Chained Forests for Fast Subsumption Matching

Adj. Prof. Sasu Tarkoma
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
sasu.tarkoma@tml.hut.fi

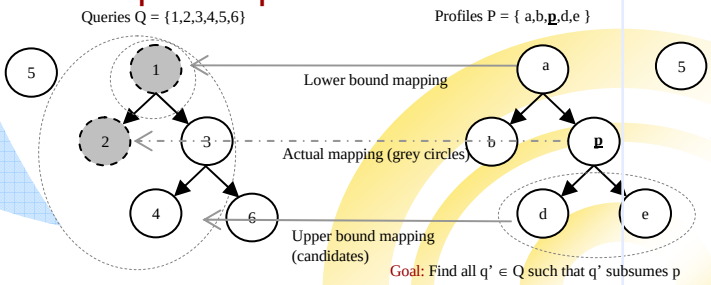
Introduction

- **Content provisioning and delivery are becoming increasingly popular and important.**
- We present a novel scheme for the maintenance and matching of partial orders.
- Partial order derives from the subsumption relation inherent the collection of objects being matched.
- The proposed chaining technique has applications in information routing, collection tracking, and peer-to-peer information exchange.
- Track result set of a continuous query with insertions and deletions.

Overview

- Content is defined using profiles. Queries select a subset of profiles.
- Assumption: There are subsumption relations in the input sets.
- Idea: build forests of each set (profiles/queries) and maintain mappings between these sets that determines subsumption between sets.
- Essentially chain two forests with the mappings. The mechanism generalizes to a chain of forests.
- Research question: How to minimize the number of subsumption test operations when inserting/deleting a profile or a query?

Add profile p

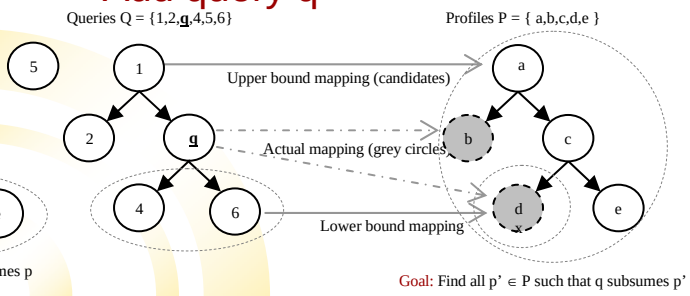


Optimizing computation of the mappings for p:

$M_{PQ}(children_P(p))$ *subsumes* $M_{PQ}(p)$

$M_{PQ}(p)$ *subsumes* $M_{PQ}(parent_P(p))$

Add query q



Optimizing computation of the mappings for q:

$M_{QP}(parent_Q(q))$ *subsumes* $M_{QP}(q)$

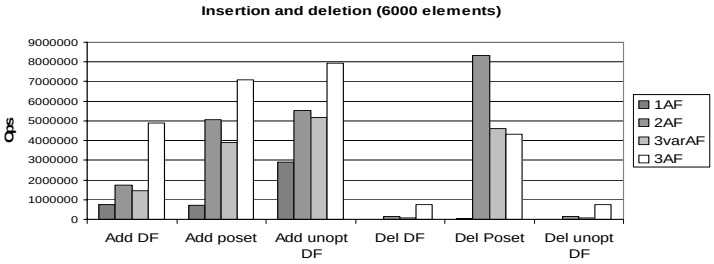
$M_{QP}(q)$ *subsumes* $M_{QP}(children_Q(q))$

Algorithm and Optimizations

- Add profile: All elements that subsume the input profile are found by traversing the query forest towards subsuming elements.
- Add query: All subsumed elements are found by traversing the profile forest. All matching elements and their children all added to the result set.
- Optimizations: Find candidate set for testing inclusion into the result set. Determine upper and lower bounds for the result set and inspect their intersection.
- Experimental results suggest significant improvement with the optimization.

Results

- Generalizes to multiple sets, each corresponding to a forest in a chain of forests.
- Separates mappings from the forests.
- Improved insertion and lookup cost. Improved or degraded deletion cost and structure size depending on the workload. Sparse bitstrings may be used to alleviate space concerns.
- Nature of optimizations suggest that copes well with self-similar workload.
- Future work investigates how to compact mappings.



	DF	Poset	Forest
1AF	1760497	14023	3484
2AF	544265	243826	5998
3varAF	779112	185338	5852
3AF	87026	208447	6000

Size after inserting 6000 elements

	DF	Poset	Forest
1AF	655	5724	317695
2AF	675	3066	637056
3varAF	667	3200	406348
3AF	719	673	937292

Time (ms) for 30 000 lookups for 3000 elements