CoSent: a Cooperative Sentinel for Intelligent Information Systems

Wesley W. Chu and *Wenlei Mao*
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{wwc, wenlei}@cs.ucla.edu

March 3, 2000

Abstract

Trigger mechanisms in commercial database systems (*e.g.*, Oracle, Sybase, Teradata) process active event-condition-action (ECA) rules on low-level events, such as insert, update, and delete. These triggers are based on exact trigger conditions and actions. However, real world trigger conditions and actions are often inexact and uncertain. They are often represented by concepts. Further, the trigger semantics are user and context sensitive. To handle such inexact and uncertain conditions, we introduce a new knowledge-based active database technology called Cooperative Sentinel (CoSent) that supports active rules with conceptual terms (e.g. heavy, large) and approximate operators (e.g. similar to, near by). As a result, rule specification is more intuitive and rule maintenance is easier.

1 Introduction

Active rules (also called trigger rules, ECA rules *etc.*) are used in active database management systems (ADBMSs) [WC96, Pat99, PD99a] to specify automatic reactions to database events. When a trigger event occurs, if trigger conditions hold, appropriate actions are taken. Traditional active database systems process active rules with exact trigger conditions and actions. However, real world trigger conditions and actions are often inexact, uncertain, and represented by concepts. Further, these conditions and actions are user and context sensitive. For example, a pilot would like to be notified if a *bad weather* forecast is reported in the region of his interest. Here "bad weather" is a concept whose semantics depends on the user type ("pilot" in this case) and the application context (*e.g.*, "mission type"). The concepts can be derived from the available data. Therefore, to handle such inexact and uncertain conditions, we apply knowledge-based cooperative query answering techniques [CMB93, CYC+96] to the active database system to support active rules with cooperative features such as conceptual terms and approximate operators which enable us to specify inexactness and uncertainty in rule conditions and actions. For example, if *bad* weather is reported in a certain region, we would like to inform the *nearby* commanders, where "bad" is a conceptual condition and "nearby" is an approximate operator in the rule action.

The paper is organized as follows: Section 2 presents cooperative technology which consists of automatic generation of knowledge base from data source and multilevel knowledge representation – TAH, and knowledge-based cooperative query relaxation. Section 3 applies cooperative query relaxation technology into active rules to provide English-like rule specification capability. Section 4 presents the architecture of and data flow in Cooperative Sentinel (CoSent). Section 5 summarizes the knowledge model, execution model and management model of CoSent. Section 6 describes

CoSent implementation and experience. Sections 7 and 8 presents comparison with related works and the conclusion.

2 Cooperative Query Relaxation

Knowledge-based query relaxation is used in intelligent information systems such as CoBase [CMB93, CYC+96]. High-level concepts can be specified in query condition. Relaxation increases the search scope of the query condition to provide approximate matching when no exact match can be found. The result is a user-friendly cooperative database system.

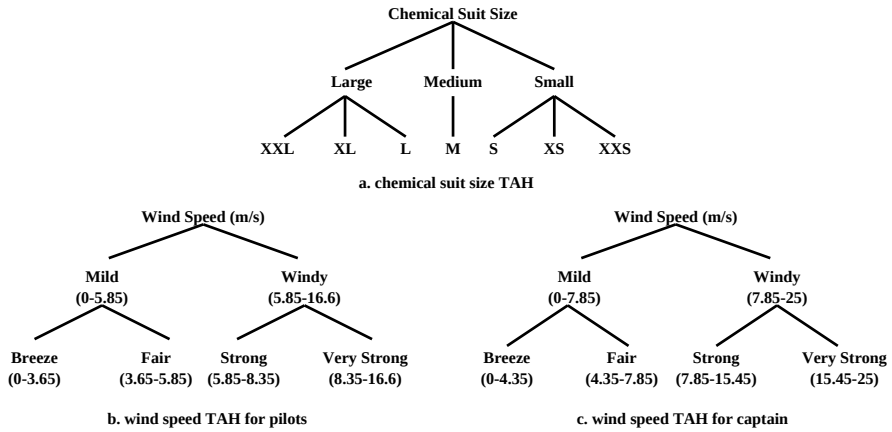

Figure 1: Examples of Type Abstraction Hierarchies

2.1 Type Abstraction Hierarchy (TAH)

We use a novel multi-level tree structure called the Type Abstraction Hierarchy (TAH) [CYC+96] for knowledge representation. Figure 1 presents some simplified example TAHs. Higher level nodes in the TAH represent more general information than that of the lower level nodes. *Conceptual terms* can be defined on the TAH nodes. As a result, queries with conceptual conditions can be specified and processed. For example, in the query "find large size chemical suits", the conceptual term "large" can be transformed into XXL, XL, or L as shown in Figure 1a. The query condition can be generalized (enlarge scope) by moving up and specialized (reduce scope) by moving down the TAH. The relaxation process is repeated until enough answers are returned.

Automatic Construction of TAHs from Data Sources TAHs can be generated and maintained automatically based on the data instances. For numerical attribute values, Distribution Sensitive Clustering algorithm (DISC) method considers both frequency and value distributions of data, making the discovered concepts more content sensitive [CCHY96]. To optimize the quality of generated hierarchies, DISC partitions the data set of one or more attributes selected by the user into clusters such that the average relaxation error (the value difference between the instance value and the target value) is minimized [CCHY96]. For non-numerical attribute values, our Pattern Based Knowledge Induction (PBKI) method first derives IF-THEN rules for all pairs of attributes. Based on the derived rules, PBKI then computes co-occurrences between all pairs of values for a single attribute [MC93]. Starting with each value as a cluster, PBKI recursively merges clusters

with the highest co-occurrences to form larger clusters. The merging process is terminated when there is only one cluster left.

Both DISC and PBKI have polynomial time complexity, making them scalable to large application domains. DISC and PBKI have been used to generate TAHs for a large transportation database consisting of 94 relations, the largest one of which has 12 attributes and 195,598 tuples. The computation time is less than a minute on a Sun SPARC 10 Workstation to generate TAHs for most cases except a few with a large number of tuples which takes about 20 minutes.

TAH Generator, Editor, Directory, and Manager After the user selects the attributes of interest from the data table, TAH's can be generated automatically [CCHY96, MC93]. The TAHs can be visualized and edited by the user. Domain experts can annotate conceptual terms on the TAH node based on the cover range of that node.

Within the CoBase server, two principal components handle the type abstraction hierarchies: the TAH Directory and the TAH Manager. The TAH Directory contains descriptions of all TAHs maintained in CoBase, which follows a file directory structure. The TAH Manager provides access to the directory and retrieves TAHs based on query conditions and the user type.

2.2 Approximate Operators

In addition to providing *implicit* condition relaxation via TAHs, relaxation can also be specified *explicitly* through the use of *approximate operators* such as `approximate`, `near to`, `similar to`, *etc.*. The `approximate` operator relaxes the specified values to an approximate range. For example, "`approximate` 6:00am" can be relaxed to (5:00am,7:00am). The `near to` operator can be used for geographical nearness specification. The `similar to` operator can be used to find objects similar to the given target object based on a set of pre-specified attributes. Weights can be assigned to the set of attributes in accordance to their relative importance. The returned answer sets are ranked based on a pre-specified measure that evaluates the nearness of the answers from the target object.

2.3 Relaxation Control

Relaxation control operators such as `relaxation order`, `not relaxable`, `use tah`, `preference list`, `unacceptable list` and `relaxation level` can be used to control the relaxation process. These relaxation control operators can be specified in the query. If relaxation control is not specified then default relaxation controls will be used based on user type [CYC+96].

3 Cooperative Active Rules

Traditional active rules require precise specification of trigger conditions and actions. Rule creators need to have detailed knowledge (schema as well as data value) about the underlying databases to specify the active rules. However, such detailed knowledge is often difficult to obtain. Furthermore, the rule designers and users' comprehension of a trigger condition may be inexact and are user and context sensitive. To remedy these shortcomings, we allow users to specify English-like active rules that contain conceptual terms and approximate operators. We use knowledge-based relaxation techniques to transform the English-like active rules to low-level rules that can be processed in commercial database triggering systems. In the following section, we shall extend the cooperative technology to the trigger condition and action specification.

3.1 Cooperative Trigger Condition Specification

Conceptual Terms in Trigger Condition Specification In traditional active rules, the trigger conditions are specified by precise values, *e.g.*, "the wave height is 3 meters and the wind speed is 16 meters per second". However, the user usually has only an approximate estimate of the situation. Therefore, humans prefer to specify trigger conditions with conceptual terms and approximate operators. Further, trigger conditions are user and context sensitive. Consider the following example, "if the weather in Bizerte is *very bad* then notify the user." "Very bad" is a conceptual term which depends on the user type and context. For an airplane pilot, "weather is very bad" is translated into "the wind speed is *very strong* and the visibility is *very poor*"; and for a ship captain, it is translated into "the wind speed is *very strong* and the wave height is *very high*". "Very strong", "very high" and "very poor" are conceptual terms. These terms can be handled by customizing the TAHs for different user types and contexts. These conceptual terms can be encoded in the TAH nodes in the corresponding TAHs. For example, "very strong" wind speed for an airplane pilot and a ship captain have different interpretations as shown in Figures 1b and 1c. Based on the wind speed TAH for airplane pilots the "very bad weather" is translated into "the wind speed is between 8.35 and 16.6 meters per second, and the visibility is less than 10 meters." Based on the wind speed TAH for ship captains, the above trigger condition is translated into "the wind speed is in the range 15.45 to 25 meters per second, and the wave height is greater than 5 meters." Note that the conceptual term "very bad weather" also translate to different conditions for different user types. Further, English-like rules with conceptual terms can be reused by different users and contexts. As a result, it reduces the number of rules in the system and eases rule maintenance.

Approximate Operators in Trigger Condition Specification The approximate operators `approximate`, `near to`, `similar to` *etc.*, can also be used in trigger condition specification to enhance the expressiveness of the trigger condition specification. For example, a ship captain wants to be informed if the weather condition is bad *near to* Bizerte *approximately* on 10/1/99. The approximate operators `near to` and `approximate` can be used to specify the trigger value ranges. The introduction of such operators greatly ease the trigger condition specification. The rule creator need not know the exact range values. Rather, the domain knowledge are encoded in the corresponding TAHs by the domain expert according to user type and context.

Relaxation Controls in Trigger Condition Specification The conceptual terms and approximate operators in trigger condition specification are converted to concrete value ranges and data sets based on TAHs. If no TAH is specified, CoSent will select a set of default TAHs based on the user type and context. The rule creator can also use specific TAHs to match his intention by using the relaxation control operator `use tah`. Further, the rule creator can also use the relaxation control operator `relaxation level` to control the relaxation of trigger conditions.

3.2 Cooperative Rule Action Specification

By introducing cooperative features to rule action specification, the rule creators can rely on CoBase [CMB93] to relax the action condition. For example, consider the rule "if not enough large-sized chemical suits at the warehouse in city X, then find 10,000 units of large-sized chemical suits from depots *near to* city X." In the action part of the above example, if there are less than 10,000 large-sized chemical suits available, CoBase will *implicitly* relax "10,000" to "approximately 10,000" and "large-sized" to "medium-sized" or "extra-large". The location relaxation is *explicitly*

specified by the approximate operator `near to`. Relaxation control operators such as `relaxation order` can also be specified. The absence of such relaxation control operators in the above rule requires CoBase to select a default relaxation order based on the user type.

3.3 Rule Template Construction and Rule Finalization

Rule template construction and rule finalization are introduced to assist rule specification. Cooperative sentinel users can be classified into *rule template writers* and *high-level users*. Rule template writers are familiar with rule syntax, and are responsible for generating rule templates that represent generic responses to generic situations. Unspecified parameters in rule conditions and actions might include cooperative features. Rule template writers generate rule templates after consultation with the high-level user familiar with the domain. Based on domain knowledge, the high-level user specifies parameter values for the conceptual terms in the rule template. Geographic location parameters can be specified on a map interface using the drag-and-drop method.

When all the rule parameters are specified, the rule is forwarded to the CoSent Server and conceptual terms and approximate operators are converted into value ranges or data sets according to knowledge represented in the TAHs. The rule is then presented to the high-level users for fine-tuning before actually installed into CoSent.

4 CoSent Architecture and Data Flow

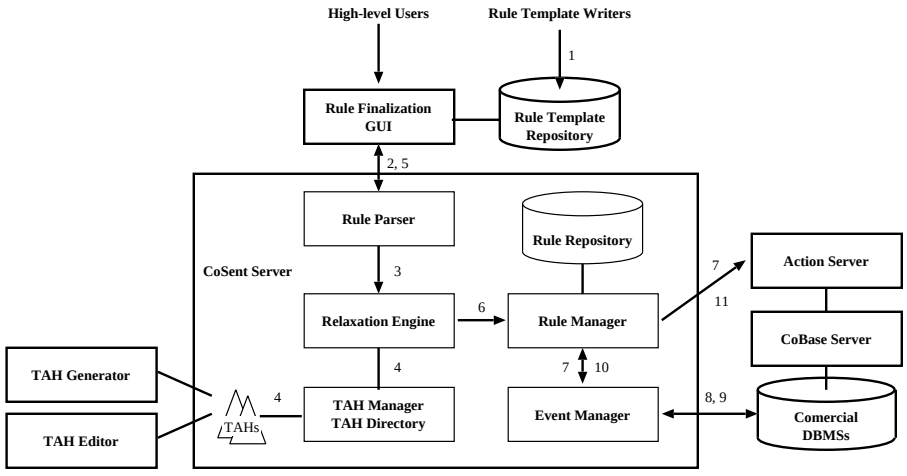

Figure 2: CoSent Architecture and Data Flow

CoSent consists of the Rule Finalization GUI, CoSent Server, TAH Generator, TAH Editor, and Action Server (Figure 2).

The CoSent data flow consists of the knowledge acquisition phase, rule specification phase, and rule triggering phase.

Knowledge Acquisition Phase The TAHs are generated off-line. The domain knowledge can be edited by domain experts using the TAH Editor.

Rule Specification Phase Following the CoSent rule template specification API, the rule template writers generate a set of rule templates which are stored in the Rule Template Repository (1). The Rule Finalization GUI presents the available rule templates to the high-level for

parameter specification. The resulting rule is then forwarded to the CoSent Server for processing (2). The Rule Parser in the CoSent Server generates an internal rule representation (3). The Relaxation Engine consults the TAH Manager to convert cooperative features in the rules to the corresponding value ranges or data sets (4). The converted active rule is returned to the high-level users for further fine-tuning (5). The finalized rule is then forwarded to the Rule Manager (6). The Rule Manager dispatches the event specification to the Event Manager, and the action specification to the Action Server (7). The Event Manager then installs the primitive event specifications to the trigger mechanism of the underlying databases (8).

Rule Triggering Phase The Event Manager is informed when primitive events are detected in the databases (9). The Event Manager then evaluates the event condition and informs the Rule Manager of those events that satisfy the rule conditions (10). The Rule Manager selects a rule from the set of rules with satisfied conditions, and then notifies the Action Server to execute the action (11).

5 Model Specification of CoSent

In this section, we shall present the CoSent model specification based on the active database knowledge model and execution model specified in [PDW+93, PD99b].

Event	Type	Primitive, Composite
	Source	Structure Operation, Clock
	Granularity	Member
	Role	Mandatory
Condition	Role	Optional
	Context	Bind_E, DB_C
Action	Options	Structure Operation, External
	Context	Bind_E, Bind_C, DB_A

Table 1: Knowledge Model Specification of CoSent

Let us describe the event, condition, and action of CoSent (Table 1). CoSent supports both primitive events and composite events. Primitive events include database operations such as insertion, deletion, and update on database tables, as well as simple temporal events. Events are defined for each tuple affected by an operation. A rule without a condition is equivalent to a rule with a condition that always evaluates to `true`. The event condition evaluation is based on data binding when the event occurs (Bind_E) and the database state when the condition is evaluated (DB_C). Rule action can be database operations or external calls. Information used in the rule action comes from data binding when the triggering event occurs (Bind_E), data binding when the condition is evaluated (Bind_C), and the database state when the rule action is executed (DB_A).

Table 2 presents the execution model of CoSent. The condition evaluation occurs in a transaction that is detached from the transaction in which the event occurs. Likewise, the action execution occurs in a transaction that is detached from the transaction in which the condition is evaluated. In CoSent layered architecture, the primitive event detection, rule condition evaluation, and rule action execution are processed separately in the underlying databases, CoSent Server, and Action Server. As a result, CoSent provides the detached condition evaluation and detached action execution.

CoSent provides rule template API (Programming Language) as well as SQL-like rule language

Condition Mode	Detached
Action Mode	Detached
Transaction Granularity	Tuple
Net-effect Policy	No
Cycle Policy	Recursive
Priority	Numerical, Static
Scheduling	All Sequential
Error Handling	Abort

Table 2: Execution Model Specification of CoSent

Description	Programming Language, Query Language, English-like Language
Operation	Activate, Deactivate
Adaptability	Run Time
Data Model	Relational

Table 3: Management Model Specification of CoSent

(Query Language). A unique feature of CoSent is the English-like rule specification language. Rules in CoSent can be activated and deactivated by the users. Because of the separation of the CoSent Server and Action Server, rules with new actions can be added without interfering event detection. Therefore, CoSent rules can be constructed during run time even with new action specifications. Although CoSent is based on relational databases, the English-like rule specification can also be adapted to other sentinel systems (*e.g.*, object-oriented database systems, sensor nets *etc.*).

6 Implementation and Experience

We have implemented CoSent which operates on Sun Solaris as well as the Windows NT systems, at UCLA. The data sources include Oracle 7.3, Oracle 8, and SyBase database systems. The trigger processing agent is implemented in C++. Orbix CORBA is used for agent communication. A Java based Template Finalization Interface that includes map display based on ESRI MapObject is used for rule specification and activation. Java based TAH Generator and Editor are used to generate and edit TAHs. We have measured the performance of the trigger system on a database which consists of 250 tables, with an average 5,000 tuples per table. 150 English-like rules were used and tested. The measured average delay between database update and action notification is less than 1 second for an average of 3-level rule complexities. Conceptual terms (*e.g.*, bad weather, significantly low) and approximate operators (*e.g.*, `near to` and `similar to`) are used to specify cooperative active rules. Relaxation control operators such as `use tah` and `relaxation level` are also provided to further control the relaxation process.

We have resolved the following list of problems during the implementation.

1. Since neither Oracle nor SyBase provides message passing mechanism to communicate between the application process and triggers, we had to use an ad hoc method to implement the notification agents. The problem is more pronounced when porting our system from Solaris to NT systems. A methodology for notification agent construction is necessary for different data sources.

2. Since our goal is for CoSent to operate on top of commercial database systems, we did not modify the internal triggering mechanism of such systems. As a result, the transition information was not available when a simple database event was notified to the event manager. In order to access transition information during event condition evaluation, the database trigger has to preserve the transition information in a transition table.

3. The action processing was an integral part of the CoSent Server in the initial development. Whenever the user inserted a new action procedure, the entire system had to be brought down and recompiled. This is clearly not acceptable for mission critical applications. Our new design separates the Action Server from the CoSent Server, thus allowing seamless addition of new action procedures.

7 Comparison with Related Works

There are recent works on fuzzy triggers that merges fuzzy logic concepts into active database capabilities [BW97, BKPW97, WB98] to provide high-level active rules. However, fuzzy trigger requires the user to specify fuzzy member functions which is a difficult task.

In our approach, the knowledge base (TAHs) is automatically generated from the mining of databases based on user type and context. Domain experts can then encode the high-level concepts on the TAH nodes for users to specify high-level rules. Further, relaxation control can also be specified in the rules to control the relaxation process. In addition, CoSent also support approximate operators in the rule.

8 Conclusion

We have proposed and implemented a Cooperative Sentinel (CoSent) that supports English-like ECA rules which contains conceptual terms and approximate operators. Based on the domain knowledge represented in Type Abstraction Hierarchies (TAHs), the conceptual terms and approximate operators can be translated into range values and can be input to the triggering mechanism of commercial RDBMS. TAH can be generated automatically from data sources, thus CoSent is scalable to large systems. Allowing user to specifying the trigger conditions and actions in English-like rules mimic the human cognitive process, which not only increases the expressibility but also greatly eases rule specification. Further, high-level rules are generic in nature and can be shared by users, which reduces the number of rules in the system and thus eases rule maintenance.

References

[BKPW97] Tarik Bouaziz, Janne Karvonen, Anton Pesonen, and Antoni Wolski. Design and implementation of tempo fuzzy triggers. In *Proceedings of the 8th International Conference on Database and Expert Systems Applications*, pages 91–100, Toulouse, France, September 1997.

[BW97] Tarik Bouaziz and Antoni Wolski. Applying fuzzy events to approximate reasoning in active database. In *Proceedings of the 6th International Conference on Fuzzy Systems*, Barcelona, Catalonia, Spain, July 1997.

[CCHY96] Wesley W. Chu, Kuorong Chiang, Chih Cheng Hsu, and Henrick Yau. An error-based conceptual clustering method for providing approximate query answers. *Communications of ACM, Virtual Extension Edition (http://www.acm.org/cacm/extension)*, 39(12):216–230, December 1996.

[CMB93] Wesley W. Chu, M. A. Merzbacher, and L. Berkovich. The design and implementation of CoBase. In *Proceedings of ACM SIGMOD 93*, pages 517–522, Washington D. C., USA, May 1993.

[CYC+96] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. CoBase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6(11), 1996.

[Inf99] Informix Corperation. *Informix Guide to SQL, Syntax*, December 1999.

[ISO99] ISO-ANSI. *(ISO-ANSI Working Draft) Foundation (SQL/Foundation)*, March 1999. ftp://jerry.ece.umassd.edu/isowg3/dbl/BASEdocs/public/sql-foundation-wd-1999-03.pdf.

[MC93] Matthew Merzbacher and Wesley W. Chu. Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery*, Washington, DC, 1993.

[Ora99] Oracle Corporation. *Oracle8i Concepts, Release 8.1.5, A67781-01*, February 1999.

[Pat99] Norman W. Paton, editor. *Active Rules in Database Systems*. Monographs in Computer Science. Springer-Verlag, 1999.

[PD99a] Norman W. Paton and Oscar Díaz. Active database systems. *Computing Surveys*, 31(1):63–103, 1999.

[PD99b] Norman W. Paton and Oscar Díaz. Introduction. In Paton [Pat99], chapter 1, pages 1–27.

[PDW+93] Norman W. Paton, Oscar Díaz, M. Howard Williams, Jack Campin, Andrew Dinn, and Arturo Jaime. Dimensions of active behaviour. In *Rules in Database Systems, Proceedings of the 1st International Workshop on Rules in Database Systems*, pages 40–57, Edinburgh, Scotland, August 1993.

[Syb99] Sybase Inc. *Sybase Adaptive Server Enterprise Reference Manual, Version 12*, 1999.

[WB98] Antoni Wolski and Tarik Bouaziz. Fuzzy triggers: Incorporating imprecise reasoning into active databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 108–115, Orlando, Florida, February 1998.

[WC96] Jennifer Widom and Stefano Ceri. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1996.