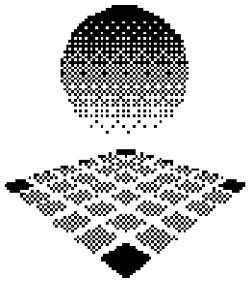


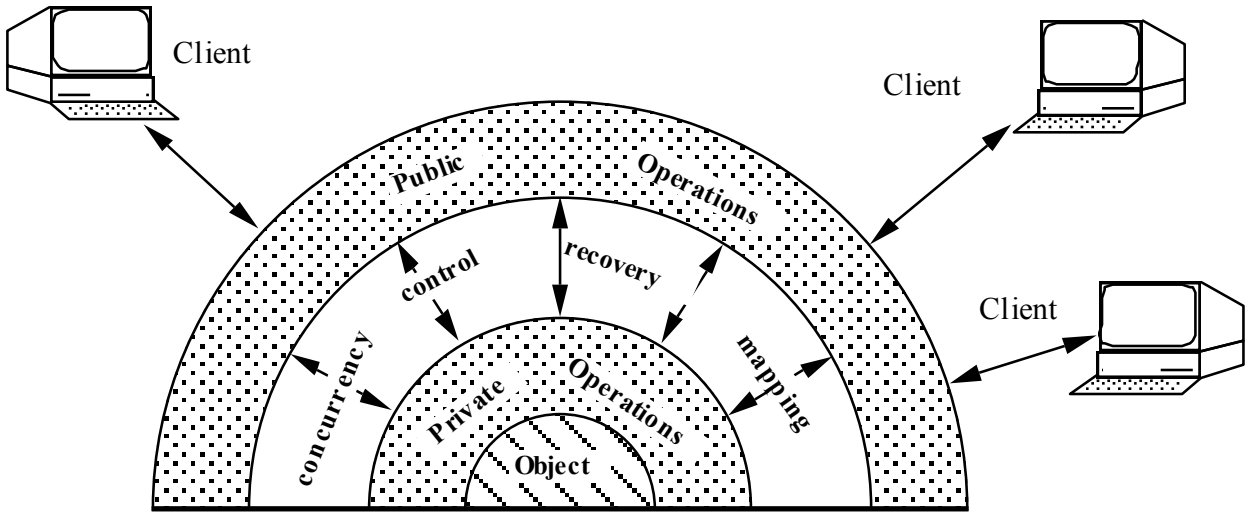
UNIVERSITÉ MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC
Laboratoire de Systèmes Informatiques

Tuple-based Abstract Data Types: Full Parallelism

Malta, C., Martinez, J.

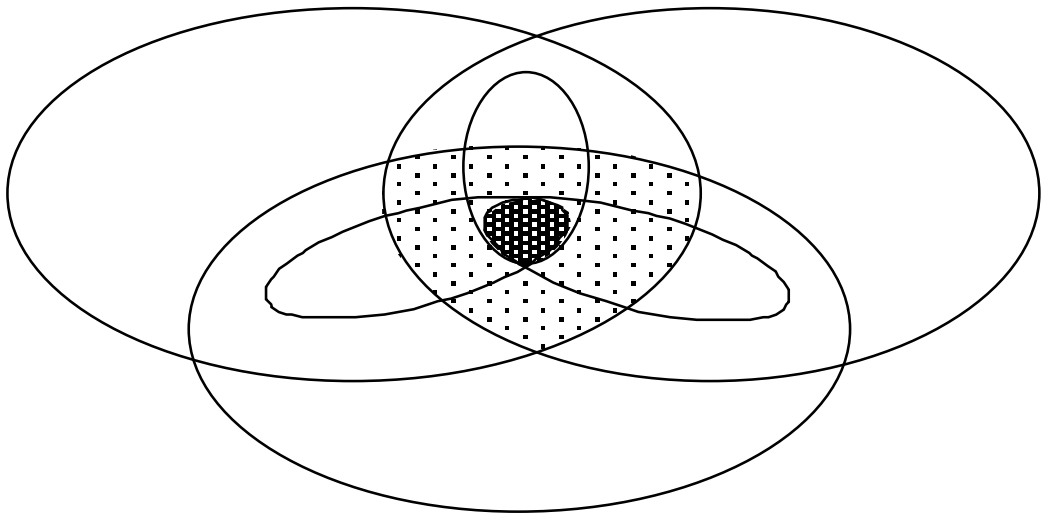
A general framework using serializability and 2PL

Goal: To provide maximal commutativity.



Used to implement several ADTs of interest: boolean, set, map, b-tree, and several kinds of counters.
Though implemention of the operations is simplified, there is still a lot of work to do in between the two interfaces.

A strong limitation for *arbitrary* objects



The major critics against implementing fine commutativity relations is that for *arbitrary* objects the behaviour is comparable to the one obtained with compatibility (only read and write access modes.)

The Idea

Use the ***syntactic*** information given in part

by the structure of the object

and in part

by the implementation of its operations

to provide ***automatic*** concurrency control and recovery.

Domain of Application

Tuple-based ADTs because

fields of a tuple are strongly connected
(e. g., by functional dependencies),

therefore, it is often unnecessary to look for interesting commutativity relations.

Examples: Two extremes

The "Address" ADT:

```
tuple of
    Number: integer;
    Street: string;
    ZIP: string;
    City: string;
end tuple;
```

The "Square" ADT

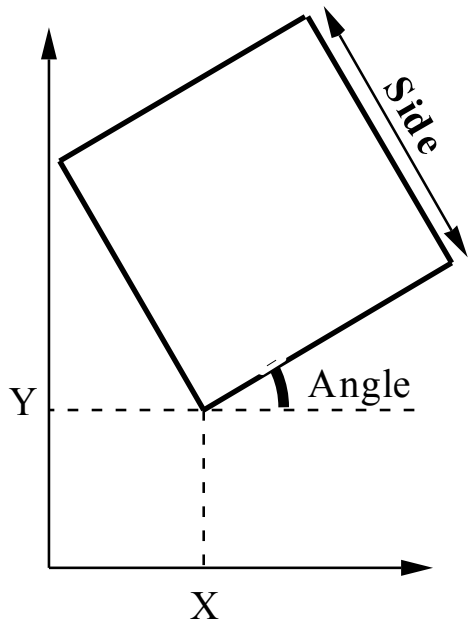
Access Vectors: The "Square" ADT

A square is represented by four attributes:

X, Y, Side, and Angle

and operations may be:

Move, Rotate, Extend, Display, ...



Rotate (Da) **is**
    if Da **mod** 2p ≠ 0
    then Angle := (Angle + Da) **mod** 2p
Extend (DS) **is**
    if DS ≠ 0 **thenif** Side + DS ≥ 0
                then Side += DS
                else Side := 0

DAV	X	Y	Side	Angle
Rotate	Null	Null	Null	Write
Extend	Null	Null	Write	Null

lemma 1

A set of concurrent operations pairwise commute if, and only if, for each field:

- there is at most one writer and no reader or,
- there is exclusively readers or,
- there is neither writers nor readers.

DAV	X	Y	Side	Angle
Rotate	Null	Null	Null	Write
Extend	Null	Null	Write	Null
Move	Write	Write	Null	Null

1 writer	**1 writer**	**1 writer**	**1 writer**
no reader	no reader	no reader	no reader

corollary 1

The actual execution of the operations can be done is *full parallelism*, (i. e., without controlling its atomicity), because this kind of commutativity is just extended compatibility.

corollary 2

Commutativity of an incoming operation can be controlled in *constant time* by using two control vectors: one for controlling the presence of a writer, another for counting the number of readers.

Controlling Operations: Downgrading

Problem

DAV being defined *a priori*, at compile-time, are pessimistic and do not provide really commutativity but just extended compatibility.

Solution

We introduce Dynamic Access Vectors (DynDAV), computed at run-time, which detect which fields have not been actually used.

property 1

DynDAV(OP) ≤ DAV(OP)

DAV(S.Move) = (Write, Write, Null, Null)
DynDAV(S.Move(5,0)) = (Write, Null, Null, Null)

DynDAV	X	Y	Side	Angle
Rotate(2p)	Null	Null	Null	Null
Extend(0)	Null	Null	Null	Null
Move(5,0)	Write	Null	Null	Null

1 writer	no writer	no writer	no writer
no reader	no reader	no reader	no reader

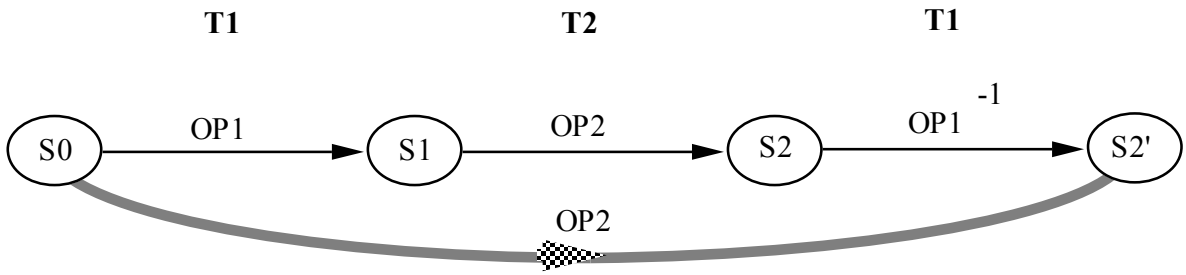
lemma 3

Downgrading does not invalidate serializability.

lemma 4

DynDAV and downgrading offer a special kind of conditional commutativity.

Controlling Operations: Inverse Operations



property 2

$DAV(OP^{-1}) \leq DynDAV(OP) \leq DAV(OP)$

DAV(OP) = (Write, Null, Read, Write, Read, Read)

DynDAV(OP) = (Read, Null, Null, Write, Read, Null)

$DAV(OP^{-1})$ = (Null, Null, Null, Write, Null, Null)

corollary 3

Inverse operations need not be controlled at all and can also be executed in full parallelism.

[Eswaran et al. 76]

Access vectors as here.
Not implemented in System R. Reasons may be that:
- access vectors have to be generated at run-time;
- decomposition of objects on several relations achieves a rough form of access vectors.

[Noe et al. 87]

Only Dynamic Access Vectors.
Only from the recovery point of view.

[Badrinath & Ramamritham 88]

A graph structure rather than vectors.
Commutativity extracted from specifications.
Only from the concurrency control point of view.

[Agrawal & El Abbadi 92]

In object-oriented databases.
For concurrent class definition modifications.
Only from the concurrency control point of view.

Summary

Automatic technique on tuple-based ADTs.

Not to take the place of the previous framework.

Eliminates the drawbacks of other proposals.

Offers full parallelism.

Issues

Multiprocessor machines.

Neither multilevel transactions, nor ARIES allow this kind of concurrency.

Extension to an object-oriented data model (i. e., essentially inheritance) are in progress.