Sixth Int. Conf. on IFSs, Varna, 13-14 Sept. 2002
NIFS 8 (2002), 3, 95-100

Intuitionistic Fuzzy Generalized Nets in Analyzing Transaction Database Systems with Continuous Deadlock Detection

Boyan Kolev[1], Panagiotis Chountas[2], Ilias Petrounias[3]

[1] - Centre for Biomedical Engineering - Bulgarian Academy of Sciences,
Acad.G.Bonchev Str., Bl.105, Sofia-1113, BULGARIA, e-mail: bobby_kolev@yahoo.co.uk
[2] - Mechatronics Group, Dept. of Computer Science, Univ. of Westminster,
London, HA1 3TP, UK, e-mail: chountp@wmin.ac.uk
[3] - Department of Computation UMIST, Manchester PO BOX 88 M60 1QD, UK

Abstract

This paper presents an intuitionistic fuzzy generalized net model of a transaction database system with continuous deadlock detection, which uses the 2PL protocol. We define probabilities for a transaction to be granted a requested lock, held back by another transaction or deadlocked, which are integrated with the intuitionistic fuzzy predicates. We can use this model to simulate transaction processing and to analyze the efficient time for useful work and the time wasted in holding back transactions.

1. INTRODUCTION

The best known protocol that guarantees serializability in executing transactions is two-phase locking (2PL) protocol. It concerns the positioning of the lock and unlock operations in every transaction. A transaction follows the 2PL protocol if all locking operations precede the first unlocking operation in the transaction [1]. Every transaction using this protocol can be divided into two phases:

- a growing phase, in which it acquires all the locks but cannot release any lock and
- a shrinking phase, in which it releases all locks but cannot acquire any new lock.

Using the 2PL protocol, however, we can cause a "deadlock" situation – two or more transactions are each waiting for locks held by the other to be released. The most common strategy for detecting deadlock situations is using the wait-for-graph (WFG), which describes the transaction dependencies. In the WFG each node corresponds to a transaction and a directed edge from node T_i and T_j exists if transaction T_i is waiting to lock an item that is currently locked by T_j. In this construction deadlock exists if and only if the WFG contains a cycle. The WFG could be checked continuously or periodically. With continuous deadlock detection the WFG is checked at each blocking of a transaction (after adding an edge to the graph). This algorithm is not so efficient for database systems, in which deadlocks occur rarely. With periodic detection deadlocks can be checked periodically after adding a few edges to the graph.

We should differentiate between lock-owners (transactions that have performed at least one lock) and non-lock-owners, because a lock owner is a potential participant in deadlock situation.

In [2] Ing-Ray Chen described the locking processes in a transaction database system with 2PL protocol using stochastic Petri nets (SPN). Now we can describe them with intuitionistic fuzzy generalized nets (IFGN, see [3]). In an IFGN, the probabilities in the SPN can be transformed to intuitionistic fuzzy predicates, which let the tokens move from one place to another. We can also assign characteristics to each token, thus making the computation of probabilities more dynamic.

2. CONTINUOUS DEADLOCK DETECTION ALGORITHM

First we will build the state machine that describes the process for one transaction. The state machine is described with the graph depicted in *Fig. 1*

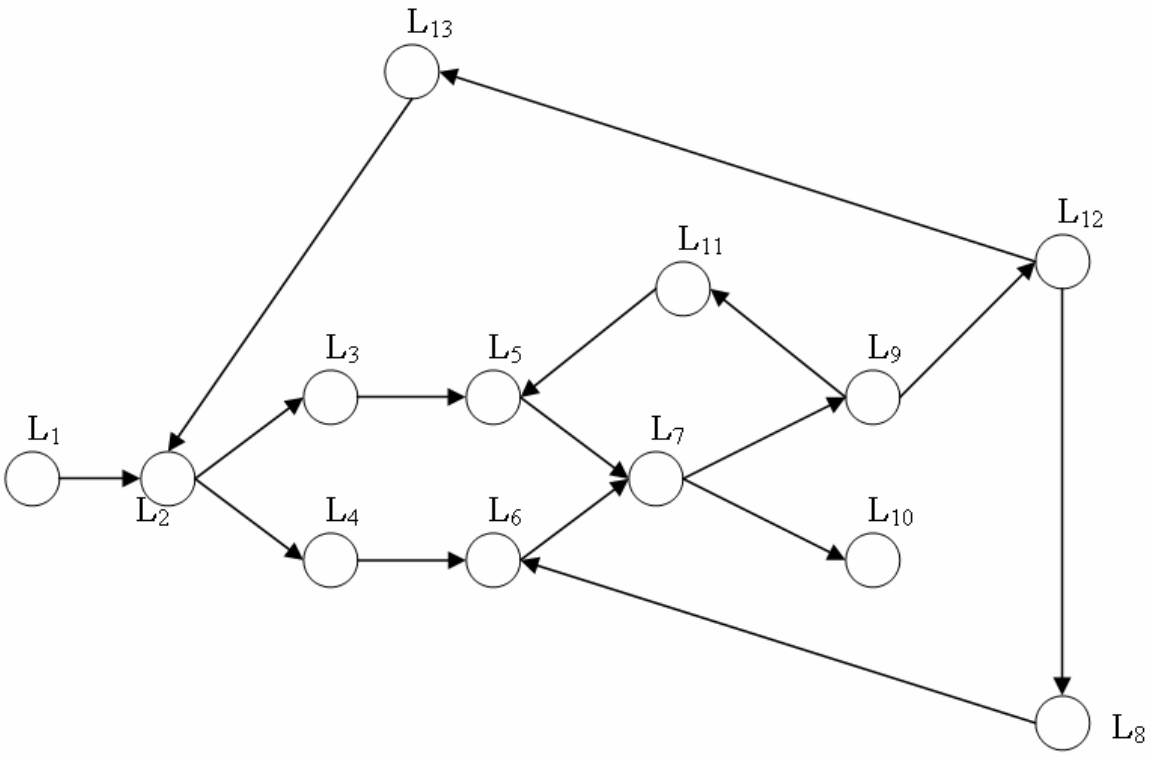

Fig. 1

The meanings of the states used in the state machine are as follows:

L_1 = the transaction is started
L_2 = the transaction is requesting its first lock
L_3 = the requested lock is granted
L_4 = the transaction is blocked
L_5 = the transaction is retrieving data items after the lock is granted
L_6 = the transaction is retrieving data items after the lock is granted
L_7 = the transaction is doing useful work

L_9 = the transaction is requesting its subsequent lock
L_{10} = the transaction is releasing all of the locks and committing
L_{11} = the subsequent requested lock is granted
L_{12} = executing a deadlock detection algorithm
L_{13} = deadlock – the transaction is restarted
L_8 = the transaction is blocked after a subsequent lock request

We should define some local (for each transaction) and global (for the entire system) characteristics, which are used in computing the probabilities for some events, as well as some time characteristics, which determine how long the transaction should stay in a certain state.

- Local characteristics:
  - *NL* (number of locks) – the total number of locks for the transaction
  - *LC* (lock counter) – the current number of locks (incremented at each lock request)
- Global characteristics:
  - N – total number of transactions
  - *NLO* – number of lock owners
  - NL_{db} – the total number of locks for the database
  - NL_{LO} – the total number of locks for all lock owners
- Time characteristics:
  - T_{lreq} – service demand of CPU for processing a lock request
  - T_{lset} – service demand of CPU for setting a lock
  - T_{CPU} – service demand of CPU per visit by a transaction
  - T_{lrel} – service demand of CPU for releasing a lock
  - T_{dm} – service demand of data manager per visit by a transaction
  - T_{node} – service demand of CPU for checking a node in the WFG
  - T_{lock} – wait time for a lock by a blocked transaction (how long a transaction should wait for a lock to be granted)
- Probabilities:
  - P_1 – probability that when a (non-lock-owner) transaction requests its first lock, the lock is granted
  - P_2 – probability that when a (lock-owner) transaction requests a subsequent lock, the lock is granted
  - P_d – probability of deadlock

The probabilities for moving from one state to another are as follows:

- From L_2 to L_3 – probability is P_1
- From L_2 to L_4 – probability is $1 - P_1$
- From L_9 to L_{11} – probability is P_2
- From L_9 to L_{12} – probability is $1 - P_2$
- From L_{12} to L_{13} – probability is P_d
- From L_{12} to L_8 – probability is $1 - P_d$

3. THE IFGN MODEL

Now we can create the topological structure of the IFGN model, using the algorithm described in [4] (*Fig. 2*)

The tokens enter the net at place L_1 with characteristics *NL* and *LC*. Each token corresponds to a transaction in the database system. *NL* and *LC* mean respectively number of locks and lock counter. *LC* is initialized with 0 and is incremented at each lock request. We should consider the characteristics *NL* and *LC* as input parameters as well as all time characteristics. All other parameters are computed as follows:

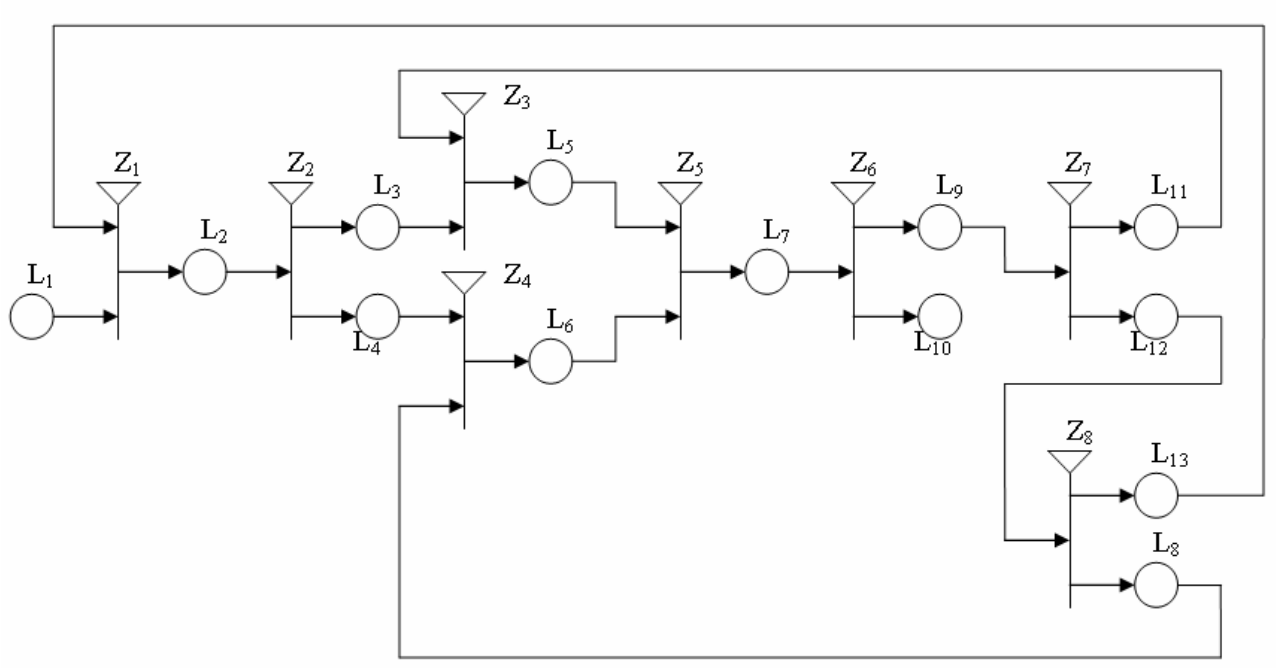

Fig. 2

(1) N is the total number of tokens in all places of the IFGN.
(2) NLO is the total number of tokens in places L_5, L_6, L_7, L_8, L_9, L_{11} and L_{12}
(3) NL_{db} is the sum of the characteristics *NL* for all tokens in the IFGN
(4) NL_{LO} is the sum of the characteristics *LC* for all tokens in the IFGN
(5) $P_1 = (NL_{db} - NLO) / NL_{db}$
(6) $P_2 = (NL_{db} - NL_{LO}) / (NL_{db} - LC)$
(7) $P_d = (1 - P_2^{LC-1}) / (NLO - 1)$

In place L_2 the transaction is requesting its first lock, where the token should stay there T_{lreq} time units and then move to place L_3 if the lock is granted (with probability P_1) or to place L_4 if it isn't (with probability $1\text{-}P_1$). After leaving L_2 the characteristic *LC* of the token is incremented. The intuitionistic fuzzy predicate matrix for transition Z_2 is the following:

$$r_2 = \begin{array}{c|cc} & L_3 & L_4 \\ \hline L_2 & W_1 & \neg W_1 \end{array},$$

where the intuitionistic fuzzy predicate W_1 is:

(8) $W_1 = (P_1, 1\text{-}P_1)$.

After L_7 a token should move either to L_9 (if it needs another lock) or to L_{10} (if it has no more locks to request for). The predicate matrix for transition Z_6 is the following:

$$r_6 = \begin{array}{c|cc} & L_9 & L_{10} \\ \hline L_7 & W_3 & \neg W_3 \end{array},$$

where the predicate W_3 is:

(9) $W_3 = true$ if $LC < NL$.

After L_9 a token should move either to L_{11} (if the requested lock is granted) or to L_{12} (if it isn't). Then the characteristic LC of the token is incremented. The intuitionistic fuzzy predicate matrix for transition Z_7 is the following:

$$r_7 = \begin{array}{c|cc} & L_{11} & L_{12} \\ \hline L_9 & W_2 & \neg W_2 \end{array},$$

where the intuitionistic fuzzy predicate W_2 is:

(10) $W_2 = (P_2, 1\text{-}P_2)$.

After L_{12} a token should move either to L_{13} (if the transaction is deadlocked) or to L_8 (if it isn't). The intuitionistic fuzzy predicate matrix for transition Z_8 is the following:

$$r_8 = \begin{array}{c|cc} & L_{13} & L_8 \\ \hline L_{12} & W_d & \neg W_d \end{array},$$

where the intuitionistic fuzzy predicate W_d is:

(11) $W_d = (P_d, 1\text{-}P_d)$.

We should complicate the predicates in order to allow the tokens to stay in some of the places for a certain period of time. The times for each place are as follows:

- in L_2 a token stays T_{lreq} time units
- in L_3 a token stays T_{lset} time units
- in L_4 a token stays T_{lock} time units
- in L_5 a token stays T_{dm} time units
- in L_6 a token stays T_{dm} time units
- in L_7 a token stays T_{CPU} time units
- in L_9 a token stays T_{lreq} time units
- in L_{10} a token stays $NL.T_{lrel}$ time units
- in L_{11} a token stays T_{lset} time units
- in L_{12} a token stays T_{dl} time units, where T_{dl} is the service demand of CPU for executing a continuous deadlock detection algorithm ($T_{dl} = N.T_{node}$)
- in L_8 a token stays T_{lock} time units
- in L_{13} a token stays $LC.T_{lrel}$ time units

4. CONCLUSION

This model gives us the possibility to analyze the behavior of a transaction database system using two-phase locking protocol with continuous deadlock detection. We can define some performance characteristics, which are easily computable using the model:

- X_t – throughput of terminating transactions, i.e. number of successfully terminated transactions per time unit
- X_a – throughput of aborting transactions, i.e. number of aborted transactions per time unit
- U_{CPU} – percentage of CPU time for doing useful computation on data items (when a token is in place L_7)

These performance characteristics could be used for comparison between database systems.

References

[1] Connoly T., C. Begg, A. Strachan. *Database Systems: A Practical Approach to Design, Implementation and management*, Addison-Wesley, Harlow, England, 1998.

[2] Chen I. R. Stochastic Petri Net Analysis of Deadlock Detection Algorithms in Transaction Database Systems with Dynamic Locking. *The Computer Journal*, Vol. 38, 1995, No. 9, 717-733.

[3] Atanassov K. *Generalized nets*. World Scientific, Singapore, 1991.

[4] Kolev, B. "An Algorithm for Transforming a Graph to a Generalized Net". In: - *Proceeding of the First International Workshop on Generalized Nets*, Sofia, 9 July 2000, 26-28.

[5] Atanassov K. *Intuitionistic Fuzzy Sets*, Springer-Verlag, Heidelberg, 1999.

[6] Pun K. H., G. G. Belford. Performance Study of Two-Phase Locking in Single-site Database Systems, *IEEE Trans. Software Eng.*, 13, 1311-1328.