\} \mid x' \text{ is a neighborhood of } x \text{ in } X,\ y \in V_Y,\ (\varphi_Y(y), \varphi_X(x')) \in conn\},$$
$$\varphi_Z(u) = \begin{cases} \varphi_{X-x}(u) & \text{if } x \in V_{X-x}, \text{ and} \\ \varphi_Y(u) & \text{if } x \in V_Y. \end{cases}$$

Then we say that "X *concretely derives* Z (*in* G, *by the production* p)", and denoted by $X \Rightarrow_{G\ p} Z$ or simply by $X \Rightarrow_p Z$. A sequence of successive concrete derivation steps in G

$$D : X_0 \Rightarrow_{p_{i_1}} X_1 \Rightarrow_{p_{i_2}} \cdots \Rightarrow_{p_{i_n}} X_n$$

where $n \geq 1$ and the sets V_{X_0} and V_{Y_i} $(1 \leq i \leq n)$ are pairwise disjoint, is referred to as *a concrete derivation in G (from X_0 to X_n)*, and the sequence of applied productions $(p_{i_1}, p_{i_2}, \cdots, p_{i_n})$ is termed *applied productions (in the derivation)*.

A graph X *directly derives* a graph Z *(in G)*, denoted by $X \rightsquigarrow_G Z$, if there is a graph Z' such that Z' is isomorphic to Z and X concretely derives Z' in G. The symbol $\rightsquigarrow_G^*$ denotes the reflexive and transitive closure of $\rightsquigarrow_G$. If $X \rightsquigarrow_G^* Z$, then we say that X *derives* Z *(in G)*. The *graph language generated by* G, denoted by $L(G)$, is the set $\{X \in \mathcal{G}_\Delta \mid Z_{ax} \rightsquigarrow_G^* X\}$. A set L of graphs is *an NLC graph language*, if there is an NLC graph grammar G such that $L = L(G)$.

Definition 2.3 A *regular* NLC (RNLC) graph grammar is an NLC graph grammar $G = (\Sigma, \Delta, P, conn, Z_{ax})$ such that every production of G is either of the form (A, H) or the form (A, a), where $A \in \Sigma - \Delta$, $a \in \Delta$, $H = (V, E)$ is a graph such that $V = \{x, y\}$, $\varphi_H(x) = B \in \Sigma - \Delta$, $\varphi_H(y) = a$, $E = \{\{x, y\}\}$.

We denote the production which has form of (A, H) by $(A, \{B, a\})$. An RNLC graph grammar G is "symmetric" if $(a, b) \in conn$ iff $(b, a) \in conn$. For any RNLC graph grammar G, $|G|$ denotes the sum of the number of labels and the number of productions.

Definition 2.4 A graph grammar G is said to be *restricted RNLC graph grammar* if G has the following properties:
(1) G is a symmetric RNLC graph grammar,
(2) for all nonterminal label A and all terminal label a, $(a, A) \in conn$,
(3) the start graph (axiom) consists of a node.

Example 2.1 Let $G_{ex} = (\Sigma, \Delta, P, conn, Z_{ax})$ be a restricted RNLC graph grammar, where $\Sigma = \{S, A, B, C, D, E, F, G, a_1, a_2, a_3\}$, $\Delta = \{a_1, a_2, a_3\}$, $conn = \{(a_1, a_2), (a_2, a_1), (a_2, a_3), (a_3, a_2)\} \cup \{(X, Y) | X$ is a nonterminal label, Y is a terminal label $\}$ $(\cup\{(Y, X) | X$ is a nonterminal label, Y is a terminal label $\}$ $)$, Z_{ax} consists of a node with label S, and P is depicted in Fig. 1.

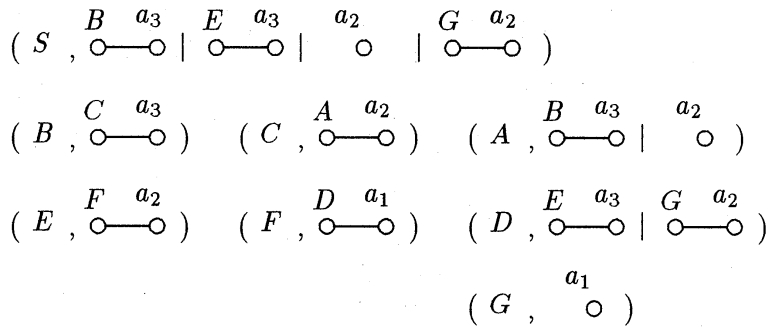

Figure 1:

Then graphs depicted in Fig. 2 can be generated by G_{ex}.

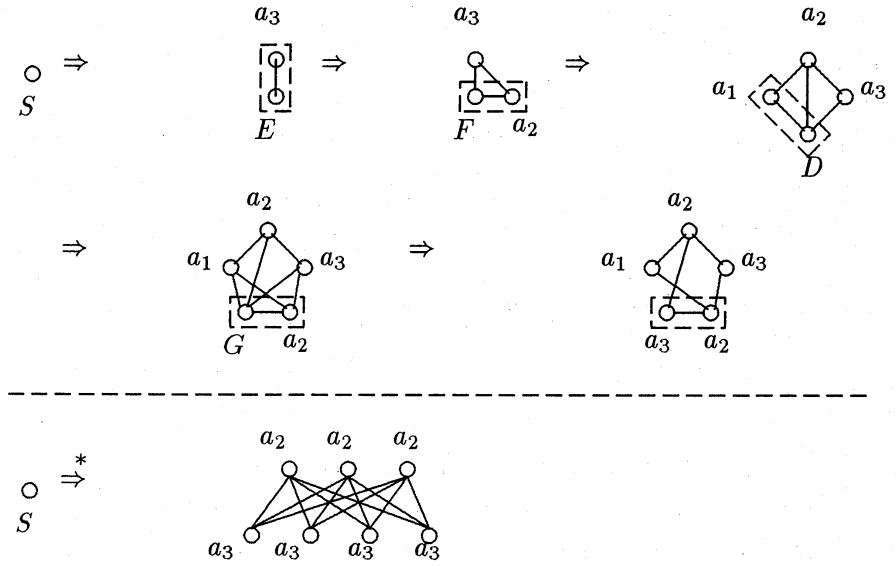

Figure 2:

2.2 Parikh mapping of restricted RNLC graph languages

In this section, we give the definition of Parikh mapping of restricted RNLC graph languages and some related properties which will be used in this paper.

Definition 2.5 Let us fix the set of terminal labels $\Delta = \{a_1, \ldots, a_n\}$ and define a mapping, called *Parikh mapping*, from $\mathcal{G}_\Delta$ into $\mathbf{N}^n$ given by

$$\Psi(H) = (\sharp_{a_1}(H), \sharp_{a_2}(H), \ldots, \sharp_{a_n}(H)),$$

where $\sharp_{a_i}(H)$ denotes the number of occurrences of a_i in the graph H. For any $L \subseteq \mathcal{G}_\Delta$, define $\Psi(L) = \{\Psi(H) \mid H \in L\}$. $\Psi(L)$ is called *Parikh image* of L.

Definition 2.6 A set of the form

$$M = \{\alpha_0 + n_1\alpha_1 + \cdots + n_m\alpha_m : n_j \geq 0 \text{ for } 1 \leq j \leq m\},$$

where $\alpha_0, \ldots, \alpha_m$ are elements of $\mathbf{N}^n$, is said to be a *linear* subset of $\mathbf{N}^n$. Each α_i is called *period* of M. A *semilinear* is a finite union of linear sets.

Proposition 2.1 *For any RNLC graph language L, $\Psi(L)$ is a semilinear set.*

Proof. Two languages L_1 and L_2 are called *letter equivalent* if $\Psi(L_1) = \Psi(L_2)$. It is known that for any language L $\Psi(L)$ is a semilinear set if and only if L is a letter equivalent to a regular set. It is clear that any graph language generated by an RNLC graph grammar is letter equivalent to a regular set by Definition 2.3. □

For any terminal label $a \in \Sigma$, a production $p \in P$ is called *a-production* if a appears in right-hand side of p.

Proposition 2.2 *Let G_1, G_2 be restricted RNLC graph grammars with a common connection relation, $D_1 = (p_{i_1}, p_{i_2}, \ldots, p_{i_n})$ be an applied productions in a derivation in G_1 and $D_2 = (q_{j_1}, q_{j_2}, \cdots, q_{j_n})$ be an applied productions in a derivation in G_2. If for any terminal label a the number of appearances of a-production in D_1 is equal to one of a-production in D_2, then the graph by derived by D_1 is isomorphism to the graph derived by D_2.*

Proof. Let G be a restricted RNLC graph grammar with a connection relation $conn$ and H be a graph generated by G. And let a and b be node labels in H, x be a node with label a in H and y be a node with label b in H.

By the conditions (1) and (3) of Definition 2.4, without loss of generality we can assume that x is yield before y is yield. Let z be the nonterminal node such that y is yield by rewriting of z.

First we will show the following Fact 1 :

Fact 1 : x is adjacent to y iff $(a,b) \in conn$.

For such nodes x, y, and z, the following property hold :

x is adjacent to y iff

(1) x is adjacent to z, and

(2) $(b,a) \in conn$.

The condition (2) of Definition 2.4 guarantees that x is adjacent to z. Hence Fact 1 holds.

Let H_1 be the graph generated by the derivation D_1 and H_2 be the graph generated by D_2. Since for any terminal label a the number of appearances of a-production in D_1 is equal to one of a-production in D_2, we can obtain the following Fact 2 :

Fact 2 : $\Psi(H_1) = \Psi(H_2)$.

Therefore, from fact 1 and 2, H_1 is isomorphic to H_2. □

The above proposition means that a graph in a restricted RNLC graph language is characterized by the connection relation and the number of applications of a-production for each terminal label a. Hence a restricted RNLC graph language is also characterized by the number of nodes with label a for each terminal label a when the connection relation is fixed. Therefore, we can obtain the following lemma.

Lemma 2.3 *Let G_1, G_2 be restricted RNLC graph grammars with a common connection relation. If $\Psi(L(G_1)) = \Psi(L(G_2))$ then $L(G_1) = L(G_2)$.*

Proof. It is sufficient to show $L(G_1) \subseteq L(G_2)$. Let H be a graph in $L(G_1)$. From $\Psi(L(G_1)) = \Psi(L(G_2))$, there exist a graph H' in $L(G_2)$ such that $\Psi(H') = \Psi(H)$. Hence, from Proposition 2.2 and the condition that G_1, G_2 have a common connection relation, H' is isomorphism to H. □

A regular string grammar G is *CNF left linear* if every production of G is either of the form $A \to Ba$ or the form $A \to a$ for some nonterminal A, B and terminal a.

Lemma 2.4 *Let L be a restricted RNLC graph language with a set of terminal labels Δ. When a semilinear set S which is the Parikh image of L and conn are given, one can construct a restricted RNLC graph grammar $G = (NT \cup \Delta, \Delta, P, conn, Z)$ such that $L = L(G)$.*

Proof. We present a polynomial time algorithm MG to construct a restricted RNLC graph grammar $G = (NT \cup \Delta, \Delta, P, conn, Z)$ such that $L = L(G)$ when a semilinear set S such that $S = \Phi(L)$ are given as input. Assume that $S = M_1 \cup \cdots \cup M_m$. Let $M_i = \{\alpha_{i_0} + n_{i_1}\alpha_{i_1} + \cdots + n_{i_{r_i}}\alpha_{i_{r_i}} \mid n_{i_j} \geq 0$ for $1 \leq j \leq r_i\}$ for each $1 \leq i \leq m$. Let $y_{i_0}, y_{i_1}, \cdots, y_{i_{r_i}}$ be strings in Σ^* whose images under Ψ are, respectively, $\alpha_{i_0}, \alpha_{i_1}, \cdots, \alpha_{i_{r_i}}$. First MG constructs left linear string grammar G'' with the productions:

$S \to A_1 \mid \cdots \mid A_m$,

$A_i \to A_i\ y_{i_j} \mid y_{i_0}$ for all $1 \leq i \leq m$, $1 \leq j \leq r_i$.

Next MG constructs a CNF left linear string grammar G' such that $L(G') = L(G'')$ from G''. Finally, MG constructs graph productions of P form each string production in G' by following way ; Transform a string production form of $A \to Ba_i$ $(A \to a_i)$ into the graph production form of

$(A, \{B, a_i\})$ ((A, a_i) respectively). Let NT be the set of nonterminal labels appearing in P. MG outputs $G = (NT \cup \Delta, \Delta, P, conn, Z)$. It is clear that MG halts in polynomial time. From Lemma 2.3, we can obtain that the output G of MG generates L. □

The above G_{ex} in the example has the following semilinear set $M_1 \cup M_2$:

$$M_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + n_{1_1} \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + n_{2_1} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

From above lemma, if we know appropriate *conn*, we can obtain a restricted RNLC graph grammar such that $L = L(G)$.

3 Learning of restricted RNLC graph languages

3.1 The problem

Angluin introduced the notion exact-learning via queries[2, 3]. In this paper, we consider the problem of learning restricted RNLC graph languages via queries. Suppose L_u be a given unknown target graph language. We assume that the set of terminal labels Δ are known.

The following two types of queries are used as access of an unknown target graph languages in learning procedure:

1. *Restricted subset queries:* The input is a conjectured restricted RNLC graph grammar G_c and the output is *yes* if $L(G_c) \subseteq L_u$ and *no* otherwise.

2. *Restricted superset queries:* The input is a conjectured restricted RNLC graph grammar G_c and the output is *yes* if $L_u \subseteq L(G_c)$ and *no* otherwise.

The answer for unrestricted version of each type of queries is not only *yes* or *no* but also with a *counterexample* in case of *no*, where the counterexample is an element in $L_u - L(G_c)$ ($L(G_c) - L_u$) if the type of the query is subset query (superset query respectively).

In the rest of this section, we consider the problem to construct a restricted RNLC graph grammar G which generates the unknown graph languages using restricted subset queries and restricted superset queries. When we consider the efficiency of learning procedure, the size of the learning problem have to be decide. Usually, the size of minimum representation of target language is used as the size of the learning problem. (For example, each of regular grammar, regular expression, and deterministic finite automata is a representation set for regular sets.) Occasionally, the efficiency of learning procedure depends on what representation set are used (for example, see [3]). For instance, the learning problem of regular sets from membership and equivalence queries[1] using deterministic finite automata is efficient[2], but if **RP** $\neq$ **NP**, one using nondeterministic finite automata is not efficient[4]. We use the restricted RNLC graph grammars as the representation of graph languages L and $\min\{|G| | L(G) = L\}$ as the size of L.

[1] Let L_u be an unknown target language. *Membership query*: The input is an instance x and the output is *yes* if $x \in L_u$ and *no* otherwise. *Equivalence query*: The input is a conjectured representation L_c and the output is *yes* if $L_u = L_c$ and *no* otherwise. When the answer is *no*, a *counterexample* is also supplied, that is, an instance $x \in (L_u - L_c) \cup (L_c - L_u)$. The choice of counterexamples is assumed to be arbitrary.

3.2 The learning algorithm

Here, we demonstrate our learning algorithm for restricted RNLC graph languages using restricted superset and restricted subset queries, which is called LA-rRNLC. Takada showed the learning algorithm LA-SLS for the family of semilinear sets from restricted subset and restricted superset queries[11]. Our learning procedure LA-rRNLC uses Takada's LA-SLS as a subprocedure. Let L be an unknown target restricted RNLC graph language. Since the set of terminal labels Δ is fixed, the number of connection relations k is $2^{|\Delta|^{C_2}}$. Let $conn_1, conn_2, \cdots, conn_k$ be distinct connection relations each other. First, for any $i \in \{1, \ldots, k\}$, we define a learning algorithm LA-rRNLC_$conn_i$ to output a restricted RNLC graph grammar G which generates an unknown target restricted RNLC graph language, when $conn_i$ is the real connection rule. Each LA-rRNLC_$conn_i$ simulates LA-SLS. When LA-SLS makes a restricted subset (superset) query with a semilinear set S_c, LA-rRNLC_$conn_i$ constructs a restricted RNLC graph grammar $G_c = (NT \cup \Delta, \Delta, P, conn_i, Z)$ from S using the procedure MG described in the proof of Lemma 2.4, makes a restricted subset (superset) query with G_c, and return the answer of the query to LA-SLS (See Fig. 3). If $conn_i$ is equivalent to

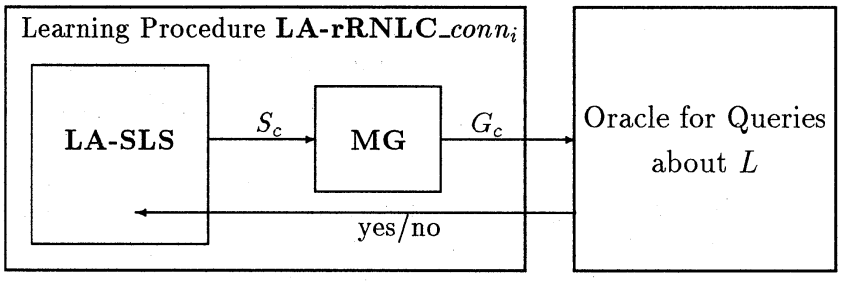

Figure 3:

the connection relation of the unknown target, LA-rRNLC_$conn_i$ outputs a restricted RNLC graph grammar G such that $L = L(G)$ and consumes polynomial time of the running time of LA-SLS from Lemma 2.4.

Our learning algorithm LA-rRNLC is as follows:

Procedure LA-rRNLC

$j := 1$
repeat
j-th step of LA-rRNLC_$conn_1$
j-th step of LA-rRNLC_$conn_2$
⋮
j-th step of LA-rRNLC_$conn_i$
⋮
j-th step of LA-rRNLC_$conn_k$
$j := j + 1$
until some LA-rRNLC_$conn_i$ halts and outputs a grammar G
output G
end.

LA-rRNLC halts and outputs a correct grammar, because $conn_i$ is equivalent to the connection relation of the unknown target. Hence, we can obtain the following lemma.

Lemma 3.1 *Using restricted superset queries and restricted subset queries for an unknown restricted RNLC graph language L_u, the learning algorithm LA-rRNLC eventually terminates and outputs a restricted RNLC graph grammar G such that $L(G) = L_u$*

Let t be a nonnegative integer. We say that a semilinear set S is *t-periods semilinear set* if S is represented by a finite union of linear sets which have at most t periods. Takada showed that the learning algorithm LA-SLS identifies any t-periods semilinear sets S of $\mathbf{N}^k$ from restricted subset and restricted super set queries in polynomial time where k is a fixed dimension[11]. From this fact and Lemma 3.1, we obtain the following theorem.

Theorem 3.2 *Let t be a fixed nonnegative integer. For any restricted RNLC language L_u whose Parikh image has at most t-periods, the learning algorithm LA-rRNLC halts and outputs a restricted RNLC graph grammar G such that $L(G) = L_u$ using restricted superset queries and restricted subset queries in polynomial time.*

参考文献

[1] IJ.J. Aalbersherg, A. Ehrenfeucht, and G. Rozenber, On the membership problem for regular DNLC grammars, *Discrete Applied Mathematics* 13:79-85 (1986).

[2] D. Angluin, Learning regular sets from queries and counter examples, *Information and Computation*, 75 (1987), pp.87-106.

[3] D. Angluin, Queries and concept learning, *Machine Learning*, 2 (1987), pp.319-342.

[4] Angluin, D. and M. Kharitonov: When Won't Membership Queries Help? in *Proc. of 23nd STOC*, ACM, 1991, pp.444-454.

[5] M. Bauderon, and B. Courcelle, Graph expressions and graph rewritings, *Math. Systems Theory.* 20:83-127 (1987).

[6] F.J. Brandenburg and K. Skodinis, Edge-marking Graph Automata and Linear Graph Languages, unpublished manuscript (1994).

[7] C. Domingo and J. Shawe-Taylor, Positive and Negative Results for Learning Minor Closed Graph Classes, unpublished manuscript (1995).

[8] J.V. Leeuwen, Graph algorithms, in *Handbook of theoretical computer science vol. A*, (ed. J.V. Leeuwen), (1990), pp.527-631.

[9] D. Janssens and G. Rozenberg, On the structure of node label controlled graph languages, *Inform. Sci.* 20:191-216 (1980).

[10] G. Rozenberg and E. Welzl, Boundary NLC graph grammars - basic definitions, normal forms, and complexity, *Inform. Control* 69:136-167 (1986).

[11] Y. Takada, Learning semilinear sets from examples and via queries, *Theoretical Computer Science* 104 (1992), pp.207-233.

[12] T. Uchida, T. Shoudai, and S. Miyano, Parallel Algorithms for Refutation Tree Problem on Formal Graph Systems, *IEICE TRANS. INF. & SYST.*, vol.E78-D, no.2 (1995), pp.99-112.

[13] K.Yamazaki, A normal form problem for the unlabeled boundary NLC graph languages, *Inform. Comput.* 120:1-11 (1995).

[14] K.Yamazaki and T.Yaku, A Pumping Lemma and Structure of Derivations in the Boundary NLC Graph Languages, *Inform. Sci.* 75:81-97(1993).