Information Technology Journal 6 (4): 509-517, 2007
ISSN 1812-5638

A Transaction Description Model and Properties for P2P Computing

[1]Yubin Guo, [1]Jianqing Xi, [1]Deyou Tang and [2]Ximing Li
[1]School of Computer Science and Engineering,
South China University of Technology, Guang Zhou, 510641, China
[2]Modern Education Technology Center, South China University of Agriculture,
Guang Zhou, 510640, China

Abstract: P2P system is a very active research field due to the popularity and the widespread use of these systems today and their potential use in future. P2P networks are loosely coupled system without central control where peers have more autonomy. Therefore transaction management is crucial to applications such as e-commerce, process support systems etc. This study intends to utilize more semantics of application in transaction management for P2P environment. A transaction description model based on Colored Petri Net (CPN) is proposed. In this model, a transaction is composed of subtransactions that have been put together as building blocks according to the semantics of application. And transaction is denoted as a CPN, Transaction Net (TN) that is the combination of some sub-nets and the sub-nets are TNs of its subtransactions. Operators to construct a transaction from known ones as subtransactions, including sequence, concurrency, iteration and alternative are defined formally. Using this CPN, relations of subtransactions can be illustrated by the structure of TN and dynamic properties of transaction can be simulated by executing the model. Properties of the model are analyzed. Algorithm to execute a compound transaction is presented also.

Key words: P2P computing, colored petri net (CPN), transaction description model, transaction net

INTRODUCTION

Transaction technology is a vital infrastructure for concurrent control and recovery. It is widely used to guarantee correctness and reliability for distributed applications. The most classic transaction model is page model (Bernstein *et al*., 1987; Papadimitriou 1986; Gray and Peuter, 1993) (named as Read/Write model also). In this model, transaction is defined as a partial order set of read and write operations and all transactions satisfy the well-known ACID properties.

Object Model (Beeri *et al*., 1989) is the substitution and extension of page model. It has provided a framework to describe operations on any kinds of objects. A transaction is composed of operations that can invoke operations of other objects. The structure of transaction is a tree with participant objects as nodes and invocation relation as edges. Nested Transaction Model (Moss, 1987) is the earliest instance of object model. And Multilevel Transaction (Weikum, 1991) is another significant instance.

Many other transaction models for distributed systems are proposed also, for example saga (Molina and Salem, 1987), flexible transaction model (Zhang *et al*., 1994) etc. The model saga is proposed to structure long running processes. Flexible transaction model is used in multidatabase transaction management. In Alonso *et al*. (1997 and 1999) proposed a new transaction model and correctness criterion, for composite transitional systems. Composite system in (Alonso *et al*., 1999) is component based applications in which component has its own transaction management logic. The main contribution of this model is to establish the correctness conditions between operations of the same transaction, as well as between conflicting operations of different transaction, in a uniform way. And there theory is implemented in a lightweight transaction server for Plug-and-Play internet data management, named as CheeTah (Guy, 2000).

As to transaction management, a novel distributed serialization graph-based approach to concurrency control and recovery in peer-to-peer environments is presented (Haller *et al*., 2004). The uniqueness of the proposed protocol is that it ensures global correctness without relying on a global, up-to-date serialization graph. Mesaros *et al*. (2005) proposed a system for executing transactions on top of structured peer-to-peer network. The system ensures the ACID properties of transactional

Corresponding Author: Yubin Guo, School of Computer Science and Engineering, South China University of Technology, Guang Zhou, 510641, China

systems (Mesaros *et al*., 2005). But to common transaction management in P2P networks, they are far from mature. Gribble *et al.* (2001) have summarized properties of P2P system. Accordingly to survive P2P environment, transactions must have some new features.

- Synchronization and collaboration signify much to transaction management in each peer for it must have its own transaction management. And emphasis of a global transaction is organizing transaction mechanisms of peers together to accomplish task.
- A transaction on a certain peer is friable for the peer can join and leave at will that is the dynamic property of P2P network.
- But to a global transaction, the degree of reliability and flexibility is heightened. For it can be fulfilled by submitting functionally equivalent transactions (or sub-transactions) to several available peers.
- Transactions and subtransactions must have autonomy to some extent. So they can execute independently on independent peers without a central coordinator.

Weikum and Vossen (2002) concluded that semantics characteristic of operation is vital to enhance performance of transaction in Object Model. Some works have been done to introduce semantics of application into transaction management like relative serializability (Agrawal *et al*., 1994). This paper intends to utilize more semantics of application in transaction management. A transaction description model is proposed in which a transaction consists of subtransactions combined together by semantics of application. A transaction is denoted as Transaction Net (TN) that is the combination of some sub-nets. And the sub-nets are TNs of subtransactions. Operators to integrate subtransactions into a transaction including sequence, concurrency, iteration and alternative are defined formally. Using CPN as modeling tool, the structure of TN illustrates the relationship of subtransactions and dynamic properties of transaction can be simulated and analyzed by executing the CPN model.

TRANSACTION DESCRIPTION MODEL

A brief introduction of CPN: Petri net is a promising graphical and mathematical modeling tool that is in particular well-suited for concurrency and conflict description. Colored Petri Nets (Jensen, 1997) (CP-nets or CPN) is a high level Petri net that combining the advantage of Petri net and programming language. It can describe complex system with graphs, analyze static and dynamic characteristics and formalize reasoning. Many analyzing methods can be used to analyze properties of simulated system like occurrence graph, invariants and so on.

CPN Tools (CPN Tools homepage) is a tool for editing, simulating and analyzing CPN. The tool features incremental syntax checking and code generation that take place while a net is being constructed. A simulator can simulate the execution of CPN model efficiently. And properties of CPN such as boundedness and liveness are analyzed using a state space tools. For more elaborate information, the reader is reffered to Jensen (1997, CPN Tools homepage). In this paper all CPN model are created and analyzed by using CPN Tools. The following definition of CPN is given by Jensen (1997).

Definition 1: A CPN is a tuple CPN = $(\Sigma, P, T, A, N, C, G, E, I)$ where:

- Σ is a finite set of non-empty types, also called color sets.
- P is a finite set of places.
- T is a finite set of transitions.
- A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \varnothing$.
- N is a node function. It is defined from N: $A \rightarrow (P \times T \cup T \times P)$
- C is a color function. C: $P \rightarrow \Sigma$.
- G is a guard function. It is defined from T into expressions such that:
  $t \in T$: $(Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma)$.
- E is an arc expression function. It is defined from A into expressions such that: $\forall a \in A$: $(Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma)$ p is the place of N(a).
- I is an initialization function. It is defined from P into closed expressions such that: $\forall p \in P$: $(Type(I(p)) = C(p)_{MS})$.

To a node $x \in P \cup T$: ${}^{\bullet}x = \{y|(y, x) \in F\}$, $x^{\bullet} = \{y|(x, y) \in F\}$. A transition t is enabled at marking M, if in M each $p \in {}^{\bullet}t$ contains one token at least. If t is enabled, it can occur and a new marking M′ will be generated at M, represented by M(t>M′, where $p \in {}^{\bullet}t - t^{\bullet}$, M′(p)=M(p)-1; when $p \in t^{\bullet} - {}^{\bullet}t$, M′(p)=M(p)+1; otherwise, M′(p)=M(p).

Transaction description model: In our model, a transaction is denoted by a Transaction Net (TN). TN is a CPN model with an input and an output place. The structure of TN describes relationship between subtransactions. And the execution of TN simulates execution of the transaction.

To define the model formally, the Color Set of TN is presented at first.

Definition 2: Color set Σ is a set including elements as follows:
Color INPUT = string;
Color OUTPUT = with committed, aborted;
Color CONDS=Booleanexpression;
Color BOOLVAL= bool

Color INPUT is the color for input of transaction that is simplified as string. Color OUTPUT is for output of transaction that denotes whether this transaction is committed or not. Color CONDS is Boolean expression for condition. And BOOLVAL is const value for Boolean expressions.

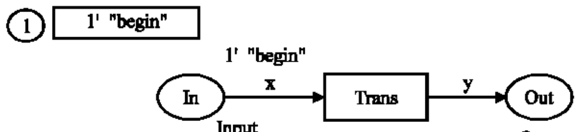
Definition 3: A Transaction Net (TN) is a CPN with a input place and an output place, i.e., TN = (Σ, P, T, A, N, C, G, E, I, In, Out)

- Symbols Σ, P, T, A, N, C, G, E have the same meaning as in definition 1.
- In is the input place of TN with $\forall t \in T$: $(t, in) \notin A$
- Out is the output place of TN with $\forall t \in T$: $(Out, t) \notin A$
- I is an initialization function. It is defined from P into closed expressions such that:
    - I(In) = 1\`x, with x is a variable of color INPUT
    - $\forall p \in P$, $p \neq In$: I(p) = NULL;

Figure 1 shows the TN of a transaction As in definition 3, place In is used to store input tokens and place Out is for deposit result of execution. We defined a homonymic transition to simulate the transaction trans. The transaction is ready to execute when there is a token of color input in place In. At that time, the TN can execute by occurring enabled transitions. The execution will terminate when there is a token in place Out.

In CPN Tools, there is a code segment for each transition. User can program to accomplish functions of simulated transaction. In our model the function of transition is to get executing result of transaction from a file named inputfile.txt. Different status can be simulated by placing different contents in this file.

To a complex transaction with subtransactions, Fig. 1 only gives holistic information of the transaction. And the simulating transition can be seen as a substitution transition (Jensen, 1997). Substitution transition is a special transition related to a separated CPN that gives detailed description of it. So to a complex transaction with subtransactions, the TN is a hierarchical CPN. But it is proved (Jensen, 1997) that to each hierarchical CPN, there is an equivalent CPN and vice verse. So we give no much consideration to hierarchical CPN in this study.



```
Input
Input ( );
Output ( );
action
let
   val file_id - TextIO, openIn("inputfile,txt")
   val y - TextIO, input(file_id)
   val_ - "aborted"
 in
  TextIO, closeIn(file_id)
end;
```

Fig. 1: TN of transaction trans

Definition 4: A transaction is a tuple T= (Id, Addr, I, O, ST, TN)
where
Id is the identifier of a basic transaction.
Addr is the peer address where this transaction can be called.
I is the input place of TN
O is the output place of TN.
ST is a set of all subtransactions
TN is a transaction net modeling the logical relations of subtransactions in ST and dynamic behaviors of the transaction.
Transaction can be divided into two types as follows.

Definition 5: Let T = (Id, Addr, I, O, ST, TN) be a transaction. T is a Basic Transaction, abbreviated to BT, if

- ST = ∅
- In TN, P = {In, Out} and |T| = 1, i.e., no subtransaction included.

And T is a *Compound Transaction*, abbreviated to CT, otherwise.

BT is the base of transaction description model. It can be executed without starting other transactions. And it can be with ACID properties if necessary. On the contrary, a CT can not be executed without starting subtransactions. The subtransactions of a CT can be both BTs and CTs. That is to say the structure of transaction is nested or multi-level.

Transaction computation: The way that brings subtransactions together to construct a transaction is defined as operations including sequence, concurrency, alternative and iteration. That is the method to create new transactions using existing ones as building blocks.

Let T_i = (Id_i, $Addr_i$, I_i, O_i, ST_i, Tn_i) (i = 1,2) be two transactions.

Definition 6: Sequence operator (In Fig. 2)
$T = T_1 \cdot T_2 = (Id, Addr, I, O, ST, TN)$, where
Id is the Identifier of new transaction;
Addr is the peer where to invocate it;
I is input of T including input arguments in proper form.
O is output of T to denote the result of executing in proper form.
$ST = ST_1 \cup ST_2$
$TN = (\Sigma, P, T, A, N, C, G, E, I, In, Out)$ where
P, T, A, N, C, E are shown in Fig 2.
$In = In_1$; $Out = Out_2$;$G(c_1) = G(c_2) = G(c_3) = null$;

Sequence operation specifies the execution of two transactions, t_1 and t_2, one after another. That is t_1 must be finished before t_2 starting.

The arc function in Fig. 2 is an example. It means when t_1 is committed (A token committed in place Out_1), Transition t_2 can be execute. Else, if t_1 aborts (A token aborted in place Out_1), t_2 would not execute. People can set arc functions with contrast semantics.

In a TN, transitions can be divided into two types. Transitions to simulate transactions are transaction transitions. And transitions to combine subtransactions together are called control transitions. In our model, code segment of each transaction transition is the same to code segment of transition trans in Fig. 1, so that it will be omitted in following figures.

Definition 7: Concurrency operator || (In Fig. 3)
$T = T_1 \| T_2 = (Id, Addr, I, O, ST, HCPN)$, where
Id is the Identifier of new transaction;
Addr is the peer where to invocate it;
I is input of T including input arguments in proper form.
O is output of T to denote the result of executing in proper form.
$ST = ST_1 \cup ST_2$
$TN = (\Sigma, P, T, A, N, C, G, E, In, Out, I)$
P, T, A, N, C, E are shown in Fig. 3.
$In = In_1$; $Out = Out_2$;$G(c_1) = G(c_2) = G(c_3) = null$;

Transition c_1 creates input conditions of transaction t_1 and t_2. And transition c_2 merges outputs of transaction t_1 and t_2 to create result in place out. Concurrency operation allows two transactions, t_1 and t_2, execute in any order. The execution of one transaction has no influence to another.

Definition 8: Alternative operator ⊕ (In Fig. 4)

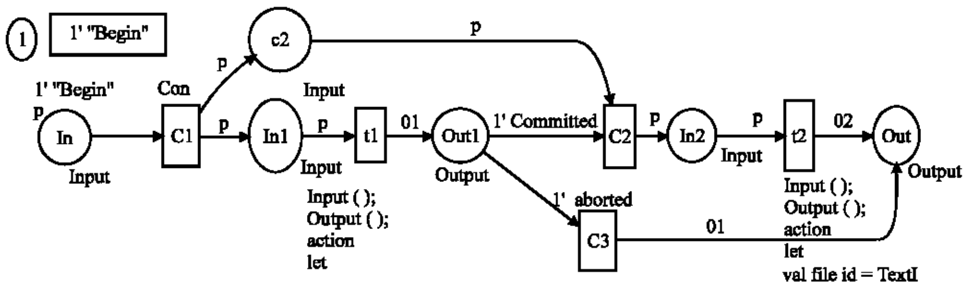


Fig. 2: Sequence operation of TN

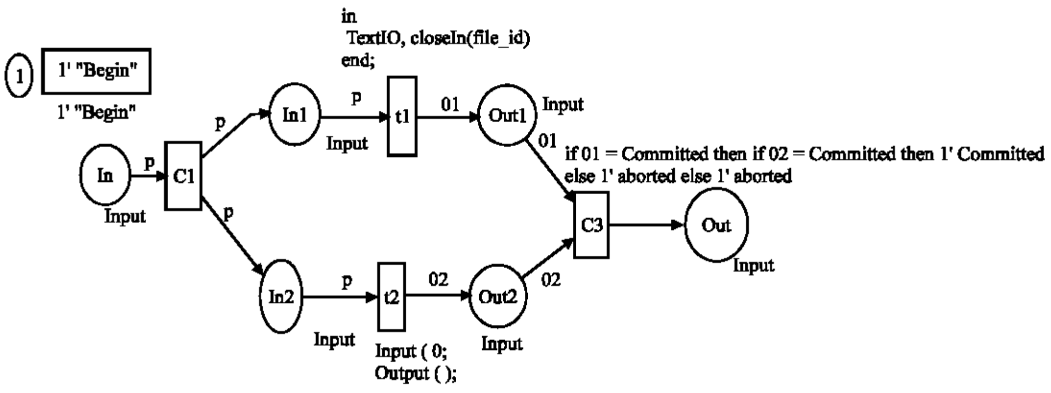


Fig. 3: Concurrency operation of transaction

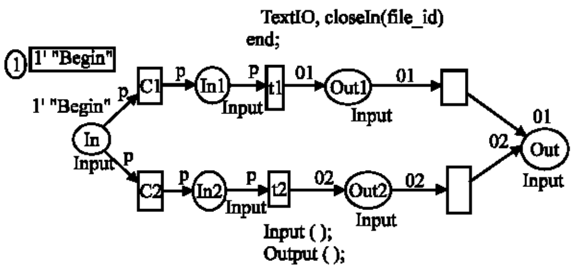


Fig. 4: Alternative operation of transaction

$T = T_1 \oplus T_2$ = (Id, Addr, I, O, ST,TN), where
Id is the Identifier of new transaction;
Addr is the peer where to invocate it;
I is input of T including input arguments in proper form.
O is output of T to denote the result of executing in proper form.
$ST = ST_1 \cup ST_2$
TN = (Σ,P,T,A,N,C,G,E,In,Out,I)
P, T, A, N, C, E are shown in Fig. 5.
In = In_1; Out = Out_2; $G(c_1)$ = boolexpression; $G(c_2) = \neg G(c_1)$;$G(c_3)$ = null;

Alternative operation of t_1 and t_2 means either t_1 or t_2, but not both, can be executed. TN of $T_1 \oplus T_2$ is shown in Fig. 4. From input condition, only one transition, c_1 or c_2, is enabled (This is decided by guard functions $G(c_1)$ and $G(c_2)$. And there functions are defined by semantics of application). Suppose c_1 is enabled, then transaction t_1 can occur and the result of t_1 would be deposited into place Out_1. Consequently transition c_3 is enabled and will occur to transfer result into place Out. The functions of transition c_2 and c_4 are similar to c_1 and c_3, respectively.

Definition 9: Iteration operator n (In Fig. 5)
$T = nT_1$ = (Id, Addr, I, O, ST,TN), where
Id is the Identifier of new transaction;
Addr is the peer where to invoke it; I is input of T including input arguments in proper form.
O is output of T to denote the result of executing in proper form.
$ST = ST_1$
TN = (Σ,P,T,F,C,G,E,In,Out,I)
P, T, A, N, C, E are shown in Fig. 4.
In = In_1; Out = Out_2; $G(c_1) = G(c_2) = G(c_3)$ = null;

Iteration operation models execution of a transaction followed by itself for several times. In Fig. 5, the function of transition c_1 is initializing the iteration that is putting proper tokens into In_1 and p_1. Token in place p_1 denotes iteration condition. Transition c_3 judges whether the iteration condition is match or not. According to judgment, a token true or false is put into place p_2. Transition c_4 will occur when the condition of iteration is match and will put a token into place In_1. Transition c_2 can occur when transaction t_1 is aborted and create a token aborted in place Out. Transition c_5 would occur and send token committed into place Out when the iteration finished.

Operators ·, ||, ⊕ are defined with two operands. It is easy to popularize to multi operands.

PROPERTIES OF THE MODEL

Here, we present the algebraic and dynamic properties of the operations. The functionally equivalent of a transaction is defined at first.

Definition 10: Let T_i = (Id_i, $Addr_i$, I_i, O_i, ST_i, Tn_i) (i = 1,2) be two transactions. T_1 is functionally equivalent to T_2, denoted as $T_1 \cong T_2$, iff $I_1 = I_2 \Rightarrow O_1 = O_2$.

In definition 10, two transactions are functionally equivalent if they can get same result from the same input. Let T_1, T_2 be functionally equivalent (They are denoted as functional equivalence). The difference between T_1, T_2 lies in two aspects, different set of subtransactions with some subtransactions having been alternated by their functionally equivalents and different structure of TNs.

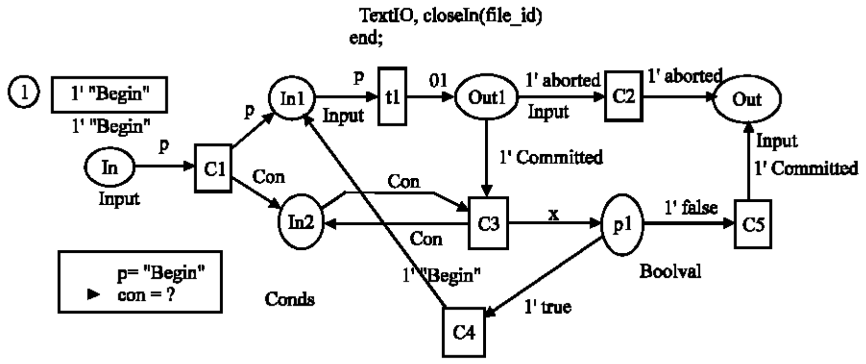


Fig. 5: Iteration operation of transactions

As illustrated in Fig. 6, transaction T and T′ have the same set of subtransactions {t_1, t_2, t_3} but different structure of TN. To same input, they get the same output. Therefore T and T′ are functionally equivalence. In P2P environment, the degree of reliability of a certain transaction on a certain peer is low. However, if functionally equivalent transactions are given for vital subtransactions, the rate of survival of the whole CT will be heightened. Similarly, to accomplish a task, if some equivalent transactions are start, it is most likely that the task would be accomplished. At this sense, our model is fault-tolerant.

Theorem 1: Let I be the set of all transaction, then operators ·, ||, n and ⊕ has algebraic properties as follows:

$$(t_1 \cdot t_2) \cdot t_3 \cong t_1 \cdot (t_2 \cdot t_3) \quad (1)$$

$$(t_1 \oplus t_2) \oplus t_3 \cong t_1 \oplus (t_2 \oplus t_3) \quad (2)$$

$$(t_1 \| t_2) \| t_3 \cong t_1 \| (t_2 \| t_3) \quad (3)$$

$$t_1 \oplus t_2 \cong t_2 \oplus t_1 \quad (4)$$

$$t_1 \| t_2 \cong t_2 \| t_1 \quad (5)$$

$$t_1 \cdot (t_2 \| t_3) \cong (t_1 \| t_2) \cdot (t_1 \| t_3) \quad (6)$$

$$t_1 \cdot (t_2 \oplus t_3) \cong (t_1 \cdot t_2) \oplus (t_1 \cdot t_3) \quad (7)$$

$$t_1 \| (t_2 \oplus t_3) \cong (t_1 \| t_2) \oplus (t_1 \| t_3) \quad (8)$$

$$n(t_1 \| t_2) \cong nt_1 \| nt_2 \quad (9)$$

Formula (1) is an immediate consequence of example in Fig. 6 for T = $t_1 \cdot (t_2 \cdot t_3)$ and T′ = $(t_1 \cdot t_2) \cdot t_3$. Other formulas can be proved similarly.

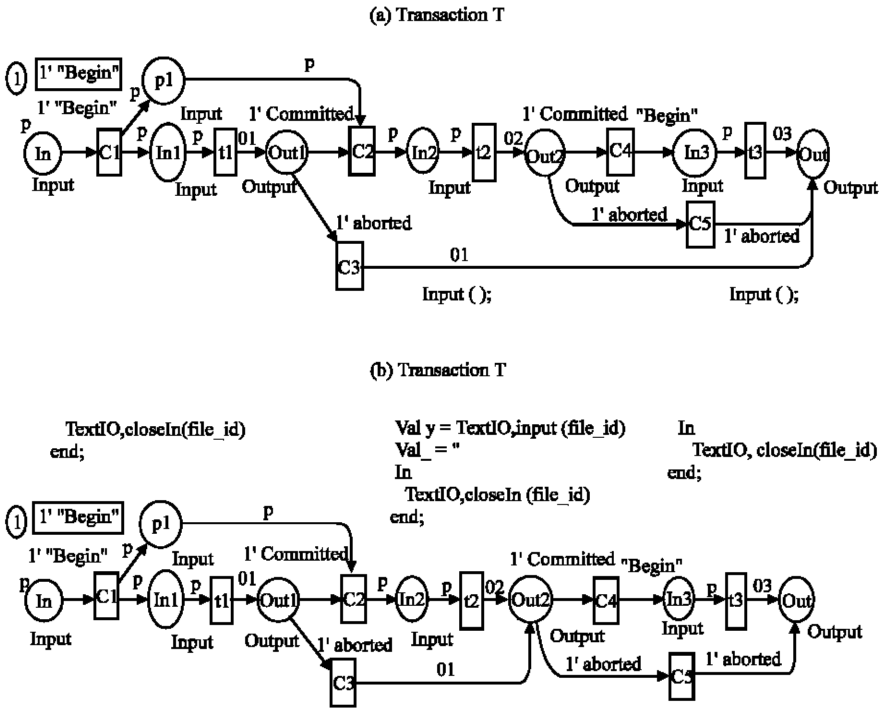
Using those algebraic properties, TN of CTs can be simplified. Then the complexity of model would be decreased.

Corollary 1: Let Γ be the set of all transactions, <Γ,.> is a semigroup and <Γ,||>, <Γ,⊕> are all changeable semigroup.

Proof: From theorem 1 formula (1), sequence operator · is associative. And from definition 6, the result of sequence operation T is a transaction constructed from $T_1 \cdot T_2$. So the sequence operator is closed to Γ. Therefore < Γ,. > is a semigroup. In the same way, we can prove that < Γ, || >, < Γ, ⊕ > are semigroups. And from Theorem 1 formula (4) (5), < Γ, || > and < Γ, ⊕ > are changeable semigroup. _

Theorem 2: Let Γ be the set of all transactions, O = {·, ||, ⊕, n } be the operator set. O is closed to Γ.

Proof: From corollary 1, operator ·, || and ⊕ are closed to Γ. And from definition 9, n is closed to Γ also. Here we give the priority of those operators: n has the highest and the other three has equal priority. The composition of operator means a transaction was constructed from many other subtransactions through multi operations consequently. Now it is not difficult to prove this theorem by induction on number of operations.



```
TextIO,closeIn(file_id)
end;
```

```
Val y = TextIO,input (file_id)
Val_ = "
In
  TextIO,closeIn (file_id)
end;
```

```
In
  TextIO, closeIn(file_id)
end;
```

Fig. 6: Two functionally equivalent transactions

Theorem 3: $< \Gamma, O >$ is congruence and completeness.

Proof: We prove congruence property at first, using operator || as example.

Let $T_i = (Id_i, Addr_i, I_i, O_i, ST_i, Tn_i)$ $(i = 1,2)$ be two transactions. From definition 9, $T = T_1 \| T_2 = (Id, Addr, I, O, ST, TN)$, ST, TN of T can be derived from T_1 and T_2. And T is a transaction complying with definition 4. So operator || has sense to all transactions in Γ. In other words, concurrency operator || is congruence.

Similarly it can be proved that all operators in O are congruence. So that the transaction expression constructed from operators in O and transactions in Γ is congruence.

Now we prove completeness of the model.

From Milner (1989), the basic behavior characteristic of network system is concurrency. The concurrency operator || in O is defined to express concurrency computation. In this model both concurrency computation framework and sequence computation framework are necessary. However concurrency computation framework cannot be deduced from sequence computation framework. So concurrency operator || is necessary.

To sequence computation framework, any program can be represented by a combination of the control elements sequence, branch and loop (Bohm and Jacopini, 1966). It is obvious that operators $\cdot$, $\oplus$ and n, is defined to express sequence, branch and loop structure respectively. So operators $\cdot$, $\oplus$ and n, are sufficient to express sequence computation framework. From above, $<\Gamma, O>$ is congruence and completeness.

Theorem 4: Let $T = (Id,Addr,I,O,ST,TN)$, be a transaction and $TN = (P,T,F,C,G,E,I,In,Out)$. TN is bounded.

Proof: From definition 4 and 5, it is easy to say that TN of a BT is bounded.

From definitions 6, 7, 8, 9 and transition firing rules of CPN, it is not difficult to prove that is true for TN of transactions constructed with only one operation. Using induction on the number of composing operations (any operator in $O = \{\cdot,\|,\oplus,n\}$), the conclusion holds.

To analyze the execution of TN, the status of termination is defined as follows.

Definition 11: Let $TN = (\Sigma, P, T, A, N, C, G, E, I, In, Out)$ be a Transaction net, a marking M is called terminate marking if $M(Out) \neq NULL$. And the set of all terminate marking of TN is called terminate marking set of TN, denoted as M_e.

Furthermore, to TN of a transaction, a transition sequence can occur from initial marking M_s to a terminal marking. That is:

Theorem 5: Let transaction $T = (Id,Addr,I,O,ST,TN)$ with $TN = (P,T,F,C,G,E,I, In,Out)$. Under the initial marking M_s $(\forall p \in P:M_s()) = I(p)$:
$\exists \boldsymbol{\sigma} \in T^*:M_s(\boldsymbol{\sigma}>M \wedge M \in M_e$ and $\forall M \in R(M_s)-M_e: \exists t \in T:M(t>$
This theorem can be proved using induction on times of composing operation (including all operators in $O = \{\cdot,\|,\oplus,n\}$). The proof is abbreviated.

From theorems above, we can know that any transaction can be composed from operators in O. And each well-formed CT can be executed and terminate nomally.

CONSTRUCTING AND EXECUTING OF CTS

This section presents a formal method to construct CT.

Definition 12: A CT can be acquired by the following steps.

1. Create information of Id, Addr, I, O and ST of CT
2. To each subtransaction in ST, get Id, Addr, I, O and ST of it from according peer and create CPN model (as in Fig. 1) for it.
3. Construct operation expression of CT with operators in $\{\cdot,\|,\oplus,n\}$ according to semantics of application.
4. Using functional equivalence to simplify the operation expression in step 3. And build TN of transactions.

From definition 12, a CT can be obtained. The execution of a CT is showed in the following algorithm.

Algorithm 1. Executing algorithm CT
```
{$ET=\{\forall t \in T:M_0(t>\}$, $M=M_0$
//ET is the set of enabled transitions
while ET< >NULL {
get $t_i \in ET$
if ($t_i$ is a control transition) {
    $t_i$ fire;
    $M` = M(t_i>$;
    $M = M`$;
    $ET = \{\forall t \in T:M`(t>\}$;
}else {
    start a transaction call at according peer and waiting
    for results
    $M`({}^{\bullet}t) = M({}^{\bullet}t)-1`input$;
    //set output token according to the result
        // of subtransaction
    if the subtransaction failed
        $M(t^{\bullet}) = 1`aborted$
    else $M`(t^{\bullet}) = 1`committed$
     if $M \in M_e$ exit
```

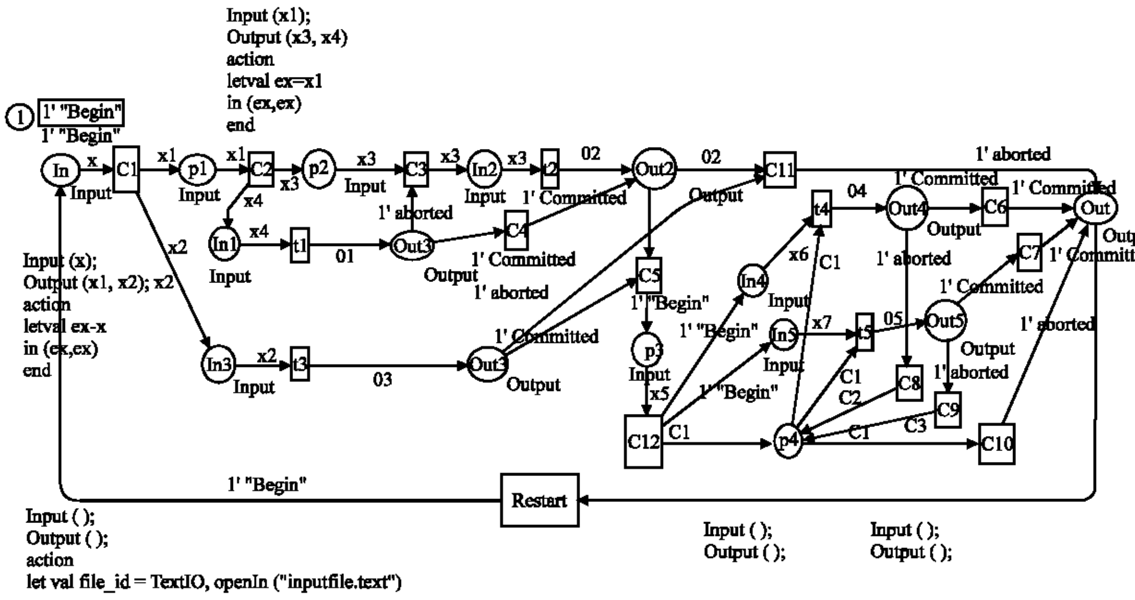


Fig. 7: CPN model of travel reservation

```
        $M = M`, ET = \{\forall t \in T: M`(t >\};$
        }//end of else
      } // end of while
        if ($M \in M_e$ and M(Out) = committed) {
              commit subtransactions waiting for global
              decision;
              return (committed);
      } else {
          abort subtransactions that waiting for global
          decision;
          compensate or committed ones;
          return (aborted )
      }//end of if
} //end of algorithm.
```

The correctness and terminability can be guaranteed by Theorem 4 and 5.

Here is an example of how the transaction description model is used. This is a typical scenario for booking a trip. The trip incorporates a flight, hotel and car rental reservation and bookings. To consumer, the whole travel is in one transaction and there is a travel agent acts as an intermediary of the consumers behalf. To the flight booking request, there are two airways A, B available and the client prefer airway A. To the hotel reservation, there are two choices C, D and the client give no priority for choice. And to the car renting, only one choice is available.

Example 1: Let T_1 be subtransaction of booking flight from Airway A; T_2 be subtransaction of booking from Airway B; T_3, subtransaction of renting a car; and T_4, T_5, subtransactions of reserving hotel C, D respectively. As illustrated in Fig. 7, the whole transaction can be seen as a CT composed of 2 sequence subtransactions, travel tool booking and hotel reservation. Travel tool booking is constructed of two parallel subtransaction flight booking and car renting. And the subtransaction flight booking includes two subtransactions T_1 and T_2. In transaction expression, that is $T = ((T_1 \cdot T_2) \| T_3) \cdot (T_4 \oplus T_5)$.

Following Algorithm 1, the model can execute. Under initial marking, ET = $\{c_1\}$, viz. c_1 can occur. And the occurring of c_1 would create input tokens of sub-transactions flight booking and car renting. At that time, ET = $\{c_2, T_3\}$. Let c_2 occur firstly, input condition of T_1 will be got. Let T_1 occur, according to the result of T_1, ET = $\{c_3, T_3\}$ (T_1 aborted) or $\{c_4, T_3\}$(T_1 committed). Step by step analogously, the model can execute to a final marking. Transition restart is an additional Control transition that has no effect on simulated transaction. With this transition, a new reservation can start after one reservation finishes.

From reports of CPN Tools, the CPN model is bounded and terminable that means the process of trip reservation is valid. And if necessary, all instance of the model can be acquired from occurrence graph created by CPN tools.

CONCLUSION

In this study, the characteristics of P2P transaction are summarized at first. Then a transaction description model to utilize more semantics of application is proposed. Transaction in this model is a multi-level framework with subtransactions combining together in a formal way. Operators to construct a transaction from subtransactions are defined as sequence, concurrency, iteration and alternative. Using CPN as the modeling tool, the structure of transaction net illustrates the relationships of subtransactions and dynamic properties of transaction can be simulated and analyzed by executing the CPN model. Properties of the model are analyzed. Algorithms to construct and execute compound transaction are also presented.

Compare to traditional transaction model (Weikum, 1991; Moss, 1987), we describe more logical relations among subtransactions that comes from application logic. For in our comprehension, transaction is more that a sequence of subtransactions. And an instance of transaction is the executing sequence of it. That is just similar to the relation between program and process.

But as to constructing a transaction processing system, present study is primarily. As to atomicity of transaction, we divide a transaction into many atomistic fragments. And the atomicity of each atomistic fragment must be guaranteed. Definition of atomicity and a new correctness criterion are given in another paper. Mechanism for concurrency control and recovery are in process. And a prototype system is constructing to implement and verify the model recently. That work including creating the CPN model atomically, giving the relationship between properties of CPN model (for instance liveness, boundedness and fairness) and properties of transaction and so on.

ACKNOWLEDGMENT

This study was supported by the China Ministry of Education (Grant No.D4109029), Guangdong High-Tech Program (G03B2040770) and Guangdong Natural Science Foundation (B6480598).

REFERENCES

Agrawal, D., J.L. Bruno, A. Abbadi and V. Krishnaswamy, 1994. Relative Serializability: An Approach for Relaxing Atomicity of Transactions. In: Vianu, V. (Ed.) ACM Principles of Database Systems. ACM, New York, pp: 139-149.

Alonso, G., S. Blott, A. Fessler and H. Schek, 1997. Correctness and parallelism in composite systems. PODS '97. 1997.

Alonso, G., A. Fessler, G. Pardon and H. Schek, 1999. Correctness in General Configurations of Transactional Components. PODS '99 Philadelphia PA.

Beeri, G., P.A. Berstein and N. Goodman, 1989. A model for concurrency in nested transaction systems. J. ACM, 36: 230-269.

Bernstein, P.A., V. Hadzilacos and N. Goodman, 1987. Concurrency Control and Recovery in Database Systems. MA: Addison-Wesley.

Bohm, C. and Jacopini, 1966. G. Flow diagram, Turing machine and languages with two formation rules(R).USA: CACM, 9: 465-477.

CPN Tools homepage. http://wiki.daimi.au.dk/cpntools/cpntools.wiki

Gray, J. and A. Peuter, 1993. Transaction Processing: Concepts and Techniques. San Francosco: Morgan Kaufmann.

Gribble, S., A. Halevy and Z. Ives *et al*., 2001. What can Database do for Peer-to-peer. In: Proceeding of 1st Workshop on the WEB and Databases (WebDB).

Guy, P., 2000. Composite Systems: Decentralized Nested Transactions. Ph.D. Thesis 2000. Swiss Federal Institute of Technology Zurich.

Haller, K., H. Schuldt and C. Turker, 2004. A fully decentralized approach to coordinating transactional processes in peer-to-peer environments. Technical Report # 463, 2004. ETH Zurich.

Jensen, K., 1997. Colored Petri nets: Basic concepts, analysis methods and practical use. Volume 1, Basic Concepts. Monographs in Theoretical Computer Science. Berlin, Heidelberg, New York: Springer-Verlag, 2nd corrected printing 1997.

Mesaros, V., R.T. Colle, K. Glynn and P. Van Roy, 2005. A Transactional System for Structured Overlay Networks. Research Report RR2005-01, Universite catholique de Louvain. March 2005. Available at ftp://ftp.info.ucl.ac.be/pub/reports/2005/rr2005-01.pdf

Milner, R., 1989. Communication and Concurrency. Prentice Hall, London.

Molina, G. and K. Salem, 1987. Sagas. ACM SIGMOD Record, 16: 249-259.

Moss, J.E.B., 1987. Nested Trasactions: An Introduction. In: Bhargava, B. (Ed.), Concurrency Control and Reliability in Distributed Systems. New York: Van Nostrand Reinhold, pp: 395-425.

Papadimitriou, C.H., 1986. The theory of Database Concurrency Control. Rockville, MD: Computer Science Press.

Weikum, G., 1991. Principles and Realization Strategies of Multilevel Transaction Management. ACM Transactions on Database Systems, 16: 132-180.

Weikum, G. and G. Vossen, 2002. Transaction Information Systems Theory, Algorithms and the Practice of Concurrency Control and Recovery. (ISBN 1-55860, 508-8).

Zhang, A., M. Nodine, B. Bhargava and O. Bukhres, 1994. Ensuring relaxed atomicity for flexible transactions in multi-database systems. ACM SIGMOD Record, 23: 67-78.