

# A Fault-tolerant Three-way Merge for XML and HTML

Tancred Lindholm and Torsten Ruger

Helsinki Institute for Information Technology

<http://www.hiit.fi>

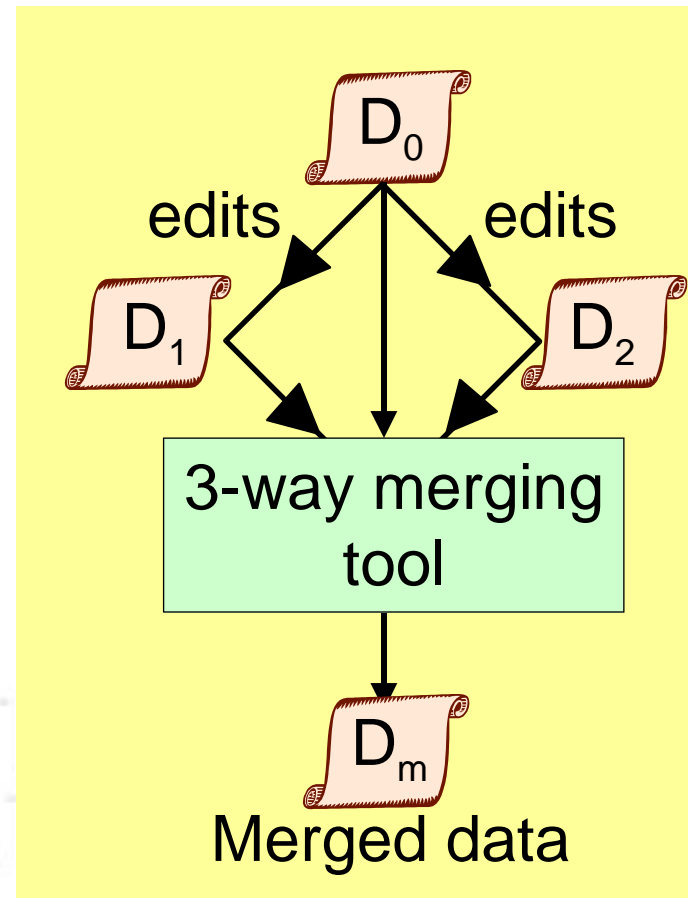
{tancred.lindholm,torsten.rueger}@hiit.fi

# Introduction

- Research problem
  - two (several) copies of an XML/HTML document are independently modified
  - we want to integrate the edits into a single copy
- We want to support **moved content**. The traditional `diff/patch` combination does not handle that well.
- We process the XML/HTML documents as **text strings**, despite their inherent tree structure.
- ➔ our tool becomes **resilient to markup syntax errors**
- ➔ we may use fast **string matching algorithms**
- We wanted to see how far this “wrong” approach (in terms of the data model) would take us

# Three-way Merging

- Assume  $D_0$  is the *base* data set. Two copies of  $D_0$  are modified, yielding  $D_1$  and  $D_2$ .
- $D_1$  and  $D_2$  can be reconciled by a three-way merge of  $D_0$ ,  $D_1$  and  $D_2$
- The **three-way merge**  $D_m$  is the result of applying the changes between  $D_0$  and  $D_1$  as well as between  $D_0$  and  $D_2$  to  $D_0$ .
- Changes are **detected** by comparing  $D_0$  and  $D_1$  as well as  $D_0$  and  $D_2$  →  
no edit history needed



# Example

- $D_0$ : `<div><p>That echoes my thoughts</p>  
<p>So nice to run into you</p></div>`
- $D_1$ : `<div><p>That echoes my thoughts</p>  
<p>So nice to run</p></div>`  
delete `<p> order`
- $D_2$ : `<div><p>So nice to run into you</p>  
<p>That echoes his  
thoughts</p></div>`
- Proposed  $D_m$ : `<div><p>So nice to run</p>  
<p>That echoes his thoughts</p></div>`

# Design principles from Use Cases

- No single “right” way to merge
- We wanted a way that is **useful in practice**
- To find merging patterns we studied hand-made use cases
  - In each case  $D_0, D_1, D_2$  and  $D_m$  were hand-crafted
  - Insert, update, delete and move of tags, attributes and text
  - 97 such cases written
  - We also used the cases for the “3dm” tool
  - Our cases more verbose, less self similar than the 3dm cases

# Merging Principles

- Useful to break up the text in  $D_1$  or  $D_2$  into regions from  $D_0$  and regions of inserted text

```
<div><p>That echoes my thoughts</p><p>So nice to run </p></div>
```

- **Inserted** regions in  $D_1$  or  $D_2$  should appear in  $D_m$
- Regions **deleted** from  $D_1$  or  $D_2$  should not appear in  $D_m$
- **Moved** (reordered with respect to  $D_0$ ) regions should appear in its changed context in  $D_m$ , where
- “*context*” is defined as the predecessor and successor of the region
  - The definition allows merging overlapping moves

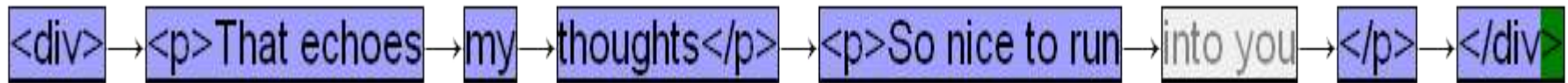
# Matching

- Use hash-based greedy matching to find largest common regions between  $D_1/D_2$  and  $D_0$ .
- Ensure mapping is one-to-one
  - no copied regions, no glued regions
- Normalize region boundaries, so that no region boundary (of one document) occurs inside a region (of the other document).
- Needed because changes happen at region boundaries
- Normalized whitespace yields better matches

`<div> <p>That echoes my thoughts</p> <p>So nice to run </p> </div>`

# Merging

- Traverse regions from  $D_1$  and  $D_2$ , always using the **changed** order (arrows on red)
- At each point, a region from  $D_1$  and  $D_2$  is merged. We always use the region that represents a change (i.e. insert or delete)



• E.g. `</div>` + `</div>` = `</div>`

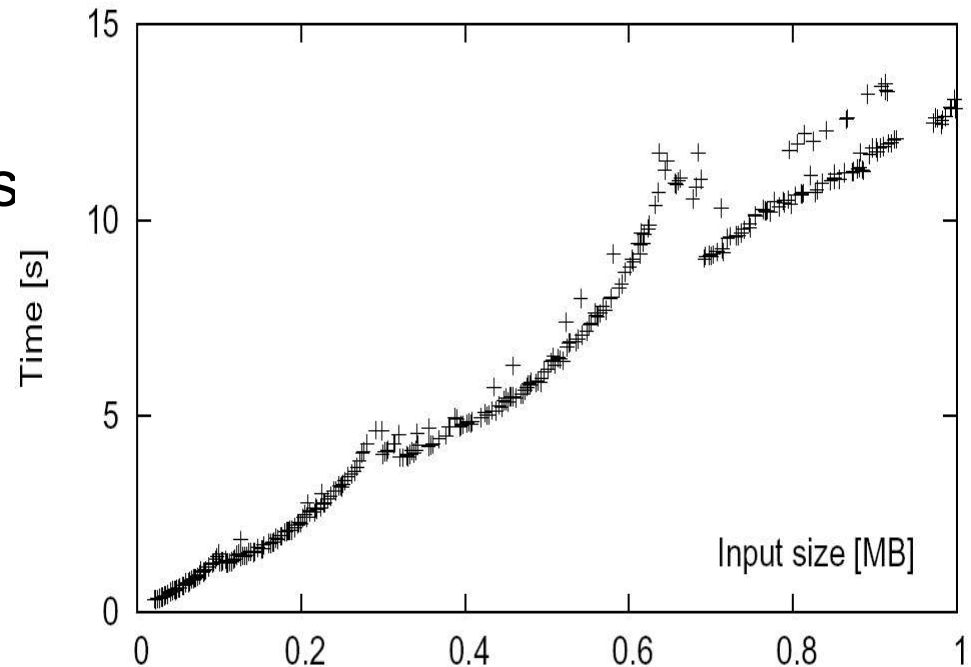
`into you` + `into you` = [] (no text)

# Conflicts

- Detected conflicts should correspond to the user's notion of a conflict
  - Don't find conflict where there are none
  - Don't sweep true conflicts under the rug!
  - I.e. if the same text is moved in both  $D_1$  and  $D_2$ , we should not guess what the user intends!
- We recognize conflicting reordering of regions
- Conflicting updates to the same text generates an “consecutive insert” warning:  
merge(“a nce day”, “a nice day”, “a splendid day”)

# Evaluation

- 97/97 study cases successful
- 18/19 3dm small cases successful
- 5/8 large 3dm cases successful
- The failed cases were due to bad matchings
- ➔ we need to improve the matching heuristics
- ➔ no problems with the string-based technique



- Satisfactory scalability

# In conclusion...

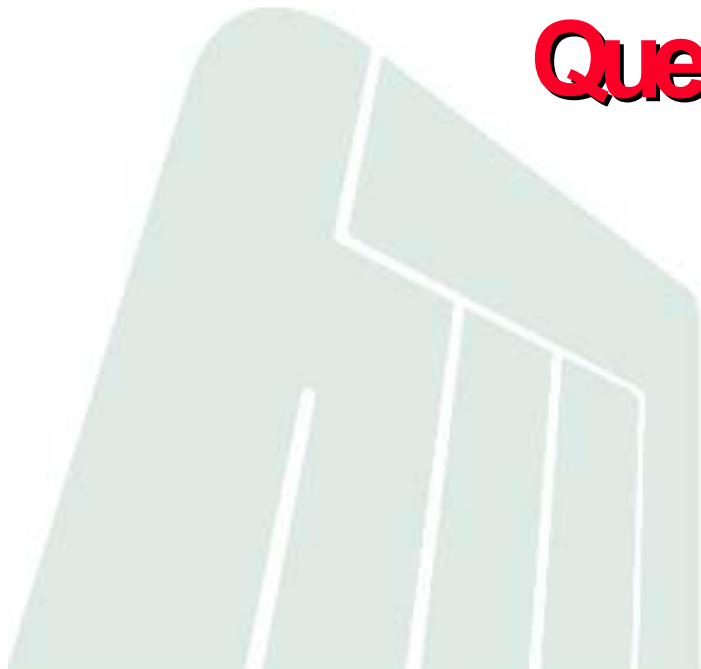
- Initial research on a three-way merge that
  - supports XML/HTML document restructuring
  - handles malformed input
  - uses string-based techniques
- Simple design
- Encouraging results
  - but matching remains a problem



HELSINKI  
INSTITUTE FOR  
INFORMATION  
TECHNOLOGY

**Thank you!**

**Questions, please?**



HELSINKI  
INSTITUTE FOR  
INFORMATION  
TECHNOLOGY