Operational transform
Map operations

31 Aug 2005
David Barrett-Lennard

Abstract
This paper provides a solution to the operational transform of map operations.

Introduction
The document state is assumed to be a map parameterised over some key element type K and Value type V. For example, in C++ std::map<K,V> is an implementation of such a map using a red black tree.

Each key is mapped to a particular value. Keys in the map must be unique.

K,V are value types, and are assumed to support equality comparison operators. In this paper we study the operational transform of the following operations

- ins(k,v) - insert key k with value v into the map. This operation can always be generated, irrespective of whether an entry with the given key is already present in the map. In that case the new entry replaces the existing entry.
- del(k) - delete the entry with given key from the map (if present). It is allowable for no entry with the given key to be present when this operation is executed.

We need convergence at quiescence even though operations are executed in different orders at different sites. Specifically we require properties TP1 and TP2 [2].

Properties required to achieve convergence

TP1
$\forall\ O_1, O_2 \quad S + [O_1\ IT(O_2, O_1)] = S + [\ O_2\ IT(O_1, O_2)\ ]$

TP2
$\forall\ O_1, O_2, O_3, \quad IT(\ IT(\ O_3, O_1),\ IT(O_2, O_1)) = IT(\ IT(\ O_3, O_2),\ IT(O_1, O_2))$

Discussion
ins(k,v) works like an assignment operation. It overwrites any existing value identified by the given key with a new value.

Consider that we think of the value type V as including a special value φ that means there is no entry in the map. Then del(k) is equivalent to ins(k,φ). This allows us to think in terms of a single operation that has assignment semantics on the value for a given key.

[1] provides an efficient solution to the IT/ET of assignment operations.

Solution
Let an operation contain the following fields

Field	Description
ins	Boolean flag to indicate whether this is an insert or delete operation
q	q-position used to select a unique winner amongst competing operations for

	setting the value for a given key. Initialised to zero when the operation is first generated. Zero indicates that the operation is enabled.
k	Key
v	Value (only relevant for insert operation)

The following C++ code shows the implementation of IT/ET

```
void IT(MapOp& O1, const MapOp& O2)
{
    if (O1.k == O2.k && ((O2.q < O1.q || O2.q == O1.q && O2.id < O1.id)) ++O1.q;
}

void ET(MapOp& O1, const MapOp& O2)
{
    if (O1.k == O2.k && O2.q < O1.q) --O1.q;
}
```

Future work

Undo needs to be supported. This requires storage of the previous value in the map, so it can be re-instated. Note that the solution already appears in [1]. It is obvious how it would be applied to assignment on map entries.

For performance this solution should be extended to support operations that insert or delete multiple elements at once. This also allows for operations to be closed under merging. Furthermore it will be possible to compress merged operations by eliminating elements that are inserted then deleted by the operation. Compression can also discard disabled operations.

References

[1]	David Barrett-Lennard. Operational transform - Assignment operations. Aug 2005