Monitoring of Timing Constraints with Confidence Threshold Requirements *

Chan-Gun Lee, Aloysius K. Mok
Department of Computer Sciences,
The University of Texas at Austin
{cglee,mok}@cs.utexas.edu

Prabhudev Konana
Department of MSIS,
The University of Texas at Austin
pkonana@mail.utexas.edu

Abstract

We propose an algorithm for monitoring timing constraints to satisfy confidence threshold requirements when there is uncertainty in the exact timing of event occurrences. In our model, a timed event trace is examined for possible satisfaction/violation with respect to a given set of timing constraints. Every event occurrence has a timestamp given by a time interval. Assuming that the time of occurrence is uniformly distributed over the time interval, our algorithm determines whether the probability that a timing constraint has been satisfied exceeds a specified threshold value. Timing constraints are composed of deadline and delay constraints for which satisfaction probabilities are defined. A confidence threshold is a minimum satisfaction probability of the timing constraint. A timing constraint is violated if the confidence threshold is not reached by the timed event trace. We present a PTime monitoring algorithm for detecting timing violation by finding the earliest expiration time (EET) of the deadline timer for each of the cases $P = 100\%$, $50\% \leq P < 100\%$, *and* $0\% < P < 50\%$, *where* P *is the confidence threshold of the timing constraint. We give a derivation of the implicit constraints needed for computing the EET, and we show how to use an all-pairs shortest path algorithm to compute the implicit constraints.*

1. Introduction

Monitoring timing constraints for violation or satisfaction is important in real-time systems for many purposes. The satisfaction of a timing constraint may for example signal a mode change by the software. Hard real-time systems are designed to avoid violating any timing constraints since a violation may lead to catastrophic system failures. In such cases, a responsible system architect must try to design for defense in depth by executing masking and compensatory actions to mitigate the damage caused by the system's failure to meet a critical timing constraint, e.g., to mask the failure of hardware components by a software mode change. Timing failures are well recognized as an important class of faults in fault-tolerance system models in [3]. Hence, the design of detectors for timing failures has been a topic of interest for both the real-time system and fault-tolerant system communities.

In this paper, we continue to extend our previous work in detecting timing constraint satisfaction/violation by incorporating quantitatively into our analysis the uncertainty in when events occur exactly. Our approach is event-based, and our timing constraints are expressed in terms of the temporal distance between event pairs. In general, there are two types of simple timing constraints from which more complex timing constraints can be constructed: a deadline timing constraint specifies a maximum separation between a pair of events; a delay timing constraint specifies a minimum separation. More complex timing constraints are composed of logical combinations of deadline and delay timing constraints.

The time of occurrence of an instance of an event is a value in the domain of time which may be a discrete or dense set; we consider dense time in this paper. The time of occurrence is captured by a timestamp. If the exact time value is known, the timestamp denotes a point in the time domain. However, in practice the exact time of occurrence may not be available because of the lack of precision in the measurement system, but it is usually possible to bracket the time of occurrence to within a range [13]. Timestamps are then given by a pair of time values that bound the interval in which the event is thought to have occurred. Some researchers, especially those in active database and artificial intelligence [17, 19, 4] prefer to deal with timestamps as intervals in order to allow more convenient semantics for describing their applications. Interval timestamps can be useful in a distributed environment where the clock uncertainty is unavoidable. Pietzuch et al. use the interval to represent the timestamp of the event and define proper ordering op-

* This research is supported partially by a National Science Foundation CAREER Award under contract No. IIS-9875746, by ONR grant N00014-03-1-0705 and by NSF grant CCR-0207853

Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS'03)
0-7695-2044-8/03 $ 17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

erators for the event composition in a distributed environment [16].

In our previous work [13], we presented a monitoring algorithm for timing constraints that uses interval timestamps. The analysis performed by that algorithm is qualitative and supports two modalities for detecting timing constraint satisfaction/violation: *certain and possible*.

In this paper, we extend our previous work in [13] by allowing users to specify a *confidence threshold* which denotes the minimum acceptable probability with which a timing constraint is satisfied by the observed event occurrences in a computation. A violation of a timing constraint is deemed to have occurred when the probability of the timing constraint being satisfied falls below the confidence threshold, as computed from the observed timestamp values.

The following example illustrates a practical situation where the event monitor capable of detecting satisfactions/violations of timing constraints with confidence thresholds is useful.

Example 1 *Consider a sensor-based monitoring system composed of two sensor devices and a transmitter. Each of the sensors S_1 and S_2 monitors a different signal and sends it to the transmitter in the form of detecting event E_1 or E_2 with interval timestamps. Suppose the event E_1 from S_1 conveys key information and event E_2 from S_2 conveys only supporting information. Furthermore, only event pairs (E_1, E_2) for which the difference in the time of occurrence between E_1, E_2 is within δ milliseconds should be transmitted. Owing to the high cost of information transmission, the transmitter is to relay only E_1 to the user whenever the transmitter cannot receive with a high probability (say, above 80%) the corresponding event E_2 in time to satisfy the maximum allowable separation between E_1 and E_2.*

In this paper, we assume that the exact time of occurrence of an event is uniformly distributed over the interval of the timestamp of the event. With the uniform distribution assumption, we calculate the satisfaction probabilities for both the deadline constraint and the delay constraint. In our analysis, we compute the earliest expiration time (EET) of a timer which waits for an event instance appearing in a timing constraint to confirm the satisfaction/violation of the timing constraint, as a function of the desired confidence threshold P. The key to our analysis involves the derivation of *implicit constraints* from the input set of timing constraints. We show that implicit constraints can be computed by using an all-pairs shortest-path algorithm.

The rest of this paper is organized as follows. Section 2 reviews related work. In Section 3, we introduce the event model of our system and a set of event functions with which we define the syntax of timing constraints and the probability of deadline/delay satisfaction from interval timestamps. Section 4 illustrates an efficient monitoring algorithm for a simple timing constraint. Section 5 presents an all pairs shortest path algorithm to derive implicit constraints from a set of timing constraints. Section 6 is the conclusion.

2. Related work

Research in event monitoring has been reported in the real-time systems literature as well as other areas such as fault tolerance, active databases etc. In the real-time systems area, Chodrow et al. observed that the *derivation of implicit constraints* is essential for catching timing violations as early as possible in their seminal paper [2]. They also proposed a graph-based monitoring algorithm for detecting violations of timing constraints. Jahanian et al. extended their work [2] to the distributed system environment in [7]. They showed that the minimization of the number of messages to be forwarded among processors for detecting violations in timing constraints as early as possible is an NP-hard problem. In [15, 14], Mok and Liu presented a timing constraint monitoring algorithm which uncovers most hidden information required to detect violations of timing constraints as early as possible at event compilation time.

Recently, we introduced an algorithm for monitoring timing constraints in [13] which extended the algorithm in [15, 14] to process events whose timestamps are time intervals. We proposed two new modalities, *certain and possible* to allow a user to specify his desired degree of certainty whether a timing violation has occurred.

Besides the real-time systems area, much research has been done in the active database systems and distributed systems areas under the topic of *event detection*. To name a few, Ode [6], SAMOS [5] and Sentinel [1] are active database systems designed to detect composite events as well as primitive events. However, most active database systems that have been proposed have limited functionalities for monitoring of timing constraints efficiently.

In the distributed systems area a number of efforts have been made toward an event monitoring system. The GEM (Generalized Event Monitor) [12] implements a generalized language for event monitoring in a distributed environment. This system permits time intervals in the event specification. Schwiderski proposed a new composition method for the timestamps of events in a distributed system and defined several temporal operators to order the timestamps based on the $2g_g$ precedence in [18]. Liebig et al. proposed a new timestamping method in which timestamps of events are modeled by accuracy intervals with reliable error bounds [11].

Applications of the event monitoring technique have been discussed recently in several papers. An E-Brokerage system based on a real-time composite event monitoring system was explained in [9]. Lee et al. applied event moni-

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

toring techniques to verify the run-time properties of a program in [10]. An extended work for providing the steering functionality as well as monitoring and checking was done in [8].

3. Timing constraints with confidence thresholds

We use the event model proposed in [13] to capture the uncertainties in event occurrence times. In this event model, the timestamp of an event is represented as a time interval, which consists of a pair of time values: the start and the end time. The pair brackets the time of occurrence of an event. Time values may be in the domain of non-negative integers or reals. In what follows, we present the definition of a timestamp and event functions of our event model.

Definition 1 *A timestamp consists of a pair of time points: (min_time, max_time) where min_time is the earliest possible time of occurrence of the event and max_time is the latest possible time of occurrence of the event.*

Definition 2 *min function:* $min(I)$ *= min_time where* I *is the timestamp given by (min_time, max_time).*

Definition 3 *max function:* $max(I)$ *= max_time where* I *is the timestamp given by (min_time, max_time).*

Definition 4 *length function:* $len(I) = max(I) - min(I)$ *where* I *is the timestamp given by (min_time, max_time).*

Definition 5 π *is the maximum length of any timestamp in the system. We assume that* π *is bounded and known to the monitoring system.*

Definition 6 $@$ *function:* $@(e, i)$ *= timestamp of the* i*th instance of event* e*.*

Definition 7 *A timing constraint on time intervals with a confidence threshold is given by:*

$$I_1 + d \geq I_2 \text{ with } P$$

where I_1 *and* I_2 *are timestamps.* d *is an integer constant representing a deadline (*≥ 0*) or a delay (*< 0*).* P *is a confidence threshold ranging from 0% to 100% .*

Example 2 *The following constraint specifies that there is a deadline constraint 30 from event* $(e1, i)$ *to event* $(e2, i)$ *with a confidence threshold 70%.*

$$@(e1, i) + 30 \geq @(e2, i) \text{ with } 70\%$$

In our system, the distribution of the event occurrence time in the timestamp is assumed to be uniform. Theorem 1 and Theorem 2 present the calculation of the satisfaction probability of a deadline constraint and a delay constraint respectively.

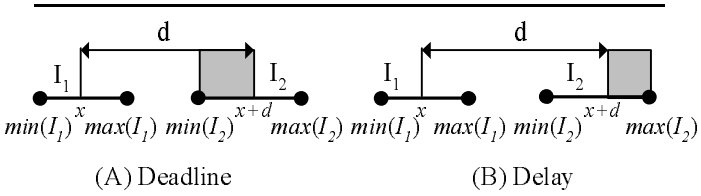


Figure 1. Satisfaction probability

Theorem 1 *Given a deadline constraint,* $c^+ : I_1 + d \geq I_2$ *where* $d \geq 0$*, the satisfaction probability of* c^+*,* $SP\ |_{c^+}$ *is given by the expression:*

$$\frac{\int_{min(I_1)}^{max(I_1)} MIN(MAX(x + d - min(I_2), 0), len(I_2))dx}{len(I_1)len(I_2)}$$

Theorem 2 *Given a delay constraint,* $c^- : I_2 + d \geq I_1$ *where* $d < 0$*, the satisfaction probability of* c^-*,* $SP\ |_{c^-}$ *is given by the expression:*

$$\frac{\int_{min(I_1)}^{max(I_1)} MIN(MAX(max(I_2) - (x + d), 0), len(I_2))dx}{len(I_1)len(I_2)}$$

In Theorem 1 and Theorem 2, $MIN(a, b)$ returns b if $b < a$ or returns a otherwise. $MAX(a, b)$ returns b if $b > a$ or returns a otherwise. To calculate the satisfaction probability of the deadline constraint c^+, we need to calculate for each time point x in I_1 the length in interval I_2 for which c^+ is satisfied and integrate with respect to x, for $min(I_1) \leq x \leq max(I_1)$. The resulting integral divided by the length of I_1 times the length of I_2 yields the probability that the two interval timestamps jointly satisfy the deadline. Figure 1(A) illustrates this idea: the gray box denotes the region that satisfies the deadline constraint assuming that the event occurrence time is exactly x, where $min(I_1) \leq x \leq max(I_1)$. Similarly, Figure 1(B) illustrates the calculation of the satisfaction probability of a delay constraint c^-.

Corollary 1 *Given a deadline constraint,* $c^+ : I_1 + d \geq I_2$ *where* $d \geq 0$ *and a delay constraint,* $c^- : I_2 + d \geq I_1$ *where* $d < 0$*, we have* $SP\ |_{c^+} = 1 - SP\ |_{c^-}$*.*
Proof:
This directly follows from Theorem 1 and Theorem 2.
□

Definition 8 *A violation of a timing constraint is said to occur when the satisfaction probability of the timing constraint is less than the specified confidence threshold.*

Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS'03)
0-7695-2044-8/03 $ 17.00 © 2003 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

4. An efficient monitoring algorithm for a specified confidence threshold

4.1. Simpler formulae for satisfaction probabilities

Although it is conceptually simple to compute the satisfaction probability of the deadline constraint and that of the delay constraint from the Theorem 1 and Theorem 2, the formulae in those theorems are not easy to use in practice since they include MIN and MAX functions with integrals.

We can simplify and make the analysis more efficient by dividing our analysis into cases which are based on the configuration of the time intervals with respect to their deadlines (or delays), as illustrated in Figure 2. Throughout this subsection, we assume that we have a deadline constraint $c : I_1 + d \geq I_2$ where $d \geq 0$.

The cases corresponding to a division of the configuration space into regions are as follows: $\alpha = (max_1, min_2)^*$, $\beta = [min_2, max_2]$, and $\gamma = (max_2, \infty)$. Note that only region β includes the two end points. α is defined to be empty when $min_2 < max_1$. We say that a timing constraint c is in XY configuration whenever X is the region where $min_1 + d$ belongs and Y is the region where $max_1 + d$ belongs. For example, Figure 3 illustrates a timing constraint in the $\beta\gamma$ configuration. At first glance, there can be nine possible configurations, $\alpha\alpha$, $\alpha\beta$, $\alpha\gamma$, $\beta\alpha$, $\beta\beta$, $\beta\gamma$, $\gamma\alpha$, $\gamma\beta$, *and* $\gamma\gamma$. However, it is evident that the configurations $\beta\alpha$, $\gamma\alpha$, *and* $\gamma\beta$ are not permissible because it is always true that $min_1 + d < max_1 + d$. Two trivial cases are $\alpha\alpha$ and $\gamma\gamma$. In $\alpha\alpha$ configuration, the satisfaction probability is zero because there is no possibility that any pair of time points, each of which is from time interval I_1 and I_2 respectively, can satisfy the deadline. Likewise, for the $\gamma\gamma$ configuration, every pair of time points from respectively the intervals I_1 and I_2 must satisfy the deadline; therefore the satisfaction probability is 1.0 for the $\gamma\gamma$ configuration.

In what follows, we present simpler formulae for the satisfaction probability of the deadline constraint c in configurations $\alpha\beta$, $\alpha\gamma$, $\beta\beta$, and $\beta\gamma$.

$\alpha\beta$ configuration:

$$\frac{\int_{min_2-d}^{max_1}(x + d - min_2)dx}{len(I_1)len(I_2)}$$

$$= \frac{(min_2 - (d + max_1))^2}{2len(I_1)len(I_2)}$$

∗ For the sake of brevity, we use min_k to denote $min(I_k)$. For example, min_1 means $min(I_1)$. A similar notation max_k is used for $max(I_k)$.

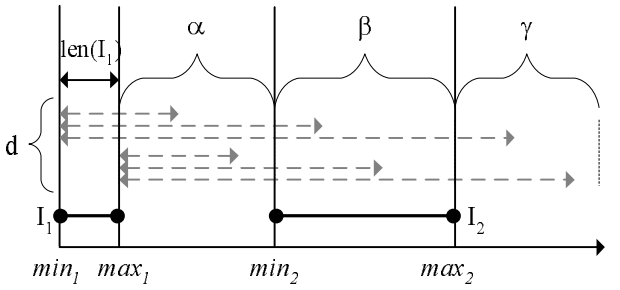


Figure 2. Regions

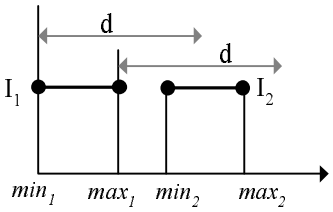


Figure 3. $\beta\gamma$ configuration

$\alpha\gamma$ configuration:

$$\frac{\int_{min_2-d}^{max_2-d}(x + d - min_2)dx + (max_1 - (max_2 - d))len(I_2)}{len(I_1)len(I_2)}$$

$$= \frac{2d + 2max_1 - min_2 - max_2}{2len(I_1)}$$

$\beta\beta$ configuration:

$$\frac{\int_{min_1}^{max_1}(x + d - min_2)dx}{len(I_1)len(I_2)}$$

$$= \frac{2d + max_1 + min_1 - 2min_2}{2len(I_2)}$$

$\beta\gamma$ configuration:

$$\frac{\int_{min_1}^{max_2-d}(x + d - min_2)dx + (max_1 - (max_2 - d))len(I_2)}{len(I_1)len(I_2)}$$

$$= 1 - \frac{(d - max_2 + min_1)^2}{2len(I_1)len(I_2)}$$

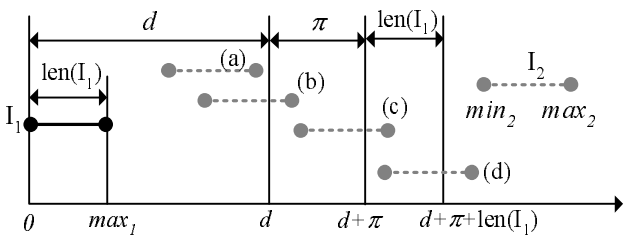
4.2. Efficient deadline monitoring

In this subsection, we design an efficient monitoring algorithm for a simple deadline timing constraint with a confidence threshold.

Throughout this paper, we shall assume for the sake of simplicity that we detect an event when its timestamp is determined; in other words, the detection time of an event instance with timestamp I is max(I).

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.



Deadline Satisfaction	100%	50%	0%
Delay Satisfaction	0%	50%	100%

Figure 4. Earliest expiration times (EET) from $min(I_1)$

Definition 9 *Consider a timing constraint c: $I_1 + d \geq I_2$ with P where $d \geq 0$. The earliest expiration time of a deadline timer from a time point t for c, $EET(t) \mid_c$, is the time difference between t and the earliest time t' when the event monitor can safely claim that c is violated in case the event instance corresponding to I_2 does not occur by t'.*

Knowledge of EET enables the event monitor to stop waiting for an event that is needed for the satisfaction of a timing constraint whenever the timing constraint is already violated by the time.

Figure 4 summarizes Theorem 3, Theorem 4, and Theorem 5, assuimg that min(I_1) is 0. For a deadline constraint $c_1 : I_1 + d \geq I_2$ where $d \geq 0$, each value to the right of the label Deadline Satisfaction in the figure denotes the maximum satisfaction probability which c_1 can achieve as long as $max(I_2)$ does not exceed the corresponding time values d, $d+\pi$†, $d+\pi+len(I_1)$. For example, the satisfaction probability of the timing constraint of case (b) is lower than 100%. If the monitored timing constraint requires 100% and the event instance corresponding to I_2 does not occur by d, the event monitor can claim that the timing constraint cannot be satisfied at time d without further waiting for the event instance. For a delay constraint $c_2 : I_2 + d \geq I_1$ where $d < 0$, each value to the right of the label Delay Satisfaction in the figure denotes the minimum satisfaction probability which c_2 can achieve if $max(I_2)$ exceeds the corresponding time point d, $d + \pi$, $d + \pi + len(I_1)$. For example, the satisfaction probability of the timing constraint of case (c) is at least 50%.

Theorem 3 *Given a timing constraint c: $I_1 + d \geq I_2$ with P where $P = 100\%$ and $d \geq 0$, we have $EET(min(I_1)) \mid_c = d$.*
Proof:
$EET(min(I_1)) \mid_c$ cannot be smaller than d because $SP \mid_c = 100\%$ always when $max(I_2) = d + min(I_1)$. In case $max(I_2) > d + min(I_1)$, the satisfaction probability cannot be 100%.

□

† π is the maximum length of the timestamp in the system. Refer to Definition 5 for details.

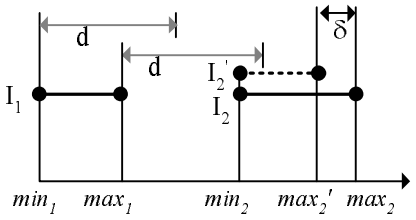


Figure 5. Shortening a time interval in $\alpha\beta$ configuration

Theorem 4 *Given a timing constraint c: $I_1 + d \geq I_2$ with P where $0\% < P < 50\%$ and $d \geq 0$, we have $EET(min(I_1)) \mid_c = d + len(I_1) + \pi$.*
Proof:
$EET(min(I_1)) \mid_c$ cannot be smaller than $d + len(I_1) + \pi$ because there exists a case where $max(I_2) = d + len(I_1) + \pi$, $min(I_2) = d + len(I_1)$ and $SP \mid_c > 0\%$. In case $max(I_2) > d+len(I_1)+\pi+min(I_1)$, $SP \mid_c = 0\%$ holds always.

□

Lemma 1 and Lemma 2 are needed to prove that the satisfaction probability of a deadline constraint in $\alpha\beta$ configuration is maximized when the timestamps of the corresponding event instances are spanned as shown in Figure 6.

Lemma 1 *Suppose that we have a deadline constraint, $c : I_1 + d \geq I_2$ where $d \geq 0$ and c is in $\alpha\beta$ configuration. Let p be the satisfaction probability of c. While preserving the configuration $\alpha\beta$, moving max_2 towards I_1 by decreasing the value of max_2 will result in $p' > p$ where p' is the satisfaction probability of the modified constraint.*
Proof:
Suppose we have a time interval I_2' for which max_2' is smaller than max_2 by δ as illustrated in Figure 5.

$$p = \frac{(min_2 - (d + max_1))^2}{2len(I_1)len(I_2)}$$

$$p' = \frac{(min_2 - (d + max_1))^2}{2len(I_1)(len(I_2) - \delta)}$$

$$p' - p = \frac{\delta(min_2 - (d + max_1))^2}{2len(I_1)len(I_2)len(I_2')} > 0$$

because $\delta > 0$, $(min_2 - (d + max_1))^2 > 0$ (by $min_2 \leq d + min_1 < d + max_1$), $len(I_1) > 0$, $len(I_2) > 0$, and $len(I_2') > 0$.

□

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

Lemma 2 *Suppose that we have a deadline constraint, c : $I_1 + d \geq I_2$ where $d \geq 0$ and c is in $\alpha\beta$ configuration. Let p be the satisfaction probability of c. While preserving the configuration $\alpha\beta$, moving min_2 towards I_1 by decreasing the value of min_2 will result in $p' > p$, where p' is the satisfaction probability of the modified constraint.*
Proof:
Proof is similar to that of Lemma 1.

Theorem 5 *Given a timing constraint c: $I_1 + d \geq I_2$ with P where $50\% \leq P < 100\%$ and $d \geq 0$, we have $EET(min(I_1)) \mid_c = d + \pi$.*
Proof:
The theorem we need to prove can be rewritten as follows: In case $max_2 > d + \pi + min_1$, the satisfaction probability of the timing constraint is lower than 50%.
Since the configurations $\alpha\gamma$, $\beta\gamma$, and $\gamma\gamma$ require $max_2 < d + \pi + min_1$, they cannot satisfy the condition of the theorem. Case $\alpha\alpha$ trivially satisfies the theorem because its satisfaction probability is zero. Case $\beta\beta$ requires the condition $d + min_1 \geq min_2$ which cannot satisfy $max_2 > min_1 + d + \pi$, since $max(I) - min(I) \leq \pi$ holds for any time interval I. Therefore, we only need to prove this theorem for configuration $\alpha\beta$.

By Lemma 1 and Lemma 2, the satisfaction probability of the case: $min_2 = min_1 + d + \varepsilon$ and $max_2 = min_1 + d + len(I_1)$ where ε is a very small positive value, is the maximum in the configuration $\alpha\beta$ as illustrated in Figure 6.

The maximum satisfaction probability of a timing constraint in configuration $\alpha\beta$ is

$$\begin{aligned} & \frac{(min_2 - (d + max_1))^2}{2len(I_1)len(I_2)} \\ = & \frac{(d + \varepsilon + min_1 - (d + max_1))^2}{2(max_1 - min_1)(max_1 - min_1 - \varepsilon)} \\ = & \frac{max_1 - min_1 - \varepsilon}{2(max_1 - min_1)} \\ = & \frac{1}{2} - \frac{\varepsilon}{2len(I_1)} < \frac{1}{2} \end{aligned}$$

□

Theorem 6 *The condition $c_1 : min(I_1) + d \geq max(I_2)$ is equivalent to the timing constraint $c_2 : I_1 + d \geq I_2$ with 100% confidence.*
Proof:
The satisfaction of c_1 implies the satisfaction of c_2 and vice versa. The violation of c_1 implies the violation of c_2 and vice versa.

□

Corollary 2 *Consider a timing constraint c: $I_1 + d \geq I_2$ with P where $d \geq 0$. Assuming that $len(I_1)$ is unknown,*

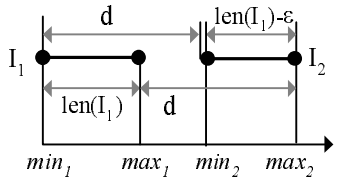


Figure 6. Maximum satisfaction probability in $\alpha\beta$ configuration

$EET(max(I_1)) \mid_c$ is given by:

$$EET(max(I_1)) \mid_c = \begin{cases} d + \pi & \text{if } P < 100\% \\ d & \text{otherwise} \end{cases}$$

Proof:
This follows from Theorem 3, Theorem 4, and Theorem 5.

□

5. Deriving implicit constraints

The ability to detect a timing constraint violation as early as possible at run time can be a critical requirement for some real-time systems. For early detection, we need to derive implicit constraints from the given timing constraint set. Various derivation algorithms for implicit constraints from a set of timing constraint on time points were discussed in [2, 14]. A derivation algorithm where the timestamps of events are time intervals was presented in [13]. We now extend the algorithm in [13] to cover the case where the timing constraints have confidence threshold by incorporating the findings from the previous section.

Before detailing the derivation of implicit constraints from timing constraints on time intervals with confidence thresholds, let us use an example to illustrate how we can derive an implicit constraint from timing constraints on time points and why they are useful in detecting a violation.

Example 3 *For this example only, we assume that the timestamp of an event is a time point and not a time interval, and we do not consider modalities. Suppose we have the following two timing constraints.*

$$\begin{aligned} @(e1, i) + 100 & \geq @(e2, i) \\ @(e2, i) - 95 & \geq @(e3, i) \end{aligned}$$

We can derive an implicit constraint $(e1, i) + 5 \geq @(e3, i)$ by simply merging two timing constraints. This derived implicit constraint can help us to detect a timing violation as early as possible. Suppose that @(e1,i) occurs at time 0. If the event instance $(e3, i)$ does not happen until time=5, then we can conclude that at least one of original timing constraints will be violated eventually.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

If we do not exploit the implicit constraint, we would have to wait until time=100 to catch a violation of the first timing constraint in case (e2,i) does not occur by that time.

5.1. Deriving an implicit constraint

Definition 10 *A violation of an implicit constraint occurs if at least one of the explicit timing constraints from which the implicit constraint is derived is also violated eventually.*

The following theorem derives from a set of timing constraints, an implicit constraint which satisfies the violation definition of implicit constraints.

Theorem 7 *Suppose we have a set of n timing constraints on time intervals with confidence thresholds as follows:*

$$\begin{aligned} c_1 : I_1 + d_1 &\geq I_2 \text{ with } P_1 \\ c_2 : I_2 + d_2 &\geq I_3 \text{ with } P_2 \\ c_3 : I_3 + d_3 &\geq I_4 \text{ with } P_3 \\ &\cdot \\ &\cdot \\ c_{n-1} : I_{n-1} + d_{n-1} &\geq I_n \text{ with } P_{n-1} \\ c_n : I_n + d_n &\geq I_{n+1} \text{ with } P_n \end{aligned}$$

where I_k is a time interval where $1 \leq k \leq n+1$. d_k is a deadline or a delay, and P_k is a confidence threshold where $1 \leq k \leq n$. Then the implicit constraint between I_1 and I_{n+1} from c_1, c_2, c_3, ..., c_n is given by:

$$I_1 + \sum_{k=2}^{n} d_k + m\pi + EET(min(I_1)) \mid_{c_1} \geq I_{n+1} \text{ with } 100\%$$

where m is the number of $P_k \neq 100\%$ where $2 \leq k \leq n$.
Proof (by induction):
Base Step: Suppose that the number of timing constraints is one, i.e., $n = 1$. In case $P_1 = 100\%$, the constraint c_1 requires a condition $min(I_1)+d_1 \geq max(I_2)$. By Theorem 6, the equivalent constraint c_i to this condition is $I_1 + d_1 \geq I_2$ with 100%. In case $50\% \leq P_1 < 100\%$, the constraint c_1 requires a condition $min(I_1) + d_1 + \pi \geq max(I_2)$. By Theorem 6, the equivalent constraint c_i to this condition is $I_1 + d_1 + \pi \geq I_2$ with 100%. In case $P_1 < 50\%$, the constraint c_1 requires a condition $min(I_1)+d_1+\pi+len(I_1) \geq max(I_2)$. By Theorem 6, the equivalent constraint c_i to this condition is $I_1 + d_1 + \pi + len(I_1) \geq I_2$ with 100%. In all cases, the constraint c_i from each case is the same as the implicit constraint, $I_1 + EET(min(I_1)) \mid_{c_1} \geq I_2$ with 100% , which is the result from this theorem.
Suppose that this theorem holds when the number of timing constraints is no bigger than $n-1$.
Induction Step: We consider the case when the number of timing constraints is n. Suppose m' is the number of $P_k \neq 100\%$ where $2 \leq k \leq n-1$. By the induction hypothesis, the implicit constraint from $c_1, c_2, ..., c_{n-1}$ is $I_1 + \sum_{k=2}^{n-1} d_k + m'\pi + EET(min(I_1)) \mid_{c_1} \geq I_n$ with 100%. The implicit constraint requires the condition q_1: $min(I_1)+\sum_{k=2}^{n-1} d_k + m'\pi + EET(min(I_1)) \mid_{c_1} \geq max(I_n)$.
If $P_n = 100\%$, c_n requires the condition q_2: $max(I_n) + d_n \geq max(I_{n+1})$. By merging q_1 and q_2, we get $min(I_1)+\sum_{k=2}^{n} d_k + m'\pi + EET(min(I_1)) \mid_{c_1} \geq max(I_n)$, which is equivalent to the timing constraint $c_i = I_1 + \sum_{k=2}^{n} d_k + m'\pi + EET(min(I_1)) \mid_{c_1} \geq I_n$ with 100% by Theorem 6.
If $P_n < 100\%$, c_n requires the condition q_2 : $max(I_n) + d_n + \pi \geq max(I_{n+1})$. By merging q_1 and q_2, we get $min(I_1) + \sum_{k=2}^{n} d_k + (m'+1)\pi + EET(min(I_1)) \mid_{c_1} \geq max(I_n)$, which is equivalent to $c_i = I_1 + \sum_{k=2}^{n} d_k + (m'+1)\pi + EET(min(I_1)) \mid_{c_1} \geq I_n$ with 100% by Theorem 6.
In all cases, the constraint c_i from each case is equivalent to the implicit constraint calculated from this theorem.

□

5.2. Using all-pairs shortest path algorithm

In order to facilitate the derivation of the shortest implicit constraints from a set of timing constraints, an efficient technique is to use an all-pairs shortest-path algorithms as the core engine, e.g., [14, 15, 13]. Those solutions are based on the Floyd-Warshall algorithm because of its simplicity of implementation and the efficiency benefits from matrix-based operations. Our proposed algorithm is also extended from the Floyd-Warshall algorithm.

We introduce a data type SPATH to store information about the shortest path identified up to the current point and a merge operator $\oplus$ which combines two SPATH instances into a single SPATH instance.

The SPATH type data structure has the following three attributes.

dsum a summation of deadlines and delays of the timing constraints in the shortest path.

count the number of constraints for which confidence thresholds are less than 100% in the shortest path.

belowh initialized to 1 if $P_1 < 50\%$ and initialized to 0 otherwise where P_1 is the confidence threshold of the first constraint in the shortest path.

Definition 11 *The merge operator $\oplus$ is defined as follows.*

$$\begin{aligned} &SPATH(dsum_1, count_1, belowh_1) \\ &\oplus SPATH(dsum_2, count_2, belowh_2) \\ &= SPATH(dsum_1 + dsum_2, count_1 + count_2, belowh_1) \end{aligned}$$

Notice that only $belowh_1$ is copied into the merged SPATH instance in combining two SPATH instances. This is be-

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

cause we need to keep the information whether the confidence threshold of the first constraint is below 50% as indicated in Theorem 7.

As usual, we assume that the set of timing constraints from which we want to derive implicit constraints are represented as a directed graph (the constraint graph) where each timestamp in the timing constraints is represented by a vertex. Together with its confidence threshold, a delay or a deadline becomes a weighed edge.

Example 4 *Figure 7(A) shows a graph corresponding to the following set of timing constraints.*

$$\begin{aligned} @(e1,i)+150 &\geq @(e2,i) \textit{ with 25\%} \\ @(e2,i)-70 &\geq @(e3,i) \textit{ with 100\%} \\ @(e3,i)+100 &\geq @(e4,i) \textit{ with 75\%} \\ @(e2,i)+50 &\geq @(e4,i) \textit{ with 20\%} \end{aligned}$$

Corollary 3 *Suppose we derive an implicit constraint c_1 : $I_1 + d_1 \geq I_2$ with 100%, for the timestamp I_1 and I_2 by Theorem 7. If we have another implicit constraint c_1' : $I_1 + d_1' \geq I_2$ with 100% where $d_1 \geq d_1'$, then c_1 is unnecessary.*
Proof:
The satisfaction of c_1' always implies the satisfaction of c_1 and the violation of c_1 always implies the violation of c_1'. Therefore, c_1 is unnecessary.

□

Corollary 4 *Suppose we derive an implicit constraint c_1 : $I_1 + d_1 + len(I_1) \geq I_2$ with 100%, for the timestamp I_1 and I_2 by Theorem 7. If we have another implicit constraint c_1' : $I_1 + d_1' + len(I_1) \geq I_2$ with 100% where $d_1 \geq d_1'$, then c_1 is unnecessary.*
Proof:
Proof can be done similarly as in Corollary 3.

□

```
AllPairsShortestPath(n, d, P)
1.  for i = 1 to n
2.    for j = 1 to n
3.      D^0_ij = SPATH(∞, 1, 0)
4.      D'^0_ij = SPATH(∞, 1, 1)
5.      if d_ij ≠ ∞
6.        if (P_ij = 100%) D^0_ij = SPATH(d_ij, 0, 0)
7.        else if (P_ij ≥ 50%) D^0_ij = SPATH(d_ij, 1, 0)
8.        else D'^0_ij = SPATH(d_ij, 1, 1)
9.  for k = 1 to n
10.   for i = 1 to n
11.     for j = 1 to n
12.       newpath = D^{k-1}_ik ⊕ MinCost(D^{k-1}_kj, D'^{k-1}_kj)
13.       newpath' = D'^{k-1}_ik ⊕ MinCost(D^{k-1}_kj, D'^{k-1}_kj)
14.       D^k_ij = MinCost(newpath, D^{k-1}_ij)
15.       D'^k_ij = MinCost(newpath', D'^{k-1}_ij)
```

```
MinCost(SPATH a, SPATH b)
1. if ((b.dsum + b.count * π) < (a.dsum + a.count * π))
   return b
2. else return a
```

In the algorithm, n is the number of vertices in the constraint graph, d_{ij} is a deadline or delay from V_i to V_j, and P_{ij} is a confidence threshold from V_i to V_j. It is clear that the run-time complexity of the all pairs shortest path algorithm presented above is O(n^3). As explained in Corollary 3 and Corollary 4, two SPATH instances D_{ij} and D'_{ij} are maintained for each pair of vertices. D_{ij} keeps track of the implicit constraint of which the confidence threshold of the first constraint is at least 50%. The confidence threshold of the first constraint of the implicit constraint in D'_{ij} is less than 50%.

The $MinCost$ function returns a SPATH instance of which cost is the minimum among the arguments to the function. The reader may wonder why only the $dsum$ and $count$ attributes are considered in comparing the cost of two SPATH instances in the $MinCost$ function. The calls to the $MinCost$ function in line 12 and line 13 in $AllPairsShortestPath$ function are valid because the constraints represented by neither D^k_{kj} nor D'^k_{kj} will be used as a header of the shortest path. By Theorem 7, the confidence threshold of only the first constraint is needed for calculating an implicit constraint. Since both $belowh$ in $newpath$ and D^k_{ij} should be 0 by program construction, we do not need to consider $belowh$ in the comparison. Similarly, both $belowh$ in $newpath'$ and D'^k_{ij} should be 1 by program construction, and there is no need to consider $belowh$ in the comparison.

At the end of the execution of the algorithm, each D^n_{ij} and D'^n_{ij} contains information about the shortest implicit constraint from vertex i to vertex j in the form of SPATH($dsum$, $count$, $belowh$) which represents an implicit constraint:

$$I_i + dsum + count * \pi + belowh * len(I_i) \geq I_j \text{ with } 100\%$$

where I_i is the timestamp of the event corresponding to vertex i and I_j is the timestamp of the event corresponding to vertex j.

The value $belowh$ in D'^n_{ij} is always 1, which means that the shortest path information contained in D'^n_{ij} depends on the length of the time interval of the event instance corresponding to the vertex i. Figure 7(B) shows all implicit constraints calculated from the set of timing constraints shown in Example 4. The derived implicit constraints are represented in the triple of ($dsum$, $count$, $belowh$). In this example, π is assumed to be 10.

Since for a pair of vertices, at most two implicit constraints can be derived from the algorithm, there can be up to three constraints for a pair of vertices if we also count the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

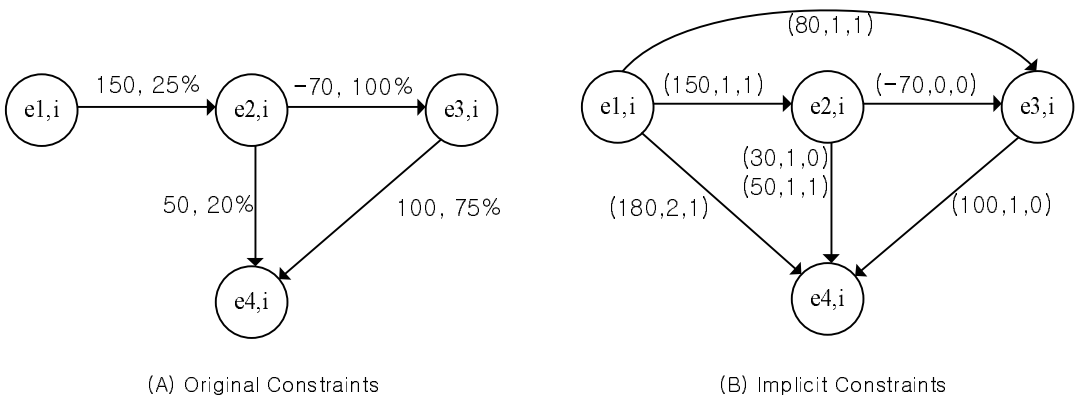


Figure 7. A graph representation

original constraint. In general, we can discard timing constraints that can be covered by other timing constraints. For example, we have following three timing constraints from vertex $(e2, i)$ to vertex $(e4, i)$:

(a) $@(e2, i) + 50 \geq @(e4, i)$ with 20%
(b) $@(e2, i) + 50 + 10 + len(@(e2, i)) \geq @(e4, i)$ with 100%
(c) $@(e2, i) + 30 + 10 \geq @(e4, i)$ with 100%

We can discard (a) in favor of (c) because the satisfaction of (c) always means the satisfaction of (a) and the violation of (a) means the violation of (c). Similarly, (b) can also be discarded in favor of (c).

6. Conclusion

In this paper, we have extended our previous work in [13] to allow users to specify a quantitative measure on the degree of certainty with which timing satisfaction/violation can be detected. To this end, we introduce a *confidence threshold* which is included in the specification of a timing constraint to specify the desired degree of certainty, ranging from 0% to 100%. In case the confidence threshold of a timing constraint cannot be inferred from the timestamps of an observed event trace, a violation of the timing constraints is issued. For computational purposes, we define and derive the concept of satisfaction probabilities for both the deadline constraint and the delay constraint. The all-pairs shortest-path algorithm is extended to facilitate the derivation of implicit constraints. A non-obvious result from our analysis is that the earliest expiration time (EET) of the deadline timers for timing constraints with $P = 100\%$, $50\% \leq P < 100\%$, and $0\% < P < 50\%$ are all different. For future work, we plan to investigate more exact bounds of the EET for each confidence threshold value so that we can extend the pruning techniques to more efficiently discard unnecessary constraints after deriving the implicit constraints.

References

[1] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S. K. Kim. Composite events for active databases: semantics, contexts and detection. In *Proc. of the International Conference on Very Large Data Bases(VLDB)*, pages 606–617, August 1994.

[2] S. E. Chodrow, F. Jahanian, and M. Donner. Run-time monitoring of real-time systems. In *Proc. of IEEE Real-Time Systems Symposium(RTSS)*, pages 74–83, December 1991.

[3] F. Cristian. Understanding fault-tolerant distributed systems. Technical report, Technical Report RJ 6980, IBM, April 1990.

[4] A. Galton. Time and change for ai. In D. Gabbay, C. Hogger, and J.Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4 (Epistemic and Temporal Reasoning), pages 175–240. Clarendon Press, 1995.

[5] S. Gatziu, A. Geppert, and K. Dittrich. Detecting Composite Events in Active Database Systems Using Petri Nets. In *Proc. of the 4th International Workshop on Research Issues in Data Engineering*, pages 2–9, Feburary 1994.

[6] N. H. Gehani, H. Jagadish, and O. Shmueli. Composite Event Specification in Active Databases: Model and Implementation. In *Proc. of the International Conference on Very Large Data Bases(VLDB)*, pages 327–338, August 1992.

[7] F. Jahanian, R. Rajkumar, and S. Raju. Run-time monitoring of timing constraints in distributed real-time systems. Technical report, Technical Report CSE-TR 212-94, University of Michigan, April 1994.

[8] M. Kim, I. Lee, U. Sammapun, J. Shin, and O. Sokolsky. Monitoring, checking, and steering of real-time systems. *Electronic Notes in Theoretical Computer Science*, 70(4), 2002.

[9] P. Konana, A. K. Mok, C.-G. Lee, H. Woo, and G. Liu. Implementation and performance evaluation of a real-time e-

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.

brokerage system. In *Proc. of IEEE Real-Time Systems Symposium(RTSS)*, pages 109–118, December 2000.

[10] I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan. Runtime Assurance Based On Formal Specifications. In *Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 56–65, July 1999.

[11] C. Liebig, M. Cila, and A. Buchmann. Event Composition in Time-dependent Distributed Systems. In *Proc. of the 4th IFCIS International Conference on Cooperative Information Systems (CoopIS 99)*, pages 70–78. IEEE Computer Society, 1999.

[12] M. Mansouri-Samani and M. Sloman. GEM: A Generalized Event Monitoring Language for Distributed Systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(2):96–108, June 1997.

[13] A. K. Mok, C.-G. Lee, H. Woo, and P. Konana. The Monitoring of Timing Constraints on Time Intervals. In *Proc. of IEEE Real-Time Systems Symposium(RTSS)*, pages 191–200, December 2002.

[14] A. K. Mok and G. Liu. Early Detection of Timing Constraint Violation at Runtime. In *Proc. of IEEE Real-Time Systems Symposium(RTSS)*, pages 176–185, December 1997.

[15] A. K. Mok and G. Liu. Efficient Runtime Monitoring of Timing Constraints. In *Proc. of Real-Time Technology and Applications Symposium(RTAS)*, pages 252–262, June 1997.

[16] P. R. Pietzuch, B. Shand, and J. Bacon. A Framework for Event Composition in Distributed Systems. In *Proc. of the 4th ACM/IFIP/USENIX Int. Conf. on Middleware (Middleware '03)*, pages 62–82. Springer, June 2003.

[17] C. L. Roncancio. Toward Duration-Based, Constrained and Dynamic Event Types. *Lecture Notes in Computer Science. Proc. of the Second International Workshop on Active, Real-Time and Temporal Database Systems (ARTDB'97)*, 1553:176–193, 1997.

[18] S. Schwiderski. *Monitoring the behaviour of distributed systems*. PhD thesis, University of London, 1996.

[19] P. Terenziani. Is point-based semantics always adequate for temporal databases. In *7th International Workshop on Temporal Representation and Reasoning (TIME-00)*, pages 191–199, 2000.

APPENDIX

In this appendix, we validate the computational method for deriving the satisfaction probability in Theorem 1 and the result in Theorem 5 by simulation. All of the simulation programs are written in Java and use java.util.Random package which contains a uniform random number generator.

In order to verify the correctness of the satisfaction probability of the deadline constraint in Theorem 1, we generated randomly 500 timing constraints $c_k : I_k + d_k \geq I'_k$ where $1 \leq k \leq 500$. For each timing interval pair of (I_k, I'_k), we generated randomly 10000 pairs of time points ($t_k^m, t_k'^m$) where $min(I_k) \leq t_k^m \leq max(I_k)$ and $min(I'_k) \leq t_k'^m \leq max(I'_k)$ and $1 \leq m \leq 10000$. Given a timing constraint c_k and pairs of time points ($t_k^1, t_k'^1$), ($t_k^2, t_k'^2$), ..., ($t_k^{10000}, t_k'^{10000}$), the actual satisfaction ratio ASR_k is determined as follows:

$$\frac{\text{The number of pairs } (t_k^m, t_k'^m) \text{ satisfying } t_k^m + d_k \geq t_k'^m}{10000}$$

where $1 \leq m \leq 10000$.

Given a timing constraint c_k, the calculated probability CP_k is determined simply by $SP\,|_{c_k}$ in Theorem 1.

Figure 8 shows a result where a point (CP_k, ASR_k) is plotted for $1 \leq k \leq 500$. Since CP_k is an anticipated probability by Theorem 1 and ASR_k is the actual satisfaction probability from the event simulation, the resulting straight line $y = x$ from the simulation agrees with Theorem 1.

With a similar setup, we calculated the satisfaction ratio for each configuration. Table 1 shows the minimum, the maximum, and the average satisfaction ratio for each configuration. As stated in Theorem 5, the maximum satisfaction ratio of $\alpha\beta$ configuration does exceed 50%.

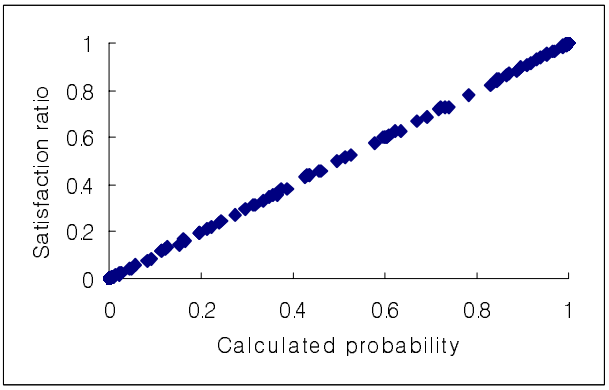


Figure 8. Calculated probability v.s. actual satisfaction ratio

	$\alpha\alpha$	$\alpha\beta$	$\beta\beta$	$\alpha\gamma$	$\beta\gamma$	$\gamma\gamma$
min	0	0	0.005	0.008	0.502	1
max	0	0.498	0.994	0.997	1	1
avg	0	0.112	0.491	0.505	0.885	1

Table 1. Actual satisfaction ratio in each configuration

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on April 30, 2009 at 10:55 from IEEE Xplore. Restrictions apply.