An Active Rule Base Simulator based on Petri Nets

Joselito Medina-Marín and Xiaoou Li

[1] Sección de Computación, Departamento de Ingeniería Eléctrica
[2] CINVESTAV-IPN, Av. IPN No. 2508, C.P. 07360, Mexico City, México

Abstract. Event-condition-action rules, in active database systems, should be performed carefully, because their firings can produce inconsistent states in database systems. In this paper, an ECA rule base simulator is described, named ECAPNSim, which uses a Conditional Colored Petri Net model to depict ECA rules. It can model ECA rules, simulate their behavior, and perform a static analysis.
Keywords: ECA rule, active database system, Petri nets.

1 Introduction

Active behavior of a database system (DBS) can be defined through a base of active rules. The model most widely used is the *event-condition-action (ECA) rule* model, whose general form is:

on *event* e_1 **if** *condition* c_1 **then** *action* a_1

This model works in the following way: when an event e_1 occurs, if condition c_1 is evaluated, then action a_1 is executed inside the database (DB).

Events in ADBs can be of two types: *primitive or composite*. A *primitive event* is generated by the execution of an operation over the DB information (insert, delete, update, or select), a transaction, a clock event (which can be absolute, relative, or periodic), or the occurrence of a DB external event. On the other hand, *composite events* (disjunction, conjunction, sequence, closure, times, negation, last, simultaneous, and any) are formed by the occurrence of a combination of primitive and/or composite events.

In this work, an ECA rule base simulator is presented, which is named ECAPNSim. In order to model an ECA rule base, ECAPNSim uses a Petri Net (PN) approach, which is based in a PN extension named Conditional Colored Petri Net (CCPN). CCPN provides properties of ECA rules, therefore, ECA rule bases can be converted into a CCPN and then, ECA rule base simulation can be performed by using ECAPNSim.

2 Conditional Colored Petri Net

Petri Nets are a graphical and mathematical tool for modeling concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic systems. Petri net may be extended widely and applied in every area with logic relations. Up to now, few researches have adopted Petri nets as ECA rule specification language [1], [2]. SAMOS is

Medina-Mauın J. and Li X. (2005).
An Active Rule Base Simulator based on Petri Nets.
In *Proceedings of the 3rd International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 96-101
DOI: 10.5220/0002566500960101

a successful ADB system, which partially uses Petri nets for composite event detection and termination analysis. But, the framework is not Petri-net-based [3]. Conditional Colored Petri Net (CCPN) is an extended PN model, which was modified in order to support features of ECA rules [4]. In a CCPN it is very easy to detect the existence of both relationships and dependencies between two or more rules according to its graphics representation. Moreover, both primitive and composite events can be modeled.

2.1 General description

In a CCPN, ECA rule event e is stored as a place p_1, conditional part c is stored inside a transition t, and the action rule a, because of its similarity to an event, is stored in a place p_2. Therefore, if t is the transition where is stored the condition of rule r, then ${}^\bullet t = \{p_1\}$, and $t^\bullet = \{p_2\}$, where ${}^\bullet t$ is the set of the input places of t, and $t^\bullet$ is the set of the output places of t.

During CCPN execution, events occurring inside DB are detected by the CCPN, and if there is a CCPN place p_1 representing to detected event e then a token is generated with information about e (e.g. the record of an employee) and with a timestamp according to the time when the event was raised. By CCPN execution, the new token is sent to transition t_1, ${}^\bullet t_1 = \{p_1\}$, and the condition c stored in t_1 is evaluated against token information. If token information is not enough to evaluate c then a query to DB is executed to know the DB state and perform the evaluation to c. If c is evaluated to true then one token with information about the rule action a is generated and it is sent to place p_2, $t_1^\bullet = \{p_2\}$, which represents the ECA rule action a.

Composite events, that deal with time interval, evaluate token timestamp, and if it belongs to composite event interval, the token is sent to its corresponding transition.

Formally, a CCPN is defined as follows [5]:

Definition 1. *A conditional colored Petri net (CCPN) is a 11-tuple*
$CCPN = \{\Sigma, P, T, A, N, C, Con, Action, D, \tau, I\}$

where

(i) Σ is a finite set of non-empty types, called color sets.

(ii) P is a finite set of places. P is divided into subsets, i.e., $P = P_{prim} \cup P_{comp} \cup P_{virtual} \cup P_{copy}$, where P_{prim} represents primitive events and it is depicted graphically as a single circle. P_{comp} represents composite events *negation, sequence, closure, last, history, and simultaneous* and it is depicted graphically as a double circle. $P_{virtual}$ represents composite events *conjunction, disjunction, and any* and it is depicted graphically as a single dashed circle. And, P_{copy}, is a set which is used when two or more rules are triggered by the same event and it is depicted graphically as a double circle where the interior circle is a dashed one.

(iii) T is a finite set of transitions. $T = T_{rule} \cup T_{copy} \cup T_{comp}$., where T_{rule} represents the set of rule type transitions and it is depicted graphically as a rectangle. T_{copy} is the set of copy type transitions and it is depicted graphically as a single bar. T_{comp} is the set of composite type transitions and it is depicted graphically as a double bar.

(iv) A is a finite set of arcs.

(v) N is a node function. It is defined from A to $P \times T \cup T \times P$.

(vi) C is a color function. It is defined from P to Σ.

(vii) Con is a condition function. It evaluates either the rule condition if $t \in T_{rule}$ or it evaluates the time interval when $t \in T_{comp}$.

(viii) $Action$ is an action function. It creates tokens according to action rules.

(ix) D is a time interval function.

(x) τ is a timestamp function.

(xi) I is an initialization function.

2.2 CCPN execution

CCPN execution is modified because CCPN token takes time information and CCPN transitions are evaluated only in certain intervals, furthermore transitions $t \in T_{rule}$ are evaluated with the conditional part of ECA rule. [5] In CCPN, composite transitions and rule transitions fire conditionally. A composite (rule) transition fires once it is enabled and the temporal (rule) condition is satisfied.

Definition 2. *In CCPN, a token element is a triple (p, c, stamp) where $p \in P$, $c \in C(p)$, and stamp specifies the natural time when the token is deposited into place p. The sets of all markings is denoted by M.*

Definition 3. *Transition $t \in T$ is enabled at a marking M iff*

1). $\forall p \in {}^\bullet t :\mid M(p) \mid = 0, type(t) = Negation$

2). $\forall p \in {}^\bullet t :\mid M(p) \mid \geq 1$, else

3 ECAPNSim

Active rules development is an activity that needs to be performed carefully. Nowadays, there are few systems [2] which perform the analysis and debug the ECA rule base. Most of commercial ADBs [7], [8] provide a syntax to ECA rule definition, however static analysis of ECA rules cannot be performed inside these systems and the ECA rule definition is only in a text way.

ECAPNSim (ECA Petri Nets Simulator) was developed under MAC OS X Server in Java. Taking advantage of Java portability, ECAPNSim can be executed in different operating systems. As an engine of ECA rules, ECAPNSim can be connected with any relational database systems such as Postgres, MS Access, Oracle, and Visual Fox Pro.

3.1 ECAPNSim architecture

ECAPNSim architecture consists of two building blocks: *ECAPNSim Kernel* and *ECAPNSim tools environment. ECAPNSim Kernel* provides active functionality to passive database. it consists of CCPN Rule Manager, CCPN rule base, Composite Event Detector, and Rule Execution Component. *ECAPNSim tools environment* has a set of tools used by the ECA rule developers. Tools environment is composed by ECA rule editor, analyzer of no-termination problem, converter of ECA rules to CCPN, CCPN visualizer/editor and explanation components, termination analyzer and runtime tools.

ECAPNSim offers two modalities. In **Simulation mode**, an user can simulate the behavior of the ECA rule base modeled by depositing tokens into the CCPN manually. And, in **Real mode**, the CCPN is executed by state modification of the connecting DBS.

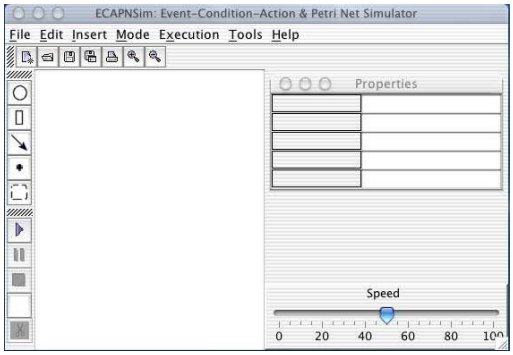

Fig. 1. ECAPNSim graphical interface.

3.2 ECAPNSim Design

ECAPNSim offers a graphical and visual interface to represent ECA rule bases by CCPN model. Like any PN editor, ECAPNSim simulates the behavior of ECA rules by executing the CCPN model. Meanwhile simulation is running, problems like no-termination and confluence can be observed obviously in the CCPN, hence ECA rule developer can modify the rule base to improve it.

The core of ECAPNSim is CCPN models. ECAPNSim contains a module to generate a CCPN structure from an ECA rule base definition written in a text file automatically. Or a CCPN model can be edited directly from a ECAPNSim user.

ECAPNSim supports CCPN design and edition from an ECA rule base, which can be moved to another position in the visualization panel. Moreover, because of there are large ECA rule bases, ECAPNSim will generate large CCPN structures, so it has zoom buttons to either increase or decrease the CCPN size. Simulation speed can be controlled through a slide. Finally, the graphical interface has tools and icons to edit a CCPN, simulate a CCPN behavior, and CCPN file management. (figure 1).

4 Example

In order to illustrate the use of ECAPNSim, a small ECA rule base is presented. The rules are concerning to a small DB with three tables EMP, BONUS, and SALES. Table EMP whose attributes are the employee's id, the employee's name, the employee's rank, and the employee's salary. Table BONUS whose attributes are the employee's id for the corresponding employee, and the bonus amount. Table SALES whose attributes are the employee's id, the month when the sales were made, and the number of products that the employee sold. Table definition is as follows: EMP(emp_id, name, rank, salary), BONUS(emp_id, amount), SALES(emp_id, month, number);

There are three events that will be monitored by the ECA rule base, a) when the attribute `BONUS.amount` is modified e0 : update_BONUS_amount); b) when the attribute `EMP.rank` is modified (e1 : update_EMP_rank), too; and c) when the sales of a month are registered into the DB and a new record is added in table SALES (e2 :

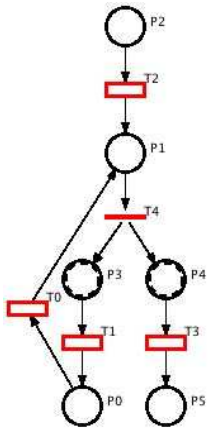

Fig. 2. CCPN generated by ECAPNSim.

insert_SALES). Four ECA rules are used to illustrate this example, which are related to DB structure described above. ECA rules description is presented in the next rows:

Rule 1: When an employee's bonus is increased by more than 100, then the employee's rank is increased by 1. Rule 2: When an employee's rank is updated, then increased by 1, then the employee's bonus is increased by 10 times the new rank. Rule 3: When an employee posts sales greater than 50 and its rank lower than 15, his bonus is decreased by 100. Rule 4: When an employee's rank reaches 15, increases the employee's salary by 10%.

In figure 2 can be observed that event e_0 is depicted by place p_0, event e_1 is depicted by place p_1, and event e_2 is depicted by place p_2. Places p_3, $p_4 \in T_{copy}$ because event e_1, depicted by p_1, is used in the firing of two rules (rules 2 and 4). Place p_5 is only output place, i.e. p_5 represents the action part of rule 4. Transitions $t_0, t_1, t_2, t_3 \in T_{rule}$ store rules 1,2,3, and 4, respectively; and transition $t_4 \in T_{copy}$ is used to replicate information of event e_1 in order to trigger transitions t_1 and t_3.

In order to start the CCPN simulation in ECAPNSim, a console application is started too. Console application establishes a communication process with ECAPNSim to send the SQL instructions typed in it from an user. After that, ECAPNSim takes the messages sent by console and convert them in tokens, that are placed in the corresponding CCPN places. In this case, the simulation is started by typing in the console the instruction: update bonus set amount = 150 where emp_id = 1. When ECAPNSim receives the instruction, a token is generated and it is placed in p_1. Because of p_1 is the input place of t_1, then t_1 is activated and fired, t_1 takes token information to evaluate the condition stored in it, and if it is true according to the random evaluation, then t_1 sends a token to its output place p_2. If all transitions $t \in T_{rule}$ are evaluated to true, then the possible DB states reachable from an event occurrence can be predicted.

In real mode, the evaluation of condition stored in transitions $t \in T_{rule}$ is verified against the DB state. If the instruction typed in console is executed in real mode, then token generator will put a new token in place p_0 with information about the update

command. Condition part in t_0 is evaluated to true, because amount is equal to 150, and the condition is `if amount > 100`. Then, token generator creates a new token according to action part of rule 1 stored in t_0, where the employee's rank is increased by one, i.e. the value for employee's rank now is 4. New token is sent to the output place p_1, which sends the token to t_4. t_4 replicates the token and it sends one token to p_3 and the other one to p_4. Transitions t_1 and t_3 are activated, but they are not triggered because both conditional parts of these transitions are evaluated to false against the token information.

5 Conclusion and Future Work

In this research work, an ECA rule base simulator ECAPNSim is developed. ECAPNSim uses a Conditional Colored Petri Net (CCPN) as a model to depict ECA rules. Unlike other ECA rule simulators or DB management systems, ECAPNSim is independent on the actual database system, i.e., a developed ECA rule base can work with any database system the communication bridge ODBC-JDBC between ECAPNSim and its DBMS. Through the CCPN model, ECAPNSim can simulate an ECA rule base behavior and analyze its static properties. Furthermore, the portability of the programming language Java makes it possible to connect ECAPNSim with many DBMS in different operating systems, such as Postgresql in both Linux and Macintosh platforms, and MS Access, MS Visual Fox Pro, and Oracle, in MS Windows platform, etc.

For future, ECAPNSim will be enhanced with distribution functions such as defining interarrival times among primitive events.

References

1. Zimmer, D.: "Rule Termination Analysis based on a Rule Meta Model". In: Cadlab Report 2, Cadlab, Bahnhofstr. 32, 33102 Paderborn, Germany, April 1995
2. Schlesinger, M. and Lörincze, G. : "Rule modeling and simulation in ALFRED". 3rd International workshop on Rules in Database (RIDS'97), Skövde, Sweden, June, pp. 83-99, 1997
3. Gatziu S., Dittrich K.R., "Detecting Composite Events in Active Database Systems Using Petri Nets", Proc. 4th International Workshop on Research Issues in Data Engineering: Active Database Systems, Houston, Texas, February 1994
4. Li X., Medina Marín J., and Chapa S. V., "A Structural Model of ECA Rules in Active Database", *Mexican International Conference on Artificial Intelligence 02*, Mérida, Yucatan, México, April 22-26, 2002
5. Li X., Medina Marín J., "Composite Event Specification in Active Database Systems: A Petri Net Approach", IEEE International Conference on System, Man, and Cybernetics, The Hague, The Netherlands, Oct, 2004.
6. Jensen K., "An Introduction to the Theoretical Aspects of Colored Petri Nets". *Lecture Notes in Computer Science: A Decade of Concurrency*, vol. 803, edited by J. W. de Bakker, W.-P. de Roever, G. Rozenberg , Springer-Verlag, 1994, pp. 230-272
7. Hanson E.N., "*The Design and Implementación of the Ariel Active Database Rule System*", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 1, february 1996.
8. Widom J., "*The Starburst Active Database Rule System*", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 4, August 1996.