MERGE POLICIES

Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu
CONCURRENT INTERACTION

Insert same character at same position?
CONCURRENT INTERACTION

Insert same character at same position?

Duplicate character!

Work preserving

Does not meet "user intention"
CONCURRENT INTERACTION

InsertOperation TransformInsertInsert (InsertOperation Remote, InsertOperation Local) {
    Operation Remote\[T\] = Remote.clone();
    if (Remote.index == Local.index && Remote.element.equal(Local.element))
        return NullOperation
    if (Remote.index > Local.index ||
        (Remote.index == Local.index && Remote.id < Local.id))
        Remote\[T\].index = Remote.index + 1;
    return Remote\[T\];
}
GOAL OF CONSISTENCY

- At quiescence (no user interacting) all displays are the same
  - When concurrent command is executed, could ignore it and clear object
  - Meets TP1 (and TP2!)
- Meets user intention
  - How to describe what it is?
  - Even if we cannot describe it, how to describe what algorithm does
    - More than one “reasonable” merge acceptable
  - Application-specific merger
  - Application-specific merge procedure
  - Declarative scheme?
ASSUMPTIONS AND INTENTION ISSUE

One-level sequence with inserts only.

Inserts at different positions are both accepted (as defined by initial transformation functions).

If both users insert the same element at the same position, only one is executed.

What happens when both users insert different elements at the same position?
Merge Matrix For Insertable Sequences

Client operation at same index

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Insert(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert(#)</td>
<td></td>
</tr>
</tbody>
</table>

Server Operation at Same index

Default value for flexible algorithm?

Cell Choices?

- Describes merge semantics of different insert operations at same index
- Merge matrix not relevant when indices are different or insertions are the same
- A particular algorithm can give the application and/or user a subset of choices in the merge matrix
CONCURRENT OPERATION

User intention?

Conflict!
Accept Both

Accept both as they can resolve the conflict
Accept None

Accept none as there is a conflict and cannot afford a wrong merge.
ACCEPt SERVER

Accept earlier (server) so later person can see it and correct it
Accept Client

Accept later (client) as more recent information available
Describes merge semantics of different insert operations at same index

A particular algorithm can give the application and/or user a subset of choices in the merge matrix

Merge matrix not relevant when indices are different or insertions are the same
Default for Insertable Sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Insert(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert(#)</td>
<td>Both</td>
</tr>
</tbody>
</table>

Default value for flexible algorithm

Delete and Modify Operations?

Server
Client
None
Both
# Merge Matrix for Insertable Sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Insert(#)</th>
<th>Delete(#)</th>
<th>Replace(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert(#)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete(#)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replace(#)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Describes merge semantics of different sequence operations at same index

Default values for flexible algorithm?
## Defaults for General Sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Insert(#, a)</th>
<th>Delete(#)</th>
<th>Replace(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert(#, b)</td>
<td>Both</td>
<td>Both</td>
<td>Both</td>
</tr>
<tr>
<td>Delete(#)</td>
<td>Both</td>
<td>NoOp</td>
<td>Server</td>
</tr>
<tr>
<td>Replace(#)</td>
<td>Both</td>
<td>Client</td>
<td>Server</td>
</tr>
</tbody>
</table>

- **Insert(#, a)**: Both
- **Delete(#)**: Both
- **Replace(#)**: Server

### Notes
- **Insert(#, b)**: Deletes at the same index always the same operation and hence **NoOp**.
- **Replace(#)**: Replacement means it is relevant and perhaps should not be deleted.

### Tables?

- **Server**
- **Client**
- **None**
- **Both**
General Sequences and Replace/Delete

bat

Replace(1, “c’)  Delete(1, “c’)

cat  at

cat
<table>
<thead>
<tr>
<th>Table</th>
<th>Put (key)</th>
<th>Delete(key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put (key)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete(key)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assume operations at different keys are non conflicting

Describes merge semantics of different table operations at key

Default values?

Server
Client
None
Both
# Defaults for General Tables

Putting the same value at the same key is a **NoOP**

<table>
<thead>
<tr>
<th>Table</th>
<th>Put (key)</th>
<th>Delete(key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put (key)</td>
<td>Server</td>
<td>Client</td>
</tr>
<tr>
<td>Delete(key)</td>
<td>Server</td>
<td>NoOp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
</tr>
<tr>
<td>Client</td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td>Both</td>
</tr>
</tbody>
</table>
put ("bob, false")
remove ("bob")
## Record Matrix

<table>
<thead>
<tr>
<th>Record</th>
<th>Set (Property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set (Property)</td>
<td></td>
</tr>
</tbody>
</table>

Describes merge semantics of different record operations at same property

Putting the same value at the same property is a NoOP

Default value?
# Defaults for Record

<table>
<thead>
<tr>
<th>Record</th>
<th>Set (Property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set (Property)</td>
<td>Server</td>
</tr>
</tbody>
</table>

Accept earlier (server) so later person can see it and correct it.
```
set ("ticketPrice", 24.0)
set ("ticketPrice", 24.1)
```
Accept earlier (server) so later person can see it and correct it.

**Atomic Objects**
# Atomic Matrix

<table>
<thead>
<tr>
<th>Atomic</th>
<th>Set ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set ()</td>
<td>Server</td>
</tr>
</tbody>
</table>
### General Merge Matrix

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation$_1$</th>
<th>...</th>
<th>Operation$_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation$_1$</td>
<td>Default$_{11}$</td>
<td>...</td>
<td>Default$_{1N}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Operation$_N$</td>
<td>Default$_{N1}$</td>
<td>...</td>
<td>Default$_{NN}$</td>
</tr>
</tbody>
</table>

Some type-specific operand (index, key, value) whose value determines when two dissimilar operations are compared.
**Asynchronous Buffered Changes**

**Diagram:**
- **Joe UNC**
- **Joe UNC-**
- **Joe UNC-C**
- **Joe UNC-CH**
- **Joe UNCC**
- **Joe UNCC-CH**

**Text:**
- "Joe" and "UNC" are sequences in a record with name and affiliation properties.
- Can we extend model to add the option of accepting all of server or client changes to affiliation?
- Need to capture multiple levels of changes.
Hierarchical Document

Set(Affiliation)
I(4, ‘-‘) I(5, ‘C‘) I(6, ‘H‘)

Record
Name
Sequence
J o e

Affiliation
Sequence
U N C

Set(Affiliation)
I(4, ‘C‘)

Record
Name
Sequence
J o e

Affiliation
Sequence
U N C - C H

Record
Name
Sequence
J o e

Affiliation
Sequence
U N C C
## Merge Next Level Option

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation$_1$</th>
<th>...</th>
<th>Operation$_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation$_1$</td>
<td>Default$_{11}$</td>
<td>...</td>
<td>Default$_{1N}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Operation$_N$</td>
<td>Default$_{N1}$</td>
<td>...</td>
<td>Default$_{NN}$</td>
</tr>
</tbody>
</table>

Options for Merge:
- Server
- Client
- None
- Both
- Merge
**Level 1 Step**

- **Set (Affiliation)**
  - I(4, ‘-’)
  - I(5, ‘C’)
  - I(6, ‘H’)

- **Record**
  - Joe
  - UNC

- **Set (Property)**
  - Merge

- **Record**
  - Name
    - Sequence
      - Joe
  - Affiliation
    - Sequence
      - UNC - CH

- **Set (Property)**
  - UNC C C
**Level 2 Step**

Set(Affiliation)

- I(4, ‘-’)
- I(5, ‘C’)
- I(6, ‘H’)

**Record**

- Name
- Affiliation

**Sequence**

**Name**

- Joe

**Set(Affiliation)**

- I(4, ‘C’)

**Sequence**

- Joe
- UNC
- CH

**Insert(##)**

- Both

**Delete(##)**

- Both
- NoOp
- Server

**Replace(##)**

- Both
- Client
- Server

**Hot to go to next level?**

InsertOperation TransformInsertInsert (InsertOperation R, InsertOperation L) {
    Operation \( R^T = \text{deepClone}() \);
    if ((R.index > L.index) ||
        (R.index == L.index && R.isServer()))
        \( R^T\.index = R\.index + 1 \);
    \( R^T \);
}

InsertOperation TransformInsertInsert (InsertOperation R, InsertOperation L) {
    Operation \( R^T = \text{deepClone}() \);
    if ((R.index > L.index) ||
        (R.index == L.index && !R.isServer()))
        \( R^T\.index = R\.index + 1 \);
    \( R^T \);
}

Sequence | Insert(#) | Delete(#) | Replace(#) |
----------|----------|-----------|------------|
Insert(#)  |          |           |            |
Delete(#)  |          |           |            |
Replace(#) |          |           |            |

Procedural

Declarative

Declarative is higher level allows easy customization

But it is less expressive

Server
Client
None
Both
Merge Matrix

Covered all asynchronous merge policies known in 94-97

Had mechanisms to extend the default matrix
Reference for Merge Matrix

- Munson and Dewan ‘94, ’97
- Showed that all merge procedures at that time for spreadsheets, file systems, databases, .. could be supported using the merge matrix
- Merge matrix entry could itself have merge procedures in it for a specific combination of operations or level