Vector Time

David Barrett-Lennard
26 Feb 2008

Site identifiers
It is assumed that each site is uniquely identified with a site identifier. Let S be the set of site identifiers.

Vector times
Let N be the set of natural numbers (ie non-negative integers). A *vector time* v is a map

$$v : S \to N$$

Definition: The vector time $v_\varnothing$ satisfies $\forall\ s \in S,\ v_\varnothing(s) = 0$.

Subset, intersection and union on vector times
Definition: For vector times v_1, v_2, $v = v_1 \uparrow v_2$ denotes the vector time satisfying

$$\forall\ s \in S,\ v(s) = \max(\ v_1(s), v_2(s)\ )$$

Definition: For vector times v_1, v_2, $v = v_1 \downarrow v_2$ denotes the vector time satisfying

$$\forall\ s \in S,\ v(s) = \min(\ v_1(s), v_2(s)\ )$$

Definition: For vector times v_1, v_2, we write $v_1 \le v_2$ if $\forall\ s \in S,\ v_1(s) \le v_2(s)$

Note

- $\forall v,\ v \downarrow v = v$
- $\forall v,\ v \uparrow v = v$
- $\forall v_1, v_2,\ v_1 \downarrow v_2 = v_2 \downarrow v_1$
- $\forall v_1, v_2,\ v_1 \uparrow v_2 = v_2 \uparrow v_1$
- $\forall v_1, v_2, v_3,\ (v_1 \downarrow v_2) \downarrow v_3 = v_1 \downarrow (v_2 \downarrow v_3)$
- $\forall v_1, v_2, v_3,\ (v_1 \uparrow v_2) \uparrow v_3 = v_1 \uparrow (v_2 \uparrow v_3)$
- $\forall v_1, v_2, v_3,\ (v_1 \downarrow v_2) \uparrow v_3 = (v_1 \uparrow v_3) \downarrow (v_2 \uparrow v_3)$
- $\forall v_1, v_2, v_3,\ (v_1 \uparrow v_2) \downarrow v_3 = (v_1 \downarrow v_3) \uparrow (v_2 \downarrow v_3)$
- $\forall v,\ v_\varnothing \le v$
- $\forall v,\ (v_\varnothing \uparrow v) = v$
- $\forall v,\ (v_\varnothing \downarrow v) = v_\varnothing$
- $\forall v_1, v_2,\ (v_1 \downarrow v_2) \le v_1$
- $\forall v_1, v_2,\ v_1 \le (v_1 \uparrow v_2)$
- $\forall v,\ v \le v$
- $v_1 \le v_2\ \&\ v_2 \le v_1\ \Rightarrow\ v_1 = v_2$
- $v_1 \le v_2\ \&\ v_2 \le v_3\ \Rightarrow\ v_1 \le v_3$
- $v_1 \le v\ \&\ v_2 \le v\ \Rightarrow\ (v_1 \uparrow v_2) \le v$

Identification of operations
Each operation is assumed to be generated at exactly one site. Each site independently assigns a zero based sequence number t to each operation generated at that site.

Definition: Let op(s,t) denote the operation generated at site s with sequence number t.

Extent of a vector time

Definition: The *extent* of vector time v is the set of operations denoted by $\chi(v)$ satisfying

$$\chi(v) = \{ op(s,t) \mid t < v(s) \}$$

This definition relates to the whole purpose for vector times - for a given operation O, we describe its execution context using the vector time v satisfying $ec(O) = \chi(v)$. This is a summary of the complete set of operations that have been executed prior to O.

Note the following

- $\chi(v_2) \setminus \chi(v_1) = \{ op(s,t) \mid v_1(s) \leq t < v_2(s) \}$
- $\chi(v_\varnothing) = \{\}$
- $\chi(v_1 \uparrow v_2) = \chi(v_1) \cup \chi(v_2)$
- $\chi(v_1 \downarrow v_2) = \chi(v_1) \cap \chi(v_2)$
- $v_1 \leq v_2 \Leftrightarrow \chi(v_1) \subseteq \chi(v_2)$

Precedes relation on operations

Definition: Let O_b be an operation originally generated on site B. We write $O_a \rightarrow O_b$ if O_a was executed on B before O_b was generated on B.

Definition: O_a and O_b are *concurrent* (written $O_a \parallel O_b$) if $O_a \neq O_b$ and neither $O_a \rightarrow O_b$ nor $O_b \rightarrow O_a$

Definition: The system enforces *Causality Preservation* if

$$O_a \rightarrow O_b \Rightarrow O_a \text{ is executed before } O_b \text{ on all sites}$$

Claim: Causality preservation $\Rightarrow$ $\rightarrow$ is a transitive relation

Proof: Suppose $O_a \rightarrow O_b$ and $O_b \rightarrow O_c$. Let O_c be generated on site C. $O_b \rightarrow O_c$ so O_b was executed on site C before O_c was generated. By causality preservation and $O_a \rightarrow O_b$, we know that O_a is executed before O_b at all sites, and in particular at site C. Therefore O_a was executed at site C before O_c was generated at site C, so we deduce $O_a \rightarrow O_c$.

Execution context of an operation

Definition: Vector time gc(O) denotes the *generation context* of operation O such that $\chi(gc(O))$ is the set of operations that had been executed prior to the original generation of O.

It follows from the definitions that

$$O_a \rightarrow O_b \Leftrightarrow O_a \in \chi(gc(O_b))$$

Definition: Vector time ec(O) denotes the *execution context* of operation O such that $\chi(ec(O))$ is the set of operations that have been performed prior to execution of O.

Claim: Causality preservation $\Rightarrow$ $gc(O) \leq ec(O)$

Claim: $O_a \rightarrow O_b \Rightarrow O_a \in \chi(ec(O_b))$

Proof:

$$O_a \rightarrow O_b \Leftrightarrow O_a \in \chi(gc(O_b))$$
$$\Rightarrow O_a \in \chi(ec(O_b))$$

It is assumed that a site generates operations in the context of previously generated operations at that site. Formally this means

$$t_1 < t_2 \;\Leftrightarrow\; op(s,t_1) \rightarrow op(s,t_2)$$

Claim: $O_a \parallel O_b \;\Rightarrow\; O_a.s \neq O_b.s$
Proof:

Suppose $O_a \parallel O_b$ and $O_a.s = O_b.s$
if $O_a.t < O_b.t$
$O_a \rightarrow O_b$ (because $t_1 < t_2 \;\Leftrightarrow\; op(s,t_1) \rightarrow op(s,t_2)$)
$\Rightarrow$ contradiction to $O_a \parallel O_b$
else if $O_b.t < O_a.t$
$O_b \rightarrow O_a$ (because $t_1 < t_2 \;\Leftrightarrow\; op(s,t_1) \rightarrow op(s,t_2)$)
$\Rightarrow$ contradiction to $O_a \parallel O_b$
else
$O_a.t = O_b.t$
$O_a = O_b$ (because $O_a.s = O_b.s$ and $O_a.t = O_b.t$)
$\Rightarrow$ contradiction (because $O_a \parallel O_b \;\Rightarrow\; O_a \neq O_b$)

Aggregate intersection and union on sets of vector times
Definition: For set of vector times V, $v = \downarrow(V)$ denotes the vector time satisfying

$$\forall\, s \in S,\; v(s) = \min \{\, v'(s) \mid v' \in V \,\}$$

Definition: For set of vector times V, $v = \uparrow (V)$ denotes the vector time satisfying

$$\forall\, s \in S,\; v(s) = \max \{\, v'(s) \mid v' \in V \,\}$$

Causally valid vector times
Definition. Vector time v is *causally valid* if

$$O_1 \in \chi\,(v),\; O_2 \notin \chi\,(v) \;\Rightarrow\; \neg\,(O_2 \rightarrow O_1)$$

or equivalently

$$O_2 \in \chi\,(v) \text{ and } O_1 \rightarrow O_2 \;\Rightarrow\; O_1 \in \chi\,(v)$$

Taking the negation, vector time v breaks causality if

$$\exists \text{ operations } O_1, O_2 \text{ st } O_1 \notin \chi\,(v) \text{ and } O_2 \in \chi\,(v) \text{ and } O_1 \rightarrow O_2$$

Claim: Causality preservation $\Rightarrow$ for every operation, ec(O) is causally valid

Each site is assumed to have a linear sequence of operations that have been executed at that site called a *history buffer*. For a given site, let v_h denote the vector time whose extent describes the current contents of the history buffer.

Let v be a causally valid vector time satisfying $v \leq v_h$. It can be proven that it is possible to transpose adjacent, concurrent operations within the history buffer to separate it into a prefix and suffix such that

the set of operations in the prefix equals $\chi(v)$ and the suffix corresponds to $\chi(v_h) \setminus \chi(v)$. Furthermore no operation in the suffix causally precedes an operation in the prefix.

More specifically it won't be necessary to ever transpose a pair of contextually serialised operations $[O_1, O_2]$ where $O_1 \rightarrow O_2$.

Claim: v_1, v_2 are causally valid $\Rightarrow$ $v_1 \downarrow v_2$ is causally valid
Proof:

Let $v = v_1 \downarrow v_2$
Let O_1, O_2 be operations with $O_2 \in \chi(v)$ and $O_1 \rightarrow O_2$
Need to show $O_1 \in \chi(v)$

$\chi(v) = \chi(v_1 \downarrow v_2) = \chi(v_1) \cap \chi(v_2)$
$O_2 \in \chi(v_1)$ (because $O_2 \in \chi(v) = \chi(v_1) \cap \chi(v_2)$)
$O_2 \in \chi(v_2)$ (because $O_2 \in \chi(v) = \chi(v_1) \cap \chi(v_2)$)
$O_1 \in \chi(v_1)$ (because $O_2 \in \chi(v_1)$ and v_1 is causally valid)
$O_1 \in \chi(v_2)$ (because $O_2 \in \chi(v_2)$ and v_2 is causally valid)
$O_1 \in \chi(v_1) \cap \chi(v_2)$ (because $O_1 \in \chi(v_1)$ and $O_1 \in \chi(v_2)$)
$O_1 \in \chi(v)$ (because $\chi(v) = \chi(v_1) \cap \chi(v_2)$)

Claim: v_1, v_2 are causally valid $\Rightarrow$ $v_1 \uparrow v_2$ is causally valid
Proof: Let $v = v_1 \uparrow v_2$

Let O_1, O_2 be operations with $O_2 \in \chi(v)$ and $O_1 \rightarrow O_2$
Need to show $O_1 \in \chi(v)$

$\chi(v) = \chi(v_1 \uparrow v_2) = \chi(v_1) \cup \chi(v_2)$
if $O_2 \in \chi(v_1)$

$O_1 \in \chi(v_1)$ (because $O_2 \in \chi(v_1)$ and v_1 is causally valid)
$O_1 \in \chi(v)$ (because $\chi(v_1) \subseteq \chi(v_1) \cup \chi(v_2) = \chi(v)$)

else

$O_2 \in \chi(v_2)$ (because $O_2 \in \chi(v_1) \cup \chi(v_2)$ and $O_2 \notin \chi(v_1)$)
$O_1 \in \chi(v_2)$ (because $O_2 \in \chi(v_2)$ and v_2 is causally valid)
$O_1 \in \chi(v)$ (because $\chi(v_2) \subseteq \chi(v_1) \cup \chi(v_2) = \chi(v)$)

It follows therefore that

$\forall v \in V$, v is causally valid $\Rightarrow$ $\downarrow(V)$ and $\uparrow(V)$ are causally valid

Formulation that allows more sites over time

Usually a vector time is defined with respect to a fixed number of sites in the system. In practice we need a definition that naturally allows for new sites to be added over time.

Definition: A *vector time* v is a set of (s,t) values where s is a site identifier and t is a positive integer, and the s values are never repeated. We write sites(v) for the set of s values in v. We write v(s) to retrieve t for given s. v(s) is defined to be zero for $s \notin$ sites(v)

It is straightforward to adjust the previous definitions to avoid reference to some fixed set of states S.

$v_1 \leq v_2 \Leftrightarrow \forall s \in \text{sites}(v_1),\ v_1(s) \leq v_2(s)$

For vector times v_1, v_2, $v = v_1 \uparrow v_2$ denotes the vector time satisfying

$sites(v) = sites(v_1) \cup sites(v_2)$
$\forall\ s \in sites(v),\ v(s) = \max(\ v_1(s), v_2(s)\ )$

For vector times v_1, v_2, $v = v_1 \downarrow v_2$ denotes the vector time satisfying

$sites(v) = sites(v_1) \cap sites(v_2)$
$\forall\ s \in sites(v),\ v(s) = \min(\ v_1(s), v_2(s)\ )$

Delta vector times
A delta vector time represents a change to a vector time which can be useful to reduce network bandwidth.

Definition: A *delta vector time* $\Delta v = v_2 - v_1$ where v_1 and v_2 are vector times is defined as follows

$$\Delta v = \{ (s,t) \in v_2 \mid t \neq v_1(s) \} \cup \{ (s,0) \mid s \in sites(v_1) \setminus sites(v_2) \}$$

and $sites(\Delta v) = \{ s \mid (s,t) \in \Delta v \}$

We define $v_1 + \Delta v$ as follows.

$$v_1 + \Delta v = \{ (s,t) \in v_1 \mid s \notin sites(\Delta v) \} \cup \{ (s,t) \in \Delta v \mid t > 0 \}$$

Claim: $v_1 + (v_2 - v_1) = v_2$

Proof:
$sites(v_2 - v_1) = \{ s \mid (s,t) \in v_2 \text{ and } t \neq v_1(s) \} \cup sites(v_1) \setminus sites(v_2)$

($\subseteq$) Suppose $(s,t) \in v_1 + (v_2 - v_1)$
so $(s,t) \in (\{ (s,t) \in v_1 \mid s \notin sites(v_2 - v_1) \} \cup \{ (s,t) \in (v_2 - v_1) \mid t > 0 \})$

if $(s,t) \in (v_2 - v_1)$ and $t > 0$
$(s,t) \in v_2$ and $t \neq v_1(s)$
else
$(s,t) \in v_1$ and $s \notin sites(v_2 - v_1)$
$s \in sites(v_1)$ (because $(s,t) \in v_1$)
$t > 0$ (because $(s,t) \in v_1$)
$(s,t) \notin (v_2 - v_1)$
if $s \notin sites(v_2)$
$s \in sites(v_1) \setminus sites(v_2)$
$s \in sites(v_2 - v_1) \Rightarrow$ contradiction
so $s \in sites(v_2)$
Suppose $v_1(s) = t \neq v_2(s)$
$(s, v_2(s)) \in (v_2 - v_1)$
so $s \in sites(v_2 - v_1) \Rightarrow$ contradiction
so $v_2(s) = t$
so $(s,t) \in v_2$

($\supseteq$) Suppose $(s,t) \in v_2$
if $t \neq v_1(s)$

$(s,t) \in v_2\text{-}v_1$
$t > 0$ (because $(s,t) \in v_2$)
$(s,t) \in v_1 + (v_2\text{-}v_1)$

else

$t = v_1(s)$
so $(s,t) \in v_1$
Suppose $s \in sites(v_2\text{-}v_1)$
$((s,t) \in v_2$ and $t \neq v_1(s))$ or $(s \in sites(v_1)$ and $s \notin sites(v_2))$
$\Rightarrow$ contradiction
So $s \notin sites(v_2\text{-}v_1)$
So $(s,t) \in \{ (s,t) \in v_1 \mid s \notin sites(v_2\text{-}v_1) \}$
So $(s,t) \in v_1+(v_2\text{-}v_1)$

We can define addition of delta vector times $\Delta v = \Delta v_1 + \Delta v_2$, satisfying (for any given vector time v)

$$(v + \Delta v_1) + \Delta v_2 = v + (\Delta v_1 + \Delta v_2)$$

Note that addition of delta vector times doesn't commute.

It can be show that

$$\Delta v_1 + \Delta v_2 = \Delta v_2 \cup \{ (s,t) \in \Delta v_1 \mid s \notin sites(\Delta v_2) \}$$

Vector time implementation
To facilitate fast look up of a vector time, a suitable implementation is a red-black tree. However a delta vector time has no need for fast look up, and therefore may be more efficiently stored using a variable size array of (s,t) pairs.

We can define an add method on a vector time in pseudo code as follows

```
v.add(s,t)
{
    if (v.hasentry(s))
    {
        v.remove(s);
    }
    if (t > 0)
    {
        v.insert(s,t);
    }
}
```

Applying a delta to a vector time simply involves calling add(s,t) for each (s,t) in the delta.

We can define an add method on a delta vector time in pseudo code as follows

```
Δv.add(s,t)
{
    if (Δv.hasentry(s))
    {
        Δv.remove(s);
    }
    Δv.insert(s,t);
}
```

This may lead to a delta that has more (s,t) entries that it needs - because it doesn't check for redundancy w.r.t the original vector time to which the delta applies.

References

[1]	*Time, clocks, and the ordering of events in a distributed system* Communications of the ACM 21 (7): 558-565. Leslie Lamport (1978).