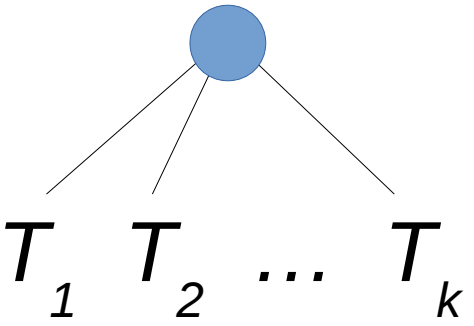
Testing for C1P using PQ-Trees

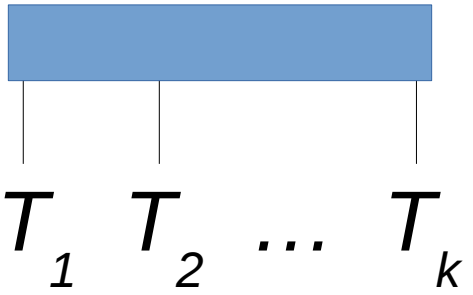
- C1P: order U so that certain sets $S \subset U$ are consecutive
- PQ-Tree: represents admissible permutations
- Online algorithm: tree changes with every new set added
- Reduction algorihtm: linear in |S| under amortized analysis
- Size of input: $n + m + f$

PQ-Trees

- Universal set U
- Leaves: elements of U
- Internal nodes:
  - P nodes



  - Q nodes



Proper PQ-Trees

- Every element of U appears exactly once as a leaf
- Every P node has at least 2 childern
- Every Q Node has at least 3 children

Frontier

- Reading the leaves from left to right

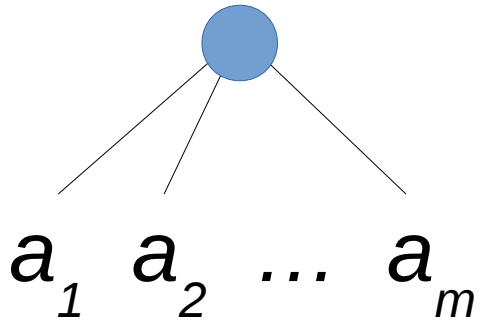
Equivalent trees

- Notation: $T_1 = T_2$
- T_1 and T_2 are equivalent when one can be transformed into the other by zero or more equivalence transformations:
  - arbitrarily permute the children of a P node
  - reverse the children of a Q node

CONSISTENT(T) = {FRONTIER(T') | T' = T}

Universal tree and null tree

- Universal tree: |U| = m



- Null tree = ∅ CONSISTENT(∅) = ∅

Modern implementation

PQTree<E>

universalSet: set of <E>
root: PQNode<E>
---
proper(): boolean
frontier(): list of <E>
equivalent(PQTree<E>):
boolean
consistent(): set of
<list of <E>>
isUniversal(): boolean
reduce(set of <E>)

PQNode<E>

parent: PQNode<E>
childList: list of
<PQNode>
type: Enum(P,Q,L)

Reducing PQ-trees

- Input: PQ-tree T, set $S \subseteq U$
- Bottom-up: starts at leaves; process each node only after all its children have been processed
- Processing each node:
  - Find a pattern that applies to it
  - Replace the pattern by the replacement
- Stop when:
  - No pattern for a node: return null tree
  - After processing a node that “contains” S

Modern implementation (2)

PQTree<E>

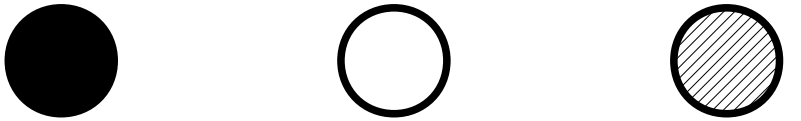
universalSet: set of <E>
root: PQNode<E>
---
proper(): boolean
frontier(): list of <E>
equivalent(PQTree<E>):
    boolean
consistent(): set of
    <list of <E>>
isUniversal(): boolean
reduce(set of <E>)

PQNode<E>

parent: PQNode<E>
childList: list of
    <PQNode>
type: Enum(P,Q,L)
---
findPattern(): Pattern
apply(Pattern)
contains(set of <E>):
    boolean

Processing a node with respect to S

- Full, empty, partial



- Pertinent: full or partial
- Pertinent subtree: contains S
- ROOT(T,S): lowest node whose frontier contains S

Modern implementation (3)

PQTree<E>

universalSet: set of <E>
root: PQNode<E>
---
proper(): boolean
frontier(): list of <E>
equivalent(PQTree<E>):
    boolean
consistent(): set of
    <list of <E>>
isUniversal(): boolean
reduce(set of <E>)
root(set of <E>): PQNode<E>

PQNode<E>

parent: PQNode<E>
childList: list of
    <PQNode>
type: Enum(P,Q,L)
---
findPattern(): Pattern
apply(Pattern)
contains(set of <E>):
    boolean
frontier(): list of <E>

Templates: pattern and replacement

- Goal: after replacement, the frontier of the subtree rooted at this node, as well as all equivalent subtrees, have all their pertinent leaves consecutive
- Affects node and its children only
- Type of node and label of children (full, empty, partial) matter
- Types of children do not matter

any type

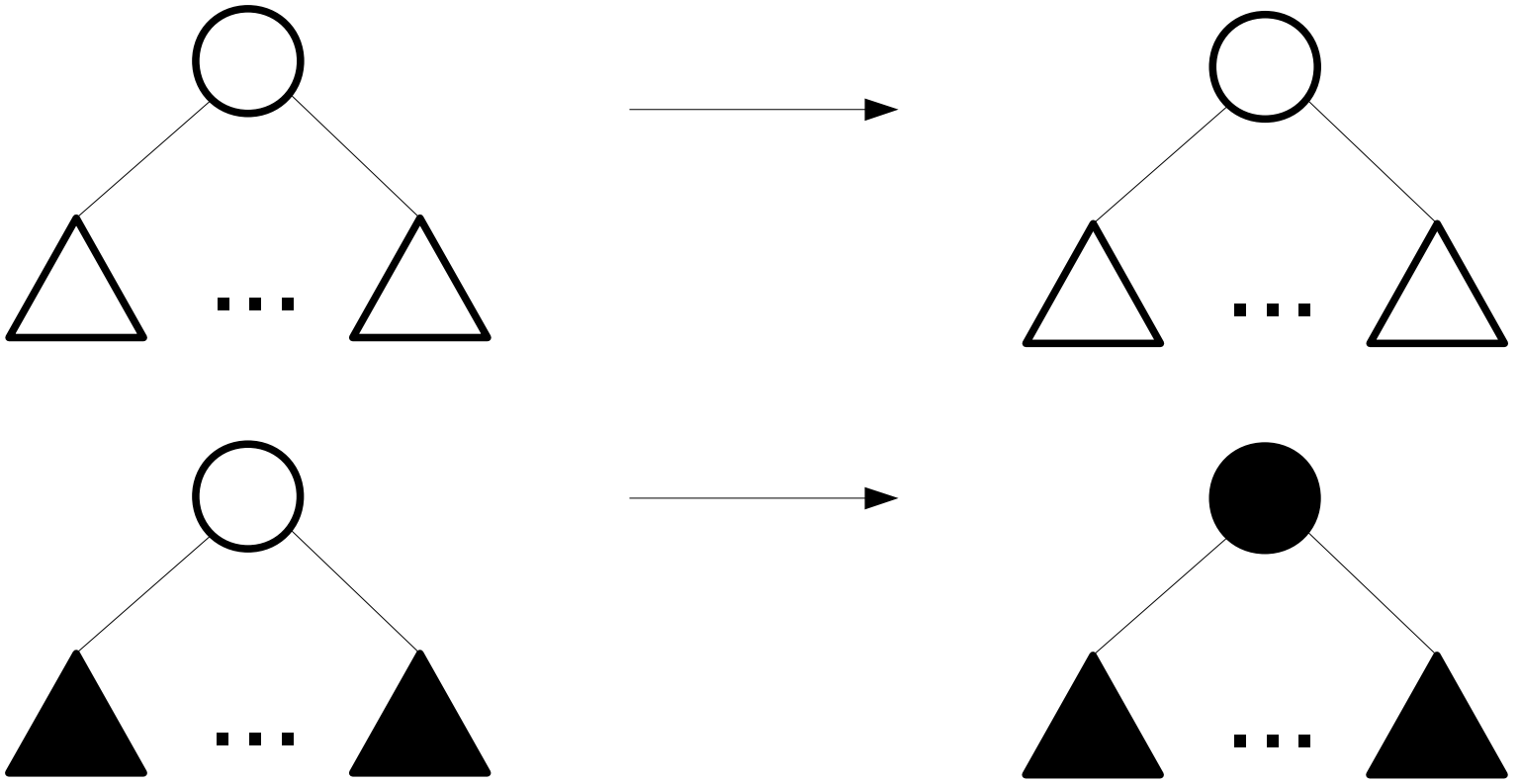
Templates for leaves

- Just label as empty or full

Templates for P-nodes

Pattern

Replacement



Templates for P-nodes (cont.)

- Templates when some children are empty and some full
  - Depend on node being ROOT(T,S) or not
  - If node is not ROOT(T,S), creates a partial P-node
- Templates when there are partial children
  - One or two partial children ok
  - More than two partial children: no pattern
- All templates: apply to equivalent nodes as well

Templates for Q-nodes

- Templates for all children empty or all children full
- Templates for some children empty and some full
- Depend on node being ROOT(T,S) or not
- Templates for partial children
  - One or two partial children ok
  - More than two partial children: no pattern
- All templates: apply to equivalent nodes as well

Efficient implementation

- Need to avoid empty subtrees to guarantee amortized |S| time
- Prunned pertinent subtree
- Two bottom-up phases
  - Bubbling up: identifying prunned pertinent nodes
  - Reduce: reducing prunned pertinent nodes
- Problem with parent pointers: blocking
- NORM(T): potential for amortized analysis