Operational Transformation on Text

David Barrett-Lennard

22 Oct 2016

# Intentions of operations

The *intentions* of insert and delete operations on text can be defined in terms of the relative positions of the characters. One can ask whether it is possible to preserve these intentions as concurrent operations are merged.

abc

Insert intention

abc

x

"x inserted after a, before b,c"

Delete intention

"b not present"

# A merge can always preserve these intentions

It is always possible to preserve these intentions when merging concurrent insert and delete operations. In that sense there are never conflicts when merging operations on text!

The merge result is *the union of all the insertions minus the union of all the deletions*

abcde

O2

O1

O2

O4

O3

O3

This is unambiguous except for insertions at the same position. In that case the CEDA merge algorithms determine the insertion order from the total order on the site identifiers.

There can be cases where merging of operations doesn’t preserve the intentions of the users at some “higher” semantic level. Such conflicts fall outside the scope of the CEDA implementation of OT. In general the merge of concurrent operations is *always* defined, even if there are real conflicts between users. Users are expected to edit the result of the merge (perhaps interactively) to resolve conflicts.

# Example with two sites

As a simple example consider two sites which independently insert a single character into a string. Initially the string is "b" and one site inserts an "a" before and another site inserts a "c" after. Obviously after merging these operations the result should be "abc".

b

abc

a

c

0 1 2

Initial state

Final state

(zero based index positions)

The following state transition diagram shows the four possible states of the string variable at the corners of a square. The operations represent transitions between these states. There are two paths from the initial state in the bottom left corner to the final state in the top right corner. Notice that the insertion position of "c" needs to be shifted to the right when it is applied in the context of having first inserted the "a".

bc

abc

ab

b

insert c at **1**

insert c at **2**

insert a at 0

insert a at 0

Initial state

Final state

Insertion position transformed

# Example with three sites

We now consider a third concurrent operation generated by a third site which deletes the "b" character. The final state should clearly be "ac".

b

ac

a

c

0 1

Initial state

Final state

(zero based index positions)

In the earlier example the state transition diagram (i.e. lattice) was a square. The additional concurrent operation means the lattice now has a cube structure. The previous lattice corresponds to the front face of the cube.

bc

c

ac

abc

a

ab

b

delete b at 0

delete b at 0

delete b at 1

delete b at 1

insert c at **0**

insert c at 1

insert c at 2

insert c at 1

insert a at 0

insert a at 0

insert a at **0**

insert a at 0

3 operations

23 = 8 states

3! = 6 paths

Final state

Initial state

There are 6 paths through the cube all starting in the front-left-bottom corner and ending on the back-right-top corner. The insertion/deletion positions of the operations are transformed as appropriate so that sites can apply the three operations in any order and they all end up with the same result in the replicated text document.

# TP2 puzzle

There is a troubling ambiguity in the state where the deletion of a character has meant that the two insertions positions now coincide

b

ac

a

c

Initial state

Final state

This ambiguity occurs on the back face of the cube

c

ac

a

insert c at **0**

insert c at 1

insert a at 0

insert a at **0**

This is related to the problem of meeting a mathematical condition in the literature called TP2.

Unlike a number of other implementations of OT (which can give the result ca instead of ac), the CEDA implementation of OT resolves the ambiguity correctly and satisfies TP2.