Modeling and Analysis of Temporal Failure and Degradation Behavior of Critical Infrastructure Systems

Mostafa Bassiouni and Ratan Guha
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL
email:- {bassi, guha}@cs.ucf.edu

Abstract

In this paper, we present an approach for modeling and analyzing the temporal failure and degradation behavior of critical infrastructure systems (CISs) using advanced temporal database management systems. We classify the possible failure and/or degraded performance of CISs into different temporal categories, namely, crisp or exact intervals, non-vanishing imprecise intervals and vanishing imprecise intervals. The three temporal operators: Union (OR), Overlap (AND) and Not are extended to operate on the above categories of precise and imprecise intervals. The temporal operators are used recursively to capture the fault tolerance topology of CIS. For example, if a component of CIS has built-in redundancy for fault tolerance, the fault behavior of this component propagates to the outside only when all the redundant units of this component fail simultaneously. In this case, the failure temporal expressions of the redundant units are joined by temporal Overlap operators to indicate that the failure of the composite component is contingent on the failure of all units. We briefly show how query languages with temporal extensions can be used to obtain useful answers for time-related queries and retrieve useful information about the exact and potential time points for degraded modes of operation. The storage overhead of incorporating the imprecise intervals in a temporal database is analyzed.

Keywords: Critical infrastructure systems, faults and vulnerabilities, temporal behavior, temporal databases, time impreciseness.

1. Introduction

Critical infrastructure systems (CISs) are complex networked systems that rely on distributed data processing hardware and software to accomplish the coordination and interfacing among CIS sub-systems. As a result, a typical CIS is usually exposed to significant potential reliability and security problems. Extensive research is needed for the assessment of CIS vulnerabilities and the development of prescriptive hardening procedures, design principles and control strategies to mitigate the effects of accidental failures and deliberate attacks on CIS. It is envisioned that a vast array of design and database tools would be needed to support the various measurements, design, development and testing activities needed for these tasks. In this paper, we motivate the use of temporal databases [OZSO95, BASS99, BETT98, BOHL98, KAKO01, SNOD98, TANS93] as a potentially useful tool that can help the effort of analyzing the temporal characteristics of CIS sub-systems. We first provide a short overview of temporal databases and then proceed to discuss their applications to CIS.

2. Temporal Databases and Imprecise Time Representations

We shall use the same general notation used in the literature on temporal databases and tuple/attribute-based timestamping [LORE93, NAVA93, BASS94, TANS97]. Time is encoded as a sequence of integer points and the symbol "NOW" is used to indicate the integer representing the current value of time. Conforming with the notation we used in [BASS94], the temporal comparison operators (e.g., $>_T$, $<_T$, $=_T$, $\geq_T$) return the set of intervals during which the comparison relation is satisfied. The notations $\cap_t$, $\cup_t$ and NOT_T will be used to denote the temporal versions of Overlap, Union and Not. If S1 and S2 are two sets of time intervals, then the expression S1$\cap_t$ S2 returns the time sub-intervals that are contained in both S1 and S2 (i.e., it returns their overlap). The expression S1 $\cup_t$ S2 returns the union of the intervals of S1 and S2. The temporal unary

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

operator NOT_T returns the set of intervals which is the complement of its operand with respect to the universal interval <0,∞>. Below are some examples of temporal expressions and their values.

Expression	Value
{<1,2>, <5, 14>} $\cap_t$ {<1,3>, <6,∞>}	{<1,2>, <6,14>}
{<2, 5>, <9, 10>} $\cup_t$ {<4, 7>, <14, ∞>}	{<2, 7>, <9, 10>, <14, ∞>}
NOT_T {<0,5>, <8, ∞>}	<6,7>

The symbol ∞ is used in this paper to represent the largest value of time applicable to the database system at hand. For example, ∞ may be represented by the largest integer value. If the database does not deal with future events, ∞ may be taken to mean the current (highest) timestamp, which is often denoted by the symbol "NOW".

In many real life situations, the time boundary of events may not be exactly known. For example, the available information known about a certain component, say component X, in CIS is that X had degraded performance for a period of unknown length starting at time 4 and ending no later than time 8. Based on this knowledge, the possible duration of the degraded mode must be one of the following five intervals: <4,4>, <4,5>, <4,6>, <4,7> or <4,8>. Any one of these five intervals has the potential of being the actual (precise) value of the duration of degradation. If this knowledge is to be stored in a traditional temporal database (i.e., one that can only handle precise or well-defined intervals), we will then be forced to associate the degraded mode with some selected well-defined interval, say <4,8>. In this case, the following query

> *"retrieve the names of components that operated normally at time 8"*

will not retrieve X because we already associated the precise interval <4,8> as the duration of X's degradation. Component X, however, may have had normal operation at time 8; the reply to the above query should have included X as a potential candidate satisfying the query. In [BASS99], three models to handle time impreciseness are presented and their formal properties are discussed. We shall use the second model presented in [BASS99] in all the discussions and examples given in this paper.

In this paper, we shall use the "*" symbol to indicate time uncertainty. The * symbol will be used within time intervals to indicate the range of uncertainty. For example, the notation 3*8 indicates that the time point under consideration could be anywhere in the range from 3 to 8 inclusive. The * symbol will be also used as a prefix before operators to indicate that the operation must take into account the uncertainty of the time values of the operands. For example, when evaluating the Boolean condition "T *= 1990", all potential (fuzzy) values of T need to be considered. More explanation of the * operator will be given in the various sections of the paper.

3. Applying Fuzzy Temporal Databases to CIS

Imprecise intervals are divided into two categories as in Model II presented in [BASS99]: non-vanishing and vanishing.

Non-Vanishing Intervals

The general format of a non-vanishing imprecise interval is of the form

<a*b, c*d>

where a ≤ b, c ≤ d, a ≤ c and b ≤ d. The meaning of this interval is that the start time of the event is between a and b and the end time is between c and d. If one of the end points is precise, we use the conventional notation of representing this point by a single integer. Thus the interval <8*12, 20*20> will be denoted by <8*12,20>. If a=0, or d=∞, then a shorthand notation may be used by omitting the value of a or d. Similarly if b=d, the value of b may be omitted. For example, the interval <3*,7> is equivalent to <3*7,7>, <*3,7> is equivalent to <0*3,7> and <*8,20*> is equivalent to <0*8,20* ∞>. Here are two more examples with brief explanation:

<*3,8> The finish time of this interval is precise but the starting time may have occurred at time 0, 1, 2 or 3. Possible values of this

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

time interval are <0,8>, <1,8>, <2,8> or <3,8>.

<3*5,8*9> Both the starting and finish points of this event are imprecise. Possible values of this time interval are <3,8>, <4,8>, <5,8>, <3,9>, <4,9> or <5,9>.

Vanishing Intervals

An interval is allowed to vanish if its potential values progressively shrink to a single point. Thus the interval <5*6,8> is not amenable to be vanishing, but the interval <5*8,8> , also denoted <5*,8>, can be made a vanishing interval by including its range in square brackets. This gives the vanishing interval <[5*,8]> whose set of potential values is { <5,8>, <6,8>, <7,8>, <8,8> , Φ}. In general, a vanishing interval can be one of the following types <[a*,b]>, <[a,*b]> and <[a*,*b]>.

We now can classify the possible failure and/or degraded performance of CISs into the following temporal categories:

a) crisp or exact intervals such as the case of precisely known and precisely executd down times for scheduled maintenance. Examples of this type are the intervals <3,8> and <11,11>.

b) non-vanishing imprecise intervals such as the case of rush time congestion in transportation systems or peak traffic times in communication networks; in this case, the start and finish endpoints of the congestion period may each have a temporal range whose length reflects the level of uncertainty or fuzziness with respect to the actual duration of the congestion. Examples of this type are the intervals <*2, 7*9> and <4*7, 10>.

c) vanishing imprecise intervals where the two endpoints of the event have a sliding range of uncertainty that permits the length of the event to shrink to zero, i.e., the event may not even happen. The interval <[3*,6]> is an example of this type.

With the above model, the temporal Overlap, Union, and Not operators can be used to construct the temporal expression of a system by capturing the topology of this system, its components, and the semantics of the behavior being analyzed. It is important to notice that the three temporal operators are closed under the fuzzy model used in this paper (model II in [BASS99]). In other words, the results of applying these operators to operands satisfying this model can always be expressed as interval types belonging to this same model. For example,

$<3*8,18*22> \cap_t <5*10,16*20>$

returns <5*10,16*20>

$<10*12, 15> \cap_t <11*16,20>$

returns <[11*,15]>

Where $\cap_t$ denotes temporal overlap. Similarly,

$<1*5,10*18> \cup_t <7*11,17>$

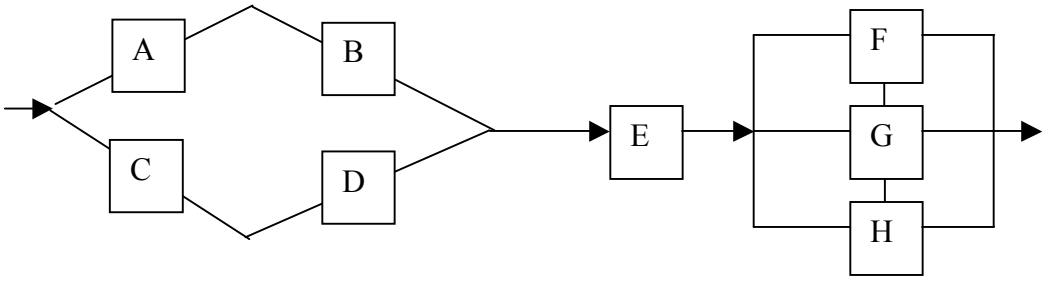
returns <1*5,17*18>

$<[5*,30]> \cup_t <10,25*28>$

returns <5*10,25*30>

where $\cup_t$ denotes temporal union.

Consider now a CIS with three subsystems. The redundancy design (for purposes of fault tolerance) of these subsystems is shown in the following parallel-series diagram.



Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

The three subsystems shown in the above diagram are:

- Subsystem 1: the bridge structured subsystem (nodes A, B, C, and D).
- Subsystem 2: the parallel-redundant subsystem (nodes F, G and H).
- Subsystem 3: the interface between the above two subsystems (node E).

Let S_K denote the fuzzy temporal expression for the error-free operation of node K. The temporal expressions S_1 and S_2 for the error-free operations of subsystems 1 and 2, respectively, are

$S_1 = (S_A \cap_t S_B) \cup_t (S_C \cap_t S_D)$
$S_2 = S_F \cup_t S_G \cup_t S_H$

The above expressions are derived from the topology of the two subsystems using straightforward logic. For example, the second subsystem is a parallel-redundant network which is able to function normally as long as at least one of the three nodes F, G or H is operating normally. Thus the time intervals for the proper operation of this subsystem are obtained by performing the temporal union on the individual temporal expressions of the three nodes. As further illustration, let

$S_F = \{<5,8*12>, <15,15>\}$
$S_G = \{<*2,6>, <12,14*17>\}$
$S_H = \{<5,9>\}$

Then the error-free operation of subsystem 2 is described by the temporal expression

$S_2 = \{<*2,9*11>, <12,15*17>\}$

The above expression implies that subsystem 2 has been operating normally at time points 2 through 9 and 12 through 15 (inclusive). These are exactly the time points during which there is at least one node (F, G, H) with normal operation. Furthermore, the expression also implies that subsystem 2 may have been operating normally during any of the intervals <0,1>, <1,1>, <10,10>, <10,11>, <16,16>, <16,17>. These are the intervals during which none of the three nodes is confirmed to have been working normally but there is at least one node which may have been operating normally.

The normal operations of the entire system given in the above figure can be described by the following temporal expression

$S = S_1 \cap_t S_E \cap_t S_2$

4. Other Considerations

4.1. Semantics of Behavior

As the above example clearly shows, the topology of the CIS redundancy architecture determines how the expressions to capture their temporal behavior are constructed. However, the semantics of the target behavior also participate critically in shaping up these temporal expressions. We illustrate this by an example.

Let R_K denote the fuzzy temporal expression for the degraded mode of operation of node K. Obviously, R_K conveys different semantics than the semantics implied by S_K (which captures the error-free behavior). The temporal expressions R_1 and R_2 for the degraded performance of subsystems 1 and 2, respectively, are

$R_1 = (R_A \cup_t R_B) \cap_t (R_C \cup_t R_D)$
$R_2 = R_F \cap_t R_G \cap_t R_H$

and the degraded performance of the entire system given in the above figure is described by the following temporal expression

$R = R_1 \cup_t R_E \cup_t R_2$

Note that because of the time impreciseness, the two expressions S and R may not be disjoint.

4.2 Analysis of the Storage Overhead

The model used in this paper has a total of five interval types: one precise interval type <a,b>, one non-vanishing imprecise type <a*b,c*d>, and three vanishing imprecise types <[a*,b]>, <[a,*b]> and <[a*,*b]>. Notice that the interval <a*b,c*d> is said to be valid if $a \le b$, $c \le d$, $a \le c$ and $b \le d$. Thus <3*7, 5*8> is a valid interval and its set of potential values includes <6,7> and <5,5> but does not include <3,4> or <8,9>. A three-bit flag per interval is needed to distinguish between the five interval types. In addition, the non-vanishing type requires and additional overhead of two integers to represent the two additional time points needed to specify the range of impreciseness.

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

4.3. Query Languages for Fuzzy Temporal DB

New operators can be introduced to help users formulate their qualifications and construct complex queries, e.g., based on best or worst case time scenarios. The simplest form of extension is to provide two modes of the retrieval query [BASS99]:

- the first mode uses precise time intervals and hence retrieves entities or attributes that are guaranteed to satisfy the query qualification. This mode is assumed to be the default mode and can therefore use the existing syntax (e.g., the keyword "SELECT" in SQL or "RETRIEVE" in QUEL).
- the second mode uses imprecise time intervals and returns entities or attributes that satisfy or has a potential to satisfy the query qualification. This mode is optional and requires the use of new syntax, say the new keyword *SELECT.

For example, assume that relation TRACK has two attributes: 1) name of the component which is a constant (non-temporal) field and 2) Status which is a temporal field that comprises status values (such as "operational" or "down") and the corresponding duration. The following query retrieves the names of components whose time of failure overlaps the time of failure of Server 1.

*SELECT X.Name

FROM TRACK X, Y

WHERE (X.Status $\cap_t$ Y.Status) AND

X.Name = "Server 1"

AND VALUE(X.Status) = "down"

The built-in function VALUE extracts the status value from the composite value/time temporal field. The star operator before the keyword SELECT directs the DBMS to return both confirmed and potential candidates for the above query.

The star operator can also be applied at a finer level of control, e.g., a single temporal operator rather than to the entire query or transaction. Specifically, if the operator is prefixed with a star then this implies that the computation of this operator uses imprecise intervals. For example, the qualification

$$X.Status \ *=_T \ Y.Status$$

returns the set of intervals during which component X had the same status or potentially has the same status as that of Y. If the star operator is omitted (i.e., the qualification becomes X.Status $=_T$ Y.Status), then the returned answer will only capture confirmed equivalence of status values.

Finally, further flexibility can be given to users by providing explicit operators that manipulate imprecise intervals. For example, the conversion operators Min_Span and Max_Span can be used to map a temporal expression to its minimum confirmed time span or to its maximum possible time span, respectively. Thus we have

Min_Span (<3*5,9*15>)

returns <5,9>

Max_Span (<3*5,9*15>)

returns <3,15>

Min_Span ({<*5,9>, <10*,20>})

returns {<5,9>, <20,20> }

Max_Span ({<*5,9>, <10*,20>})

returns { <0,20> }

Based on our previous notation, temporal operators that are not star-prefixed must use precise intervals for their (default mode of) evaluation. This is done by performing the Min_Span conversion operation on the operand(s) of each operator that is not star-prefixed.

5. Conclusions

In this paper, we presented an approach for modeling and analyzing the temporal failure and degradation behavior of critical infrastructure systems (CISs). The concept of imprecise time intervals is applied to the failure behavior and/or degraded performance of CIS. Crisp or exact intervals, non-vanishing imprecise intervals and vanishing imprecise intervals are all used in the modeling of CIS behavior. The temporal operators are used to capture the fault-tolerant topology of CIS and the interactions among its sub-systems. The semantics of the behavior is also used to guide the construction of the temporal expression. The paper discussed extensions to query languages that can be used to retrieve useful information about the exact and potential time points for the normal and degraded modes of operation.

Acknowledgment

This work has been partially supported by ARO under grant DAAD19-01-1-0502. The views and

conclusions herein are those of the authors and do not represent the official policies of the funding agency or the University of Central Florida.

References

[BASS99] Bassiouni, M.A. and Llewellyn, M., "Extending temporal query languages to handle imprecise time intervals," J. Computer Languages, Vol.25, April 1999, pp. 39-54.

[BASS94] Bassiouni, M. A., Mukherjee, A. and Llewellyn, M. J., "Design and Implementation of Extended Boolean and Comparison Operators for Time-Oriented Query Languages," The Computer Journal, Vol.37, No.7, 1994, pp. 576-587.

[BETT98] Bettini, C. et. al. "Temporal Semantic Assumptions and Their Use in databases," IEEE Transaction on Knowledge and Data Engineering, Vol. 10, No. 2, 1998, pp. 277-296.

[BOHL98] Bohlen M, et al. "Point-versus interval-based temporal data models," In: Proceedings of IEEE Conf. On Data Engineering, 1998.

[KAKO01] Kakoudakis, I. and Theodoulidis, B. "TAU: Towards a Unified Temporal Information Management Framework," Bulletin of the Italian Association for Artifical Intelligence, (AI*IA), March 2001.

[LORE93] Lorentzos N. The interval-extended relational model and its applications to valid-time databases. In: Tansul A, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R, editors, Temporal databases. New York: Benjamin Cummings, 1993. P. 67-91.

[NAVA93] Navathe, Shamkant B. and Rafi Ahmed. "Temporal Extensions to the Relational Model and SQL," Temporal Databases: Theory, Design, and Implementation, Inc., 1993, pp. 92-109.

[OZSO95] Ozsoyoglu G, Snodrass RT. Temporal and real-time databases: a survey, IEEE Transactions on Knowledge and Data Engineering 1995; 7(4), pp.513-32.

[SNOD98] Snodgrass, R. "Of Duplicates and Septuplets," Database Programming &Design, June 1998, pp. 46-49.

[TANS97] Tansel A, Tin E. "The expressive power of temporal relational query languages," IEEE Transactions on Knowledge and Data Engineering 1997; 9(1).

[TANS93] Tansel, Abdullah, et al. "Part I: Extensions to the Relational Model," Temporal Databases: Theory, Design, and Implementation, eds.A. Tansel, et al., The Benjamin/Cummings Publishing Company, Inc.,1993, pp. 1-5.

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE