Measuring Triggering-Interactions Complexity on Active Databases based on Conditional Colored Petri Nets

Lorena Chavarria, Joselito Medina Marín, Xiaoou Li
Sección de Computación
Departamento de Ingeniería Eléctrica
CINVESTAV-IPN
Av. IPN 2508, México D.F., 07360, México
email: lchavarria@computacion.cs.cinvestav.mx, jmedina@computacion.cs.cinvestav.mx, lixo@cs.cinvestav.mx

Abstract—Triggering interactions between active rules of an active database system can be modeled by conditional colored Petri nets (CCPN). This paper proposes three different metrics for measuring trigger complexity based on CCPN. These metrics are triggering potential, the number of anchors and the distance. These measures are characterized above the level of the ordinal scale using the measurement theory. Validation of the proposed metrics has been conducted through comparisons with a fundamental work of the area.

I. INTRODUCTION

Software engineers have proposed large quantities of metrics for software products, processes and resources [4], [3]. However, most of the metrics focus on program characteristics, disregarding databases [5]. Although there exists some research on database, they are not convenient for active databases which allow the definition and management of triggers. Up to date, only one result can be found related to measurement of triggering-interaction complexity on active databases, that was proposed by Diaz et al [2]. However, that work is just a primitive one. They considered a simplified case of active database where events and actions are primitive events. The majority of triggers contains composite (or complex) events. Thus, the measures in [2] cannot give an analysis of complexity when composite events are included.

We have proposed conditional colored Petri net (CCPN) model for modeling and simulation of trigger interactions [1]. A CCPN model can be generated easily with our interface ECAPNSim by using a text description of triggers. After analyzing the metrics defined by Diaz et al in [2], we found that all these metrics can be defined on a CCPN model and more complicated triggers can be included too. In this paper, we will show how to measure triggering interaction complexity based on CCPN.

The rest of the paper is organized of the following way: Section II provide a brief introduction to triggers in active database and CCPN model. Metrics of complexity based on CCPN are defined and illustrated in Section III. Finally, Section IV gives a comparison between our metrics and those described in [2], and future work.

II. TRIGGERS AND CCPN

A. Triggers in Active Database System

Active databases systems (ADB) supports mechanisms that enable them to respond automatically to events that take place either inside or outside the database system itself. Generally, ADB system has a knowledge model (a description mechanism) that specifies triggers, and an execution model (a strategy in runtime) that support the active behavior [6].

A trigger consists of three components: event, condition, and action (so a trigger is also called an ECA rule), it takes the form ON event IF condition THEN action. When an event is detected, the trigger will be fired and its action will be executed if the condition part of the rule is satisfied. The event part defines the changes of the database content. An event may be database transactions such as insert or update; clock, where the event happens at a moment of time that can be absolute, relative or periodic; external, in which, the event happens by a situation outside the database (e.g., the temperature is over 30 degrees). Furthermore, an event may be of two types: primitive, when the event happens by an occurrence of anyone of the described sources; or composite, when the event happens by a combination of primitive or composite events. The condition is a boolean predicate computed over the database state. The action is an instruction that may cause changes to the database state.
The execution model indicates how a set of rules is treated at runtime. During the action execution other events can in turn be signaled that may produce a cascade rule firing. This triggering interaction complicates ADBS maintenance. Just as an early ADB project report shows "a few hundred rules will represent a real maintenance problem" [2] which represents an obstacle for active database systems to become widely used. Maintainability is contributed by three factors: understandability, modifiability and testability, which in turn are influenced by complexity [2].

B. Conditional Colored Petri Nets

Petri Nets (PN) are a graphical and mathematical tool for modeling concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic systems. PN may be extended widely and applied in every area with logic relations. The mathematical representation of the modeled system can be used to reveal important information about the system structure and dynamic behavior [7].

Colored Petri Nets (CPN) is a high-level PN which integrates the strength of PNs with the strength of programming languages. PN provides the primitives for the description of the synchronizion of concurrent processes, while programming languages provide the primitives for the definition of data types and the manipulation of their data values [10]. Thus, CPN is more suitable for ADB than an ordinary PN since it can manipulate data values. By using CPN one cannot only reveal the interrelation between ECA rules but also capture the operational semantics. Due to this idea, we developed a conditional colored Petri net (CCPN) model for specification of ECA rules [1]. For the limit of pages, here we just give a brief introduction of CCPN.

1) CCPN Structure: The logic relation of the three parts of an ECA rule can be modeled as a PN structure. The condition of the rule is modeled as a conditional transition, events and actions are modeled as input and output places of the transition, respectively. Furthermore, event and action operations can be modeled as colors of a CPN.

CCPN is a modified CPN in which CPN elements are defined with more ADB details. It was developed for ADB specially.

Definition 1: A conditional colored Petri net (CCPN) is a 12-tuple

\[
CCPN = \{\Sigma, P, T, A, W, N, C, Con, Action, D, \tau, I\}
\]

where

(i) \(\Sigma\) is a finite set of non-empty types, called color sets. Elements in a color set may be any information that an event or an action takes.

(ii) \(P\) is a finite set of places. A place may be one of the following four types: 1) primitive places represent primitive events; 2) composite places represent composite events; 3) copy places are used when one event triggers two or more rules. (An event can be shared by two or more rules, but in a PN, one token need to be duplicated for share.) A copy place takes the same information as its original one; 4) virtual places are used for accumulating different events that trigger the same rule. For example, when the event part of a rule is the composite event OR.

(iii) \(T\) is a finite set of transitions. A transition may be one of the following three types: 1) rule transitions represent a rule; 2) composite transitions represent composite event generation; 3) copy transitions duplicate one event for each triggered rule.

(iv) \(A\) is a finite set of arcs between places and transitions, such that \(P \cap T = P \cap A = T \cap A = \emptyset\).

(v) \(W\) is a weight function which assigns to each arc a weight. If the weight is not specified, it is 1.

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

(vii) \(C\) is a color function. It is defined from \(P\) to \(\Sigma\).

(viii) \(Con\) is a condition function. It is defined from either \(T_{rule}\) or \(T_{comp}\) into expressions such that

\[
\forall t \in T_{rule}, P \in t : [Type(Con(t)) = E],
\]

where \(Con\) function evaluates the rule condition.

\[
\forall t \in T_{comp}, P \in t : [Type(Con(t)) = E],
\]

where \(Con\) function evaluates the time interval of \(t\) against tokens timestamp.

(ix) \(Action\) is an action function. It is defined from \(T_{rule}\) into expressions such that:

\[
\forall t \in T_{rule}, p \in t : [Type(Action(t)) = C(p)_{mst}]
\]

(x) \(D\) is a time interval function. It is defined from \(T_{comp}\) to a time interval \([d1(t), d2(t)]\), where \(t \in T_{comp}\), and \(d1(t), d2(t)\) are the initial and final interval time, respectively.

(xi) \(\tau\) is a time stamp function. It is defined from \(M(p)\) to \(\mathbb{R}^+\), which assigns each token in place \(p\) a time stamp corresponding to natural o'clock with the form year : month : day – hour : minute : second. For example, 2003 : 11 : 10 – 11 : 16 : 46.
(xii) $I$ is an initialization function. It is defined from $P$ into closed expressions such that

$$\forall p \in P : [\text{type}(I(p)) = (C(p)_MS, r(C(p)_MS)]$$

Since we just need CCPN static structure to compute trigger interaction complexity, we will not mention the CCPN execution part.

2) Modeling ECA rules with CCPN: The following example will be used through the paper.

Example 2: The rule base is based on the following relational database:

**Tables**

Emp(Employee, Name, Salary, Boss, Depa, Bond)

Dep(Depa, Name, Budget, Prod)

Project(Proj, Name, Budget, Boss)

Work(Proj, Employee)

There are 5 rules in the rule base: Rule0, Rule1, Rule2, Rule3, Rule4.

**Rule0**

ON delete to Dep

IF true

THEN delete to Emp

where Emp.Depa = Dep.Depa

**Rule1**

ON delete to Work

IF true

THEN delete to Project

where Project.Proj = Work.Proj

**Rule2**

ON delete to Emp

IF true

THEN delete to Work

where Emp.Employee = Work.Employee

**Rule3**

ON or((insert Emp), (update Emp.Salary))

IF (Emp.Salary > 15,000)

THEN delete to Emp

where Emp.Employee = new.Emp.Employee

**Rule4**

ON and((update Dep.prod), (insert Emp))

IF (Dep. Prod > 90 & Emp.Depa = Dep.Depa)

THEN update Emp.Bono = 100

where Emp.Employee = new.Emp.Employee

The rule base of Example 1 can be mapped into a CCPN as is shown in Figure 1. The modeling process is omitted.

In Figure 1, the 5 rules typed transitions $t_0, t_1, t_2, t_3, t_4$ represent the 5 rules Rule0, Rule1, Rule2, Rule3, and Rule4. $t_5$ is a composite typed transition which represents the generation of a composite event and((update Dep.prod), (insert Emp)) of Rule4. $t_6$ duplicates $e_4$ for the need of a composite event and((update Dep.prod), (insert Emp)) of Rule4 and Rule3. $t_6$ and $t_7$ are copy typed transitions which deposit tokens to a virtual place $e_{10}$ which represents an event or((insert Emp), (update Emp.Salary) of Rule3.

A corresponding interface ECAPNSim was developed based on CCPN in Java. ECAPNSim can be connected with a traditional database system by its JDBC driver. It automatically generates a CCPN from a set of rules ECA and simulates the behavior of ADBs. See reference [8] for details.

<table>
<thead>
<tr>
<th>place</th>
<th>color</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_0$</td>
<td>delete to Dep</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_1$</td>
<td>delete to Emp</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_2$</td>
<td>delete to Trabaja</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_3$</td>
<td>delete to Proyecto</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_4$</td>
<td>insert Emp</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_5$</td>
<td>update Emp.Salario</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_6$</td>
<td>update Dep.Prod</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_7$</td>
<td>update Emp.Bono</td>
<td>primitive</td>
</tr>
<tr>
<td>$e_8$</td>
<td>insert Emp</td>
<td>copy</td>
</tr>
<tr>
<td>$e_9$</td>
<td>insert Emp</td>
<td>copy</td>
</tr>
<tr>
<td>$e_{10}$</td>
<td>or((insert Emp), (update Emp.Salario))</td>
<td>virtual</td>
</tr>
<tr>
<td>$e_{11}$</td>
<td>and((update Dep.prod), (insert Emp))</td>
<td>virtual</td>
</tr>
</tbody>
</table>

### III. Complexity Metrics Based on CCPN

The complexity of a rule can be divided into inter- and intra-rule complexity. The first one is due to rule itself,
and the second depends on the implicit interaction between the rules. Furthermore, the interaction degree determines the complexity more than the rules separately. For this reason, we focus on the investigation of triggering interaction complexity in this paper.

The same as analyzed in [2], we considered the same two important aspects in the measurement of the triggering interaction complexity:

1) The width of the argumentation, which reflects the intuition that the larger is the flow of event occurrences that conforms the rules circumstance, the more difficult will be to understand these rules.

2) The depth of the argumentation, that is, the intricacy of the line of reasoning that connects the rule with the context where its circumstance has been produced.

When measuring rules circumstance, Diaz used a simple triggering graph [2]. From a triggering graph, they found three complexity parameters that are the minimum number of anchors, the distance and the triggering potential. However, a triggering graph is very general, it does not take complete intra-information. For example, composite events cannot be found in a triggering graph. Thus, the complexity caused by generating a composite event cannot be included in the work of Diaz. Now we take into account composite events, we defined these three parameters based on CCPN. Proposing metrics could be quite straightforward, however, the challenge is for the measures to adhere to the science of measurement to be valid and accepted [4]. Thus, before the definitions we review Zuse’s measurement framework.

A. Characteristics of software metrics

Zuse’s framework is based on an extension of the classical measurement theory, which provides a sound basis for measure validation and classification (i.e. measurement scales). This framework makes a mapping between the Empirical Relational System and Numerical Relational System. A relational system consists of a set of objects and object relations.

A empirical relational system is defined as:

$$ A = (A, \bullet \geq, o) $$

where $A$ is a nonempty set of objects, $\bullet \geq$ is an empirical relation on $A$, and $o$ is a closed binary (concatenation) operation on $A$. the concatenation operation $o$ should follow:

$$ A1oA2 \in A, \forall A1, A2 \in A $$

$$ f(A1, A2) : AxA \rightarrow A \text{ where function } f \text{ is a combination rule} $$

$$ u(A1oA2) = f(u(A1), u(A2)) $$

A numerical relational system can be defined as a triplet

$$ B = (\mathbb{R}, \geq, +) $$

where $\mathbb{R}$ are the real numbers, $\geq$ a relation on $\mathbb{R}$, and + a closed binary operation on $\mathbb{R}$. A measure is a mapping:

$$ u : A \rightarrow \mathbb{R} \text{ such that:} $$

$$ a \geq b \Leftrightarrow u(a) \geq u(b); \forall a, b \in A $$

Then, the triple $(A, B, u)$ is called a scale.

Once the mapping is established, both mathematics and statistics can be used to process the information.

Measurement theory gives also conditions for the translation of numerical statements back to empirical statements. In order to check whether the measure satisfies the user needs, Zuse proposes an internal validation, based on the comparison between the empirical interpretation of numbers and the empirical statements in the real world. On this framework, Zuse defines a set of axioms for measures which gives rise to distinct structures. See [4] for details for Zuse’s formal framework properties.

B. Characterization of ADB Complexity Metrics

We agree Diaz’s definition on the Empirical Relational System. For ECA rules, the Empirical Relational System, $A = (R, \bullet \geq, o)$ was defined as follows:

1) $R$ is a nonempty set of rules;

2) $\bullet \geq$ is the empirical relation "more or equally complex than" on $R$;

3) $o$ is a closed binary (concatenation) operation on $R$ such that concatenation of rules $R_1(E_1, C_1, A_1)$ and $R_2(E_2, C_2, A_2)$ produces a rule $R_3$, where:

(a) event $E_3$ is obtained as $E_1 \text{ OR } E_2$;

(b) condition $C_3$ is obtained as $(C_1 \text{ AND } e_1) \text{ OR } (C_2 \text{ AND } e_2)$ where $e_i$ is a Boolean variable where true (false) value indicates the occurrence (absence) of the $E_i$ event;

(c) action $A_3$ can be defined as: IF $e_1$ then $A_1$, If $e_2$ THEN $A_2$

To define the measures, we need the following concepts:

Definition 3: Initial place (anchor). An initial place, in a CCPN, is a place that only has output arcs and no input arcs. For example, places $e_0$, $e_4$, $e_5$ and $e_6$ are initial places.

Definition 4: Route (RU $(p_i, t_j)$). Given an initial place $p_i$ and a rule typed transition $t_j$ in a CCPN, a route $RU$ from $p_i$ to $t_j$. $RU(p_i, t_j)$ is a finite sequence of pairs of arcs that connects places and transitions and vice versa, as the following way:
\[ RU(p_i,t_j) = (p_i,t_k) \rightarrow (t_k,p_n) \rightarrow \ldots \rightarrow (p_s,t_j) \]

where:
- \( p_i \) is an initial place;
- \( t_j \) is a rule typed transition;
- \( p_r \) is a place, \( r = n, \ldots, s; \)
- \( t_d \) is a transition, \( d = k, \ldots, m; \)
- \( (p_r,t_d) \) is an input arc;
- \( (t_d,p_r) \) is an output arc.

For example, in Figure 1, a route from \( e_0 \) to \( t_1 \) is
\[ RU(e_0,t_1) = (e_0,e_0) \rightarrow (e_0,e_1) \rightarrow (e_1,t_2) \rightarrow (t_2,e_2) \rightarrow (e_2,t_1). \]
Some routes of the CCPN of example 1 are listed on Table II.

Table II: Example of route of each rule typed transition of example 1.

<table>
<thead>
<tr>
<th>transition route</th>
<th>example 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 ) ( (e_0,e_0) )</td>
<td>( e_0 \rightarrow t_1 )</td>
</tr>
<tr>
<td>( t_1 ) ( (e_0,e_0) \rightarrow (t_0,e_1) \rightarrow (e_1,t_2) \rightarrow (t_2,e_2) \rightarrow (e_2,t_1) )</td>
<td>( (e_0,t_0) \rightarrow (t_0,e_1) \rightarrow (e_1,t_2) \rightarrow (t_2,e_2) \rightarrow (e_2,t_1) )</td>
</tr>
</tbody>
</table>

Definition 5: Transition Route \((RU_tr(p_i,t_j))\). Given a route \( RU(p_i,t_j) \) where \( p_i \) is an initial place and \( t_j \) is a rule typed transition, we can extract a transition route \( RU_tr(p_i,t_j) \) by taking off all the non-transition elements from the route. Nevertheless, since Copy typed transitions do not raise complexity, they are taken off from the transition route too. For a rule typed transition \( t_j \), it may have more than one transition route if there are more than one route from an initial place to it.

For example, a transition route for transition \( t_1 \) in Figure 1 is: \( t_0 \rightarrow t_2 \rightarrow t_1 \).

Definition 6: Distance of a transition route \((D_tr(p_i,t_j))\). The distance of a transition route \( D_tr(p_i,t_j) \) is defined as the sum of the input arcs weights of all transitions on the route, i.e., \( \forall t \in RU_tr(p_i,t_j), D_tr(p_i,t_j) = \sum_{t \in RU_tr(p_i,t_j)} w(p_k,t) \).

For example, a transition route of \( t_1 \), \( t_0 \rightarrow t_2 \rightarrow t_1 \), \( D_tr(e_0,t_1) = w(e_0,t_0) + w(e_1,t_2) + w(e_2,t_1) = 3 \).

Suppose that the corresponding transition of rule \( R_j \) is \( t_j \).

Definition 7: The minimum number of anchors, \( NA \), required to encompass the whole set of potential causes of rule \( R_j \). In CCPN, an anchor is an initial place on a route from \( p_i \) to \( t_j \).

Definition 8: The distance, \( D \). This measure corresponds to the longest distance of all transition routes to \( t_j \), i.e., \( D = \max_{p \in \text{route}} D_tr(p_i,t_j) \).

Definition 9: The triggering potential, \( TP \). Given a CCPN, the triggering potential for rule \( R_j \) is the quotient between the number of potential causes of \( t_j \) and \( t_j \)'s event cardinality. This measure attempts to reflect the number of \( t_j \)'s inputs by giving an indication of the potential different inputs which can make \( t_j \) to be enabled.

According to the definition of \( TP \), we need to analyze number of causes and event cardinality firstly. Let \( p \in t_j \), potential causes of \( t_j \) can be computed by the following algorithm:

Case I: \( p \) is primitive
- If \( p = \emptyset \), \( cause = cause + 1 \)
- Otherwise, \( cause = cause + \sum_{t \in p} w(t,p) + 1 \)

Case II: \( p \) is composite
- \( cause = cause + 1 \)

Case III: \( p \) is virtual,
- \( cause = cause + \sum_{t \in p} w(t,p) \)

The event cardinality of a rule is 1 if the event of the rule is primitive. If the event of a rule consist of a composite event, then the cardinality is the sum of the number of primitive events that compose the composite event. Table III shows our measures of Example 1.

Table III: The measurement of each rule of Example 1.

<table>
<thead>
<tr>
<th>transition route</th>
<th>rule name</th>
<th>NA</th>
<th>D</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 ) Rule0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_1 ) Rule1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( t_2 ) Rule2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( t_3 ) Rule3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( t_4 ) Rule4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

These measurements were made using the given definitions directly. However, the computation will be difficult when the rule base is large. So a formal method is necessary. Remember that PNs has a mathematical representation, that is its incidence matrix. Using the incidence matrix all above parameters can be obtained easily. See [9] for details.

1) Verification of the proposed measures: 1. Measure \( NA \) is a mapping, \( NA : R \rightarrow \mathbb{R} \) so that:
\[ R_i \preceq R_j \Leftrightarrow NA(R_i) \geq NA(R_j) \quad \forall R_i, R_j \in \mathbb{R} \]

The operation of concatenation \( o \) is defined as:
\[ NA(R_i o R_j) = NA(R_i) + NA(R_j) - NA(R_i \cap R_j) \]
where \( NA(R_i \cap R_j) \) is the number of anchors that \( R_i \) and \( R_j \) share.

2. Measure \( D \) is a mapping, \( D : R \rightarrow \mathbb{R} \) so that:

\[
R_i \cdot R_j \iff D(R_i) > D(R_j), \quad \forall R_i, R_j \in R
\]

The operation of concatenation \( o \) is defined as:

\[
D(R_i o R_j) = MAX(D(R_i), D(R_j))
\]

3. Measure \( TP \) is a mapping, \( TP : R \rightarrow \mathbb{R} \) so that:

\[
R_i \cdot R_j \iff TP(R_i) > TP(R_j), \quad \forall R_i, R_j \in R
\]

The operation of concatenation \( o \) is defined as:

\[
TP(R_i o R_j) = \frac{causeE(R_i) + causeE(R_j)}{card(R_i) + card(R_j) - card(R_i \cap R_j)}
\]

where \( \text{card()} \) is the event cardinality.

Table IV shows the result of the verification of the proposed measures on Zuse’s framework. The conclusion of this table is that the proposed measures can be measured on an ordinal scale since the parameters do not fulfill all the axioms but fulfill the axioms of weak order, transitivity and order.

Table IV: The verification result of the measures on Zuse’s framework.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>NA</th>
<th>D</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom 1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Axiom 2</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Axiom 3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Axiom 4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Axiom 5</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Axiom 6</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IC 1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IC 2</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IC 3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IC 4</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MRB 1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MRB 2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MRB 3</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MRB 4</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MRB 5</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

IV. Conclusion

The only existing relational work is [2]. Here we give a brief comparison with it.

Firstly, let us see the triggering graph of Example 1 as shown in Figure 2. It is not difficult to observe that this graph cannot represent composite events.

Comparing with [2], our measurement has the following advantages:

1) CCPN integrates a graphical and mathematical representation in one model, this assures that our measurement is consistent.

2) A CCPN model contains more rule information than a triggering graph.

3) Our measurement is more precise since composite events complexity are considered.

4) Computation of the metrics based on incidence matrix is much easier than those based on a triggering graph.

![Figure 2. The triggering graph of example 1](attachment:image.png)

CCPN was developed and implemented into a graphical interface, named ECAPNSim, in our previous work. Complexity analysis can be integrated into ECAPNSim too, this is under doing. In the future, we will make an empirical validation with some real database system.

REFERENCES


