Enhancement of User's Call Logging facilities using Push Down Automata (PDA) with Real Time Constraint Notation (RTCN)

Vivek Kr. Singh[1], S. P. Tripathi [2], J. B. Singh[3] and R. P. Agarwal[4]

[1] School of Computer Science & Information Technology, Shobhit University, Research Scholar Meerut, U.P. 250110, India

[2] Department of Computer Science & Engineering, Gautam Buddh Technical University, IET Lucknow, U.P. 226021, India

[3] School of Computer Engineering & Information Technology, Shobhit University, Professor Meerut, U.P. 250110, India

[4] School of Computer Engineering & Information Technology, Shobhit University, Professor Meerut, U.P. 250110, India

Abstract

This paper highlights the use of Push – Down Automata (PDA) in storing and maintaining the call logs. The special feature about this paper is to maintain incoming call record from the different mobile service provider in a mobile in clustered way to the user. It focuses on real time constraint notation being applied to the push down automata for formal verification of the model.

Keywords: *Push Down Automata (PDA), Object Constraint Language(OCL)*

1. Introduction

The focus of this paper is to provide a framework that can record the calls coming to a particular mobile service provider with special characteristics of logging the service provider of each call. In order to achieve this goal we have identified two basic steps, the first step deals with the development of Ubiquitous computing environment as the calls coming will be from various sources. Secondly, development of push down automata model for call logs database management system. Since OCL [5] is being applied on Object Oriented Language [2] we will be representing the entire communication framework into Object Oriented Language. In database application it is beneficial as a part of database schema.

2. An Introduction to Push Down Automata (PDA):

A Push Down Automata (PDA) is finite automata with auxiliary storage devices called *stacks*. A pushdown automaton (PDA) may be pictured as a finite automaton with the **stack** or **pushdown store** onto which symbols may be 'pushed' or from which they may be 'popped'

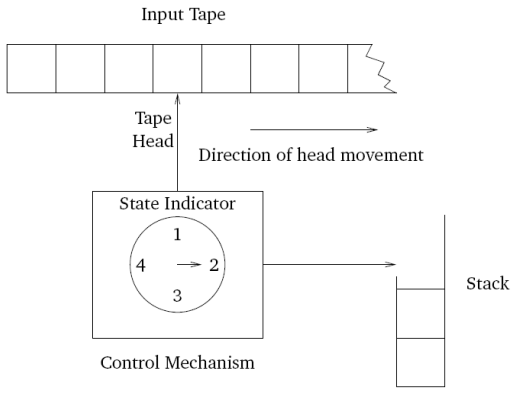

Fig. 01 : Push Down Automata Model

2.1 Transitions in Pushdown Automata

A Transitions in Push Down Automata (PDA) is depends on

1. The current state of the machine;
2. The symbol read from the input tape;
3. The top symbol on the stack.

Outcomes of the transition are

1. After reading the input alphabet current state may be transit to new state or on same state (loop condition)
2. On the top of the Stack new Stack Alphabet may be Pushed or current Top of the Stack Alphabet may be popped
3. No change on the Top of the Stack

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

2.2 Formal Definition of Pushdown Automata (PDA)

A Pushdown Automata (PDA) M is represented by seven tuples as (Q, Σ, Γ, δ, q_0, Z_0, F) where

1. Q is finite set of states
2. Σ is finite set of input alphabets
3. Γ is finite set of Stack alphabets
4. q_0 is the initial state and $q_0 \in Q$
5. F is the finite set of final states and $F \in Q$
6. δ is the Transition Function defined as

$$\delta: Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow (Q \times \Gamma^*)$$

Transtion Fuction δ can defined the moves as two types : First type of moves in which PDA reads one input symbol from the input tape. Let $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$ and

$$\delta(q, a, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), (q_3, \gamma_3), \ldots, (q_m, \gamma_m)\} \ldots. (1)$$

where $q_1, q_2, q_3, \ldots. q_m \in Q$ and $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_m \in \Gamma^*$
Equation (1) can be defined as : PDA is in state q, reads input alphabet **a** from the input tape and Z is the top symbol of the stack, PDA can change its state to any one of $q_1, q_2, q_3, \ldots., q_m$ and replace top of the stack by $\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_m$ i.e. if the next state is q_1, Z will be replaced by γ_1, if the next state is q_2, Z will be replaced by γ_2 …. and so on.
Second type of moves in which no input symbol is read from the input tape i.e. existence of null string (ε) then in this case Equation (1) can be represented as

$$\delta(q, \varepsilon, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), (q_3, \gamma_3), \ldots, (q_m, \gamma_m)\} \ldots. (2)$$

2.3 Instantaneous Description (ID) :

A configuration of PDA at a given instant of time is called instantaneous description (ID), is defined to be a member of $Q \times \Sigma \times \Gamma^*$: The first component is the state of the machine, the second is the portion of the input yet to be read and third is the contents of the stack i.e. for every $(q, \gamma) \in \delta(p, a, Z)$ and for every $x \in \Sigma^*$ and $\alpha \in \Gamma^*$ we define

$$(p, ax, Z\alpha) \vdash (q, x, \gamma a)$$

2.4 Acceptance by PDA through Empty Stack:

Let a string w is accepted by Empty Stack or Null Stack by M if and only if

$(q_0, w, Z_0) \vdash (q, \varepsilon, \varepsilon)$ for some $q \in Q$

A Push Down Automata accepted by empty stack is represented as :

$M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, \Phi\}$

3. An Introduction to Object Constraint Language (OCL) & Real Time Constraint Notation (RTCN):

The Object Constraint Language (OCL) is a modeling language with which we can build software models. It is defined as a standard "add-on" to the Unified Modeling Language (UML), the Object Management Group (OMG) standard for object-oriented analysis and design. Every expression written in OCL relies on the types (i.e., the classes, interfaces, and so on) that are defined in the UML diagrams. The use of OCL therefore includes the use of at least some aspects of UML.
OCL expressions can be used anywhere in the model to indicate a value. A value can be a simple value, such as an integer, but it may also be a reference to an object, a collection of values, or a collection of references to objects. An OCL expression can represent, e.g., a boolean value used as a condition in a statechart, or a message in an interaction diagram. An OCL expression can be used to refer to a specific object in an interaction or object diagram.
An outstanding characteristic of OCL is its mathematical foundation. It is based on mathematical set theory and predicate logic, and it has a formal mathematical semantics.
A real-time constraint notations are used in object-oriented language that provide sufficient real-time specifications as one may expect from a real-time language, while integrating these specifications within the object-oriented tapestry [6,7,8,9]. The significant aspects in object oriented real-time modeling can be identified as:
1. The use of inheritance and redefinition of real-time constraints through the inheritance hierarchy and extension of the inheritance of the state and behavior of a class to include the definition of temporal constraints.
2. The reuse of the temporal constraint specifications through the inheritance mechanism and across class boundaries. These temporal constraints can be referenced to formulate new constraints in a consistent manner in classes.
3. Time-abstraction seems like a natural approach to specifying timing constraints, where the constraint is defined at the class definition and then associated with the class implementation. The temporal behavior of an object is made independent of the object's actual implementation by relating the temporal constraint to a labeled code block..[14]

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

4. Proposed Work: Push Down Automata Model for Mobile Call Logs Storage in Clustered way:

Working of Push Down Automata Model for Mobile Call Logs Storage in Cluster way can be represent by the flow chart as shown in Fig. 2

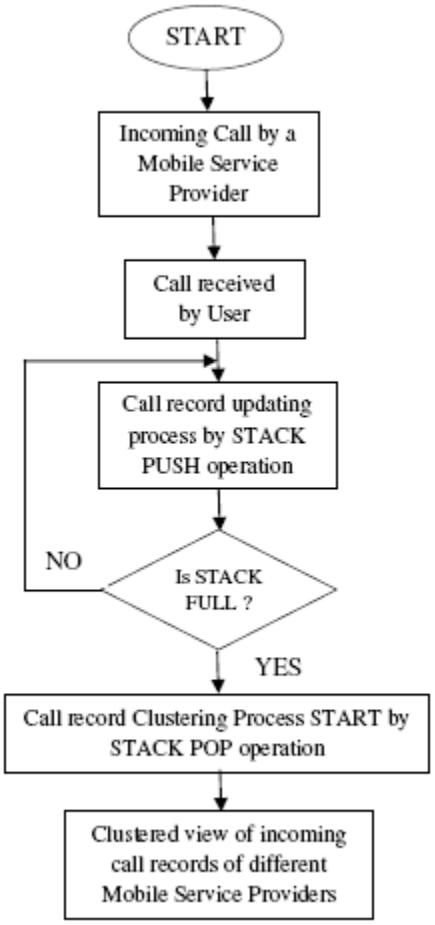

Fig. 2 : Flow Chart for Working of Push Down Automata Model for Mobile Incoming Call Logs Storage in Cluster way

Flow Chart for working of Push Down Automata Model for Mobile Incoming Call Logs Storage in Cluster way is divided into following steps:

Step1: Occurrence of Incoming Call on user mobile through a Mobile Service Provider

Step2: Call received by the User

Step3: Call record updating process by STACK PUSH operation

Step4: Check for STACK condition : (FULL / NOT FULL)

Step5: Two cases may occur:

IF YES: Call record clustering process start by STACK POP operation

IF NO : GO TO STEP 3

Step6 : User View : Clustered view of incoming call records of different Mobile Service Providers

Steps 01 to 06 as mentioned above can be modeled with the help of Push Down Automata. We are proposed a model for storage of Mobile Call logs for different Mobile services in cluster way for a summary to know how many calls are received by various types of mobile service.

To achieve this goal first we design the Push Down Automata (PDA) as

$M_{MOBILE} = \{Q_{MOBILE}, \Sigma_{MOBILE}, \Gamma_{MOBILE}, \delta_{MOBILE}, q_0, Z_0, \Phi\}$

$\mathbf{Q_{MOBILE}}$: is the set of states for maintaining the call logs from different mobile service provider and it is represented by two states as

$\mathbf{q_{push}}$ **:** is the state when the incoming calls are received by the user and stack alphabets assigned to different service provider is PUSHED on the stack and at that time stack is not FULL i.e. (STACK ≠ FULL)

$\mathbf{q_{pop}}$ **:** is the state when the STACK of call log memory is full (STACK = FULL) and user is the process to get a clustered view of call records for incoming calls from different mobile service providers with the help of POP operation of STACK alphabets allocated to different mobile service provider calls

$\mathbf{\Sigma_{MOBILE}}$ **:** is finite set of input alphabet for incoming calls for different mobile service provider i.e.

$\mathbf{\Sigma_{MOBILE} = \{a, b, c, d\}}$ where

a represents : BSNL service provider
b represents : VODAPHONE service provider
c represents : RELIANCE service provider
d represents : IDEA service provider

$\mathbf{\Gamma_{MOBILE}}$ **:** is finite set of STACK alphabets for incoming calls for different mobile service providers i.e.

$\Gamma_{MOBILE} = \{Z_{BSNL}, Z_{VODAPHONE}, Z_{RELIANCE}, Z_{IDEA}, Z_0\}$

For STACK operation (PUSH) when a call received from different mobile service provider is represented as:

$\mathbf{Z_{BSNL}}$ **:** Z_{BSNL} stack alphabet is pushed on the STACK when a call is received from BSNL service provider

$\mathbf{Z_{VODAPHONE}}$ **:** $Z_{VODAPHONE}$ stack alphabet is pushed on the STACK when a call is received from VODAPHONE service provider

$\mathbf{Z_{RELIANCE}}$ **:** $Z_{RELIANCE}$ stack alphabet is pushed on the STACK when a call is received from RELIANCE service provider

$\mathbf{Z_{IDEA}}$ **:** Z_{IDEA} stack alphabet is pushed on the STACK when a call is received from IDEA service provider

q_0 : is the initial state when mobile phone is just switched ON and transit to q_{push} when the first call is received after the mobile is switched ON

Z_0 : is the initial STACK alphabet

$\mathbf{\delta_{MOBILE}}$: is the Transition Function to update the STACK when different types of calls are received from different mobile service providers and is defined as:

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

δ_{MOBILE} : Transition Function is defined as
$Q_{MOBILE} \times (\Sigma_{MOBILE}) \times \Gamma_{MOBILE} \rightarrow (Q_{MOBILE} \times \Gamma_{MOBILE}^{*})$

4.1 Push Down Automata (PDA) Transitions for Incoming Call Updation:

Fig. 3 represents the incoming call updation by PUSH operation according to call received through a specific Mobile Service Provider.

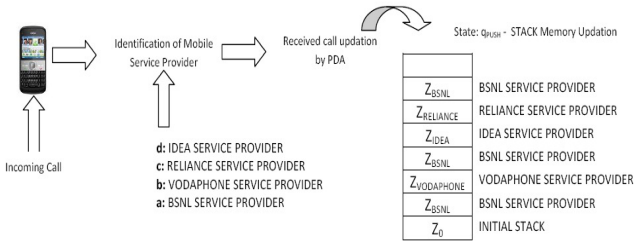

Fig. 3 : Incoming Call Record Updation

According STACK record as mentioned in Figure(3), sequence of incoming calls are :
BSNL(a), VODAPHONE(b), BSNL(a), IDEA(d), DOCOMO(c), BSNL(a)
Then according to above sequence of call received the input string identified on the input tape is

abadca ……. (3)

PDA Transitions for input string as mentioned in (3) is :

1. $\delta(q_0, a, Z_0) = (q_{PUSH}, Z_{BSNL}Z_0)$
2. $\delta(q_{PUSH}, b, Z_{BSNL}) = (q_{PUSH}, Z_{VODAPHONE}Z_{BSNL})$
3. $\delta(q_{PUSH}, a, Z_{VODAPHONE}) = (q_{PUSH}, Z_{BSNL}Z_{VODAPHONE})$
4. $\delta(q_{PUSH}, d, Z_{BSNL}) = (q_{PUSH}, Z_{IDEA}Z_{BSNL})$
5. $\delta(q_{PUSH}, c, Z_{IDEA}) = (q_{PUSH}, Z_{RELIANCE}Z_{IDEA})$
6. $\delta(q_{PUSH}, a, Z_{RELIANCE}) = (q_{PUSH}, Z_{BSNL}Z_{DOCOMO})$

4.2 Instantaneous Description (ID) for Incoming Call updation:

As per the rules Instantaneous Description (ID) for the input string as mentioned in (3) is :
q_0, abadca, $Z_0 \vdash q_{PUSH}$, badca, $Z_{BSNL}Z_0 \vdash q_{PUSH}$, adca, $Z_{VODAPHONE}Z_{BSNL}Z_0 \vdash q_{PUSH}$, dca, $Z_{BSNL}Z_{VODAPHONE}Z_{BSNL}Z_0 \vdash q_{PUSH}$, ca, $Z_{IDEA}Z_{BSNL}Z_{VODAPHONE}Z_{BSNL}Z_0 \vdash q_{PUSH}$, a, $Z_{RELIANCE}Z_{IDEA}Z_{BSNL}Z_{VODAPHONE}Z_{BSNL}Z_0 \vdash q_{PUSH}$, ε , $Z_{BSNL}Z_{RELIANCE}Z_{IDEA}Z_{BSNL}Z_{VODAPHONE}Z_{BSNL}Z_0$

4.2 Push Down Automata (PDA) Transitions to generate Cluster view of Call Logs(Incoming) when STACK = FULL:

Incoming call log updation, as shown in Figure (3), will continue till STACK ≠ FULL. When the STACK = FULL then transition for Cluster view generation is started according to following steps:

Step 1: Suppose that as STACK = FULL then # symbol is generated on the input tape and q_{PUSH} is converted to q_{POP}

1. $(q_{PUSH}, \#, Z_{BSNL}) = (q_{POP}, Z_{BSNL})$
2. $(q_{PUSH}, \#, Z_{VODAPHONE}) = (q_{POP}, Z_{VODAPHONE})$
3. $(q_{PUSH}, \#, Z_{RELIANCE}) = (q_{POP}, Z_{RELIANCE})$
4. $(q_{PUSH}, \#, Z_{IDEA}) = (q_{POP}, Z_{IDEA})$

Step 2: It is assumed that when STACK = FULL that user will not received any incoming call till the STACK is not empty. In this case READ head will continue read # symbol on the input tape and following transitions are occur for the generation of Cluster view of incoming call logs with the help of STACK POP operation as follows:

1. $(q_{POP}, \#, Z_{BSNL}) = (q_{POP}, \varepsilon)$
2. $(q_{POP}, \#, Z_{VODAPHONE}) = (q_{POP}, \varepsilon)$
3. $(q_{POP}, \#, Z_{RELIANCE}) = (q_{POP}, \varepsilon)$
4. $(q_{POP}, \#, Z_{IDEA}) = (q_{POP}, \varepsilon)$

According to incoming call sequence as shown in equation (3), all POP operation are carried out as transition 1 to 4 mentioned above and clustered view will be generated as:

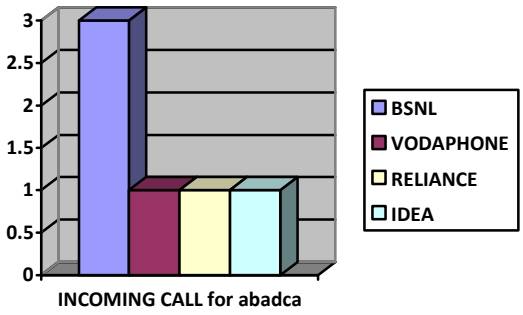

Fig. 4 : Cluster view of Incoming Calls

Fig. 4 shows that the cluster view of input string **abadca** and can be interpreted as:
No. of BSNL calls : 03 (number of a's in input string)
No. of VODAPHONE calls : 01 (number of b's in input string)
No. of RELIANCE calls : 01 (number of c's in input string)
No. of IDEA calls : 01 (number of d's in input string)
Step 3: When Top of the STACK = Z_0 it implies that all the stack alphabets corresponding to different service provider is popped out and Top of the STACK is Z_0 then state q_{POP} is transit to q_{PUSH} and mobile set is ready to receive incoming call. Step 3 condition in PDA model can be completed with the help of transition:

1. $(q_{POP}, \#, Z_0) = (q_{PUSH}, Z_0)$

4.3 Verification Using Object Constraint Language (OCL):

A constraint as given by Kleppe[11] is defined as "restriction on one or more values of an object – oriented model or system". A restriction of the multiplicity of an association can be expressed either directly by using syntactical construct of Object Oriented Notation.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

Basically three types of constraints can be classified in Database Management System:

1. **Implicit Constraint:** An Implicit Constraint, which represents an integrity rule applied on data models.
2. **Explicit Constraint:** An Explicit Constraint, which represents the Business Rules.
3. **Inherent Constraint:** An Inherent Constraint, which are specified in a schema but are assumed to hold by the definition of relational model.

The order to apply constraint on the database we have used OCL invariant on SQL Table consisting of details of the Caller, his/her number and location. The transformation pattern is given as:

Context : Session

Inv: for Incoming Calls
Inv: Self.Stack →Empty( )
Pre: Self.Stack = Q_{Mobile}
Post: Self.Value = Γ_{MOBILE}

Context : Record Call

Session.Record =
(if Q_{PUSH}! = a then 'Z_{BSNL}' or
b then '$Z_{VODAPHONE}$' or
c then '$Z_{RELIANCE}$' else
d then 'Z_{IDEA}')

Context : Cluster View Generation

Session.Cluster View =
(if STACK = FULL then
q_{PUSH} = q_{POP}
end if)
Session.READ =
(if Input string = abadca then
Count := 03 for BSNL and
Count := 01 for VODAPHONE and
Count := 01 for RELIANCE and
Count := 01 for IDEA
end if)

5. Conclusion

This paper focuses on the development of a feature in mobile phone for user to view number of incoming calls in cluster view to know how many calls are received from different mobile service providers. This process can also be implemented for outgoing calls & helps to mobile service providers to analysis comparative calls rates from different mobile service provider.

6. Reference:

[1] Bruce, K "A Pattern Language for Object RDBMS Integration, Knowledge System Group."

[2] Dresden UML Tool Set, http://www.st.inf.tu.dresden.de

[3] Melton, J "SQL's Stored procedures " A complete guide, Morgan Kaufman, 1998

[4] M. Weiser, "The Computer for the Twenty – First century" Scientific Aug 1991, pp 94 -101.

[5] Wanner; J & Kleppe, A: "The Object Constraint Language", Addison Wesley, 1999.

[6] Gligor, V. and Luckenbaugh, G., (1983) "An Assessment of the Real-time Requirements for Programming Environments and Languages", Proceedings of the IEEE Real-time System Symposium, Washington DC., 3-19.

[7] Kligerman, E., and Stoyenko, A., (1986) "Real-time Euclid: A Language for Reliable Real-time Systems", IEEE Trans. on Software Engineering, Vol. SE-12, No. 9, Sept

[8] Pereira, C., (1993) "Putting OO to work: Results from Applying the Object-Oriented Paradigm during the Development of Real-time Applications", Fifth Euromicro Workshop on Real-time Systems Proceedings. Apr. 166-170.

[9] Bihari, T., Gopinath, P., and Schwan, K., (1989)"Object-Oriented Design of Real-time Software", IEEE Software Trans., 194-201.

[10] Alexander P. Pons1 and Moiez A. Tapia2, "Real-time Constraint Specification in Object-Oriented Languages"

[11] Anneke G. Kleppe, Jos Warmer: "The Object Constraint Language: Precise Modeling with UML (Addison-Wesley Object Technology Series)"

Acknowledgments

Vivek Kumar Singh is currently working as Assistant Professor in the Department of I. T. at BBDNITM, Lucknow. He has over 11 years of teaching experience. He has done his B.Tech in Computer Science & Engineering from Purvanchal University in 2001, M.Tech from U.P.Technical University, Lucknow in 2006, he is pursuing his Ph.D. from Shobhit University, Meerut. He has published numbers of papers in referred National journals. His teaching areas are: Theory of Automata & Formal Language, Design & Analysis of Algorithm & Computer Architecture

Dr. S. P. Tripathi is currently working as Professor in Department of Computer Science & Engineering at I.E.T. Lucknow. He has over 30 years of experience. He has published numbers of papers in referred Journals.

Dr. J. B. Singh is currently working as Dean Students Welfare at Shobhit University, Meerut. He has 38 years of teaching experience and has published number of papers in referred Journals.

Dr. R. P. Agarwal is currently working as Professor & Director in School of CE&IT at Shobhit University, Meerut. He has 40 years of teaching experience and has published number of papers in referred Journals.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.