Repository - Single Char Insertion Ops
[Bogus approach]

Feb 2008
David Barrett-Lennard

Introduction
This is some analysis prior to the idea to use the Check-in Sequence Number (CISN) in order to define an ordering on the insertions in the repository that is compatible with the precedes relation.

Check-In
Consider the check-in of operation O, with execution context given by vector time v = ec(O). ie O is assumed to be executed in the context of executing all operations in $\chi(v)$. Let's assume that we only need to check in a single interval of O with given s,t,value,q.

For the given vector time v we can distinguish between assignments in $\chi(v)$ versus assignments that are not in $\chi(v)$. As an example, suppose we want to checkin an assignment with q = 1, and the assignments in the respository can be depicted as follows

```
[  x0  x1  q0   x2   x3   x4   q1   q2   q3   x5   x6   q4  ]
```

where the q_i depict assignments in $\chi(v)$ and the x_i depict assignments not in $\chi(v)$.

We interpret the insertion position q = 1 with respect to the index positions of the assignments with all the xi removed. ie

```
[  q0   q1   q2   q3   q4  ]
      |
    insertion with q = 1 must be in here
```

This implies that the new assignment must be inserted after q_0 but before q_1.

In the original full list of assignments, we actually have x_2,x_3,x_4 between q_0,q_1. Therefore there are four possible insertion positions which we have labeled p_0,p_1,p_2,p_3 in the following:

```
[    ...  q0   x2    x3    x4    q1   ...   ]
            |    |     |     |
           p0    p1    p2    p3
```

In turns out there is a unique position where the new assignment must be inserted.

From [1], $O_a \rightarrow O_b \;\Rightarrow\; O_a \in \chi(gc(O_b)) \;\Rightarrow\; O_a \in \chi(ec(O_b))$

Claim: for each i, $O \parallel x_i$.
Proof:

$x_i \notin \chi(v)$	(distinction between x_i and q_i)
$x_i \notin \chi(ec(O))$	(because v = ec(O))

$\neg(x_i \rightarrow O)$ (contrapositive of $x_i \rightarrow O \Rightarrow x_i \in \chi(ec(O))$)
Also $\neg(O \rightarrow x_i)$ (x_i checked in before O, so $O \notin \chi(gc(x_i))$)
So $O \parallel x_i$

It follows therefore that for each i, $O.s \neq x_i.s$.

Definition: We write $x_i \ll x_j$ if x_i dominates x_j. This is a total ordering on all assignments. The assignments recorded in this repository are uniquely ordered according to this relation.

ie $x_i \ll x_j \Leftrightarrow$ x_i appears on left of x_j in the repository.

Consider firstly that x_2,x_3,x_4 are mutually concurrent assignments. ie for each i,j $x_i \parallel x_j$. This implies they were generated on different sites. Furthermore the x_i will be ordered according to siteid. ie $x_i \ll x_j \Leftrightarrow x_i.s < x_j.s$. Clearly O must be inserted in order to maintain the ordering by site identifier. This uniquely defines the insertion position amongst the x_i.

Claim: $x_j \rightarrow x_i \Rightarrow x_i \ll x_j$
Proof: $x_j \rightarrow x_i$ $\Rightarrow$ x_i was executed in the context of x_j
$\Rightarrow$ x_i dominates x_j
$\Leftrightarrow$ $x_i \ll x_j$

[Note on an approach that won't work]

When we look at all the xi, we could imagine that are partitioned into groups according to the precedes relation. eg

$(x_1\ x_2\ x_3)\ (x_4)\ (x_5\ x_6\ x_7)\ (x_8\ x_9)$

where $x_3 \rightarrow x_2 \rightarrow x_1$, $x_7 \rightarrow x_6 \rightarrow x_5$, $x_9 \rightarrow x_8$

However, this is not actually the case. For example the following is possible

$x_1 \ll x_2 \ll x_3$ with $x_3 \rightarrow x_1$ & $x_1 \parallel x_2$

Proposal: Restructure this explanation as follows

1. The case of an insertion of an interval where all intervals in the repository are in X(v) --> left to right scan
2. The case of an insertion of an interval where all intervals in the repository are not in X(v) --> right to left to scan
3. Explain how to use above two solutions to solve the general case.

Note that the left to right scan is consistent with Mergeii() defined in MultiCharCDMOp.cpp which also performs a left to right scan and assumes i1 >> i2

TODO: We have created a significant source of confusion because << is used in the literature to mean contextually serialised, and we have used it here to denote the effects relation, which is completely different.

Case where all intervals in the repository are not in $\chi(v)$
Let the repository consist of intervals $x_1,x_2,...x_n$ where each $x_i \notin \chi(v)$

Claim: The insertion position of a given can be achieved with a right to left scan of the xi.

Proof: The repository $[x_1,...,x_n]$ can be considered a sequence of contextually serialised sequence of operations in reverse order $L = x_n,...,x_1$ that all insert at $q = 0$. This totally ordered sequence is compatible with partial ordering defined by the precedes relation because $x_j \rightarrow x_i \Rightarrow x_i \ll x_j$.

We can assume O also inserts a character at $q = 0$. Also $O \parallel L$ so it makes sense to consider IT(O,L) and IT(L,O). We know we simply IT O against xn then xn-1 and so on. Furthermore all the insertions are at the same q-position so therefore we compare site ids and only increment the q-position of the loser.

As a result this shows that the correct insertion position of O is determined by a right to left scan and comparison of site identifiers.

TODO: Turn this into proper mathematics!

Let L be split into prefix L_1 and suffix L_2. ie $L = L_1 + L_2$, where $L_1 >> L_2$ where >> denotes contextually serialised operations. We want to uniquely specify L_1,L_2 such that $[L_1 \ O \ L_2]$ is compatible with the effects relation. In fact

1. $\forall x_i \in L_1, \ x_i.s < O.s$
2. $L_2 \neq \{\} \Rightarrow L_2[0].s > O.s$

TODO: We need to formalise the definition of the effects relation, preferably at the level of insertions of single characters at any position (not just q = 0, as for assignments), then show how the repository is able to store all positions in according to the effects relation. The concept of q-position can be formalised as a coordinate that is derived from the effects relation for a particular consistent cut defined by a given vector time.

Maths background
In mathematics, the *transitive closure* of a binary relation R on a set X is the smallest transitive relation on X that contains R. Similarly we can define *reflexive closure*.

Definition: Set X is (non-strict) totally ordered under $\leq$ if the following hold for all a,b and c in X:

1. $a \leq b \ \& \ b \leq a \Rightarrow a = b$ (antisymmetry);

2. $a \leq b \ \& \ b \leq c \ \Rightarrow \ a \leq c$ (transitivity);
3. $a \leq b$ or $b \leq a$ (totality or completeness).

Note that (3) implies that $\leq$ is reflexive. ie for all a, $a \leq a$.

Definition: Set X is strict totally ordered under < if the following hold for all a,b and c in X:

1. $\neg(a < a)$ (irreflexive)
2. $a < b \ \Rightarrow \ \neg(b < a)$ (asymmetric)
3. $a < b \ \& \ b < c \ \Rightarrow \ a < c$ (transitivity)
4. exactly one of $a < b$ or $b < a$ or $a = b$ (trichotomous).

Note asymmetric implies irreflexive. trichotomous implies asymmetric.

Definition of the effects relation on operations that insert single characters

Let operations insert single characters. There are no deletions. We want to define a total ordering on the characters. Note that for each character there is an operation that inserted it and vice versa. Therefore an effects relation on characters is also an effects relation on operations.

When an operation O_2 is generated, we have a generation context $gc(O_2)$ and for each operation O_1 in $X(gc(O_2))$ we say $O_1 \rightarrow O_2$. For these characters we know exactly the characters to the left of O_2, and to the right of O_2.

For any pair of operations O_1,O_2,

1. if $O_1 \rightarrow O_2$ or $O_2 \rightarrow O_1$ then one of the operations was generated in the context of the other. Therefore the ordering is well defined.
2. Otherwise $O_1 \parallel O_2$. So how do we define the effects relation in that case? See below...

Definition: $O_1 \rightarrow_r O_2$ if $O_1 = O_2$ or $O_1 \rightarrow O_2$. ie this is the *reflexive closure* of the precedes relation.

Definition: $O_1 \blacktriangleright' O_2$ if

1. $O_1 \rightarrow O_2$ and O_1 was to the left of the insertion position of O_2 when O_2 was generated; or
2. $O_2 \rightarrow O_1$ and O_2 was to the right of the insertion position of O_1 when O_1 was generated; or
3. $O_1 \parallel O_2 \ \& \ gc(O_1) = gc(O_2) \ \& \ O_1.s < O_2.s$

Definition: $\blacktriangleright$ is defined as the transitive closure of $\blacktriangleright'$.

Claim: $\blacktriangleright$ is a total ordering on all the characters inserted into a given text document

Proof:

$(a < b \ \Rightarrow \ \neg(b < a))$

Suppose $O_1 \blacktriangleright O_2$. We show $\neg(O_2 \blacktriangleright O_1)$

...

$\neg(a < b)$ and $\neg(b < a) \ \Rightarrow \ a = b$

Suppose $\neg(O_1 \blacktriangleright O_2)$ and $\neg(O_2 \blacktriangleright O_1)$ and $O_1 \neq O_2$. We want a contradiction
If $O_1 \rightarrow O_2$ then
    if O_1 on left of O_2 then $O_1 \blacktriangleright' O_2 \Rightarrow O_1 \blacktriangleright O_2 \Rightarrow$ contradiction
    else if O_1 on right of O_2 then $O_2 \blacktriangleright' O_1 \Rightarrow O_2 \blacktriangleright O_1 \Rightarrow$ contradiction
Similarly if $O_2 \rightarrow O_1$ leads to contradictions
otherwise $O_1 \parallel O_2$
    $O_1.s \neq O_2.s$
    if $gc(O_1) = gc(O_2)$ then
        if $O_1.s < O_2.s$
            $O_1 \blacktriangleright' O_2 \Rightarrow O_1 \blacktriangleright O_2 \Rightarrow$ contradiction
        else
            $O_2.s < O_1.s$
            $O_2 \blacktriangleright' O_1 \Rightarrow O_2 \blacktriangleright O_1 \Rightarrow$ contradiction
    else
        < todo >

No, actually the claim is not true! Eg consider O_1,O_2,O_3 where $O_1 \rightarrow O_2$, $O_3 \parallel O_1$ and $O_3 \parallel O_2$.

O_1 = insert 'x' at 0
O_2 = insert 'y' at 1 (to be executed after O_1)
O_3 = insert 'z' at 0

O_3' = IT(O_3,O_1) = insert 'z' at 1
O_3'' = IT(O_3',O_2) = insert 'z' at 2

Note that we compared site identifiers for O_3 against both O_1 and O_2. Therefore it is not sufficient to only compare site identifiers when $gc(O_1) = gc(O_2)$.

It seems difficult to see how we can easily define the effects relation, except in terms of IT, and use TP1, TP2 to prove convergence at all sites, and hence a proof that the effects relation is well defined.

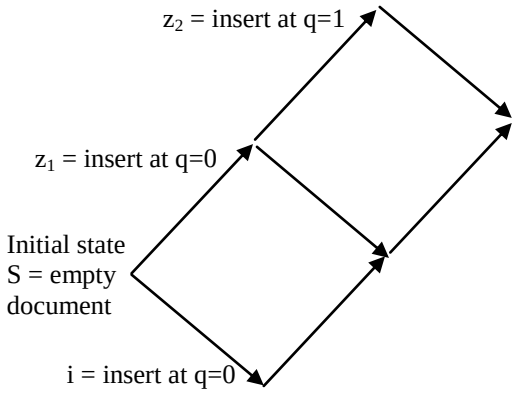
How to do repository on insertion ops
For now assume we only have single character insertion operations in the repository.

We want to insert operation i correctly in the effects document associated with operations z_1,z_2 that are each concurrent with i, where the character inserted by z_1 is always to the left of the character inserted by z_2. So we know the following

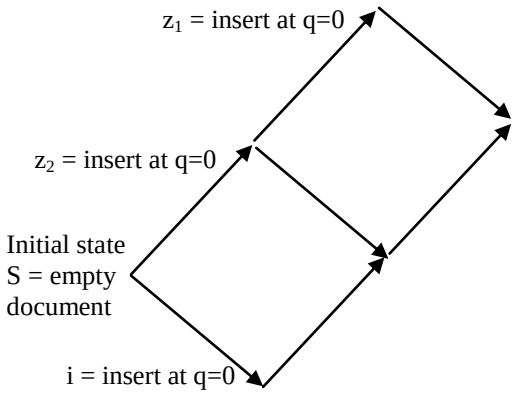
$i \parallel z_1, \quad i \parallel z_2, \quad z_1 <_r z_2$

where $<_r$ denotes the effects relation between insertion operations. Note well that we do not know whether $z_1 \rightarrow z_2$, $z_2 \rightarrow z_1$ or $z_1 \parallel z_2$.

If $z_1 \rightarrow z_2$ then we can consider execution contexts for z_1,z_2 such that they are contextually serialised as per the figure on the left, whereas when $z_2 \rightarrow z_1$ we can consider execution contexts for z_1,z_2 such that they are contextually serialised as per the figure on the right. This allows us to consider the dual IT of the contextually serialised operations with concurrent operation i.



$z_1 \rightarrow z_2$



$z_2 \rightarrow z_1$

Note in both figures we have satisfied the requirement that $z_1 <_r z_2$ (ie that the character inserted by z_1 is to the left of the character inserted by z_2).

If $z_1 \parallel z_2$ then both figures must be applicable! ie we will find that in either case we come up with the same objective criterion for where to insert i relative to z_1,z_2.

The following table summarises all the possible cases.

$z_2.s < z_1.s$	$z_1 \parallel z_2$	impossible
	$z_1 \rightarrow z_2$	if i.s < z_1.s then insert i before z_1 else insert i after z_2
	$z_2 \rightarrow z_1$	if i.s > z_2.s then insert i after z_2 else insert i before z_1
$z_1.s < z_2.s$	$z_1 \parallel z_2$	if i.s < z_1.s then insert i before z_1
	$z_1 \rightarrow z_2$	if z_1.s < i.s < z_2.s then insert i between z_1,z_2
	$z_2 \rightarrow z_1$	if z_2.s < i.s then insert i after z_2

Note that if $z_1 \parallel z_2$ then it must be the case that $z_1.s < z_2.s$.

We see that when $z_1.s < z_2.s$ the insertion position depends on maintaining the ordering on site identifiers.

However when $z_2.s < z_1.s$, it is useful to think of one of the operations as inheriting the site identifier from the other. ie if $z_1 \rightarrow z_2$ then we imagine that z_2 inherits the site identifier from z_1 (ie both z_1,z_2 have the same site identifier $z_1.s$ as far as determining the

correct insertion position for i) whereas if $z_2 \rightarrow z_1$ then we imagine that z_1 inherits the site identifier from z_2.

This leads to a site identifier propagation algorithm for the more general case of working out where to insert operation i within $[z_1,...,z_n]$. An example will illustrate the idea

	z_1	z_2	z_3	z_4	z_5	z_6
siteid	2	4	7	8	4	9

Consider that we pick any pair of adjacent operations $[z_i, z_{i+1}]$ where $z_i.s > z_{i+1}.s$. Then we propagate siteid to the left or the right depending on whether $z_i \rightarrow z_{i+1}$ or $z_{i+1} \rightarrow z_i$ (noting that $z_i \parallel z_{i+1}$ is impossible). For example $[z_4,z_5]$ meets this condition.

Suppose $z_4 \rightarrow z_5$, then we get

	z_1	z_2	z_3	z_4	z_5	z_6
siteid	2	4	7	8	8	9

Now we no longer have any adjacent operations where the siteid decreases from left to right.

If instead it had been the case that $z_5 \rightarrow z_4$, then we get

	z_1	z_2	z_3	z_4	z_5	z_6
siteid	2	4	7	4	4	9

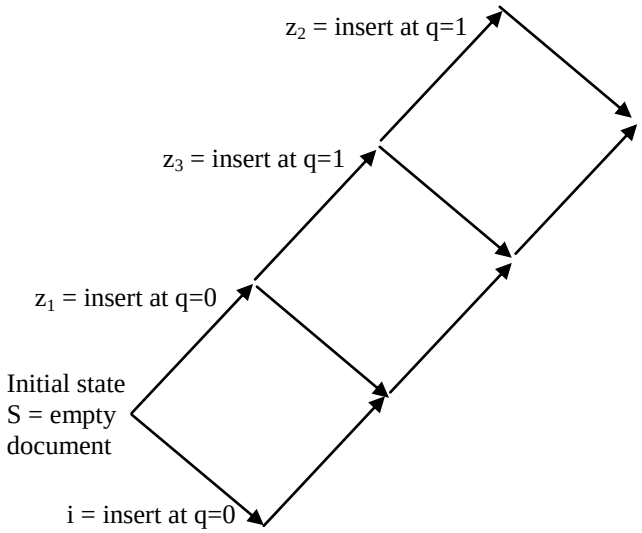
We now regard z_4,z_5 as joined at the hip as far as further id propagation goes. ie any changes to one affects the other!

Now the pair $[z_3, (z_4+z_5)]$ requires id propagation. Suppose $z_3 \rightarrow (z_4,z_5)$ then we get

	z_1	z_2	z_3	z_4	z_5	z_6
siteid	2	4	7	7	7	9

Hypothesis
During id propagation we are forming larger and larger equivalence classes. When id's are propagated, it is always to the entire equivalence class.

Example with three operations



$z_1 \rightarrow z_3$, $z_3 \rightarrow z_2$, $z_1 \rightarrow z_2$

$z_1 <_r z_2 <_r z_3$

Let the siteids be as follows

Operation	z_1	z_2	z_3
SiteId	4	6	2

Then the insertion position for operation i will be as follows

if (i.s < z_1.s) then	insert i before z_1
else	insert i after z_3

So using our concept of site id propagation, for some reason z_1 dominates both z_2 and z_3.