

Sequential Pattern Mining Using Formal language Tools

Sunil Joshi¹, Dr. R. S. Jadon² and Dr. R. C. Jain³

¹ Department of Computer Applications, Samrat Ashok Technological Institute
Vidisha, India

² Department of Computer Applications, Madhav Institute of Technology and Science
Gwalior, India

³ Department of Computer Applications, Samrat Ashok Technological Institute
Vidisha, India

Abstract

In present scenario almost every system and working is computerized and hence all information and data are being stored in Computers. Huge collections of data are emerging. Retrieval of untouched, hidden and important information from this huge data is quite tedious work. Data Mining is a great technological solution which extracts untouched, hidden and important information from vast databases to investigate noteworthy knowledge in the data warehouse. An important problem in data mining is to discover patterns in various fields like medical science, world wide web, telecommunication etc. In the field of Data Mining, Sequential pattern mining is one of the method in which we retrieve hidden pattern linked with instant or other sequences. In sequential pattern mining we extract those sequential patterns whose support count are greater than or equal to given minimum support threshold value. In current scenario users are interested in only specific and interesting pattern instead of entire probable sequential pattern. To control the exploration space users can use many heuristics which can be represented as constraints. Many algorithms have been developed in the fields of constraint mining which generate patterns as per user expectation. In the present work we will be exploring and enhancing the regular expression constraints. Regular expression is one of the constraint and number of algorithm developed for sequential pattern mining which uses regular expression as a constraint. Some constraints are neither regular nor context free like cross-serial pattern $a^n b^m c^n d^m$ used in Swiss German Data. We cannot construct equivalent deterministic finite automata (DFA) or Push down automata (PDA) for such type of patterns. We have proposed a new algorithm PMFLT (Pattern Mining using Formal Language Tools) for sequential pattern mining using formal language tools as constraints. The proposed algorithm finds only user specific frequent sequence in efficient optimized way as compared to other existing algorithm. Our experimental results clearly show

that proposed algorithm is quite enhanced and improved and generates optimum frequent sequences as per user expectation.

Keywords: *Sequential Pattern Mining, Regular Expressions, Context Free Grammars, Formal Language Tools, Deterministic Finite Automata, Push Down Automata, Turing Machine.*

1. Introduction

In present scenario almost every system and working is computerized and hence all information and data are being stored in Computers. The capacity of data storage is increasing tremendously. In the present Hi-tech age the amount of data is also increasing in line with data storage capacity. Retrieval of untouched, hidden and important information from this huge data is quite tedious work [1]. Data Mining is a great technological solution which extracts untouched, hidden and important information from vast databases to investigate noteworthy knowledge in the data warehouse [2]. An important problem in data mining is to discover patterns in various fields like medical science, world wide web, telecommunication etc.

In the field of Data Mining, Sequential pattern mining is one of the method in which we retrieve hidden pattern linked with instant or other sequences [3, 4]. In sequential pattern mining we extract those sequential patterns whose support count are greater than or equal to given minimum support threshold value. The application of sequential pattern mining is to predict customer's behavior, patient's behavior and stock market behavior based on transaction history [5, 6, 7, 8]. Several different sequential pattern mining algorithms have been developed and implemented for time series databases [9]. The main objective of these

algorithms is to improve performance and effectiveness. In current scenario users are interested in only specific and interesting pattern instead of entire probable sequential pattern [10]. In this type of Data mining number of rules are generated from specific data which are in thousand or more and most of them are not correlated and not useful to us [11]. To control the exploration space users have many heuristics which can be represented as constraints. Many algorithms have been developed in the fields of constraint mining which generate patterns as per user expectation [11, 12]. Various types of constraints are suggested in literature like: knowledge type constraint, data constraint, interesting constraint, rule constraint, item constraint, regular expression constraint etc [2, 13]. In the present work we will be exploring and enhancing the regular expression constraints.

Regular expression is one of the constraints and there are number of algorithms in sequential pattern mining which uses regular expression as a constraint [14, 15, 16]. Some constraints are not regular and we can not construct equivalent deterministic finite automata (DFA) for such type of pattern like $a^n b^n$. Recently a new work for sequential pattern mining has been developed which uses context free grammar as a constraint and develops push down automata (PDA) instead of DFA [17]. Still there are some other constraints like cross-serial pattern $a^n b^m c^n d^m$ which are neither regular nor context free [18].

We propose a new theoretical framework for sequential pattern mining using formal language tools. We propose a new algorithm PMFLT (Pattern Mining using Formal Language Tools) for sequential pattern mining using formal language tools as constraints. The proposed algorithm finds only user specific frequent sequence in efficient optimized way as compared to other existing algorithm. We first recognize category of constraint pattern as per formal language classification and then develop equivalent machine which satisfies sequences as per user constraint and determines frequent sequence whose support count is greater than minimum support value. Our experimental results clearly show that proposed algorithm is quite enhanced and improved and generates optimum frequent sequences as per user expectation.

The Paper is organized as- In section 2 we discuss basic concept of sequential pattern mining. In section 3 we discuss the working and characteristics of existing sequential pattern mining algorithms which uses regular expression constraints and context free grammar constraints. We also discuss some issues and challenges of the existing algorithms. In section 4 we discuss the formal languages tools and their classification. In section 5 we present the new algorithm framework for sequential pattern mining using formal language tools and discuss its

illustration. In section 6 we analyze the performance of proposed algorithm. In section 7 we present the conclusion.

2. Basic Concept

The primary goal of Sequential Pattern Mining algorithms is to determine frequent sequences in the given database. Algorithms for this problem are mined sequentially where dataset are ordered like in the field of bioinformatics example of sequences are amino-acids and nucleotide sequences. The problem was first introduced by Agrawal and Srikant, where the basic concepts involved in pattern detection were established [4]. In the past years, several sequential pattern mining algorithms were proposed, but not all assume the same conditions. The use of some constraints has a great impact on their design and efficiency [17, 19, 20].

Some basic definitions are needed, in order to formally introduce the problem [1, 2]:

Items: - An itemset is a non-empty subset of elements from a given set C , the item collection is known as items.

Sequence- A sequence is a list of item sets in sorted order. A sequence called maximal if it is not enclosed in any other sequence. K items sequence is called a k -sequence. The length of the sequence S is number of elements and it is denoted by $|s|$.

Support—The ratio of the number of existing frequent sequence (F) and the number of possible sequences (S) is the database density (p) and it is calculated by $p=F/S$.

Frequent Sequence- Given D as database of sequences, and a few user-specified minimum support threshold σ and constraint K . A sequence is frequent if it is satisfying the constraint and support at least σ .

3. Related Work

Recently, Regular language and Context-Free languages have been proposed [11, 12, 14, 21, 22, 23, 24, 25] to be used as a constraint to reduce the number of discovered patterns. In computing, a regular expression is an expression that describes a language (set of strings), according to specific syntax rules and is called a pattern. The need to search for regular expressions arises in many text based applications, such as document retrieval, text editing and computational biology. There are many applications of Regular expression like search and modifying text in text editors and utilities based on given patterns. Various programming languages like Java etc. provide packages of regular expressions for string search and modification. Another grammar known as CFG (Context Free Grammar) is powerful than regular grammar and most programming languages syntax are often

described by CFG. Most existing literatures for searching regular expression are usually done by converting the regular expression into a deterministic finite automaton [26, 27, 28, 29].

Throughout the last two decade, lot of people have implemented and compared several algorithms that try to solve the sequential Pattern Mining problem as efficiently as possible. Out of these several algorithms which has made a significant contribution to improve the efficiency of sequential pattern mining using regular expression constraints some are SPIRIT [14], PrefixGrowth [12], RE-Hackle [21], MILPRIT [15], SMA [16]. Regular expression generates type-3 language or regular language which is the type of formal language classification. Most of the work developed in this type-3 language category of formal language classification.

SPIRIT [14] is very popular and well known algorithm which generates pattern as per constraints given by user in the form of regular expression. It is apriori like algorithm and its working is similar to GSP [14]. It generates candidates and satisfies by regular expression constraints. It does not require any complex data structure or graph [9]. It is based on BFS which uses horizontal representation of database [28].

PrefixGrowth [11] is a pattern-growth method, like PrefixSpan that generate only those sequences which satisfies the constraint R and according to the constraint sequences that are prefixes of valid sequences.

RE-Hackle algorithm (Regular Expression Highly Adaptive Constrained Local Extractor) [21] is a highly adaptive algorithm, which represents tree structure of regular expression. An Abstract Syntax Tree which encodes the structure of the canonical form of a RE-constraint is called Hackle-tree. Each intermediated node represent operator and leaf node contain sequences which are atomic. These atomic sequences are assembled by the union, concatenations, and Keene closures to structure the first RE-constraint.

MILPRIT (Mining Interval Logic Patterns with Regular expressions constraints) [15, 22] is intended to mine first order temporal patterns, where time is represented by intervals, and regular expression constraints are used to confine the search space. It also belongs to the apriori class.

Recently **SMA** [16] proposed new algorithm which uses regular expression as a constraint and develops new Petri Net type automata. It generates only those sequences which are satisfied by Automata and whose support count are greater than or equal to minimum support threshold value. First algorithm based on the SMA: SMA-1P. This algorithm simply passes through the SMA every sequence in the input database, producing all the valid patterns, whose frequency is then counted. Second method based on frequency pruning. The method is named SMA-FC, and it

performs a number of scans of the database equals to the number of states of the given regular expression. To evaluate the effectiveness of the SMA over SPIRIT, we perform experiment with synthetic dataset data_500K-50l_20l. The C++ implementations of both algorithms and dataset are downloaded from link given in article [16]. The runtime behavior of both algorithms are shown in figure 1.

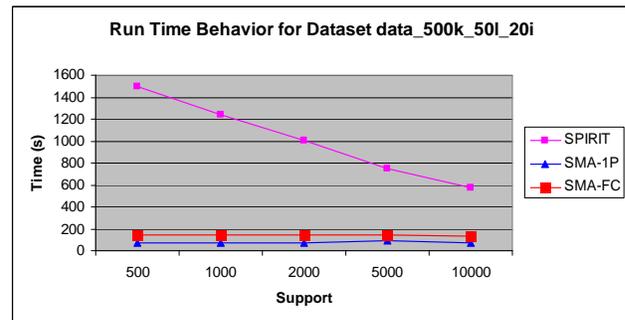


Fig. 1 The Run Time of SMA is compared with that of SPIRIT

We performed experiment on two available approaches of SMA one is SMA-1P (One Pass) and other is SMA-FC (Full Check). We Compare the result of SMA with SPIRIT (V) over Regular Expression $A*B(B/C)D^*E$. It is clear that SMA is faster than SPIRIT algorithm and also generates less number of frequent sequences. SMA is fastest and efficient method for sequential pattern mining using Regular expression constraints [16]. If pattern is $A*B(B/C)D^*E$ and Transaction $T=ACDBFAAEBFCFDDE$ then SMA generate 18 frequent sequences. The program outputs are shown in Figure 2 in terms of frequent sequences.

```

S31.TXT - Notepad
File Edit Format View Help
AAABCDDE supp:1
AAABCE supp:1
AAABCE supp:1
AABCDDE supp:1
AABCE supp:1
AABCE supp:1
ABBDE supp:1
ABBDE supp:1
ABBE supp:1
ABBE supp:1
ABCDDE supp:1
ABCDE supp:1
ABCE supp:1
BBDE supp:1
BBDE supp:1
BBE supp:1
BCDDE supp:1
BCDE supp:1
BCE supp:1
    
```

Fig. 2 Frequent Sequences are generated from SMA

Sequence Mining Automata (SMA) has lot of advantages over SPIRIT and it is proved by above experiments. If pattern is $A^nB(B/C)D^nE$ then out of 18 sequences which are shown in Figure 2 then only three sequences satisfy this constraint and they are AABCDDE, ABBDE, ABCDE. This type of constraints is context free constraints and we can not construct the Deterministic Finite Automata for such type of expression and at this point method failed because Deterministic Finite Automata cannot count the occurrences of each character. Recently some authors pointed the same type of problem in their research work for Customer Bill Payment pattern problem [17]. Antunes developed a new algorithm SpiCFLComplete which uses context free grammar as a constraint and develops push down automata instead of deterministic finite automata for accepting such types of constraints. His algorithm approach is similar to apriori approach and author claims that substitution of regular expression constraint to context free grammar constraints does not affect the performance [17]. This is very powerful and beneficial because we know that context free languages are more powerful than regular.

Still we have another problem, suppose constraint is $A^nB(B/C)D^nE$ then only two sequences satisfy constraint and they are ABBDE and ABCDE. The sequence AABCDDE is frequent but not interesting because users are interested in only those sequences in which number of A is equal to Number of D as well as number of E. This is another category of constraint and it is not regular as well as context free so we can not construct DFA or PDA to accept such types of constraints.

Number of algorithms has been developed for sequential pattern mining using regular expression and context free grammar constraint discussed above. But some constraints are neither regular nor context free. We have number of motivational examples like an example given by Shieber [18] from Swiss German and similar phenomena occurs in Dutch. Stuart shieber [18] pointed two facts case-marking and cross-serial order pattern of Swiss German data in his article. He shows in his example cross-serial dependencies Pattern like $A^nB^mC^nD^m$ and prove that it is not context free. We know that the languages which are not context free cannot make push down automata for such language. Jackson [30, 31] also emphasizes on cross-agreement pattern like $A^nB^mC^nD^m$ for RNA/DNA. He introduces ξ -calculus and open new research trends in bioinformatics and linguistics fields. English language is not regular [32] as well as context free. It has unlimited number of nested dependencies and it contains mirror image properties. If language has limitless crossing dependencies then it not context free. Cross-serial dependencies is present in Dutch [33]. Dutch structure is similar to copy-language and it is not context-free which is shown in Table I.

Table 1: Example of New Constraints

Language Structure	Pattern	Class
German	$A^nB^mC^mD^n$	Context Free
Dutch	$A^nB^mC^nD^m$	Not Context-Free
Swiss German	$A^nB^mC^nD^m$	Not Context-Free

Formal language or natural language data consists of symbols and strings which is the set of symbols. There is a need to retrieve pattern from such type of language and this type of problem will be very useful in various fields like bioinformatics, data mining etc [34]. A verb sequence of each of the sequence comes after the noun for same length in Swiss German data. They follow $A^nB^nC^n$ and $A^nB^mC^nD^m$ which is not context free but is mildly context sensitive. This cross-serial pattern $A^nB^mC^nD^m$ is also used for generating pattern for Kambi and Non-Kambi Kolam [35]. Nested embedding pattern and multiple crossing dependencies is also used in phonological words [36].

This is the motivation to explore all other categories of patterns and hence we move to higher level of Chomsky classification because such types of patterns belong to that category. Hence we develop a new framework for sequential pattern mining using formal language tools. Now we discuss formal language tools and their classification in next section.

4. Formal Language Tools

Formal language or natural language data consist of symbols and strings which is the set of symbol. Formal language is a set of words in particular pattern. Words are made by symbols collectively one after other without space. The word combination according to grammatical rules creates syntax or pattern [37]. Our work is based on uses of Formal languages tools as constraints in sequential pattern mining.

Formal Grammar is formally represented by 4-Tuple $G=(V, T, P, S)$, Where V is the set of non-terminals or variable, T is the set of constants/input/terminals, S is the start symbol of the grammar; P is the set of production rule of the form $\alpha \rightarrow \beta$, $\alpha \in (VUT)^+$, $\beta \in (VUT)^*$, α contains at least one variable.

Formal Language Tools: Formal language is character or symbol oriented language which is used in theory of computation or compiler design. Chomsky represents classification of formal language [32]. Chomsky given classification based on production Rule. This classification is also known as Chomsky hierarchy which is shown in table 2.

Table 2: Chomsky Classification with Equivalent Machines

S. No.	Type Of Grammar	Type of Language	Production Rule	Tools/ Machine
1	Type-0 Grammar or Unrestricted Grammar	Type-0 Language or Unrestricted Language or Recursively Enumerable Language	$\alpha \rightarrow \beta$ $\alpha \in (VUT)^+, \beta \in (VUT)^*$ Example: $A^n B^n C^n \mid n \geq 1$	Turing Machine
2	Type-1 Grammar or Context Sensitive Grammar	Type-1 Language or Context Sensitive Language	$\alpha \rightarrow \beta$ $\alpha \in (VUT)^+, \beta \in (VUT)^*$ $ \alpha \leq \beta $ Example: $A^n B^m C^m D^m \mid n, m \geq 1$	Linear Bounded Automata
3	Type-2 Grammar or Context Free Grammar	Type-2 Language or Context Free Language	$A \rightarrow \beta$ A is single Variable, $\beta \in (VUT)^*$ Example: $A^n B^n \mid n \geq 1$	Push Down Automata
4	Type-3 Grammar or Regular Grammar	Type-3 Language or Regular Language	$A \rightarrow \beta x$ A is single variable , $\beta \in (VUT)^*, x \in T^*$ Example: $A^* B^*$	Finite State Machine

Chomsky classifies formal language into four categories on the basis of production rules [32]. Now we discuss these categories with related work in context of pattern mining as follows

Type-3 or Regular Language → Machine works on that grammar is known as finite automata, so if we want to apply regular expression constraint then we need to develop finite automata, pass each sequence on that automata, if automata accept it then we count occurrence of sequence, if occurrence of sequence exceed support count then it is frequent. This concept used in various algorithms like SPIRIT, MIPLPRIT, SMA etc [14, 15, 16].

Type-2 or Context Free Language → Context free language or Context free grammar belong to type-2 category and machine works on that grammar is known as push down automata , so if we want to apply context free grammar constraint then we need to develop push down automata. Context free languages are more expressive than regular language i.e. $a^n b^n$ is not regular but it is context free. Antunes [17] developed a new methodology and algorithm which uses the context free grammar constraints.

Still we have another two categories of language which are more expressive than previous two languages.

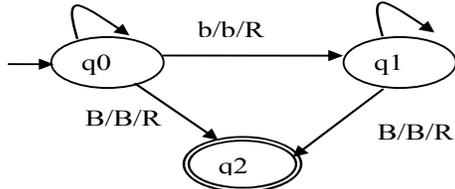
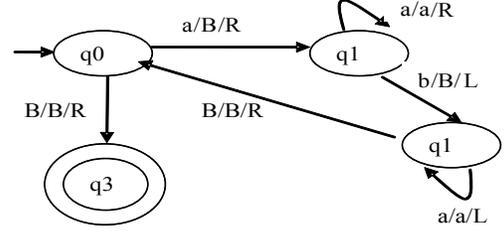
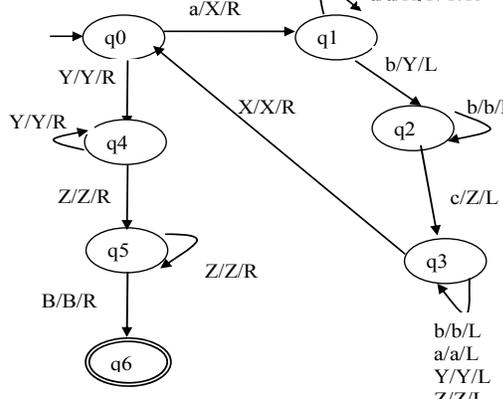
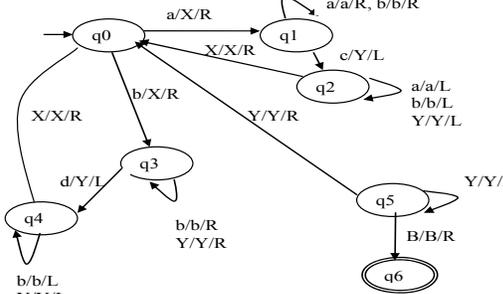
Type-1 or Context Sensitive Language → Machine works on that grammar is known as linear bounded automata or Turing machine. Context sensitive languages are more expressive then Context Free i.e. $a^n b^n c^n$ is not context free but it is context sensitive.

Type-0 Language or Type -0 Grammar – Unrestricted grammar or unrestricted language, Machine works on that grammar is known as Turing machine [38]. Unrestricted languages are more expressive than context free. The Turing machine is the formal language tools or machine of unrestricted grammar or type-0 grammar. Type-0 Language is the super set of all type-1 CSL, Type-2 CFL and Type-3 RL. So it is very popular and strong tools for accepting pattern strings of any class. It was developed by Alan Matheson Turing in the year 1936[38]. The languages accepted by Turing machine are called unrestricted language or recursive enumerable. We can construct Turing machine for all types of constraints [39]. Table 3 shows different types of constraints with grammar and their equivalent Turing machine Transition Diagram.

5. Pattern Mining using Formal Language Tools (PMFLT): A NEW Algorithm

The discussion of previous section motivates us to use formal language tools as a constraint in sequential pattern mining algorithm. In this section, we present new algorithm for Sequential Pattern Mining using formal language tools which first recognize constraint category from formal language classification and then generate equivalent machine to accept such type of constraints and determine frequent sequences in optimized way. Now we discuss its methodology in subsequent section.

Table 3: Constraints Types and its Equivalent Turing Machines

Constraints Type/Class	Equivalent Grammar	Equivalent Turing Machine
a^*b^* Regular Constraints	$S \rightarrow aS/bX$ $X \rightarrow bX/\epsilon$	
$a^n b^n$ Context Free Constraints	$S \rightarrow aSb/ab$	
$a^n b^n c^n$ Context Sensitive Constraints	$S \rightarrow aXbc/abc$ $Xb \rightarrow bX$ $Xc \rightarrow Ybcc$ $bY \rightarrow Yb$ $aY \rightarrow aaX/aa$	
$a^n b^m c^n d^m$ cross-serial pattern	$S \rightarrow aXcZ/aYcZ$ $X \rightarrow aXc/aYc$ $Yc \rightarrow cY; Yb \rightarrow bY$ $YZ \rightarrow Wd; cW \rightarrow Wc$ $bW \rightarrow Wb; aW \rightarrow ab$ $YZ \rightarrow FZd; cF \rightarrow Fc$ $bF \rightarrow Fb; aF \rightarrow abY$	

5.1 The Proposed Methodology

The Proposed Algorithm works on entire database. The Algorithm requires three types of Input Parameter Database, Minimum Support and Constraint. Firstly algorithm calls the Procedure MFLT (Algorithm for equivalent machine using formal Language tools) and pass given constraints. The Procedure MFLT recognizes the category of constraint and constructs the equivalent machine. If recognition is not possible or rigid then by default construct Turing machine for given constraint because Turing machine accept all types of constraint and language.

We have used Turing machine in our research work. After this the algorithm reads each transaction one by one and generates sequence with the help of equivalent machine and store in sequence list with their frequency. At last it reads sequence from sequence list one by one and prints only those sequences whose support counter are greater than or equal to minimum support threshold value. This approach is very easy and generates frequent sequence in optimized way without candidate generation and does not require any complex graph or data structure. Our objective is to generate a sequence which satisfies given constraint and their support counts are greater than or equal to minimum support. The pseudo code of the proposed algorithm PMFLT is shown in figure 3.

Algorithm PMFLT (Pattern Mining Using Formal Language Tools)

Input: D Database, δ minimum Support, C Constraint

Output: Sequence $S \in$ Language $L(C)$ and $Support(S) \geq \delta$ minimum Support

Algorithm:-

// recognize Constraint's category from formal language Classification and construct equivalent machine

- I. $M = MFLT(C)$
- II. for all $T \in$ Database D do
- III. Generate Sequence $S_m(T)$ satisfied by Machine M
- IV. for all $s \in S_m(T)$ do
- V. if s is not in S_L then $S_L[s].counter = 1$ else $S_L[s].counter = S_L[s].counter + 1$
- VI. for all $f \in S_L$ do
- VII. if $S_L[f].count \geq \delta$ then print f

Procedure MFLT (Algorithm for equivalent machine using Formal Language Tools)

Input Parameter: Constraint C

Return Parameter: Equivalent machine M to accept Language of C

// Check the category of constraint C from Formal Language Classification

1. Switch Case Formal_Language_Category (C)
2. Case 'Type-3 Regular Language'
 - Construct Deterministic Finite Automata A_c to Accept Language of C
 - return A_c
3. Case 'Type-2 Context Free Language'
 - Construct Push Down Automata P_c to Accept Language of C
 - return P_c
4. Case 'Type-1 Context Sensitive Language'
 - Construct Turing Machine T_c to Accept Language of C
 - return T_c
5. Case 'Type-0 Unrestricted Language'
 - Construct Turing Machine T_c to Accept Language of C
 - return T_c

// if Recognition is not possible and complex

// then return Turing machine

Case Default

- Construct Turing Machine T_c to Accept Language of C
- return T_c

Fig. 3 Proposed Algorithm PMFLT Pseudo Code

5.2 Explanation of Algorithm with Example

Consider Transaction $T=AABCCDE$ and given constraint is $a^nbc^nd^n \mid n \geq 1$. We know that given constraint is not regular or not context free so we can construct Turing machine for such types of constraints. Figure 4 presented Turing Machine for given constraint.

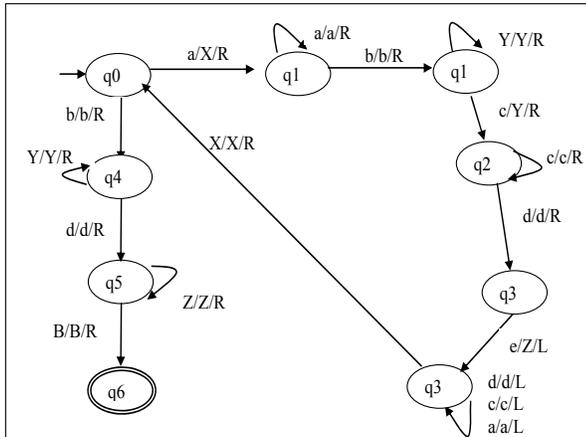


Fig. 4 Turing machine equivalent to Constraint $a^nbc^nd^n \mid n \geq 1$

Now we use Transaction $T=AABCCDE$ on given constraint and Run different algorithm. Table 4 presented frequent sequences generated from SMA with Regular Expression Constraint $a^*bc^*de^*$, SpiCFLComplete with Context Free Grammar Constraint $a^nbc^nd^n$, PMFLT $a^nbc^nd^n$.

Table 4: Comparison of the frequent sequences found In PMFLT with SMA and SpiCFLComplete

SMA with $a^*bc^*de^*$ Regular Expression	SpiCFLComplete with $a^nbc^nd^n$ Context Free Grammar Constraint	PMFLT with $a^nbc^nd^n$ Unrestricted Grammar Constraint
aabccd, aabccde, aabcd ,aabccde, aabd ,aabde, abccd ,abccde,abcd , ,abcde ,abd ,abde ,bccd ,bccde, bcd ,bcde ,bd ,bde	aabccd ,aabccde abcd , abccde	abcde
Total Frequent Sequence =18	Total Frequent Sequence =4	Total Frequent Sequence =1

We note that new proposed algorithm PMFLT generates optimized frequent sequences because it generates only those sequences that are satisfied by constraint. The constraint given in above problem is $a^nbc^nd^n$ in which number of a, b and c are equal. Only one sequence abcde satisfies it.

6. Performance Analysis

Our objective is not scalability even our objective is to find sequence in optimized way as per given constraint. To perform study we used constraint $a^nbc^nd^n$ and synthetic dataset data_500k_50l_20i downloaded from link given in article [16]. All experiments were performed on an Intel® Pentium® 4 CPU, 2.26 GHz and 1 GB of RAM computer on Windows XP Operating System. The Object Oriented implementation of SMA was downloaded from link given in his article [16]. We generate frequent sequences from SMA using equivalent regular expression of given constraint and then analyze generated sequence as per context free grammar constraint used in SpiCFLComplete algorithm and unrestricted constraint used in proposed algorithm PMFLT. The testing results of experiments are showed in Figure 5. In the Graph X axis represent Support value and Y axis represent Number of frequent Sequences.

It is clear that proposed algorithm work well. Our main objective is just optimization of frequent sequence generation not scalability. It is clear that most of the part of execution time is utilized in counting the support of sequence. So if candidates are already optimized then execution time is also optimized. That's why we have not analyzed the run time analysis because we have already compared and quoted that existing algorithm gives best and fast results in context of time.

Our objective is only to introduce new types of constraint and follow the existing work in new innovative direction. We presented all the category of formal language and proposed new frame work for sequential pattern mining using formal language tools.

6. Conclusion

Based on the analysis of above sequence data it is clear that if we use formal language tools as constraint in sequential pattern mining then we can reduce the explore space by categorizing constraint type and generate only those sequences which are frequent and satisfied by given constraint. Performance gives the evidence that technique is well improved.

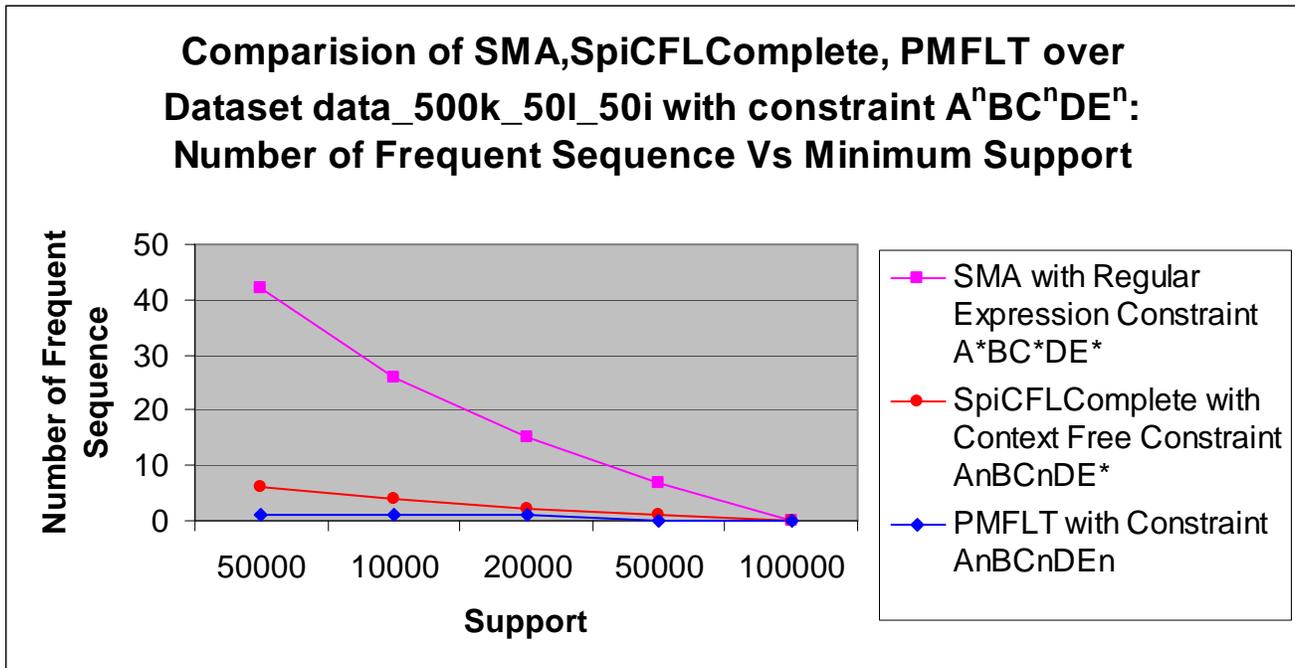


Fig. 5 Performance Result of PMFLT for dataset data_500k_50I_20i

We use Formal language tools for categorizing constraint and generate appropriate equivalent machine or if recognition is not possible then construct Turing machine because it is very powerful and accept all types of languages. It reduces unnecessary or uninteresting sequences by equivalent machine.

Our experimental results gives evidence that proposed algorithm works well. Still there is always some option for future scope. We can optimize proposed algorithm in terms of time also.

References

- [1] V. Pudi, and P. Radha Krishna, "Data Mining", Fourth Impression, India: Oxford University Press, 2011.
- [2] J. Han, and M. Kamber, "Data Mining: Concepts and Techniques", First Indian Reprint, India: Harcourt (India) Private Limited, 2002.
- [3] Q. Zhao, "Sequential Pattern Mining: A Survey", Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003118, July 2003.
- [4] R. Agrawal, and R. Shrikant, "Mining sequential patterns," *Proc. Eleventh International Conference on Data Engineering*, 3-14, March 1995
- [5] S de Amo, P. Waldecir: Mining Temporal Relational Patterns over Databases with Hybrid Time Domains, XXII Symposium, 2007.
- [6] A. Xue, S. Hong, S. Ju, W. Chen, "Application of Sequential Patterns Based on User's Interest in Intrusion Detection", *IEEE International Symposium on IT in Medicine and Education*, 1089-1093, December 2008.
- [7] I. S. Sitanggang, N. A. Husin, A. Agustina, N. Mahmoodian, "Sequential Pattern Mining on Library Transaction Data", *IEEE International Symposium in Information Technology (ITSim)*, 1-4, June 2010.
- [8] X. Tang, Q. Zeng, T. Cui, Z. Wu, "Regular Expression-based Reference Metadata Extraction from the Web", *IEEE 2nd Symposium on Web Society (SWS)*, 346-350, August 2010.
- [9] N. A. Sajid, S. Zafar, and S. Asghar, "Sequential pattern finding: A survey," In *International Conference on Information and Emerging Technologies (ICIET)*, 1-6, June 2010.
- [10] S. Sakurai, Y. Kitahara, R. Orihara, "Sequential Pattern Mining based on a New Criteria and Attribute Constraints, *IEEE Conference*, 2007.
- [11] J. Pei, J. Han, and W. Wang, "Mining Sequential Patterns with Constraints in Large Databases," in *Conference on Information and Knowledge Management*, 18-25, November 2002.
- [12] C. M. Antunes, and A. L. Oliveira, "Mining Patterns Using Relaxations of User Defined Constraints", in Proc. of 3rd *International Workshop on Knowledge Discovery in Inductive Databases (KDID 2004)*, September 2004.

- [13] J. Bisaria, N. Srivastava, K. R. Pardasani, "A Rough Set Model for Sequential Pattern Mining with Constraints", *International Journal of Computer and Network Security*, Vol 1 No.2 November 2009.
- [14] N. N. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential pattern mining with regular expression constraints," *Proc. International Conference on Very Large Databases*, 223-234, September 1999.
- [15] S. de Amo, A. Giacometti, and M. S. Santana, "Milprit: Mining interval logic patterns with regular expression constraints," In *1st Brazilian Workshop on Data Mining Algorithms*, October 2005.
- [16] R. Trasarti, F. Bonchi, and B. Goethals, "Sequence Mining Automata: a New Technique for Mining Frequent Sequences Under Regular Expressions," *Proc. Eight International Conference on Data Mining*, 1061-1066, December 2008.
- [17] C. M. Antunes, and A. L. Oliveria, "Using Context-Free Grammars to Constrain Apriori-based Algorithms for Mining Temporal Association Rules", in *Proc. 2nd Workshop on Temporal Data Mining – International Conference Knowledge Discovery and Data Mining*, pp. 11-24. July 2002.
- [18] S. M. Shieber, "Evidence against the context-freeness of natural language," *Linguistics and Philosophy*, 333–343, November 1985.
- [19] K. C. Kandpal, R. Agnihotri, "SBLOCK-A Closed Sequential Pattern Mining Algorithm", *International Journal of Computer Applications in Engineering Sciences*, Vol. 1, issue III, 296-299, September 2011.
- [20] S. Sakuria, Y. Kitahara, R. Orihara, "A Sequential Pattern Mining Method based on Sequential Interestingness", *World Academy of Science, Engineering and Technology*, Issue 23, 542-550, November 2008.
- [21] H. Albert-Lorincz, and J. Boulicaut, "Mining frequent sequential patterns under regular expressions: A highly adaptive strategy for pushing constraints," In *Proc. of 3rd SIAM International Conference on Data Mining*, 316-320, May 2003.
- [22] S. de Amo, and D. Furtado, "First-order temporal pattern mining with regular expression constraints," *Data & Knowledge Engineering*, Vol. 62, Issues 3, 401-420, September 2007.
- [23] C. M. Antunes, and A. L. Oliveria, "Sequential Pattern Mining with Approximated Constraints," in *Proc. of IADIS International Conference on Applied Computing*, 131-138, March 2004.
- [24] C. M. Antunes, and A. L. Oliveira, "Mining Patterns Using Relaxations of User Defined Constraints", in *Proc. of 3rd International Workshop on Knowledge Discovery in Inductive Databases (KDID 2004)*, September 2004.
- [25] L. I. Gomez, and A. A. Vaisman, "RE-Spam: Using Regular Expressions from Sequential Pattern Mining in Trajectory Databases," *IEEE International Conference on Data Mining Workshop*, 395-398, December 2008.
- [26] P. Hingston, "Using Finite State Automata for Sequence Mining," In *Twenty-Fifth Australasian Computer Science Conference Melbourne*, 105-110, February 2001.
- [27] R. Ivancsy, and V. Istvan, "Automata theory Approach for Solving Frequent Pattern Discovery Problems," *World Academy of Science, Engineering and Technology*, 203-208, October 2005.
- [28] R. Ivancsy, and I. Vajk, "Efficient Sequential Pattern Mining Algorithms," *WSEAS Transactions on Computers*, Vol. 4, Num. 2, 96-101, February 2005.
- [29] D. Ficara, S. Giordano, and G. Procissi, "An Improved DFA for Fast Regular Expression Matching," *ACM SIGCOMM Computer Communication*, Volume 38, Issues 5, 29-40, October 2008.
- [30] Jackson, "Adaptive Predicates in Natural Language Parsing," *Perfection*, Vol. 1 No. 4, Apr., 2000.
- [31] J. Q. Tyler, "The λ -Calculus: Manageable Context-Sensitive Parsing," Faculty of Engineering Sciences, Computer Science Division, Peru, pp. 1-14, 20 Apr. 2005.
- [32] Chomsky, "Noam Chomsky, "Syntactic structures. Mouton, The Hague. 1957.
- [33] Huybregts, "Overlapping dependencies in Dutch". *Linguistics*, 24-65, 1976.
- [34] A. Clark, C. C. Florencio, and C. Watkins, "Languages as hyperplanes: grammatical inference with string kernels," *Mach Learn*, Vol. 82, 351–373, September 2010.
- [35] T. Robinson, "Extended Pasting Scheme for Kolam Pattern Generation", *Forma by Published Science Press*, 22, 55-64, September 2007.
- [36] J. Heinz, and W. Idsardi, "Why Are sentences more complex than words" *Science*, 33-35, July 2011.
- [37] J. Hopcroft, and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison Wesley. 1979.
- [38] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society*. 2 42: 230–65. 1936.
- [39] C. H. Sumitha, O. G. Krupa, "Implementation of Recursively Languages in Universal Turing machine", *International Journal of Computer Theory and Engineering*, Vol.3 No.1 February 2011.

Sunil Joshi is presently working as a Assistant Professor, Computer Applications at Samrat Ashok Technological Institute Vidisha (M.P). He has 11 years teaching experience and 4 years research experience. His research areas include Data mining.

R S Jadon is presently working as a Head, Computer Applications at Madhav Institute of Technology and Science, Gwalior. He has 16 years research experience. He has presented research papers in more than 30 national and international conferences and published more than 30 papers in national and international journals. His research areas include Video Data Processing.

R C Jain is presently working as a Director and Head; Computer Applications at Samrat Ashok Technological Institute Vidisha He has 35 years teaching experience and 20 years research experience. He has presented research papers in more than 100 national and international Conferences and published more than 150 papers in national and international journals. His research areas include Data mining and Network security.