[TA Számítástechnikai és Automatizálási Kutató Intézete, Közlemények 27/1982

FORMALIZATION OF CONCURRENCY CONTROL IN DISTRIBUTED DATA SYSTEMS[1]

Serge M. MIRANDA

CERISS-INRIA
Université des Sciences Sociales
Place Anatole France
31040 Toulouse Cedex
France
Tel (61) 23 11 45 ext 395

ABSTRACT

Synchronization issues in distributed data bases were heavily invested n recent literature and many synchronization protocols have been designed.

However much work is still to be done in the areas of ROBUSTNESS and ECOVERY, FORMAL PERFORMANCE ANALYSIS, FORMAL SPECIFICATION AND VALIDATION, IGOROUS UNIFORMIZATION.

This paper is a contribution to the two latter points; we propose a ormal approach based on abstract data types (algebraic methodology) and evelop a uniform rigorous framework in which the synchronization protocols an be specified and validated.

We illustrate our approach on a basic protocol which is representative f a major class of solutions.

EY-WORDS: protocol formalization (specification and validation), synchronization protocol, distributed data bases, duplicated data, abstract data types.

1) This research is sponsored by INRIA-ADI (SIRIUS project) under contract # 80003.

1. INTRODUCTION

Many researchers have recently presented solutions to concurrency control for distributed data systems. These solutions take form of synchronization protocols which have been designed to coordinate the remote processes in charge of local data (called "controllers") during an update session.

Very few proposals have been made to formally specify and validate ("formalize") these protocols.

This paper is primarily concerned with this crucial aspect; we introduce a formal methodology based on algebraically-specified data types to formalize existing protocols.

This article encompasses two major sections:

- the first one presents a clear definition of mutual integrity which turns out to be the basic requirement which must be verified by a synchronization protocol and introduces our formalism.
  This concept is translated in terms of our model through mutual-integrity theorems which are recalled.

- the second section illustrates our approach with a synchronization protocol (for duplicated entities) which has been largely referenced in the literature (namely THOMAS' one).

2. INTEGRITY CONCEPTS

We shall in turn examine the integrity concept in centralized and distributed data base management systems (DBMS).

2.1 INTEGRITY IN A CENTRALIZED DBMS

A data base can be viewed as a collection of entities and constraints whose values define VALID states of the data base. The concept of integrity (consistency) of a DBMS is twofold (MIRA80-b):

- internal integrity.

- external integrity.

Internal integrity is associated with integrity constraints defined on data to meet real-world restricitons.
The concept of transaction (GRAY78) has been introduced in centralized DBMS (and naturally extended to distributed DBMS (GRAY79)) to represent the atomic interaction of the user with the data base which preserves internal integrity.

External integrity corresponds to the control of concurrent transactions which may conflict in sharing common data (problems of "lost update", "dirty read", ...) A serialization mechanism (locking is the one which has been almost exclusively elected) must be defined to ensure external integrity.

2.2 INTEGRITY IN A DISTRIBUTED DBMS

Internal integrity in a distributed DBMS is called MUTUAL INTEGRITY when remote entities are involved in a (global) transaction; identity is a particular case of an integrity constraint which leads to the well-studied problem of duplicated entities. There on, we shall mainly consider the latter aspect.

We say that a synchronization protocol ensures MUTUAL INTEGRITY when the manipulated entities converge to the same state should update activity cease. (THOM75)...

A distinction between STRONG and WEAK mutual integrity has been proposed by several authors (SEGU78), (LELA79)...

However this distinction has been rather vague or incorrect since the underlying concept was itself vague or incorrect; for example in (LELA79) or (SEGU78) the distinction is based upon the SIMULTANEITY concept among remote states and this turns out to be delicate since no site can ever know the state of the entire distributed system (MONT78),(GRAY79),...

We propose a distinction based on the AVAILABILITY concept; an entity is said to be available when it is stable (no modification in progress) and open to a retrieval access.

Mutual integrity is said to be WEAK whenever a synchronization protocol enables to have two TEMPORARILY different available versions of the same copy at a given time (a transaction may retrieve consistent entities which are not the most current); it is said to be STRONG otherwise (the retrieved data are the most current).

We can make a parallel between this definition of mutual integrity (strong and weak) and the levels of consistency defined in centralized DBMS like SYSTEM-R (GRAY75); in this latter case, strong integrity correspond to the third level, weak integrity to the second level while the first level of consistency can be considered as a "weaker" form of integrity (access to dirty data is possible at that level). Our definition of availability precludes this latter type of consistency.

External integrity refers to the control of concurrent confliciting transactions which can be initiated anywhere in the underlying network.

In centralized systems there exists a control locus where shared COMMON memory is used for coordinating concurrent conflicting transactions.

Let us consider the type of control we may have in distributed data systems.

In a distributed system, which can be defined as a collection of processes communicating only through message-passing, three types of CONTROL LOCI have been chosen for synchronization protocols:

(i) WITHIN A SITE; the concurrency control is said to be CENTRALIZED (or VERTICAL) by analogy with local systems (MENA77), (GARC79)...

(ii) WITHIN A PROCESS; there is no privileged site. Each site is functionnaly homogeneous. There exists a given process (which we call MASTER CONTROLLER) responsible for the whole synchronization session.
The control is said to be "PARTIALLY DECENTRALIZED"; the protocols of ELLIS (ELLI77)..., LE LANN (LELA76)..., BUSTA (BUST78), POPEK/MIRANDA(POPE79) ... are of this type.

(iii) WITHIN A MESSAGE; the initiator of the synchronization session and the initiator of the global update are (generally) different controllers. A special synchronization message propagates control data from controller to controller (like OK votes in THOMAS' protocol (THOM75)).
The control is said to be "FULLY DECENTRALIZED".

In the last two cases, the control is said to be HORIZONTAL; from point (i) to point (iii), the tendency is towards a reduction of the time for which a host has control over the progress of a protocol.

A "primary update token" that moves around the network and symbolizes control is a particular case of centralized control technique ("circulating centralized control") which has been proposed by several authors (WILM79-b),

The protocol we formalize in this paper presents the following characteristics:

PROTOCOL / INTEGRITY	THOMAS'
Mutual integrity	WEAK
External integrity (type of control)	Fully-decentralized control
(type of consensus)	(majority)

Figure 1.
Major characteristics of THOMAS's protocol.

3. ABSTRACT DATA TYPES

3.1 ABSTRACT-DATA TYPE CONCEPT

A data abstraction is a behavioural representation-free description often using formal notation of a data object and the operations upon it.

Data abstractions are realized in programming languages by abstract data types (ADT) which isolate the representational detail from other programs units.

Although the ADT concept has largely been adopted by language designers (WULF76), (TARD77), (GUTT78),... the concept is really language-independent and can very naturally be carried over in layered systems (operating systems, data base management systems,...).

There are two properties of ADT which appear to be appealing for distributed systems:

(i) enclosure and implementation hiding (data independence; user transparency).

(ii) abstractional power (separability of functions; flexibility).

Two families of object-oriented languages (encompassing ADT's) have been proposed, the propositional one (MILN71), (HOAR72), (WULF76) and the algebraic one (BURST77), (GOGU78), (GUTT78).
We elected the algebraic approach which seems to be more adequate to complex structure formalization (PAOL77), (LOCK78), (MELK78),...

3.2 ALGEBRAIC APPROACH (GOGU76), (TARD77), (GUTT78).

We briefly recall the major features of the algebraic approach for ADT's as presented in (GOGU76).
An ADT can be defined as a MANY-SORTED ALGEBRA:

- an algebra of ONE SORT is roughly speaking a set of objects and a family of operators on the set; the set is called the carrier of the algebra.

- MANY-SORTED ALGEBRA extends this definition by allowing the carrier of the algebra to consist of several disjoint sets; each of these sets is said to have a SORT; the operators are sorted and typed but must be closed with respect to the carrier.

Two basic kinds of sorts are involved in an ADT specification: the sort being defined and any number of sorts assumed previously defined in a similar fashion (i.e. the Boolean sort includes the usuals constants T and F, and is assumed to have been defined in its own right with the same methodology; other built-in ADT's we may use are INTEGER, QUEUE,...)

Operators of the algebra are indexed by pairs (w,s) where $w \in S^*$ (sort of the operand) and $s \in S$ (sort of the results); the symbol $\underline{\Sigma w,s}$ will be used for the set of all operations with index (w,s); $\underline{\Sigma}$ is used for the union of all the sets $\Sigma w,s$ and is called the "signature" of the algebra.

A Σ-algebra A is a family of sets (As), $s \in S$, called CARRIERS of A which is determined by a triple $<S, \Sigma, E>$ where:

S is the set of "sorts", S : {s1, s2,...si,...sn} denoting the various types of objects which are required for that definition.

Σ is the set of operations $\Sigma = \{\Sigma, w, s\}$, whose operands and results are objects making up the sorts in S (SYNTAX description)

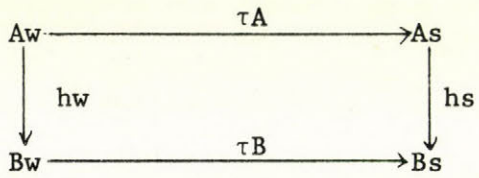
E is the set of equations which describe the semantics of each $\Sigma w,s$ of Σ; each algebraic equation (or axiom) defines the results of various combinations of operators applied upon various operands.

For every $(w,s) \in S^* \times S$ and every $\tau \in \Sigma w,s$ the function $\tau A : As1 \times As2 \times \times Asn \rightarrow As$ with w = s1,s2,...sn is called "A-operation named by τ"

Two many-sorted algebras are called Σ-algebras if they have the same signature.

A very important concept is the one of Σ-homomorphism; Definition: given two Σ-algebras A and B, a Σ-homomorphism, h:A → B, is a family of functions < hs:As → Bs, s∈S > mapping each carrier in A into the corresponding carrier in B while preserving the operations, i.e.

$\forall \tau \in \Sigma w,s$ with w=s1,s2,...sn and (a1,...an) $\in As1 \times As2 \times ..Asn$ we have $hs(\tau_A(a1,...an)) = \tau_B\ hs1(a1),...hsn(an))$; this can be visualized by the following diagram commutation:



We shall use the Σ-homomorphism concept to express:

(i) parallelism among remote operators (Σ-homomorphism)

(ii) layered abstractions between the distributed data base and the underlying transmission facility on one hand; between the distributed data base and the local DBMS on the other hand.

It is important to note that the ADT definitions of a particular type is not unique; however, it should meet the following goals:

(1) The operators should not be redundant.
(2) The equations must not be contradictory.
(3) The operators and axioms should be as simple as possible.
(4) The equations should be constructed in such a way that leads forcibly to unique reduction.
(5) We shall use a formalism close to OBJ-O (TARD77), (GOGU78), (GOGU76), (TARD79).

Which represents one of the basic language encompassing algebraically-specified data types; Guttag' systems for symbolic execution

of ADT's seems to suffer some inadequacies mainly at the sintactic level (TARD79).

3.3 OBJ-O

OBJ-O is an object-oriented language defined by GOGUEN(GOGU76) and implemented by TARDO (TARD77); this language is very close to NPL language, now called HOPE defined in (BURS77-b). In OBJ-O, an algebra is a 4-tuple:

< SORTS, OPS, VARS, SPECS > where

SORTS, OPS, SPECS correspond respectively to S,Σ,E and VARS includes the definition of the working variables used in the axioms.

The error-operators ("ERROR-OPS") and their semantics ("ERROR-SPECS") may be naturally defined in this language; an extensive discussion of "error algebras" is presented in (GOGU77).

The general syntax of an operator is given by:

OP : S1,S2,.....SN ⟶ Sp

Operator name (under bars are used for place-holders)

Operand sorts (N: arity of the OP operator)

results sort

Prefix, infix, postfix, distributed-fix declarations are possible in OBJ-O.

Each sort has an equality relation which is built-in with syntax:

-:= - : S S⟶S

Hidden operators may be declared with the key-word "HIDDEN" placed after the result sort.

We shall often use conditional equations in the algebraic specification; a formal theory of conditional equations in ADT's is developed in (THAT76).

In the specifications of a synchronization controller using ADT's, we shall neither use place-holders in OPS nor consider ERROR operators.

4. FORMALIZATION OF A SYNCHRONIZATION PROTOCOL

The concept of ADT should be the lowest common denominator which maximizes every one's naturalness at some level of expression. In the area of sycnhronization protocols, the three basic operators (framework for uniformization/abstraction) which may receive a general consensus, are:

- PREPAREG; this operator corresponds;

  (i) to the local initialization which consists of all the actions needed for the transaction to start consistency enforcement: time-stamp acquisition (THOM75), new-value computation (THOM75), ticket allocation (LELA78), priority definition (ELLI77), (BUST78), (POPE79),...

  (ii) to the global initialization which includes the strategy to resolve conflicts: first revolution in the virtual ring (ELLI77), first step of synchronization (POPE79),..., to check security (POPE79), to perform temporary update (LELA78), (POPE79),....

- SETG; this operator consists of the propagation or broadcasting of the (permanent) update.

- UNSETG; this operator concerns the (robust) commitment phase (log synchronization (POPE79),...)

The work on reliability is much less developed than that on external consistency and in many proposals, UNSETG is embedded in SETG, as in (THOM75)... This corresponds to the fact that there exists a clear dichotomy between performance-oriented and robustness-oriented protocols.

The above operators apply to a "global virtual object"; the global virtual object consists of each local object semantically tied during a manipulation. A duplicated entity may be seen as a logical entity (the global virtual object) physically stored in several remote sites.

This "locality" concept receives a formal development in (BILL79).

The global virtual object is characterized by a set of (global) states in each participating controller. Each operator maps an input global state and a set of input messages into an output state and a set of output messages.

Protocols are said to be message-driven: reception of a synchronization message causes a series of actions (and eventually a message transmission) to be executed.

This concept of (global) STATE enables us to express the semantics of the attached manipulation operators in a simple way. A change of state is the result of the execution of an operator bound to the occurence of an event (here a synchronization-message).

These states enable us to specify the different protocol steps; they allow:

(i) The simplification of operator semantics.
(ii) The simplification of the validation process.
(iii) The setting of clear re-entry points for the recovery procedure.
(iv) The expression of parallelism among remote controllers by using homomorphisms among remote states.

Our model encapsulates the functionning of a controller with a Σ-algebra, called SYNCH, whose carrier is the global-entity state and operators, (PREPAREG, SETG, UNSETG), a reduced set of primitives attached to particular message receptions.

A synchronization protocol is depicted as a set of Σ-homomorphism ALGEBRAS.

This approach offers the same advantages as Petri-nets (representation independence...) with additional ones such as:

- simple rigorous formalism for inferring proofs of correctness.

- semantic framework from which correct implementations can be derived (automatically) (GUTT78), (TARD77), (NOUR79),...

- ROBUSTNESS integration in a natural way (in (POPE79), we define the semantics of a message reception indicating the failure of a cooperating controller).

4.1 GLOBAL ENTITY STATES

For each of the three generic operators indicated previously we attach an input and an output (global) state. Among these states two of them appear to be important;

- the available state, noted F, before the actual modification (initial state) an F' after (final state); the local entity (copy) can be retrieved.
  F represents the input state of PREPAREG and F' the output state of UNSETG.

- the unstable (unavailable) state noted U which corresponds to a current modification of the local entity which cannot be retrieved.

4.2 CANONICAL REDUCTION; SERIALIZATION

Every sequence of operators of SYNC is reduced to a sequence (PREPAREG)-(SETG)-(UNSETG), called canonical reduction.

A transaction is said to be ATOMIC if canonical reductions of this transaction are identical in each controller involved in the synchronization session.

Two confliciting transactions are said to be SERIALIZED if their respective canonical reductions do not interfere in the modification phase (SETG).

4.3 MUTUAL-INTEGRITY THEOREMS

Before recalling the two mutual-integrity theorems (whose proofs are given in (MIRA79)), let us give an extension of the Σ-homomorphism definition; we say we have a TOTAL Σ-homomorphism in a synchronization protocol whose each controller is specified by a Σ-algebra SYNC if:

$\forall i$, $\forall j$ SYNCi and SYNCj are Σ-homomorphic, $i,j \in (1,n)$ with n number of duplicated entities.

THEOREM 1: Strong-mutual integrity theorem

A synchronization protocol ensures strong-mutual integrity if:

(i) transactions are atomic and there exists a TOTAL Σ-homomorphism (on output states).

(ii) conflicting transactions are serialized.

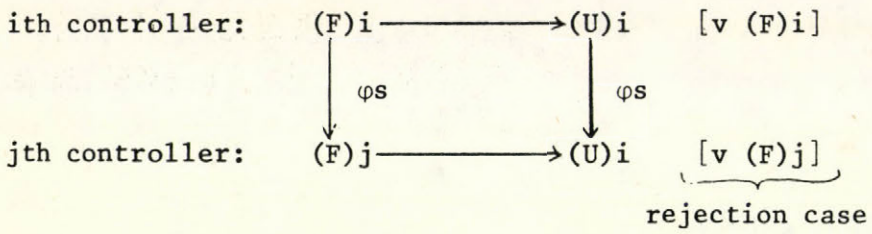
THEOREM 2: Weak-mutual integrity theorem

A synchronisation protocol ensures weak-mutual integrity if:

(i) transactions are atomic and there exists a PARTIAL Σ-homomorphism (on input states).

(ii) for every unstable state, there is a transition $U \longrightarrow F'$ on each controller.

(iii) for the Σ-algebras SYNC which are not Σ-homomorphic there exists a morphism φs ensuring the following diagram commutation.



(iv) conflicting transactions are serialized

NOTE: A morphism φs between two global-entity states represents a FUNCTIONAL CORRESPONDENCE (not necessarily a simultaneity) between the operators generating these states; as a matter of fact, the morphism is associated with the synchronization message sent during the generation of the considered state. The successive morphisms represent an event-sequencing scheme.

5. FORMALIZATION OF THOMAS' PROTOCOL

Interested readers are urged to get a look at the quoted references concerning this protocol since we shall only sum up their basic properties.

5.1 PRINCIPLE OF THOMAS' SOLUTION (THOM75), (THOM77)

THOMAS' protocol is based on a MAJORITY CONSENSUS mechanism. Each Data Base Manager Process (DBMP) - there is one DBMP in every site - VOTES on the acceptability of update requests. For a request to be accepted and applied to all data base copies, only a majority of DBMP's need to aprove it. The request is said to be RESOLVED when a majority of DBMP's accepted or rejected it.

The initical step of this protocol concerns the acquisition of the BASE-VARIABLES (BV's) by the originator of the request (called application program or A.P.) from any DBMP (INTREQ message). The DBMP replies to the AP (BVTS message) by sending it the BV values along with the attached time-stamps (a time-stamp represents the last modification date of a given BV). The AP calculates the new values of the BV called the UPDATE-VARIABLES (UV's).

The global update session is initialized by the sending of the update request (EXTREQ message) which encompasses both the set of BV's with their time-stamps and the set of UV's.

The initial version of this protocol (THOM75) used a DAISY-CHAIN during the resolution phase: each DBMP votes when receiving an EXTREQ, and forwards the request along with the accumulated votes to another DBMP that hasn't voted yet if a consensus is not reached. This procedure continues until the request is resolved (a request is said to be "pending" till this resolution). The voting rules basically amount to voting:

O (OK): Each BV is current and there is no conflict;
If a majority consensus is attained (on OK votes) the request is globally accepted; the acceptation is notified to each DBMP by the sending of the UPD message.

R (Rejected): There is an obsolete BV. The rejection is notified to each DBMP with the REJ message (one reject vote is enough to globally reject a request which could be later resubmitted).

A weak form of rejection not considered here, is proposed in (THOM77).

P (pass): Each BV is current but there exists a conflict with a higher priority pending request (which received an OK vote).

A conflict between Ti and Tj correspond to $(BV's)i \cap (UV's)j \neq \emptyset$.

If a majority consensus is obtained on PASS votes the request is globally rejected.

D (defer): defer voting when:

- either each BV is current but there exists a conflict with a lower-prority pending request.

- or request-BV time-stamps are more current than the local ones which are obsolete (a previous update was not yet performed).
  This case corresponds to the weak integrity feature.

  These deferred request are queued (in Q).

5.2 THOMAS' PROTOCOL FORMALIZATION

A controller is formalized by an ADT named SYNC whose signature is indicated in the following figure.

OBJECT	OPERATORS	COMMENTS
Global entity (GE)	PREPAREG (M, GES); SETG (M, GES); UNSETG (GES); ID (GES);	M : Messages GES: Global entity state ID: Identity operator

Fig. 2.
SYNC Signature

OBJECTS	OPERATORS	COMMENTS
Message (M) M∈ (EXTREQ, UPD, REJ)	TRANSMIT (M, PR,DBMPL, I,J); BROADCAST (M, PR, LIST); RECEIVE (M,PR,DBMPL,I,J, GES, PR); WAIT (M,PR);	DBMPL: List of controllers which voted IJ: counters LIST: identification of receiving controllers (may be "ALL") GES,PR: local parameters PR: priority we indicate the parameters of importance in a message transmission/ reception
Copy status (CS) CS (STB,USTB)	ID (CS)	
Global Request Status (GRS) GRS∈(Λ,P,A,R)	ID(GRS)	Λ : none P : pending A : accepted R : rejected
Local Request Status (LRS) LRS∈ (OK,D,PS,RJ)	ID (LRS)	OK : OK vote D : deferred PS : pass RJ : rejected
Local entity (LE)	PREPAREL (LE); SETL (LE); UNSETL (LE);	
TRANSACTION (T)	PROCESS (T)	

Fig.3.

Signatures of other involved types

Other basic operators used in the specifications are those attached to:

- INT, BOOL, like TEST (A,B): = IF A = B THEN TRUE ELSE FALSE
  SUP (A,B): = IF A > B THEN TRUE ELSE FALSE
- LIST (L) like APPEND (i,L),...
- QUEUE (Q) like ENQ,DEQ,EMPTY?,...

A global entity state (GES) is a 3-tuple <CS, GRS, LRS> which may take 4 basic forms:

(STB,Λ, -), (STB,P,-), (STB,R,-),(USTB,A,-)

where " - " represents a non-specified local parameter (OK,RJ,D, or PS).

The specifications of SYNC for THOMAS' protocol are given in Annex 1.

Parallelism

We want to express the parallelism between two participating controllers namely SYNCi and SYNCj; parallelism between controllers i and j will be depicted by a Σ-homomorphism between SYNCi and SYNCj.

In order to do so we introduce the following morphism φ associated with synchronization messages which defines a correspondance between remote global states:

$$(STB,\Lambda,-) \xrightarrow{\varphi[none]} (STB,\Lambda,-)$$

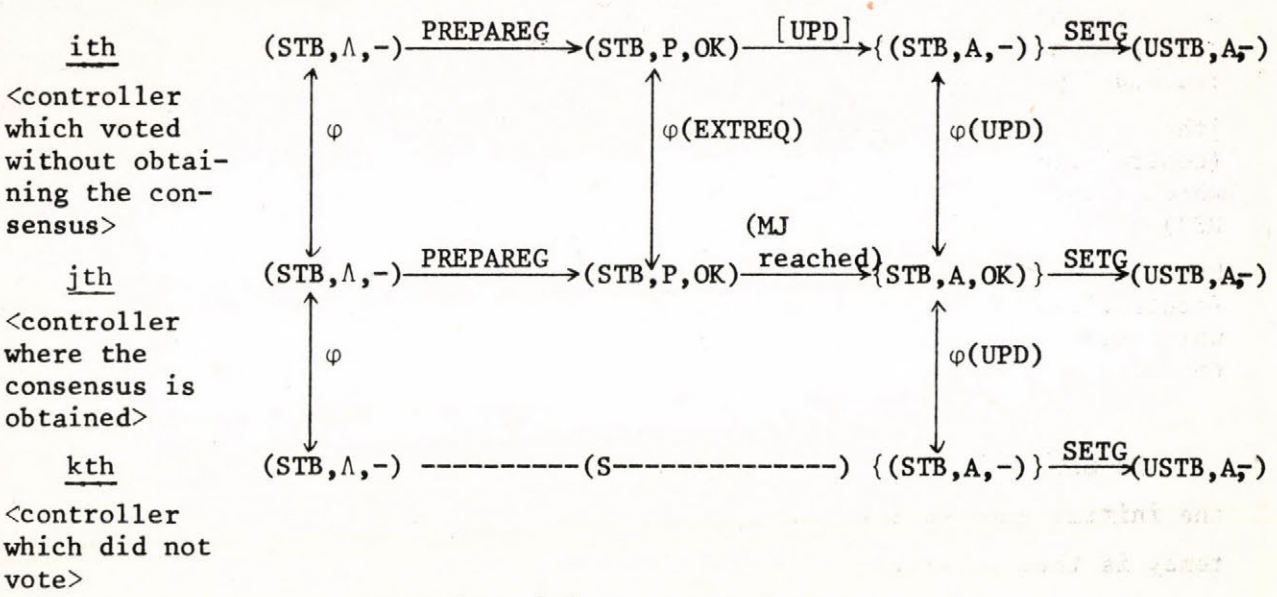
$$(STB,P,-) \xrightarrow{\varphi[EXTREQ]} (STB,P,-)$$

$$(USTB,A,-) \xrightarrow{\varphi[UPD]} (USTB,A,-)$$

Assertion 1: In an environment without concurrency conflicts, the protocol ensures weak mutual consistency

The verification of theorem 2 is straight-forward with (STB,Λ,-) = F and (USTB,A,-) = U. As a matter of fact we get the following diagram commutations (attached to a given transaction) by making use of the Σ-algebras equations.

NOTE: We represent the three types of controllers which may exist in the distributed system.

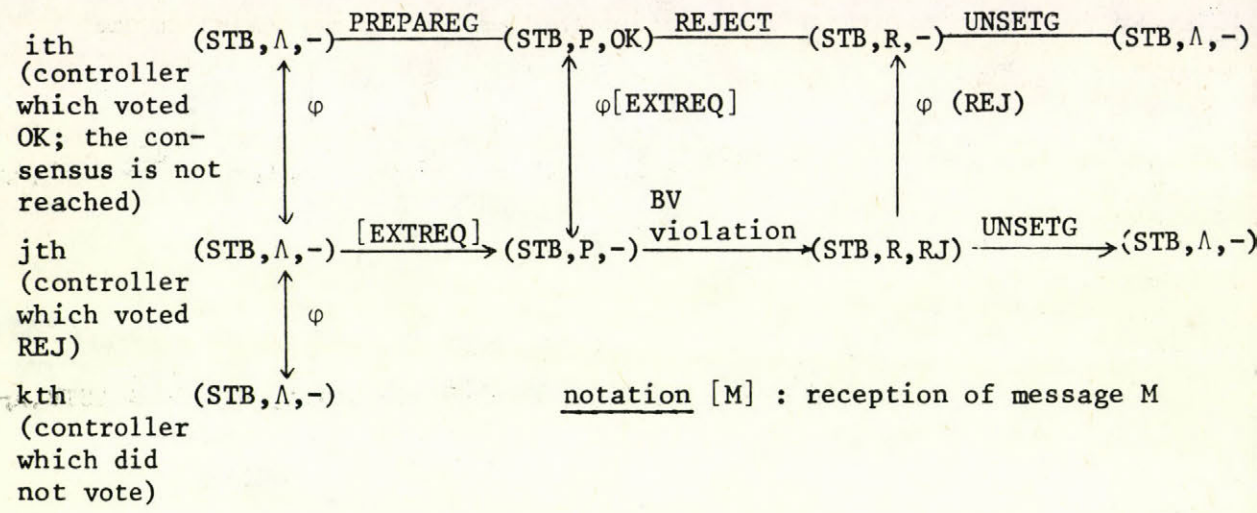


notation: [M] : reception of message M

In the following diagrams, we shall not represent the intermediate states (between {}) to alleviate the representations.

NOTE: If we integrate internal-integrity violation in this scheme we get the following commutation which ensures that the final global state (available for retrieval/update) is identical to the initial one.

We introduce the following morphism $(STB,R,-) \xrightarrow{\varphi(REJ)} (STB,R,-)$



No SETG has been performed; therefore, the final states correspond to the initial ones (before the synchronization session). The mutual consistency is then verified.

Assertion 2:

The protocol ensures weak mutual consistency when there is a finite set of concurrent conflicting transactions

Proof:

The proof of this assertion may be reduced to two concurrent transaction Ti and Tj (with PRi > PRj) since there is a total ordering of transactions.

Two major cases may occur depending on the fact that:

(i) The lower priority transaction (Tj) gets the majority consensus on OK before Ti.

(ii) Tj gets the majority consensus on PASS votes and is rejected.

We will use the indices i and j for φ to indicate the belonging of a correspondence to the ith or jth session. In ANNEX 2 Figure 4 (rejection of T and Figure 5 (rejection of Tj) depict the most general situations which can arise and show the weak mutual consistency.

6. CONCLUSION

The difference between this model and the others are a reflection of different goals; our model is an attempt to provide:

(i) a minimization of the primitive concepts: a framework for protocol uniformization/abstraction leading to the concept of synchronization-protocol transparency (this introduces a new degree of transparency to the four types of transparency presented in (TRAI79). Whatever the synchronization protocol is, we pointed out three generic primitive operators which represent the only knowledge of the inner and outer layers where the protocol is used; the operator semantics (depending on the protocol) is hidden and can be switched according to the suitability (strong or weak-mutual integrity,...) of the chosen protocol.

(ii) a way to express synchronization protocols clearly such that the effects of failures are formally specified (POPE79).

(iii) a basis from which SIMPLE (visualisable) and RIGOROUS proofs of correctness can be inferred.

(iv) a global architecture for a distributed-data-base integrity system; a synchronization protocol corresponds to a functional layer with a clear mapping to a local DBMS and to an END-to-END communication protocol.
The formalism is extendable to the functional-layer specification of a data-base-management system (models and manipulation languages) and a computer network (entities and protocols); our formalism is not constrained to synchronization protocols.
This represents a salient feature of the ADT-based approach.

(v) the expression of basic integrity concepts (mutual integrity theorems; serialization; atomicity,...).

Those points are not addresed in the other existing proposal made by ELLIS (ELLI77-b); ELLIS proposes a formalism based on L-SYSTEM for his protocol (in a fail-safe environment only); the specifications are visualizable and simple; however the proofs are very complex and dependent on the size of the network (small preferably).

There is another proposal made by WILMS (WILM79-a) with a formalism based on NUTT's networks but only concerned with the specification aspect. This article presents a new application of abstract data types; it aims at demonstrating that the ADT concept can be applied with a good profit to a typical distributed-data-base problem.

ACKNOWLEDGEMENTS

Special thanks are due to Gerry POPEK for invaluable discussions on this topic and to the numbers of the SIRIUS-INRIA group (animated by G. LE LANN) in which the ideas presented here were freely discussed and explored.

7. REFERENCES

(ALSB76) ALSBERG,P.A. "Multi-copy resiliency techniques"
University of Illinois research report, CCTC-VAD-6-505,May 76.

(BILL79) BILLER,H., EBERHARD,L. "On the evaluation of architectures and applications of distributed data base management system".
Seminar on distributed data sharing, Aix en Provence; France, May 15-17-1979.

(BURST77-a) BURSTALL,R.M., GOGUEN,J.A. "Putting theories together to make specifications".
Proc. 5th Int. Conference on aritifical intelligence.
Mit/1977, pp 1045-1058.

(BURST77-b) BURSTALL,R.M. "Design considerations for a functional programming language".
Proc. Infotech Symposium, Copenhaguen, Oct. 1977.

(BUST78) BUSTA,J.M. "Integridad externa da una base de datos repertida en una red de ordenadores general heterogénea".
PH-D Thesis, Univ.de Santiago de Compostela, Spain 1978.
(CITEMA award of the best doctoral thesis in Spain.)

(ELLI77-a) ELLIS,C.A. "A robust algorithm for updating duplicated data bases".
Workshop on distributed systems, Berkeley, 1977, pp.146-160.

(ELLI77-b) ELLIS,C.A. "Consistency and correctness of duplicate data systems".
Proc. of the 6th Symposium on OS principles, 16-18.

(GARC79) GARCIA-MOLINA,H. "Performance of update algorithms for replicated data in a distributed data base".
PH-D Thesis Univ. of Stanford, June 1977 (report n° STAN-CS, 79-744).

(GRAY75) GRAY,J. et al. "Granularity of locks and degrees of consistency in a shared data base".
Modelling in DBMS, G.M. Nijssen editor, North Holland, pp.365-394 (also IBM research report RJ 1606, 1975).

(GRAY78) GRAY,J. "Notes on data base operating systems"
Lecture notes in computer science, edited by J.Hartmanis, Springer Verlag, Berlin, New York, 1978, pp. 394-481.

(GRAY79) GRAY,J. "A discussion of distributed systems".
Proc. AICA, Bari, Oct. 10-13, 1979, pp. 204-211.

(GOGU76) GOGUEN,J.A. et al."An initial algebra approach to the specification, correctness and implementation of abstract data types".
IBM research report RC-6487, Oct. 1976.

(GOGU78) GOGUEN,J.A. "Some design principles and theory for OBJ-0 a language for expressing and executing algebraic specifications of programs".
Proc. Int. Conference on Mathematical studies of Information processing, Kyoto, Japan, 1978, pp. 429-475.

(GOGU78-a) GOGUEN,J.A., BURSTALL,R.M. "Some fundamental properties of algeraic theories: a tool for semantics of computation".
UCLA research report, July 1978..

(GOGU78-b) GOGUEN,J.A., TATCHER,J.W., WAGNER,E.G. "An initial algebra approach to the specification, correctness and implementation of ADT".
Current trends in Programming Methodology, Vol n° 4, Data Structuring (Ed. by R.Yeh) Prentice Hall 1978, pp.80-149.

(GUTT78) GUTTAG,J.V. et al. "Abstract data types and software validation"
CACM, Dec. 1978, Vol 21, number 12, pp. 1048-1064.

(HOAR72) HOARE,C.A.R. "Proof of correctness of data representations". Acta Informatica, 1,4 (1972) pp. 271-281.

(LELA76) Le LANN,G. "Introduction à l'analyse des systèmes multi-référentiels" Thèse d'Etat, Univ. de Rennes, France, Mai 1976, 198. p.

(LELA77) LE LANN,G. "A protocol to achieve distributed control in failure tolerant multi-computer systems". SIRIUS research report, IRIA, CTRI 002-1977.

(LELA78-a) LE LANN,G. "Algorithms for distributed data sharing systems which use tickets" SIRIUS research report, SYN 1003, 1978. (Proc. 3rd Berkeley Workshop, San Francisco, August 1978, pp. 259-272.

(LELA78-b) LE LANN,G. "Pseudo-dynamic resource allocation in distributed data bases" Proc. of the ICC, Kyoto, Japan, Sept. 1978

(LELA79) LE LANN,G. "Consistency and concurrency control in distributed data base systems". Presented at the ECC course on distributed data base Sheffield (U.K.) July 9-20, 1979 (SIRIUS Report, INT 1-1007.)

(MENA77) MENASCE,D., MUNTZ,R., POPEK,G. "A locking protocol for resource coordination in distributed systems" UCLA research report, 1977

(MILN71) MILNER,R. "An algebraic definition of simulation between programms". Stanford AI project, Memo AIM - 143, Feb 1971, 21 pp.

(MIRA79) MIRANDA,S., BUSTA,J. "Interference problem in distributed data systems". Proc.5th Conf. on the theory of operating systems Visegrád, Hungary, Jan 31. Feb 3, 1979.

(MIRA80-a) MIRANDA,S. "Specification and validation of two ring-structure synchronization protocols for distributed data bases". Proc.Int. Symposium on distributed data bases, Paris, March, 12-14, 1980.

(MIRA80-b) MIRANDA,S. "Aspects of data security in general-purpose data base management systems". Proc. 1980 Symp. on data security and privacy, IEEE, April 14-16, 1980, Berkeley, CA.

(MIRA80-c) MIRANDA,S. "Architecture formelle d'un système d'intégrité pour une base de données répartie". Thèse d'Etat, Univ. Paul Sabatier, Toulouse, France, Sept.1980.

(MIRA80-d) MIRANDA,S. "DLP:a fault-tolerant decentralized locking protocol for distributed data bases". Proc. FTCS,10,Kyoto, Japan. Oct. 1980.

(MONT78) MONTGOMERY,W.A. "Robust concurrency control for a distributed information system" PH-D Thesis, MIT/LOCS/TR-207, Dec 1978.

(NOUR79) NOURANI,F."Constructive extension and implementation of abstract data types and algorithms" PH-D Thesis, UCLA,ENG-7945, 172 p. August 1979.

(PAOL77-a) PAOLINI,P.,PELAGATTI,V. "Formal definition of mappings in a data base" Proc. ACM SIGMOND Conference, Toronto, 1977, pp. 40-46.

(PAOL77-b) PAOLINI,P.,PELAGATTI,V. "Formal definition of data models". Instituto d'Ellectrotecnica-Politecnico de Milano, Research report, n° 77-2, Feb. 1977.

(POPE79) POPEK,G., MIRANDA,S. "Formal solution to the external integrity problem in distributed systems using abstract data types". UCLA and CERISS research report,n° 79038, May 79 (revised version forthcoming).

(SEGU78) SEGUIN,S. "Traitement distribué d'informations réparties dans un réseau d'ordinateurs"
Thèse d.Etat, INPG, Marc 1980.

(TARD77) TARDO,J. "OBJ and the automatic implenetation of abstract data types".
PH-D Thesis, UCLA 1977

(TARD79) TARDO,J. et al. "A practical method for testing algebraic specifications".
UCLA quarterly report, Dept, of computer science, UCLA, pp. 59-71.

(THAT76) THATCHER,J.N. et al. "Specification of ADT using conditional axioms".
IBM research report, RC 6214. Sept. 1976.

(THOM75) THOMAS,R.H. "A solution to the update problem for multiple copy data bases which use distributed control"
BBN report ≠ ≠ 3340, July 1975.

(THOM77) THOMAS, R.H. "A majority consensus approach to concurrency control for multiple copy data bases"
BBN report,n° 3733, Dec. 1977, 51.p.

(TRAI79) TRAIGER,I.L. et al. "Transactions and consistency in distributeddata systems"
IBM research report, 1979, (to appear)

(WILM79-a) WILMS,P. "Etudes d'algorithmes de cohérence d'informations dupliquées et réparties" -Formalization à l'aide des rèseaux de Nutt".
Research report, INP Grenoble, RR n° 160, 160 bis, Feb. 79.

(WILM79-b) WILMS,P. et Al "A majority consensus algorithm for the consistendy of duplicated and distributed data" Research report, Univ. of Grenoble, March 79, (Proc.1st Int. Conf. on DCS, Huntsville, Alabama, Oct. 1-5, 1979).

(WULF76) WULF, W.A. et Al "Abstraction and verification in ALPHARD" Internal Report, 1976, CACM, Vol n° 20,8, pp. 553-563, August 1977.

ANNEX 1: Specifications of THOMAS' protocol in an OBJ- close language

OBJECT SYNC (ith controller)

SORTS GES,LE,INT,BOOL,T,M, PR,LIST,QUEUE,GRS,LRS,CS

OPS
PREPAREG : M × GES ⟶ GES
SETG : M × GES ⟶ GES
UNSETG : GES ⟶ GES
ID : GES ⟶ GES

VARS
PR : INT <priority (PR,i) will be noted Pi>
i : INT <controller number>
STB: CS ; USTB : CS ;
P : GRS;A : GRS; R : GRS; : GRS;
OK : LRS;D : LRS; PS : LRS; RJ : LRS;
EXTREQ : M < the EXTREQ is built by the AP with the BVTS message>
UPD : M < the updata message is broadcasted to each DBMP>
REJ : M < the REJECTION message>
We do not consider the messages exchanged with the AP
PS# : INT <counter of pass votes>
OK# : INT <counter of Ok votes>
MJ : INT <majority number>
DMPL : LIST <list of DBMPs which voted>
Q : QUEUE <queue of deferred conflicting requests>

SPECS

<EXTREQ, reception>

```
RECEIVE (EXTREQ, PRk,DBMPL,PS#, OK#,; (STB,Λ,-),-):=ID(GRS)=P;ID(GES)=(STB,P,-);
                        IF <current base variables>
                        THEN PREPAREG((EXTREQ,PRk,PS#,OK#),(STB,P,-));
                        ELSE BROADCAST (REJ,PRk,DBMPL);
                             ID(LRS)=RJ; ID(GRS)=R; ID(GES)=(STB,R,RJ);
                             UNSETG (REJ,PRk; (STB,R,RJ));
```

```
RECEIVE (EXTREQ,PRk,DBMPL,PS≠,OK≠; (STB,P,-),PRj):=ID(GRS)=P;ID(GES)=(STB,P,-);
        IF <current base variables>
        THEN IF <conflicting updates>; <(BV's)k ∩ (UV's)j≠∅>
        THEN IF  SUP (PRk,PRj) = TRUE
        THEN ID(LRS) = D; ID(GES) = (STB,P,D);
             ENQ (Q,(EXTREQ,PRk));
             TRANSMIT (EXTREQ,PRk,DBMPL,PS≠,OK≠); WAIT (EXTREQyUPDyREJ,PRk);
        ELSE IF TEST (PRk,PRj) = TRUE  revote
             THEN PREPAREG ((ESTREQ,PRk,-,OK≠),(STB,P,-));
             ELSE ID(LRS) = PS; INCR(PS≠); APPEND (i,DBMPL);
                  IF TEST (PS≠,MJ) = TRUE
                  THEN ID(GRS) = R; ID(GES) = (STB,R,PS);
                       BROADCAST (REJ,PRk,DBMPL);
                       UNSETG ((EXTREQ,PRk), (STB,R,PS));
                  ELSE TRANSMIT (EXTREQ,PR,DBMPL,PS≠,OK≠),
                       ENQ(Q,(EXTREQ,PRk)); WAIT (UPDvREJ,PRk);
             ELSE PREPAREG((EXTREQ,PRk,PS≠,OK≠), (STB,P,-));
        ELSE ID (LRS) = RJ; ID(GRS) = R; ID(GES) = (STB,R,Rj);
             BROADCAST (REJ,PRk,DBMPL);
             UNSETG((EXTREQ,PRk),(STB,R,RJ));

RECEIVE (EXTREQ,PRk,DBMPL,PS≠,OK≠); (STBvUSTAB,A,-),PRJ):=ID(LRS)=RJ;
                                    ID(GES)=(STBvUST,R,-) ID(GRS)=R;
                                    BROADCAST (REJ,PRk,DBMPL);
                                    UNSETG((EXTREQ,PRk);(STBvUSTB,R,RJ));

<UPD reception>
RECEIVE (UPD,PRk,-,-,-; (STB,ΛvP,-),PRj) :=
        IF  TEST (GRS,Λ) = TRUE
            THEN PREPAREL(LE); <the DBMP did not participate in the voting>
            ELSE (DEQ(Q,(EXTREQ,PRk)); IF Test (PRj,PRk)=FALSE THEN (ID(GRS)=R;
                                       UNSETG((EXTREQ,PRj),(STB,R,-)));
```

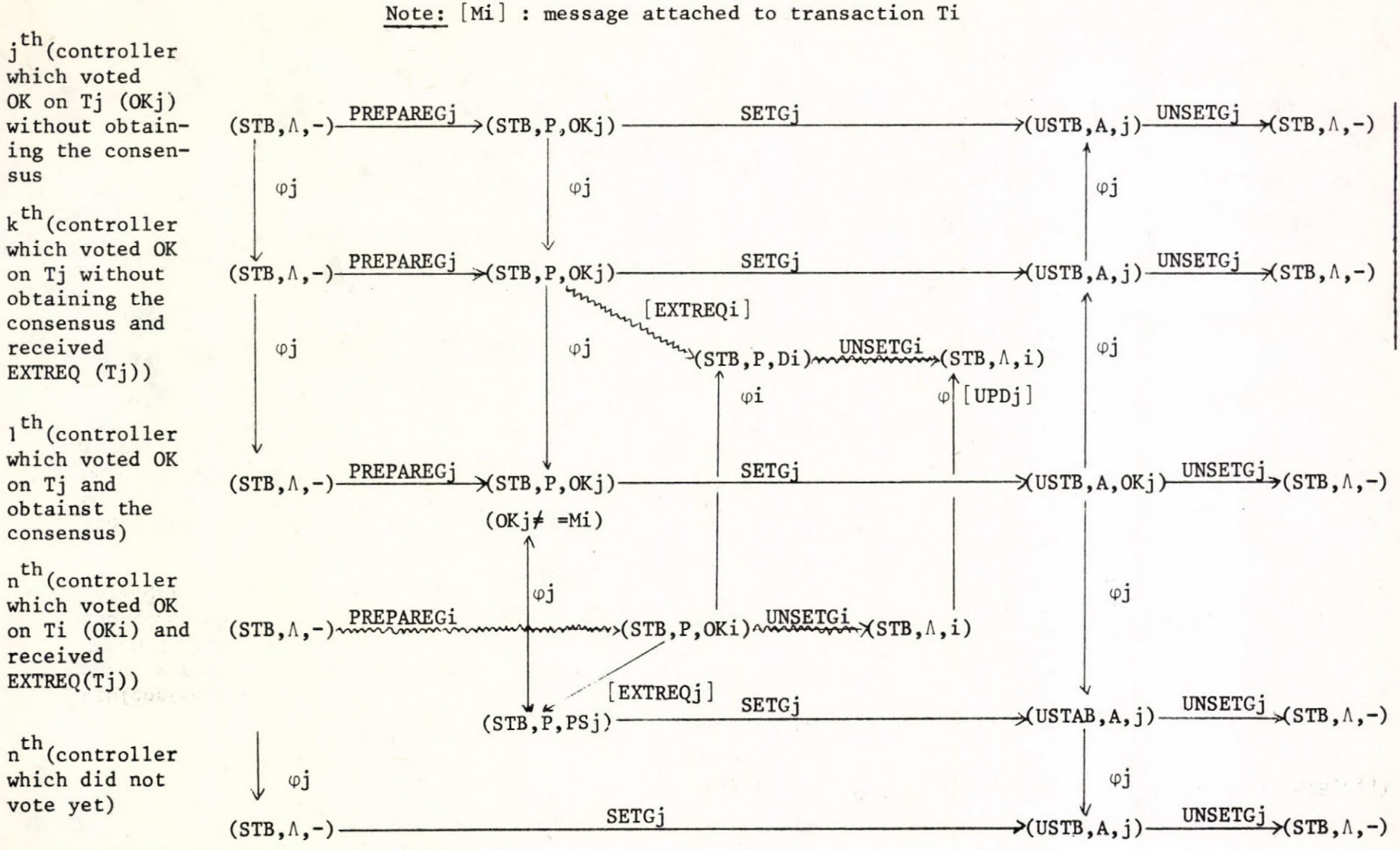
```
        ID(GRS) = A; ID(GES) = (STB,A,-);
           SETG(UPD,(STB,A,-)) ;)
           LOOP WHILE EMPTY ? (Q) = FALSE
                DEQ (Q,(EXTREQ,PRi)); ID(GRS)=R; ID(GES) = (STB,R,-);
                UNSETG((EXTREQ,PRi),(STB,R,-)); <all conflicting transactions
                                                 are rejected>
           ENDLOOP

<REJ reception>
RECEIVE (REJ,PRk,-,-,-,;(STB,P,-),PRk) : =
                        ID(GRS) = R; ID(GES) = (STB,R,-);
                        UNSETG (REJ,PRk;(STB,R,-));
                        IF EMPTY? (Q) = FALSE
                        THEN WAIT (EXTREQ,PRj); <for revote>
            <This set of operations will be referred in the diagrams as
             REJECT>
PREPAREG((EXTREQ,PRk,PS≠,OK≠),(STB,P,-)):=INCR(OK≠);ID(LRS)=OK;APPEND(i,DBMPL)
                        ID(GRS) = P; PREPAREL(LE);
                        IF TEST (OK≠,Mi) = TRUE
                        THEN ID(GRS) = A; ID(GES) = (STB,A,OK);
                             BROADCAST(UPD,PRk,ALL);
                             SETG (UPD,PRk;(STB,A,OK));
                        ELSE TRANSMIT (EXTREQ,PRk,DBMPL,PS≠,OK≠);
            ENQ(Q,(EXTREQ,PRk); WAIT (UPDvREJ,PRk);

SETG(UPD,PRk,(STB,-,-):= SETL(LE<PROCESS(T),...>; ID(CS)=USTB; ID(GRS)=A;
                                                   ID(GES) = (USTB,A,-);
                        UNSETG(-,-,;(USTB,A,-));

UNSETG(,-,-;(-,-)) : = UNSETL(LE); ID(CS) = STB; ID(GRS)=Λ; ID(GES)=(STB,Λ,-);
```

TCEJBO

Notes: (i) received messages are ignored in other configurations (in fact they could be treated like error conditions)

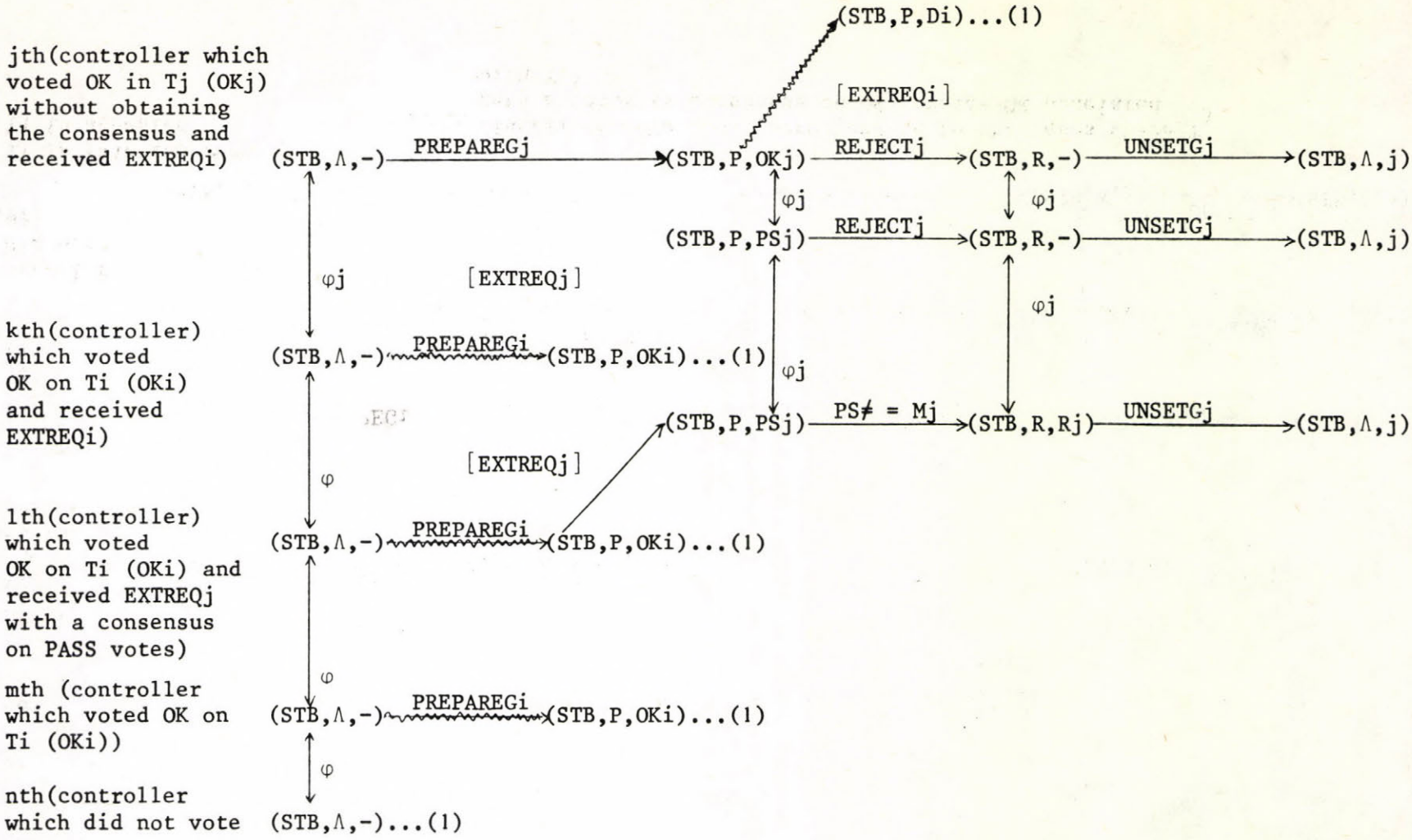
(ii) We do not consider local tests made on current BVs before local update is performed; these tests are related to robustness.

(iii) BROADCAST is here equivalent to TRANSMIT [(n-i) messages] The REj message is broadcasted to the DBMP's which voted (whose identification is in DBMPL) while the UPD message is broadcasted to every concerned DBMP.



Note: Ti is rejected when Tj is accepted

Fig.4. Diagram commutations corresponding to the cases where T_j gets a majority consensus on OKj (votes OK assciated with Tj)



Notes: (1) As soon as majority consensus is obtained on OK votes for Ti, we shall get the same diagram commutations as in the assertion 1.

(2) Here we introduced the morphism: $(STB,R,-) \xrightarrow{\varphi[REJ]} (STB,R,-)$ corresponding to the REJ message.

Figure 5. Diagram commutations corresponding to the case where Tj gets a majority consensus on PSj (Pass votes associated with Tj).

Összefoglalás

Az irodalom kiterjedten foglalkozik osztott adatbázisok szinkronizációs problémáival és sok szinkronizációs protokolt terveztek már. A cikkben absztrakt adattipusokon (algebrai módszeren) alapuló formális megközelítést mutatunk be és kidolgozunk egy egységes módszert a szinkronizációs protokolok leírására és ellenőrzésére. Az eredményeinket egy alapvető protokolon mutatjuk be, amely a megoldások egy széles osztályát reprezentálja.

Р Е З Ю М Е

Формальное описание синхронизации раздельных баз данных

В литературе обширно излагаются синхронизационные задачи разделенных баз данных. Многочисленные протоколы были проектированы. В настоящей работе предлагается формальный подход, основанный на абстрактных типах данных /на алгебраическом методе/ и разрабатывается единый метод для описания и проверки синхронизационных протоколов. Результаты иллюстрируются основным протоколом, представляющим широкий класс решений.