Learning of Partial Languages

K. Sasikala*, V.R. Dare† and D.G. Thomas‡

Abstract

In this paper we introduce a finite automaton called partial finite automaton to recognize partial languages. We have defined three classes of partial languages, viz., local partial languages, regular partial languages and partial line languages. We present an algorithm for learning local partial languages in the limit from positive data. The time taken for the algorithm to identify the regular partial languages is $O(l)$ where l is the length of a sample from positive data. Further, we have shown that the class of regular partial languages is learnable in the limit from positive data and restricted subset queries. We provide another learning algorithm to infer partial line languages.

Keywords : Partial words, partial finite automaton, partial languages, identification in the limit, learning algorithm.

1 Introduction

Partial words are strings of symbols from a finite alphabet that may have a number of "do not know" symbols. While a word can be described by a total function, a partial word can be described by a partial function.

Partial words were recently introduced by Berstel and Boasson [3] in the context of gene (or protein) comparison. Alignment of two genes (or two proteins) can be viewed as a construction of two patial words that are said to be compatible. The main motivation for the introduction of partial words came from molecular biology of nucleic acids. There, among other things, one tries to determine properties of the DNA or RNA sequences encountered in nature. Quite recently, Blanchet-Sadri [4] has made a first step towards investigating languages of partial words by introducing the concept of pcodes, which are sets of partial words preserving the uniqueness of factorization of partial words.

On the other hand within computational learning theory, there are three major established formal models for learning from examples, namely Gold's model of inductive inference or identification in the limit [5], the query learning model of Angluin [1] and the probably approximately correct (PAC) learning model of Valiant. Each model provides a learning protocol and a criterion for the success of learning. Inductive interference introduced by Gold is a model which treats learning as an infinite process.

Gold [5] has shown that any class of languages containing all the finite languages and at least one infinite language cannot be identified in the limit from positive presentations. According to this result, the class of context-free langauges (even the class of regular languages) cannot be learnt from positive presentation. Angluin [1] has given several conditions for a class of languages to be learnable in the limit from positive data and presented some learnable classes.

A linear time learning algorithm in Gold's framework of identification in the limit from positive data is given for the class of local languages and another algorithm additionally with restricted superset queries for the class of regular languages by Yokomori [10]. Motivated by the work of [4] in this paper we introduce a language of partial words. This takes us more in the direction of classical formal language theory. Partial languages are collection of partial words over $(\Sigma \cup \{\Diamond\})^*$. In section 3, we introduce a new automaton called partial finite automaton both deterministic and non-deterministic to recognize partial languages and define regular partial grammar. Motivated by the work of [10] in section 4 we define a local partial automaton to recognize local partial languages. We present an algorithm for learning of local partial languages from positive data. The time taken for the algorithm to identify the local partial langauge is $O(l)$ where l is the length of a sample from positive data. In section 5, we have shown that the class of regular

*Department of Mathematics, St. Joseph's College of Engineering, Jeppiaar Nagar, Chennai - 600 119, India

†Department of Mathematics, Madras Christian College, Chennai - 600 059, India

‡Department of Mathematics, Madras Christian College, Chennai - 600 059, India. Email : dgthomasmcc@yahoo.com

partial languages is learnable in the limit from positive data and restricted subset queries. Line languages and its geometrical representation are introduced in [8]. In section 6, the concept of line langauges is applied to partial words and their properties are discussed. Partial line languages can be inferred using IAL algorithm [9] for line languages.

2 Preliminaries and Basic Definitions

In this section we deal with basic concepts on words, partial words, finite automaton and local languages.

Let Σ be a finite non-empty set of symbols called an alphabet. Symbols in Σ are called letters and any finite string over Σ is called a word over Σ. The empty word is denoted by λ. The set of all words over Σ is denoted by Σ^*. The collection of all infinite words is denoted by Σ^ω and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. A word u is primitive, if it is not of the form v^k, for some word $v \neq u$ with $k > 1$.

A partial word is very much like a conventional word, only at some positions we do not know, which letter it has. Similar to the definition of a word as a total function from $\{0, \ldots, |w|-1\}$ to Σ, a partial word w is defined as a partial function from $\{0, \ldots, |w|-1\}$ to Σ. The positions, where $w(n)$ (the n^{th} letter of w) is not defined for $n < |w|$ are called the word's holes. If $D(w)$ stands for the domain of w, then the set of holes of w denoted by $h(w)$ is the set of numbers in $\{0, \ldots, |w|-1\} \backslash D(w)$.

A word over Σ is a partial word over Σ with an empty set of holes (we sometimes refer to words as full words). For any partial word u over Σ, $|u|$ denotes its length. In particular $|\lambda| = 0$.

We denote by W the set Σ^*, and for every integer $i \geq 1$, by W_i the set of partial words over Σ with exactly i holes. $W = \bigcup_{i \geq 0} W_i$, the set of all partial words over Σ with an arbitrary number of holes.

If u is a partial word of length n over Σ, then the companion of u (denoted by $u_\Diamond$) is the total function $u_\Diamond : \{0, 1, \ldots, n-1\} \to \Sigma \cup \{\Diamond\}$ defined by

$$u_\Diamond(i) = \begin{cases} u(i) & \text{if } i \in D(u) \\ \Diamond & \text{otherwise} \end{cases}$$

When it is more convenient, we will also refer to the companion as a partial word to simplify the syntax of our sentences. Thus we will say for example the partial word $\Diamond a \Diamond b$ instead of the partial word with companion $\Diamond a \Diamond b$.

For a subset X of W, we denote by X^* the submonoid of W generated by X. It consists of all partial words which are obtained by concatenating of elements of X.

A partial word u is a factor of the partial word v if there exist partial words x, y such that $v = xuy$. The factor u is called proper if $u \neq v$. The partial word u is a prefix (respectively suffix) of v if $x = \lambda$ (respectively $y = \lambda$). For a subset X of W, we denote by $F(X)$ the set of factors of elements of X.

$$F(X) = \{u | u \in W \text{ and there exist } x, y \in W \text{ such that } xuy \in X\}.$$

Definition 2.1 *If u and v are two partial words of equal length then u is said to be contained in v, denoted by $u \subset v$, if $D(u) \subset D(v)$ and $u(i) = v(i)$ for all $i \in D(u)$. The partial words u and v are compatible, denoted by $u \uparrow v$ if there exists a partial word w such that $u \subset w$ and $v \subset w$.*

As an example $u = aba\Diamond aa$ and $v = a\Diamond\Diamond b\Diamond a$ are two partial words that are compatible.

Definition 2.2 *An infinite partial word u over Σ is a partial map $u : N \to \Sigma$. For $1 \leq i < \infty$, if $u(i)$ is defined, then we say that i belongs to the domain of u (denoted by $D(u)$), otherwise we say that $u(i)$ is defined and i belongs to the set of holes of u (denoted by $H(u)$). An infinite word over Σ is an infinite partial word over Σ with an empty set of holes.*

Example 2.1 *An infinite word $u = aabab\Diamond ab^\omega$ is the infinite partial word with $D(u) = \{1, 2, 3, 4, 5, 7, 8, 9, ...\}$ and $H(u) = \{6\}$. The compatibility of two infinite partial words can be defined in a way similar to that of compatibility of two finite partial words.*

Definition 2.3 *A partial language over Σ is the set of partial words over Σ.*

Example 2.2

1. *$a\Diamond b(ab)^*$ is a partial language of finite partial words.*
2. *$(ba)^*\Diamond(ba)^\omega$ is a partial language of infinite partial words.*

3 Partial Finite Automaton

In this section a partial automaton to recognize a partial language is introduced.

Definition 3.1 *A finite deterministic finite partial automaton (DFPA) is a 7-tuple*
$PA = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$*, where* Q *is a finite set of states,* Q_h *is a finite set of hole states,* Σ *is a finite input alphabet,* $\Gamma = \{\Diamond\}$*, where* $\Diamond$ *is the don't care symbol not in* Σ *corresponding to a hole position in a partial word,* $\Sigma \cap \Gamma = \phi$ *and* $Q \cap Q_h = \phi$*,* q_0 *in* Q *is the initial state,* $F \subseteq Q$ *is the set of final states,* δ *is the transition function mapping* $(Q \cup Q_h) \times (\Sigma \cup \{\Diamond\}) \rightarrow Q \cup Q_h$ *such that (i)* $\delta(p, a) = q, q \in Q, p \in Q \cup Q_h, a \in \Sigma$*, (ii)* $\delta(q_h, \Diamond) = q_h, q_h \in Q_h, \Diamond \in \Gamma$*, (iii)* $\delta(p, \Diamond) = q_h, p \in Q, q_h \in Q_h, \Diamond \in \Gamma$*.*

A partial word w *is said to be accepted by a finite deterministic partial automaton*
$PA = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ *if* $\delta(q_0, w) = p$ *for some* p *in* F*. The partial language accepted by* PA *is designated* $L(PA) = \{w | \delta(q_0, w)$ *is in* $F\}$*. This is called a regular partial language.*

Example 3.1 *The following deterministic finite partial automaton accepts the partial language* $(a\Diamond b)^n, n \geq 1$*.*
$PA_1 = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ *where* $\Sigma = \{a, b\}$*,*
$Q = \{q_0, q_1, q_2, q_3\}$*,* $Q_h = \{q_h\}$*,* $\Gamma = \{\Diamond\}$ *and* $F = \{q_3\}$*. The transition diagram is shown in figure 1.*

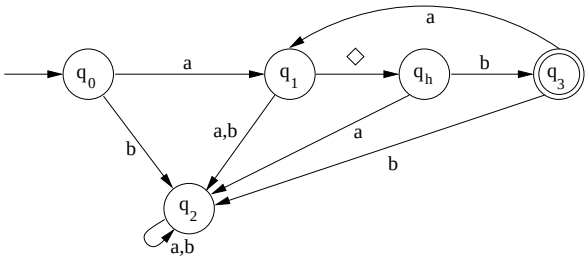

Figure 1: DPFA accepting $(a\Diamond b)^n, n \geq 1$.

Definition 3.2 *A non-deterministic finite partial automaton (NPFA) is a 7-tuple* $(Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ *where* $Q, Q_h, \Sigma, \Gamma, q_0$ *and* F *have the same meaning as for a deterministic finite partial automaton, but* δ *is a mapping from* $(Q \cup Q_h) \times (\Sigma \cup \{\Diamond\})$ *to* $2^{Q \cup Q_h}$ *where* $2^{Q \cup Q_h}$ *is the set of all subsets of* $Q \cup Q_h$*. Here (i)* $\delta(p, \Diamond) = q_h, p \in Q, q_h \in Q_h$*. (ii)* $\delta(q_h, \Diamond) = q_h, q_h \in Q_h$*.*

Example 3.2 *The following non-deterministic finite partial automaton accepts the partial language* $a^*(a\Diamond b)^n$*.*
$PA_3 = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ *where*
$Q = \{q_0, q_1, q_2\}$*,* $Q_h = \{q_h\}$*,* $\Gamma = \{\Diamond\}$*,* $\Sigma = \{a, b)$ *and* $F = \{q_2\}$*. The transition diagram is shown below*

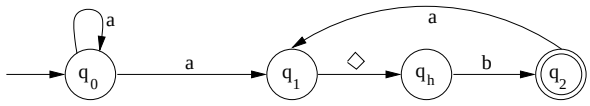

Figure 2: NFPA accepting $a^*(a\Diamond b)^n$

4 Local Partial Languages

In this section we introduce local partial automaton to recognize local partial languages. Informally speaking, a local language is described by the factors of length two of its words and we exhibit an algorithm to identify a local partial language in linear time in the length of a sample string from the positive data which is a subset of a local language.

We need the following notations. Let
$P_1(w)$ be the Prefix of w of length 1 in $(\Sigma \cup \{\Diamond\})$
$S_1(w)$ be the suffix of w of length 1 in Σ
$F_2(w)$ be the set of subwords of w of length 2.
$PF_2(w)$ be the set of subpartial words of w of length 2.
We denote by $\mathcal{PL}$ the set of all local partial languages.

Definition 4.1 *A partial language* PL *of finite partial words over* Σ *is called local, if there exists a four tuple* (I, C, H, J) *where* $I, J \subseteq \Sigma$*,* $C \subseteq \Sigma^2$*,* $H \subseteq (\Sigma \cup \{\Diamond\})^2$*, such that*

$$PL = \{w \in (\Sigma \cup \{\Diamond\})^* : P_1(w) \in I, F_2(w) \subseteq C, PF_2(w) \subseteq H, S_1(w) \in J\},$$

Example 4.1 *The partial language*
$PL = \{(a\Diamond b)^n, n \geq 1\}$ *over* $\Sigma \cup \{\Diamond\}$*, where* $\Sigma = \{a, b\}$ *is a local partial language. Here* $I = \{a\}$*,* $J = \{b\}$*,*
$C = \{ba\}$*,* $H = \{a\Diamond, \Diamond b\}$*,* $P_1(PL) = a \in I$*,*
$S_1(PL) = b \in J$*,* $F_2(PL) = \{ba\} \subseteq C$*,*
$PF_2(PL) = \{a\Diamond, \Diamond b\} \subseteq H$*.*

Example 4.2 *The finite partial language*
$PL = \{(a\Diamond a)^n, n \geq 1\}$ *is non-local.*

Definition 4.2 *A partial finite automaton*
$M = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ *is said to be local, if for every*

1. *$a \in \Sigma$, the set $\{\delta(q,a) : q \in Q \cup Q_h\}$ contains at most one element*

2. *$\Diamond \in \Gamma$, the set $\{\delta(q,\Diamond) : q \in Q \cup Q_h\}$ contains at most one element from Q_h.*

It is called standard, if it contains no transition arriving on the initial state.

Example 4.3 *The following local partial automaton accepts the local partial language*
$PL = \{(a\Diamond b)^n / n \geq 0\}$.
Let $Q = \{[\lambda], [a], [a\Diamond], [\Diamond b], [ba], [b]\}$; $q_0 = [\lambda]$
$F = \{b\}$; $\delta[[\lambda], a] = [a]$; $\delta[[a], \Diamond] = [a\Diamond]$
$\delta[a\Diamond, b] = [\Diamond b]$; $\delta[[\Diamond b], \lambda] = [b]$;
$\delta[[b], a] = [ba]$; $\delta[[ba], \Diamond] = [a\Diamond]$
$I_{RL} = \{P_1(w)/w \in R_L\} = \{a\}$
$C_{RL} = \{F_2(w)/w \in R_L\} = \{ba\}$
$J_{RL} = \{S_1(w)/w \in R_L\} = \{b\}$
$H_{RL} = \{PF_2(w)/w \in R_L\} = \{a\Diamond, \Diamond b\}$
The transition diagram of the local partial automaton is shown in figure 3.

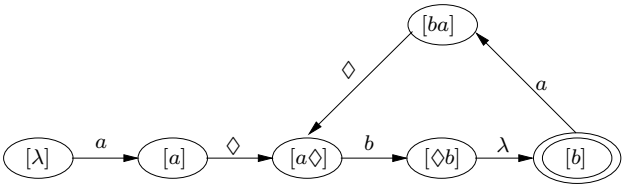

Figure 3: Local partial automaton accepting $(a\Diamond b)^n$

Theorem 4.1 *A partial local language $PL \subseteq (\Sigma \cup \Gamma)^*$ is local if and only if PL is recognized by the standard local partial automaton.*

4.1 Characteristic Sample for Local Partial Languages

Let PL be a local partial language such that $PL \subseteq (\Sigma \cup \Gamma)^*$. A finite subset R of $(\Sigma \cup \Gamma)^*$ is called characteristic sample for PL iff PL is the smallest local partial language containing R.

4.2 Test Vector Set

For a partial word w, an ordered tuple $t(w)$ defined by $t(w) = \langle\ P_1(w),\ F_2(w),\ PF_2(w),\ S_1(w)\ \rangle$ is called test vector of the partial word w. A test vector set (TVS) V_E over $\Sigma \cup \Gamma$ is any finite subset of $\{t(w)/w \in (\Sigma \cup \Gamma)^*\}$.

We now present a learning algorithm to learn the unkown local partial language U, from positive data.

Theorem 4.2 *There is an algorithm, which given any unknown local partial language U, learns in the limit a TVS V_E such that $L(V_E) = U$.*

Input : a sequence of positive presentation of U
Output : a sequence of TVSs for local partial languages
Procedure
Initialize $E = \phi$
Construct the initial TVS $V_E = \phi$
Repeat (for ever)
let V_E be the current TVS;
read the next positive example w,
scan w to compute
$t(w) = \langle P_1(w), F_2(w), PF_2(w), S_1(w)\rangle$
$E = E \cup \{w\}$
$V_E = V_E \cup \{t(w)\}$

Output V_E as a conjecture.

Theorem 4.3 *The algorithm may be implemented to run in time $O(N)$, where N is the sum of the lengths of all positive samples of the data E provided in the learning process.*

5 Towards the General Learning of Regular Partial Sets using Positive Data

It is well-known that the (full) class of regular languages of words is not learnable in the limit from only positive data [5]. So, the question arises: besides positive data, what is minimally needed to learn the class of regular languages?

Concerning the learnability on the full class of regular languages, Angluin has shown that membership queries, restricted equivalence queries and subset queries are insufficient for learning the class in the sense of exact identification [2]. More exactly, it is shown that any algorithm that exactly learns any regular language using restricted equivalence, membership, and subset queries must make at least exponential number of queries in the worst case.

We shall show that the class of regular partial languages is learnable in the limit using positive data

with restricted subset queries, ignoring the time efficiency.

We prove the following theorem connecting local partial languages and regular partial languages. This theorem helps to provide the learning algorithm for regular partial languages.

Theorem 5.1 *Given any λ-free regular partial language PL, there effectively exists a local partial language PL' and a coding h such that $PL = h(PL')$.*

Proof Let PL be a regular partial language.

Let $PA = (Q, Q_h, \Sigma, \Gamma_1, \delta, q_0, F)$ be a deterministic finite partial automaton which recognizes PL. Let
$\Gamma_1 = (Q \cup Q_h) \times (\Sigma \cup \Gamma) \times (Q \cup Q_h)$
$I = \{q_0\} \times (\Sigma \cup \Gamma) \times (Q \cup Q_h)$
$C = \{(q_1, a, q_2)(q_2, b, q_3) \in \Gamma_1^2 : q_2 = \delta(q_1, a),$
$q_3 = \delta(q_2, b)\}$
$H = \{\{(q_1, a, q_2)(q_2, \Diamond, q_h) \in \Gamma_1^2 : q_2 = \delta(q_1, a),$
$q_h = \delta(q_2, \Diamond)\}, \cup\{(q_1, \Diamond, q_h)(q_h, a, q_2) \in \Gamma_1^2 :$
$q_h = \delta(q_1, \Diamond)$ and $q = \delta(q_h, a)\},$
$\cup\{(q_1, \Diamond, q_{h_1})(q_{h_1}, \Diamond, q_{h_2}) \in \Gamma_1^2 :$
$q_{h_1} = \delta(q_1, \Diamond), q_{h_2} = \delta(q_{h_1}, \Diamond)\}\}$
$J = (Q \cup Q_h) \times (\Sigma \cup \Gamma) \times F.$
Let $PL' = \{w \in \Gamma_1^* : P_1(w) \in I, F_2(w) \subseteq C,$
$PF_2(w) \subseteq H, S_1(w) \in J\}.$
Then PL' is a local partial language.
Define $h : \Gamma_1 \to \Sigma \cup \Gamma$ by
(i) $h(q_1, a, q_2) = a, \quad q_1 \in Q \cup Q_h, q_2 \in Q.$
(ii) $h(q_1, \Diamond, q_h) = \Diamond, \quad q_1 \in Q \cup Q_h, q_h \in Q_h.$
We show that $h(PL') = PL.$
Let $w \in h(PL')$. Then $w = h(u_0)$ for some $u_0 \in PL'$.
Since $P_1(u_0) \in I, F_2(u_0) \subseteq C, PF_2(u_0) \subseteq H,$ $S_1(u_0) \in J,$
we can take $u_0 = (q_0, y_0, q_1)(q_1, y_1, q_2)\ldots$
where $q_i \in Q \cup Q_h, y_i \in \Sigma \cup \Gamma$ such that
$\delta(q_i, y_i) = q_{i+1}$ for $i = 0, 1, 2, \ldots, n-1$ and if $y_i = \Diamond$ then $q_{i+1} = q_h$.
Then $w = h(u_0) = y_0 y_1 y_2 \ldots y_n$ and
$\sigma : q_0 \overset{y_0}{\to} q_1 \overset{y_1}{\to} q_2 \overset{y_2}{\to} \cdots \overset{y_n}{\to} q_n$ is a run of PA for w.
$S_1(u_0) = y_n \in J$. $q_n \in F$. Hence $w \in PL$. Thus $h(PL') \subseteq PL$.

Conversely, assume that
$w = w_1 w_2 \ldots w_n \in PL, w_i \in \Sigma \cup \Gamma$
Let $\sigma : q_0 \overset{w_1}{\to} q_1 \overset{w_2}{\to} q_2 \overset{w_3}{\to} \cdots \overset{w_n}{\to} q_n$, where
$q_0 \in Q$, $q_i \in Q \cup Q_h$, $q_n \in Q$, be the run of PA for w with $q_n \in F$. Consider the partial word
$u_0 = (q_0, w_1, q_1)(q_1, w_2, q_2)\ldots(q_{n-1}, w_n, q_n)$
Clearly $P_1(w) \in I, F_2(w) \subseteq C, PF_2(w) \subseteq H$, $S_1(w) \in J$. Thus $w \in PL'$.
Also $h(u_0) = w_1 w_2 \ldots w_n = w$.
Therefore $w = h(u_0) \in h(PL')$ which gives $PL \subseteq h(\mathcal{P}L')$. Hence $h(PL') = PL$. □

Example 5.1 *Consider $PL = \{(a\Diamond a)^n / n \geq 1\}$. PL is a regular partial language over $\Sigma \cup \Gamma$, where $\Sigma = \{a, b\}$ and $\Gamma = \{\Diamond\}$. But PL is not a local partial language.*

A partial automaton accepting PL is $PA = (Q, Q_h, \Sigma, \Gamma, \delta, q_0, F)$ where $Q = \{q_0, q_1, q_2\}$, $Q_h = \{q_h\}$ and $F = \{q_2\}$. The transition diagram of the partial automaton PA defining δ is given below:

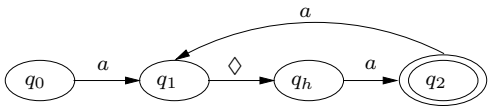

Figure 4: Partial automaton accepting $(a\Diamond a)^n$

By theorem 5.1, we can construct a local partial language PL' such that $PL = h(PL')$ as follows:
$I = \{(q_0, a, q_1)\},$
$C = \{(q_h, a, q_2)(q_2, a, q_1) : q_2 = \delta(q_h, a),$
$q_1 = \delta(q_2, a)\}$
$H = \{(q_0, a, q_1)(q_1, \Diamond, q_h), (q_1, \Diamond, q_h)(q_h, a, q_2) :$
$q_1 = \delta(q_0, a), q_h = \delta(q_1, \Diamond),\ q_2 = \delta(q_h, a)\},$
$J = \{(q_h, a, q_2)\}.$
Then $PL' = \{(q_0, a, q_1)(q_1, \Diamond, q_h)(q_h, a, q_2)\}^+$ which is a local partial language.

Remark 5.1

1. *The alphabet Γ_1 contains at most mn^2 elements, where n is the number of states of minimum partial deterministic finite automata M_U for $U, m = |\Sigma|$.*
2. *For w in U, let $d(w) = (i_0, a_1, i_1)(i_1, a_2, i_2)\ldots(i_{t-1}, a_t, i_t)$ be a partial word over Γ_1 where $a_i \in \Sigma \cup \Gamma$, $i_0 = q_0$, $i_r \in Q \cup Q_h$ and $i_t \in F$, representing a sequence of state transitions for $w = a_1 \ldots a_t$ by M_U. We call this a valid partial word for w. Let $Val(w) = \{d(w) \in \Gamma_1^* / d(w)$ is a valid partial word for $w\}$.*
3. *Let PL be a local partial language over Γ_1 such that $U = h(PL)$, and let R_L be a characteristic sample for PL. Then, there exists a finite set of positive data S_U of U such that $R_L \subseteq Val(S_U) = \bigcup_{w \in S_U} Val(w) \subseteq h^{-1}(S_U)$, where h^{-1} is the inverse of the mapping h.*

4. *If U is a target regular partial language the restricted subset query "if $L \subset U$ or not" takes as an input a set L and produces an output 'yes' of L is a subset of U and 'no' otherwise.*

5.1 Learning Algorithm for Regular Partial Languages (IAPL)

Input : Positive presentation of U, n is the number of states of the minimum DFPA for U,
Output : A sequence of conjectures of the form $h(PL(S_i))$, where $PL(S_i)$ is a local partial language generated by S_i.
Query : restricted subset query.
Procedure :

Initialize $E = \phi$
Construct the initial
tuple $S_0 = (\phi, \Sigma^2, (\Sigma \cup \Gamma)^2, \phi)$
Repeat (for ever)
let $S_i = (I_i, C_i, H_i, J_i)$ be the current tuple;
read the next positive example w;
let $val(w) = \{\alpha_1, \ldots, \alpha_t\}$
scan α_j to compute $P_1(\alpha_j), F_2(\alpha_j)$,
$PF_2(\alpha_j), S_1(\alpha_j)$
$I_{i+1} = I_i \cup P_1(\alpha_j)$
$C_{i+1} = C_i \cup F_2(\alpha_j)$
$H_{i+1} = H_i \cup PF_2(\alpha_j)$
$J_{i+1} = J_i \cup S_1(\alpha_j)$
let $S_{i+1} = (I_{i+1}, C_{i+1}, H_{i+1}, J_{i+1})$
for all i, ask if $h(PL(S_{i+1})) \subseteq U$ or not
$val(w) = val(w) - \{\alpha_j|$ the answer is 'no' $\}$
$E_i = E_i \cup val(w)$
$I_i = I_i \cup \{P_1(\alpha)|\alpha \in val(w)\}$
$C_i = C_i \cup \{F_2(\alpha)|\alpha \in val(w)\}$
$H_i = H_i \cup \{PF_2(\alpha)|\alpha \in val(w)\}$
$J_i = J_i \cup \{S_1(\alpha)|\alpha \in val(w)\}$
output $S_i = (I_i, C_i, H_i, J_i)$

6 Partial Line Languages

In this section we deal with partial line languages and their properties.

Definition 6.1 *Let u be a partial word over Σ, then the Generalized Parikh Vector (GPV) of u is $\mathbf{P}(x) = \{(p_1, p_2) \in [0,1]^2$,*

$$p_1 = \sum_{j \in A_1} \frac{1}{2^j} \ , \quad p_2 = \sum_{j \in A_2} \frac{1}{2^j}$$

where A_1 and A_2 denotes the positions of a and b in u, where the positions representing the holes are neglected.

Definition 6.2 *A partial language PL is called a partial line language if there exist a partial language line pl in R^2 such that the GPV's of partial words of PL lie on pl. The partial line pl is called partial language line.*

Correspondingly a language L is called a line language if there exist a line l in R^2 such that the GPV's of words of L lie on l. The line l is called a language line.

A partial language is said to be an infinite partial line language if it contains only infinite partial words.
A partial language is said to be finite partial line language if it contains only finite partial words.

Example 6.1 *The infinite partial line language $(ba)^*\Diamond(ba)^\omega$ lies on line $y = 2x$.*

Remark 6.1

1. *All partial words have their GPV's in the region bounded by the lines $x = 0, y = 0$ and $x + y = 1$. This region is called PL-region, shown in Figure 5, where no partial word lies on the line $x+y = 1$.*
2. *Partial words of length n lie on the lines*
$$x + y = \frac{2^n - 3}{2^n} \text{ where } n = 2, 3, ...$$
3. *Partial words are densely packed in the PL region.*

Theorem 6.1 *If the line language is of the form $L = \{w_2^*\} \cup \{w_4\}$ where $w_4 \in \Sigma^\omega$ and $|w_2| > 1$, then the corresponding partial line language is of the form $w_2^* \Diamond w_4$ only when the language line intersects the lines $x + y = \frac{2^n - 1}{2^n}$ at*
$n = 1, n_1 + 1, 2n_1 + 1, ...$ where $|w_2| = n_1$.

Proof Let $L = \{w_2^*\} \cup \{w_4\}$, where $|w_2| = n_1 > 1$ be a line language.
Let $|w_2| = n_1 > 1$, then the primitive word exists for $n = n_1$.
For $n = n_1, 2n_1, 3n_1, ...$ the points of intersection of ℓ, language line of L with $x + y = \frac{2^n - 1}{2^n}$ gives full words. Since $|w_2| > 1, n = 1$ does not give a full word and the point of intersection lie on $x + y = 1/2$. Hence the first position becomes a hole and the entire word

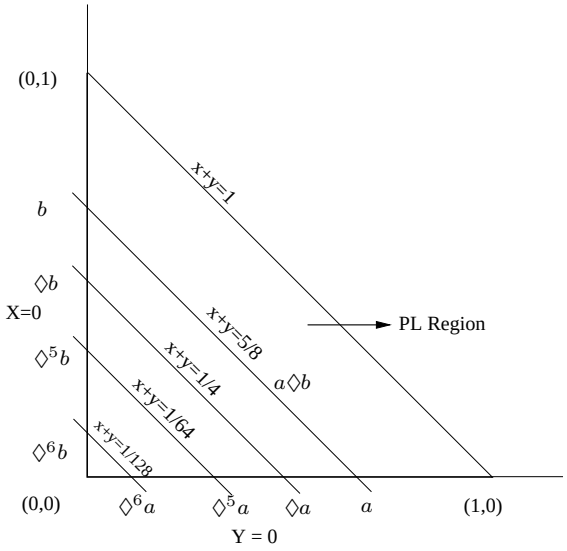

Figure 5: Diagrammatic representation of partial words

$w_2 \cup w_4$ is shifted by one place. Hence the partial word is of the form ow_4.

The next partial word of this form exists for $n = n_1 + 1$. Since there is a full word for $n = n_1$, the point of intersection of ℓ with $x + y = \frac{2^{n_1+1}-1}{2^{n_1+1}}$ gives a partial word. The full word w_2 is retained and a hole occurs in the $n_1 + 1^{th}$ place and the remaining words are shifted by one place. Hence the partial word is of the form $w_2 \Diamond w_4$. Continuing this process, we obtain the partial words are of the form $w_2^* \Diamond w_4$. □

Example 6.2 $y = 2x$
$L = \{(ba)^*\} \cup \{(ba)^\omega\}$, *where* $n_1 = 2$
For $n = 1$, *we obtain* $\Diamond(ba)^\omega$
For $n = 2$, *we obtain* ba
For $n = 3$, *we obtain* $ba\Diamond(ba)^\omega$
For $n = 1, n_1+1, 2n_1+1, ...$ *the corresponding partial words obtained are* $\Diamond(ba)^\omega, ba\Diamond(ba)^\omega, baba\Diamond(ba)^\omega, ...$
Hence in general, we get $PL = (ba)^*\Diamond(ba)^\omega$.

Theorem 6.2 *If line language is of the form* $\{w_1(w_2)^*\} \cup \{w_4\}$, $w_4 \in \Sigma^\omega$ *where* $|w_2| = 2$ *and* $|w_1| = 1$ *then the corresponding partial line language is of the form* $w_1(w_2)^*\Diamond w_4$ *only when the language line intersects the lines* $x+y = \frac{2^n-1}{2^n}$ *at* $n = 1, n_1+1, 2n_1+1, ...$

Proof Let $L = w_1(w_2)^* \cup w_4$ be a line language where $|w_2| = 2$ and $|w_1| = 1$. Let ℓ be the corresponding language line. The primitive word exists for $n = 1$. Since $|w_1| = 1$ the remaining words exist for $n = 3, 5, 7, ...$ Hence when the language line intersects $x + y = \frac{2^n-1}{2^n}$ at $n = 2, 4, 6, ...,$ the partial words exist. Since there are full words at $n = 1, 3, 5, ...$ those full words are retained followed by a hole and the remaining part of the entire word is shifted by one place hence the partial words will be of the form $w_1(w_2)^*\Diamond w_4$. □

7 Learning of Partial Line Languages

In this section, we present a learning algorithm IAL for partial line languages.

The following notations are used in the algorithm.

- common-prefix (x_1, x_2) denotes the string, which is the longest common-prefix of x_1 and x_2.
- remove-prefix (x, y) denotes the string β such that $x = y\beta$.
- common-suffix (x_1, x_2) denotes the string, which is the longest common-suffix of x_1 and x_2.
- common-subword (x_1, x_2) denotes the string, which is the longest subword x_1 and x_2.
- remove subword (x, s) denotes the string $\alpha\beta$ such that $x = \alpha s \beta$.
- mod (x_1) denotes the length of the string x_1.

Identification Algorithm (IAL)

Input : A positive presentation of a line language L.
Output : The line language L and the partial line language PL.
Procedure :
Read three positive samples x_1, x_2, x_3.
$l_1 = mod(x_1)$, $l_2 = mod(x_2)$, $l_3 = mod(x_3)$.
z_1 = common-prefix(x_1, x_2),
z_2 = common-suffix(x_1, x_2)
If $l_1 < l_2$ then
begin
I_{11} = remove-suffix(x_2, z_2),
I_{31} = remove-prefix(x_2, z_1)
end
else
begin
I_{11} = remove-suffix(x_1, z_2),
I_{31} = remove-prefix(x_1, z_1)
end

z_3 = common-prefix(x_2, x_3),
z_4 = common-suffix(x_2, x_3)
If $l_2 < l_3$ then
begin
I_{12} = remove-suffix(x_3, z_4),
I_{32} = remove-prefix(x_3, z_3)
end
else
begin
I_{12} = remove-suffix(x_2, z_4),
I_{32} = remove-prefix(x_2, z_3)
end
If $I_{11} = I_{12}$ and $I_{31} = I_{32}$ then
z_5 = common-suffix (I_{11}, I_{12})
If $z_5 = s^k$ then
begin
$w_2 = s$, w_1 = remove-suffix(I_{11}, s^k)
s_1 = common-subword (s, I_{31})
If $s \neq \lambda$ then
w_3 = remove-prefix(I_{31}, s^n)
else
w_3 = remove-prefix$(x_1, w_1 w_2^m)$
end
else
begin
z_5 = common-subword$(I_{11}, I_{31}) = s^k$,
w_1 = remove-suffix(I_{11}, s^m), $w_2 = s$
w_3 = remove-prefix(I_{31}, s^m)
end
If $w_1 = \lambda$ and $w_3 = \lambda$ then
begin
$L = \{w_2^*\} \cup \{w_2^\omega\}$
$PL = w_2^* \Diamond w_2^\omega$
end
If $w_3 = \lambda$ then
begin
$L = \{w_1 w_2^*\} \cup \{w_2^\omega\}$
$PL = w_1 w_2^* \Diamond w_2^\omega$
end

Output: $PL = w_1 w_2^* \Diamond w_2^\omega$.

Example Run

$L = a(ab)^*ba$
$x_1 = aabba, x_2 = aababbba, x_3 = aababababbba$
$l_1 = 5, l_2 = 9, l_3 = 13$
$z_1 = aab, z_2 = abba, I_{11} = aabab, I_{31} = ababba$
$z_3 = aababab, z_4 = abababba, I_{12} = aabab,$
$I_{32} = ababba$
z_5 = common-subword $(aabab, ababba) = abab$
$= (ab)^2$
$w_2 = ab$
w_1 = remove-suffix $(aabab, (ab)^2) = a$
w_3 = remove-prefix $(ababba, (ab)^2) = ba$
Hence $L = a(ab)^*ba$.
Corresponding $PL = a(ab)^* \Diamond ba$.

8 Conclusion

In this paper we have defined the three classes of partial languages and we present an algorithm for learning local partial languages in the limit from positive data. We have shown that the class of regular partial languages is learnable in the limit from positive data and restricted subset queries and we provide another learning algorithm to infer partial line languages.

References

[1] D. Angluin, "Inductive inferences of formal languages from positive data," *Information and Control*, vol. 45, 1980, pp. 117-135.

[2] D. Angluin, "Queries and Concept Learning," *Machine Learning*, vol. 2, 1998, pp. 319-342.

[3] J. Berstel and L. Boasson, "Partial words and theorem of Fine and Wilf," *Theoretical Computer Science*, vol. 218, 1999, pp. 135-141.

[4] F. Blanchet-Sadri, "Codes, orderings and partial words", *Theoretical Computer Science*, vol. 329, 2004, pp. 177-202.

[5] E.M. Gold, "Language Identification in the Limit," *Information and Control*, vol. 10, 1967, pp. 447-474.

[6] M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading Mass, 1978.

[7] A. Salomaa, *Handbook of Formal Languages*, Vol. 1, Berlin.

[8] K. Sasikala, T. Kalyani, V.R. Dare and P.J. Abisha, "Line Languages," *Electronic Notes in Discrete Mathematics*, Vol. 12, 2003.

[9] K. Sasikala, V.R. Dare and D.G. Thomas, Learning Line Languages using positive data, in "Computational Mathematics," Narosa Publishing House, 2004, pp. 153-157.

[10] T. Yokomori, "Learning local languages from positive data," Proc. Fujitsu Workshop on Computational Learning Theory, Numazu, Japan, 1989, pp. 1-16.

K. Sasikala received her M.Sc. degree in Mathematics from the Bharathidasan University in 1989 and her M.Phil. degree in Mathematics from Madras University in 1990. She is doing Ph.D. under the guidance of Dr. V.R. Dare in Madras Christian College. She is at present on the faculty of the Department of Mathematics, St. Joseph's College of Engineering, Chennai - 119.
Her research interests include in formal languages, studies on partial words and learning theory.

V. R. Dare received his M.Sc. degree and M.Phil. in Mathematics from the Madurai Kamaraj University in 1976 and 1977, respectively, and his Ph.D. degree from Madras University in 1986. He is at present on the faculty of the Department of Mathematics, Madras Christian College, Tambaram, Madras. He was a post-doctoral fellow at the Laboratory of Theoretical Computer Science and Programming (LITP), University of paris VII during 1987-1988.
His research interests include topological studies of formal languages, studies on infinite words and infinite arrays and learning theory.

D.G. Thomas received the M.Sc. degree in Mathematics in 1981 from the Madurai Kamaraj University and M.Phil. and the Ph.D. degrees in 1984 and 1996 respectively from Madras University. Since 1984, he has been on the teaching faculty of the Department of Mathematics, Madras Christian College, Tambaram, Madras.
His area of interest includes formal languages, automata, string rewriting systems, infinite and bi-infinite words, codes, learning, cryptography and picture languages.