# Inductive Inference of Context-free Languages
## — Context-free Expression Method

Takashi YOKOMORI

International Institute for Advanced Study of
Social Information Science(IIAS-SIS), FUJITSU LIMITED
140 Miyamoto, Numazu, Shizuoka 410-03 JAPAN

ABSTRACT

An inductive inference problem of context-free languages is investigated. There have been many attempts to this problem, and most of them are based on a problem setting in which a representation space for hypotheses is a class of context-free grammars. An inference algorithm given in this paper , on the contrary, employs a kind of extensions of regular expressions called context-free expressions as a representation space for context-free languages. The algorithm, based on the notion of an identification in the limit, is significantly concise when compared with existing algorithms.

## 1. Introduction

We consider the following model of inductive inference problem: Given an object L of inference, an inductive inference device (IID) tries to infer a representation H for the object from examples. It is assumed that IID has an enumeration mechanism by which any possible hypothesis from the representation space can be eventually enumerated at least once. It is also assumed that we can utilize an oracle for presenting examples concerning the object. IID asks the oracle for an example, and computes hypothesis and outputs it, and again asks another example for the next step, and this process is cycled. In a sequence of hypotheses $H_1$, $H_2$,... IID is said to identify L in the limit if there exists a positive integer n such that $H_n$ represents L and $H_{n+i}$ equals to $H_n$ for all $i>0$.

A simple algorithm for identification in the limit is the one based on the notion of identification by enumeration. Let $H_1$, H2,... be an effective enumeration of the possible hypotheses, and suppose a set of examples $e_1, e_2, \ldots, e_k$ are presented. Then, ED provides as its next output the first hypothesis which is compatible with all these examples. Under the assumption of a perfect oracle, the sequence of hypotheses converges in the limit.([Gold 1967])

In this paper, we deal with the inductive inference problem for context-free languages, and employ a representation space for hypotheses different from the ones in the existing methods. This enables us to make an elegant discussion on the problem and to obtain a simple algorithm for solving the problem.

## 2. Context-free Expressions — Extended Regular Expressions

The reader is assumed to be familiar with the rudiments in the formal language theory. (See, e.g., [Salomaa 1973] for definitions not mentioned here.)

For a given finite alphabet $\Sigma$, the set of all strings with finite length ( including zero) is denoted by $\Sigma^*$. (An empty string is denoted by $\varepsilon$.) A *language* L *over* $\Sigma$ is a subset of $\Sigma^*$. For an infinite alphabet $\Gamma$, L is a language over $\Gamma$ if L is a language over some finite subset $\Sigma$ of $\Gamma$.

The following operation plays a crucial role in this paper.

**Definition 2.1** ([Gruska 1971])
(i) Let $\sigma$ be a symbol and $L_1$, $L_2$ be languages. Then, $\sigma$-*substitution of* $L_2$ *into* $L_1$, denoted by $L_1 \uparrow {}^{\sigma}L_2$, is defined as follows:
$L_1 \uparrow {}^{\sigma}L_2 = \{x_1 y_1 \cdots x_k y_k x_{k+1} | x_1 \sigma \cdots x_k \sigma x_{k+1} \in L_1,\ \sigma$ does not occur in the word $x_1 \cdots x_{k+1}$ and $y_i \in L_2$ for $1 \leqq i \leqq k\}$
(ii) Let $\sigma$ be a symbol and L be a language. Then, $\sigma$-*iteration of* L, denoted by $L^{\sigma}$, is defined by
$L^{\sigma} = \{x \mid x \in L \cup L \uparrow {}^{\sigma}L \cup L \uparrow {}^{\sigma}L \uparrow {}^{\sigma}L \cup \cdots$, and x has no occurrence of $\sigma\}$.

**Remarks.**
(1)If $L_1$ does not contain $\sigma$, then $L_1 \uparrow {}^{\sigma}L_2 = L_1$.
(2)Let L be a language over T and suppose that T does not contain $\sigma$. Then, $L^* = (L_{\sigma} \cup \{\varepsilon\})^{\sigma}$ and $L^+ = (L_{\sigma} \cup L)^{\sigma}$.

We introduce the notion of a context-free expression which provides the representation space for context-free languages.

**Definition 2.2**([Gruska 1971])
Let $\Gamma$ be a (possibly infinite) alphabet, $\Gamma'$ be the boldface version of $\Gamma$, i.e, $\Gamma' = \{\sigma' \mid \sigma \in \Gamma\}$.
(i) A *context-free expressions over* $\Gamma$ are strings over $\Gamma \cup \Gamma' \cup \{\phi, +, (,)\}$ defined as follows:
(1) $\phi$ is a context-free expression,
(2) if $\alpha$ is in $\Gamma \cup \{\varepsilon\}$, then $\alpha$ is a context-free expression,
(3) if $E_1$, $E_2$ are context-free expressions and $\sigma \in \Gamma$, then $(E_1 + E_2)$, $E_1 E_2$, $(E_1)\sigma$ are context-free expressions,
(4) nothing else is a context-free expression.
(ii) A mapping $|\ |$ from context-free expressions to a class of languages is defined by:
(1) $|\phi| = \Phi$(empty set)
(2) $|\alpha| = \{\alpha\}$ (for $\forall \alpha \in \Gamma \cup \{\varepsilon\}$)
(3) $|E_1 + E_2| = |E_1| \cup |E_2|$, $|E_1 E_2| = |E_1||E_2|$, $|E\sigma| = |E|^{\sigma}$.

(iii) A class of languages $\Omega_\Gamma$ is defined as follows:
(1) $\Phi, \{\varepsilon\} \in \Omega_\Gamma$,
(2) if $a \in \Gamma$, then $\{a\} \in \Omega_\Gamma$,
(3) if $L_1, L_2 \in \Omega_\Gamma$ and $\sigma \in \Gamma$, then $L_1 \cup L_2$, $L_1 L_2$, and $L_1\sigma \in \Omega_\Gamma$.

**Example 2.1**

Let a,b, $\sigma$ be in $\Gamma$, then $E=(a\sigma b+ab)\sigma$ is a context-free expression. Further, $|E|=|(a\sigma b+ab)\sigma|=\{a\sigma b, ab\}^\sigma = \{a^i b^i \mid i \geq 1\}$.□

Now, the next result plays an important role in this paper.

**Theorem 2.1** ([Gruska 1971])

*Let $\Sigma$ be a finite alphabet, and let L be a language over $\Sigma$. Then, L is a context-free language if and only if there exists a finite alphabet T such that $\Sigma \subseteq T$ and $L \in \Omega_T$.*

Therefore, we have the following characterization of context-free languages in terms of context-free expressions.

**Theorem 2.2**

*Let $\Sigma$ be a finite alphabet, and let L be a language over $\Sigma$. Then, L is a context-free language if and only if there exists a finite alphabet T and a context-free expression E over T such that $\Sigma \subseteq T$ and $|E| = L$.*

*Proof.* Obvious from Theorem 2.1 and definitions.□

Thus, context-free expressions provide a way of representing context-free languages, and as is shown later, an inductive inference algorithm for context-free languages is obtained by naturally extending the one for regular sets in terms of regular expressions.

## 3. Inductive Inference of Context-free Languages

### 3.1 Inductive Inference Algorithm

In this paper, we formalize the inductive inference problem for context-free languages as follows:

**<Inductive Inference Problem for Context-free Languages>**

(1) a *semantic domain* D is the class of context-free languages,
(2) a *target* $d_0$ is a given context-free language,
(3) a *representation space* $\Omega$ is the class of context-free expressions,
(4) an *oracle* EX() gives a *complete presentation of $d_0$*, that is, for every $e \in \Omega$ such that $|e| \subseteq d_0$, EX() eventually returns "+e" at least once, and for every $e \in \Omega$ such that $|e| \not\subseteq d_0$, EX() eventually returns "−e" at least once.

Now, let $\Sigma$ be a finite alphabet over which a target context-free language is defined. We fix an infinite alphabet $\Gamma$ over which context-free expressions are defined, where $\Sigma \subseteq \Gamma$.( It is assumed that auxiliary symbols "+", "(", and ")" are not contained in $\Gamma$.)

We define an operator $\delta$ on $\Omega$, the set of all context-free expressions over $\Gamma$, as follows:

Suppose $E, E_1, E_2$ are context-free expressions over $\Gamma$. As a notation, by $E_1 \rightarrow E_2$ we denote $E_2 \in \delta(E_1)$:

(1) $\phi \rightarrow \phi\phi$
(2) $\phi \rightarrow a$ $(\forall a \in \Gamma \cup \{\varepsilon\})$
(3) $\phi \rightarrow (\phi)\sigma$ $(\forall \sigma \in \Gamma)$
(4) $\phi \rightarrow (\phi + \phi)$
(5) if $E_1 \rightarrow E$, then $E_1 + E_2 \rightarrow E + E_2$ and $E_2 + E_1 \rightarrow E_2 + E$
(6) if $E_1 \rightarrow E$, then $E_1\sigma \rightarrow E\sigma$ $(\forall \sigma \in \Gamma)$
(7) if $E_1 \rightarrow E$, then $E_1E_2 \rightarrow EE_2$ and $E_2E_1 \rightarrow E_2E$.

**Lemma 3.1**

*The operator $\delta$ defined above has the following properties:*
(i) *$\delta$ is complete for the most specific expression $\phi$ in the sense that the set $\delta^*(\phi)$ of all expressions obtainable from $\phi$ in a finite number of applications of $\delta$ includes at least one expression for every context-free language.*
(ii) *For arbitrary expressions $E_1$, $E_2$ in $\Omega$, whenever $E_1 \in \delta(E_2)$, $|E_2| \subseteq |E_1|$ holds.*

*Proof.* We prove by induction on the way of constructing expressions in Definition 2.2. (i) It suffices to show that $\Omega \subseteq \delta^*(\phi)$ holds. First, by the rewriting rule (2) above, we have that $a \in \delta(\phi)$ for $\forall a \in \Gamma \cup \{\varepsilon\}$. Now, suppose that $E_1$ and $E_2$ are in $\delta^*(\phi)$. (In what follows, by $\rightarrow^*$ we denote a finite number of applications of $\rightarrow$. Then,

$\phi \rightarrow \phi + \phi$ ( by (4) ) $\rightarrow^*$ $E_1 + \phi$ (by applying (5) together with the induction hypothesis) $\rightarrow^* E_1 + E_2$ (by applying (5) together with the induction hypothesis),
thus, we have $E_1 + E_2 \in \delta^*(\phi)$. Similarly,

$\phi \rightarrow \phi\phi$ ( by (1) ) $\rightarrow^*$ $E_1\phi$ (by applying (7) together with the induction hypothesis) $\rightarrow^* E_1E_2$ (by applying (7) together with the induction hypothesis),
hence we have $E_1E_2 \in \delta^*(\phi)$. Further, $\phi \rightarrow \phi\sigma$ ( by (3) ) and since $\phi \rightarrow^* E_1$, by applying (6) repeatedly, we have $\phi\sigma \rightarrow^* E_1\sigma$, hence $\phi \rightarrow^* E_1\sigma$, i.e. $E_1\sigma \in \delta^*(\phi)$.
(ii) Since, from the rules (1)-(4), $\delta(\phi) = \{\phi\phi, \varepsilon, \phi + \phi, \sigma, \phi\sigma(\forall\sigma \in \Gamma)\}$, we have that for each $E \in \delta(\phi)$, $\Phi = |\phi| \subseteq |E|$.

Now, let E' be in $\delta(E)$. Then, there are only three cases concerning E.(Note that if E is in $\Gamma \cup \{\varepsilon\}$, then no rule in $\delta$ is applicable to E.)

① $E = E_1 + E_2$; Let $E_i' \in \delta^*(E_i)(i = 1,2)$. Then, by the induction hypothesis, $|E_i| \subseteq |E_i'|$ holds. Hence, by (5) $|E| = |E_1 + E_2| = |E_1| \cup |E_2| \subseteq |E_1'| \cup |E_2| = |E'|$ or $|E| = |E_1 + E_2| = |E_1| \cup |E_2| \subseteq |E_1| \cup |E_2'| = |E'|$ is obtained.

② $E = E_1E_2$; By the same hypothesis,
$|E| = |E_1E_2| = |E_1|\,|E_2| \subseteq |E_1'|\,|E_2| = |E'|$ or $|E| = |E_1E_2| = |E_1|\,|E_2| \subseteq |E_1|\,|E_2'| = |E'|$ is obtained.

③ $E = E_1\sigma(\forall\sigma \in \Gamma)$; Let $E_1' \in \delta^*(E_1)$. Then, by the induction hypothesis, $|E_1| \subseteq |E_1'|$ holds. Hence by (6), $|E| = |E_1\sigma| = |E_1|^\sigma \subseteq |E_1'|^\sigma = |E'|$ is obtained. This completes the proof.□

Now, using the operator $\delta$ defined above, we obtain an algorithm for the inductive inference problem of context-free languages, which is quite simple and based on the principle of "identification by enumeration".

**Definition 3.1** (admissible presentation [Laird 1986])
(i) Let $d_0$ be a context-free language of target. An oracle EX is called *complete* and *sufficient for* $d_0$ if there exists a signed subset K of $\Omega$ satisfying the following:
(1) the set $\{E \in \Omega \mid$ for all e in K, if e is positive, then $|e| \subseteq |E|$, else $|e| \not\subseteq |E|\}$ is exactly the set $\{E \in \Omega \mid |E| = d_0\}$,
(2) for every $e \in K$ such that $|e| \subseteq d_0$, EX() eventually returns "+e" at least once, and for every $e \in K$ such that $|e| \not\subseteq d_0$, EX() eventually returns "−e" at least once.
(ii) A presentation of examples of $d_0$ is *admissible* if it has an oracle EX which is complete and sufficient for $d_0$.
**Note.** There exists such a K for our case, that is, if we take K as the set $\{E \in \Omega \mid |E|$ is a singleton, and if it is in $d_0$, then E has a sign +, otherwise it has a sign −$\}$, then K satisfies the conditions mentioned above. Hence, we may assume the existence of the admissible presentation of $d_0$.

Before presenting an algorithm, we need some preliminaries.

For a given target context-free language L, let $T = \{a_1, \ldots, a_n\}$ be the alphabet over which L is defined. Further, for each $k \geq 1$, let $\Gamma_k = T \cup \Delta_k \cup \Delta'_k \cup \{\phi, +, (, )\}$, where $\Delta_k = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, $\Delta'_k = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$.

For a given expression E and $i \geq 0$, $k \geq 1$, let $\delta(E,i,k) = \{E' \mid E \rightarrow^i E'$ and E' is a string over $\Gamma_k\}$. Then, $\delta^*(E) = \cup_{i \geq 0} \cup_{k \geq 1} \delta(E,i,k)$. We abbreviate $\delta(\phi,i,k)$ as $\delta(i,k)$. The algorithm requires an enumeration procedure which, starting with $\phi$, enumerates every expression in $\delta^*(E)$.

As a notation, we denote an element of $\delta(i,k)$ by $[E,(i,k)]$. Note that for each i,k, $\delta(i,k)$ is finite. Further, let $\delta(i+1,k) = \{E' \mid E \rightarrow E'$ and $E \in \delta(i,k)$, $E' \in \Gamma_k^*\}$, that is, an element of $\delta(i+1,k)$ is obtained from an element of $\delta(i,k)$ by applying $\delta_k$ once, where $\delta_k$ is a restriction of $\delta$ to $\Gamma_k$. Each $\delta(1,k)$ is obtained from $\phi$ by one application of $\delta_k$.

**[The outline of the algorithm]**

Now, the algorithm works as follows: By applying $\delta_k$ to $\phi$, it enumerates a hypothesis (an expression) E and stores into a queue Q in the form of $[E,(i,k)]$. then refines (generalizes) each hypothesis by examples. The enumeration is performed in the order as in

$\delta(1,1)$, $\delta(2,1)$, $\delta(1,2)$, $\delta(3,1)$, $\delta(2,2)$, $\delta(1,3)$, $\delta(4,1)$, ···. That is, the algorithm *dovetails* the enumeration of $\delta(i,1), \delta(i,2), \delta(i,3), \cdots$ with that of $\delta(1,k), \delta(2,k), \delta(3,k), \cdots$.

For a hypothesis E, if it does not cover some positive example(it is called "*too specific*"), then the algorithm generalizes E in some way. Otherwise, if its language |E| includes some negative example(it is called "*too general*"), then the algorithm simply discards it. This is repeated until a correct hypothesis is found.

The algorithm is given as Algorithm $A_1$.

For the sake of helping one understand the process of enumerating hypotheses in the algorithm, Figure 1 illustrates how the contents of a queue Q changes during the enumeration.(The contents of Q is represented as a rectangle whose length may change as the time goes. The shadow portion of Q denoting $\delta(1,k)$ is created only when $\delta(k+1,1)$ is produced from $\delta(k,1)$, and it is placed before $\delta(k+1,1)$).

**Algorithm $A_1$** (Inference Algorithm for Context-free Languages)
*Input*: A recursively enumerable set of context-free expressions $\Omega$
An enumeration operator $\delta$
An admissible presentation of a target language $d_0$
*Output*: A sequence of expressions $E_1, E_2, \ldots$ such that $E_n$ is correct for the first n examples.
*Procedure*:

```
Q←δ(1,1);{elements of δ(1,1) are stored in the queue Q}
EXAMPLES←Φ (empty set)
X←next(Q);{next removes the top element of Q}
do (forever):
EXAMPLES←EXAMPLES∪EX() {get next example}
while (let X=[Ej,(i,k)] be the j-th element of δ(i,k), then )
    ∃e∈EXAMPLES s.t. Ej is not correct for e
    if Ej is "too specific"
    then do
        if i≥2
        then if k=1 and j=1
            then append δ(1,i)δ(E1,1,1) to the tail of Q;
                 X←next(Q);
            else append δ(Ej,1,k) to the tail of Q;
                 X←next(Q);
        else append δ(Ej,1,k) to the tail of Q;
             X←next(Q);
    else (Ej is "too general")
        discard Ej;
        X←next(Q);
Output Ej as the next hypothesis.
where  E is "too specific" :=
         if ∃+e∈EXAMPLES s.t. e∉|E|, then return true
         else return false
       E is "too general" :=
         if ∃−e∈EXAMPLES s.t. e∈|E|, then return true
         else return false
```

**Theorem 3.1**

*For any given context-free language* $d_0$, *the algorithm* $A_1$ *identifies* $d_0$ *in the limit.*

*Proof.* We need to show the following two: First, the algorithm $A_1$ converges some hypothesis E, secondly, the hypothesis E is correct for the target $d_0$.

From the completeness property of $\delta$ ((i) of Lemma 3.1) and Theorem 2.2, there is a chain of generalization steps from $\phi : \phi = E_0 \rightarrow E_1 \rightarrow \cdots \rightarrow E_n = E$ and $|E| = d_0$. Here we assume that n is as small as possible for the given $d_0$. Property (ii) of Lemma 3.1 together with the minimality of n ensures that for each i, $|E_i| \subset |E_n|$. Then, there are strings $w_i$ such that $w_i \in |E|$ and $w_i \notin |E_i|$. Let $E_n$ be in $\delta(n,k)$.

[Proof for correctness] Assume the algorithm converges to some expression E: that is, there exist i,k

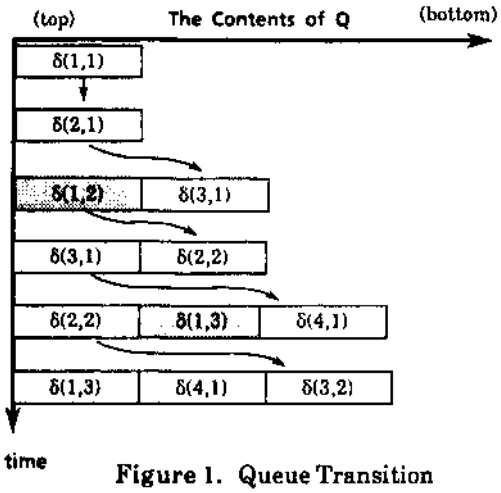


**Figure 1. Queue Transition**

≥Oand E in 8(i,k) such that E is correct for not only all examples in EXAMPLES but also every example given in the future. (That is the definition of "convergence".) Hence, E is correct for do.

[Proof for convergence] Suppose that the algorithm diverges. Then, we show, by the induction on j , that every hypothesis $E_j$ $(0 \leq j \leq n)$in the chain above appears on the top of Q and is generalized. When j = 0, it is trivial. Suppose that Ej appears on the top of Q as a hypothesis in $\delta(i,p)$ and is generalized. Since $E_{i+1} \epsilon \delta(E_i), E_{i+1} \epsilon \delta(i+1,q)$ for some q. The divergence of the algorithm implies that the finite number of expressions preceding Ej+i will all appear on the top and be generalized or simply discarded. Hence, Ei+1 eventually appears on the top of Q and is generalized duetowi+i. Thu $E_n(j=n)$ntually appears on the top. However, because of the assumption of the divergence, $E_n$ is not correct for some w, which contradicts the fact that $|E_n| = d_0$. Hence, the algorithm converges.

Thus, we conclude that the algorithm converges to a correct expression.□

3.2 Inferring Semilinear Languages

If we restrict our attention to a subclass of context-free languages called semilinear languages([Gruska 1971]), then we can easily get an operator for enumerating all expressions over T2, where T2 $= T \cup \{\sigma_1, \sigma_2\}$, and obtain more efficient and simpler inference algorithm for the class. This is due to the fact **that** Any semilinear language over T is contained in $nT_2$ > that > for any semilinear language L over T there exists a context-free expression E over T2 such thatL=\E\ holds.(Note that the rules (I)-(4) in 8 form a context-free **grammar,** provided that T is finite.)

As a natural conclusion, we have:

Theorem 3.2

There exists a simple algorithm for inferring semilinear languages in which the enumerator 8 is achieved by a context-free grammar.

Proo/KSimilar to that of Theorem 3.1 and omitted.)D

## 4. Meta Inference

As we have mentioned, an expression enumerator 8 for semilinear case is realized as a context-free grammar, which implies that an enumerator in the IID is identified with a context-free expression. Therefore, one can think of a meta inference problem in which for a given target, the meta-IID infers a representation (context-free expression, or enumerator) denoting the target from examples of expressions, where the target is a class of expressions (or, a class of semilinear languages denoted by the expressions). Note that since the input of meta-IID, which is an expression, can be regarded as a string over some alphabet, the inference schema of the meta-inference problem is structurally equivalent to that of the inference problem in the usual sense, which is illustrated by Figure 2 .

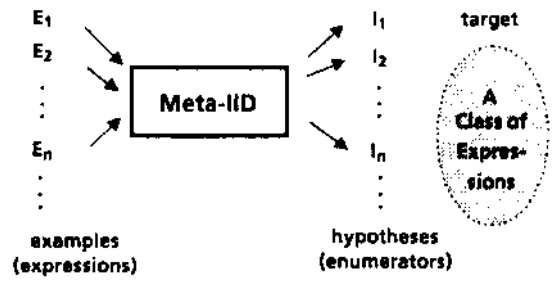


**Figure 2. Meta Inductive Inference Schema**


## ACKNOWLEDGEMENTS

**The author is grateful to Dr. T. Kitagawa, the president of IIAS-SIS, for ceaseless encouragements. He is also indebted to Dr. H.Enomoto, the director of IIAS-SIS, for providing useful reference papers as well as invaluable advice. Last but not least, discussion with the colleagues Y.Takada, Y.Sakakibara, and H. Ishizaka was very fruitful.**

**This work is a part of the major research and development of FGCS Project conducted under the program setup by MITI.**


## REFERENCES


**[I]Gold,E.M., Language Identification in the Limit, Information and Control 10, 447-474 (1967).**

**[2]Gruska,J., A Characterization of Context-free Languages, Jounal of Computer and System Sciences. 5, 353-364(1971).**

**[3JLaird,P.D., "Inductive Inference by Refinement", Technical Report TR-376, Dept. of Computer Science., Yale University, 1986.**

**[4J Salomaa,A., "Formal Languages", Academic Press, 1973.**

**[5]Yokomori,T., "Inductive Inference of Context-free Languages -Context-free Expression Method", Research Report 71, IIAS-SIS FUJITSU LIMITED, 1986.**