The lattice of database states

David Barrett-Lennard

16 Oct 2016

# Introduction

This document describes some of the mathematical aspects of the CEDA replication technology. It is only provided for information purposes for those interested in knowing more about how it works, it does not need to be understood by software developers using CEDA to build applications.

For convenience some mathematical definitions are provided in an appendix.

## State transition diagram

Given a scenario of some sites exchanging operations there is an associated state transition diagram. In the case of two sites exchanging two concurrent operations the state transition diagram is a diamond.

In the following diagram there are two sites A and B that are initially in the same state S0. Site A generates and applies operation O1 and transitions to state S1. Site B concurrently generates and applies operation O2 and transitions to state S2. Each site sends its operation to the other site. The received operation is transformed and applied and both sites enter state S3.

O1

O2

O2’

O1’

Site A

Site B

S0

S0

S1

S2

S3

S3

O1

O2

O2’

O1’

S0

S1

S2

S3

convergent

convergent

divergent

state

transition

## Correspondence between states and sets of operations

Each *state* can be labelled with the *set of operations* producing that state, so there is a one to one correspondence between states and sets of operations in the state transition diagram.

O1

O2

O2’

O1’

{}

{O1}

{O2}

{O1 O2}

With n concurrent operations there are 2n states

2 0 = 1

2 1 = 2

2 2 = 4

2 3 = 8

2 4 = 16

2 5 = 32

## Correspondence between paths and lists of operations

A *list of operations* corresponds to a *path* through the state transaction diagram. In the diamond there are two paths (along the top and along the bottom), corresponding to the two permutations of a list with two elements.

O1

O2

O2’

O1’

[O1 O2’]

[O2 O1’]

With n concurrent operations there are n! paths

0! = 1

1! = 1

2! = 2

3! = 6

4! = 24

5! = 120

## State transition diagram is the Hasse diagram of subset

It has already been pointed out that there is a one to one correspondence between states and sets of operations. The subset operation on the sets of operations gives a partial order on the states. The Hasse diagram (see the appendix for a definition) of the subset relation corresponds to what we have previously described as the state transition diagram.

{}

{O1}

{O2}

{O1 O2}

⊆ is a partial order on the states

S1

S2

S1⊆S2

When there are three concurrent operations we get a cube.

{O1 O2 O3}

{O2 O3}

{O1 O3}

{O1 O2}

{O1}

{O2}

{O3}

{}

Poset (P,⊆) is a bounded lattice

|P| = 23 = 8

Note that the powerset (set of all subsets) of {O1 O2 O3} has 8 elements:

P = { {} {O1} {O2} {O3} {O1 O2} {O1 O3} {O2 O3} {O1 O2 O3} }

## Vector time

A vector time identifies a database state by identifying the set of operations producing that state. In the following diagram the vector time (2,1,1) identifies the set of operations { a1 a2 b1 c1 }. If we think of the operations as events then a vector time is like a time stamp in the way it separates past and future events

a1

a2

a3

b1

b2

c1

c2

Site A

Site B

Site C

past

future

A vector time is a multi-dimensional notion of time and is only partially ordered. Let i be a site identifier.

v[i] = ith component of vector time v

= number of operations generated by site i in the state associated with v

Let v1, v2 be vector times. We define a partial order (≤) on vector times as follows:

v1 ≤ v2 ⇔ ∀i v1[i] ≤ v2[i]

⇔ (set of operations of v1) ⊆ (set of operations of v2)

## Intersection of two states

The intersection of two states means the operations in common (shown here in green) and corresponds to a green dotted line which "cuts" down through the left of the "cuts" of the two given states.

a1

a2

a3

b2

b3

c112

c2

Site A

Site B

Site C

(3,1,2)

(1,2,0)

a4

b4

b1

b5

(1,1,0)

## Union of two states

Similarly the union of two states corresponds to a cut down through the right of the cuts of the two states.

a1

a2

a3

b2

b3

c1

c2

Site A

Site B

Site C

a4

b4

b1

b5

(1,2,0)

(3,1,2)

(3,2,2)

## Meet and join of vector times

There are corresponding operations on vector times involving taking the minimum/maximum over the components.

v1=(1,2,0)

v2=(3,1,2)

v1↑v2 = (3,2,2)

v1↓v2 = (1,1,0)

*join* of v1 and v2

(v1↑v2)[i] = **max**(v1[i], v2[i])

*meet* of v1 and v2

(v1↓v2)[i] = **min**(v1[i], v2[i])

## Precedes relation

There is a concept of an operation happening before another operation. We say O1 precedes O2 (and write this as O1→O2) if O1 had been applied on the site where O2 was generated.

In the following diagram operations a1 and b1 *precede* a2 because both a1 and b1 had been applied on site A at the time a2 was generated on site A.

a1

a2

b1

Site A

Site B

a1 → a2

b1 → a2

“precedes”

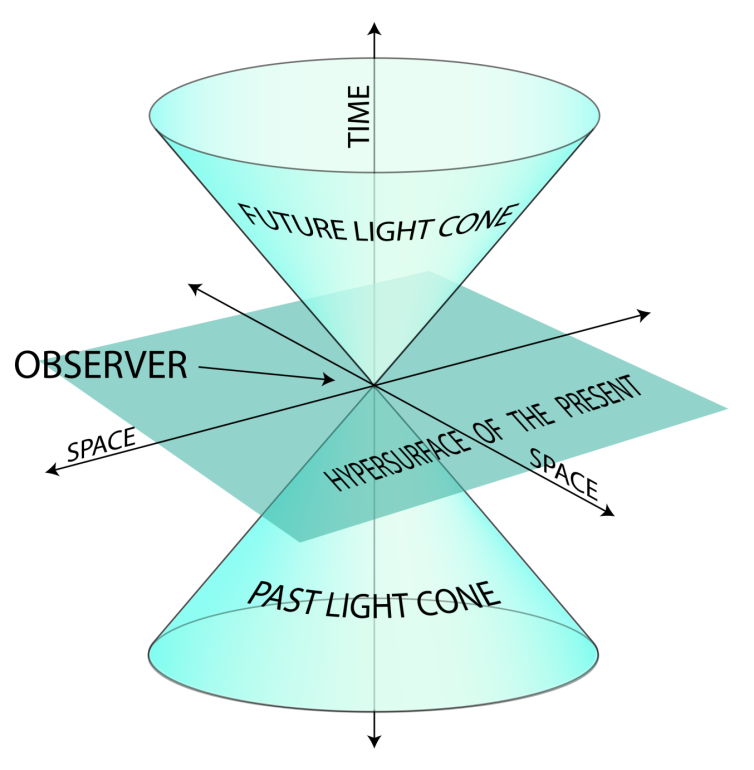
The precedes relation is a strict partial order on the operations:

* → is irreflexive
* → is asymmetric
* Causality preservation implies → is transitive

The precedes relation is not a total order, in fact we write O1||O2 when O1 and O2 are concurrent, meaning neither O1→O2 nor O2→O1.

## Analogy to special relativity

If one considers operations to be events that occur in space and time, then the precedes relation is analogous to the partial order on events in special relativity. In relativity events can only affect events on or inside its future light cone. Events that are not in each other’s light cone are “space like” and are not comparable.



[Image by K. Aainsqatsi at en.wikipedia. CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2210907]

## Don’t confuse the two posets

Consider the following example where there are two sites:

a1

a1

b1

Site A

Site B

There are two different posets which are being discussed. On the left are the operations partially ordered by the precedes relation and on the right are the states/vector times/sets of operations partially ordered by the subset relation. The Hasse diagram on the right is the state transition diagram.

a1 → a2

b1 → a2

a2

a1

b1

(operations,→)

{a1,b1}

{a1}

{b1}

{}

{a1,b1,a2}

(states,⊆)

## More complicated example

This is a more complicated example. Later in this document the state transition diagram will be shown.

a1

a2

a3

b1

b2

c1

c2

Site A

Site B

Site C

*precedes*

a1 → a2

b1 → a2

b1 → b2

c1 → b2

b1 → c2

c1 → c2

a1 → a3

a2 → a3

b1 → a3

b2 → a3

c1 → a3

*concurrent*

 a1 || b1

a1 || b2

a1 || c1

a1 || c2

b1 || c1

c1 || a2

a2 || b2

a2 || c2

a3 || c2

a1

a2

a3

b1

b2

c1

c2

Hasse diagram of (operations,→)

## Causality Preservation

Causality preservation requires the following condition to be met:

If O1→O2 then O1 is applied before O2 at all sites

In this diagram a1 and b1 both precede a2. Causality preservation requires that both a1 and b1 be applied before a2 at all sites. So there is no point sending a2 to site C before sending a1 and b1.

a1

a2

b1

Site A

Site B

Site C

a1 and b1 must be applied before a2

## States that preserve causality

There's a corresponding definition of the states which preserve causality:

S preserves causality ⇔ (O2∈S ∧ (O1→O2 ⇒ O1∈S))

The vector time shown in red doesn't preserve causality because there's an operation in its future b2 which precedes an operation a3 in its past

a1

a2

a3

b1

b2

c1

c2

(2,1,1)

Site A

Site B

Site C

(3,1,2)

## The states that preserve causality form a bounded lattice

It can be proven that the states which preserve causality are closed over intersection and union. In other words, let S1 and S2 be states then

S1 and S2 preserve causality ⇒ S1∩S2 and S1∪S2 preserve causality

S1∪S2 is the smallest set which contains both S1 and S2 so it's the supremum. S1∩S2 is the largest set contained by both S1 and S2 so it's the infimum.

S1

S2

S1∪S2

S1∩S2

supremum of {S1,S2}

infimum of {S1,S2}

S1

S2

S1∩S2

S1∪S2

The existence of the infimum and supremum means the states that preserve causality form a mathematical structure called a *lattice.* This is shown in the lower part of the picture below having 5 cubes plus a flat face for this particular scenario with three sites.

This lattice has 24 vertices (i.e. vector times or states or sets of operations) which correspond to 24 different cuts in the top picture (only three are shown with green dotted lines). The red vector time doesn't preserve causality and therefore doesn't correspond to a point on the lattice in the picture below.

Site A is applying operations in a particular order and this corresponds to one of the paths (the one drawn in orange) which is zig-zagging its ways though the lattice, starting at (0,0,0).

a1

a2

a3

b1

b2

22

c1

c2

Site A

Site B

Site C

A

C

B

(0,0,0)

(0,0,0)

(3,2,2)

(3,2,2)

## Paths through the lattice

A path through the lattice is defined by an ordered list of operations. In the following diagram two alternative paths are shown in red and green.

a1 b1 a2 c1 b2 a3 c2

A

C

B

a1 → a2

b1 → a2

b1 → b2

c1 → b2

b1 → c2

c1 → c2

a1 → a3

a2 → a3

b1 → a3

b2 → a3

c1 → a3

c1 b1 a1 c2 a2 b2 a3

constrains order

(0,0,0)

(3,2,2)

Causality preservation ⇒ total order respects the partial order defined by the precedes relation.

i.e. O1→O2 ⇒ O1 appears before O2 in every list

Even though there are 7 operations in this example, there are not 7!=5040 paths through the lattice because the great majority of them break causality. There are 12+40+40+6+12+4 = 114 paths which can be taken by a site, depending on the order in which it receives the operations. These are the permutations of the 7 operations which respect the partial order given by the precedes relation.

Note as well that there are not 27 = 128 states, there are only 24 states.

## Appendix: Mathematical background

### Partially ordered set (mathematics)

(P,≤) is a poset if binary relation ≤ on P is a partial order:

* x≤x (reflexive)
* x≤y & y≤x ⇒ x=y (antisymmetric)
* x≤y & y≤z ⇒ x≤z (transitive)

≤ is a total order if also:

* x≤y or y≤x

< is a strict partial order if

* not x<x (irreflexive)
* x<y ⇒ not y<x (asymmetric)
* x<y & y<z ⇒ x<z (transitive)

### Hasse diagram of a finite poset

The Hasse diagram of a finite poset involves taking the transitive reduction which eliminates "redundant" edges in the graph. We say y covers x if x<y and ∄z x<z<y

a

b

c

a < b

b < c

a < c

a

b

c

Hasse diagram

c covers b

b covers a

transitive reduction

transitive closure

### Infimum and supremum

The infimum and supremum of a given subset of a poset are unique if they exist. The supremum is the lowest of all the upper bounds and the infimum is the greatest of all the lower bounds.

Let (P,≤) be a poset. Let S⊆P

u is an upper bound of S if x∈S ⇒ x≤u

s is the supremum of S if

* s is upper bound of S and
* u is upper bound of S ⇒ s≤u

upper bounds

supremum

S

### Lattice (mathematics)

A lattice is a poset in which every two elements have a supremum and an infimum

s = sup {x, y} means:

* x≤s & y≤s; and (s is an upper bound)
* ∃u x≤u & y≤u ⇒ s≤u (s is the least upper bound)

A bounded lattice has least and greatest elements