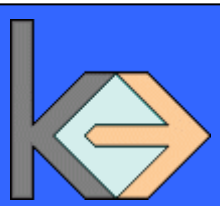
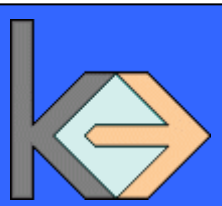


Efficient Discovery of Frequent Unordered Trees

uFREQT

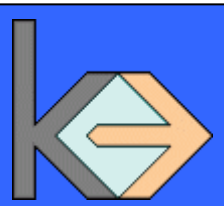
Veronika Kostadinova, Fan Zhang

Content



- Motivation
- Basic Definitions
- Enumeration
- Tree Counting
- Conclusion

Motivation

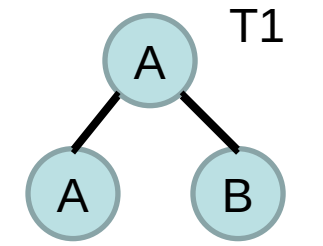


- In einigen strukturierten Datenbank ist der Kinderordnung nicht relevant
- Die Kinderordnung ist nicht vorgegeben
- Algorithmen für geordnete Bäume können nicht direkt verwendet werden.

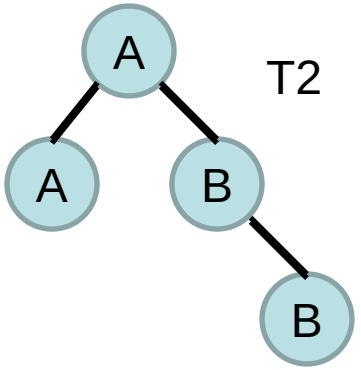
- => uFreqt

Definitionen

- Pre-Order String:
  - DFS
  - l(υ)=(depth(υ), label(υ))
- Prefix:
  - Tree T_1, T_2
  - $l(T_1)$ ist Prefix von $l(T_2)$



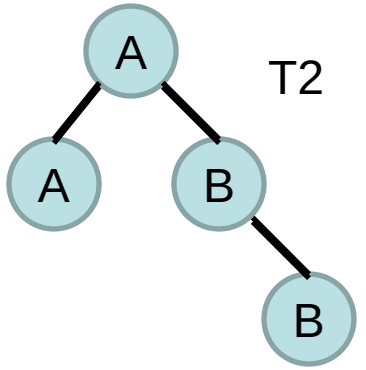
(0,A)(1,A)(1,B)



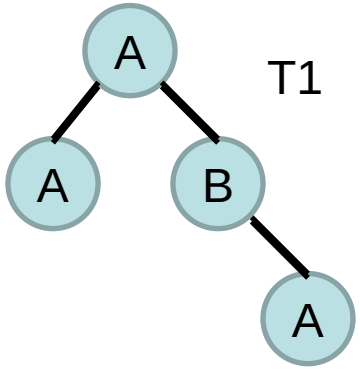
(0,A)(1,A)(1,B)(2,B)

Definitionen

- Vergleich von Tupeln:
- $(d_1,l_1) < (d_2,l_2)$,
  - gdw. $d_1 > d_2$ oder
  - $l1 < l_2$ if $d_1 = d_2$
- Vergleich von Trees:
- $l(T_1) < l(T_2)$
  - T_2 ist Prefix von T_1 oder
  - $(d_1,l_1) \in T_1 < (d_2,l_2) \in T_2$
  - (d_1,l_1) ist linkste Position, wo $l(T_1)$ und $l(T_2)$ unterschiedlich sind



(0,A)(1,A)(1,B)(2,B)

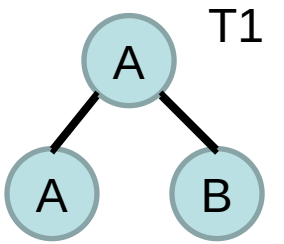


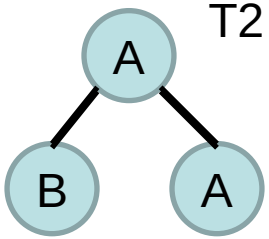
(0,A)(1,A)(1,B)(2,A)

Definitionen

- Normalform:
  - l (subtree (υ)) ⩽ l(subtree(υ ´)) wobei υ ´ ist next_sibling(υ)
  - Wenn ein geoordnete Tree in NF, es representiert alle aquivalente ungeordnete Trees
  - Beispie: T1, T2 sind äquivalente ungeordnete Trees

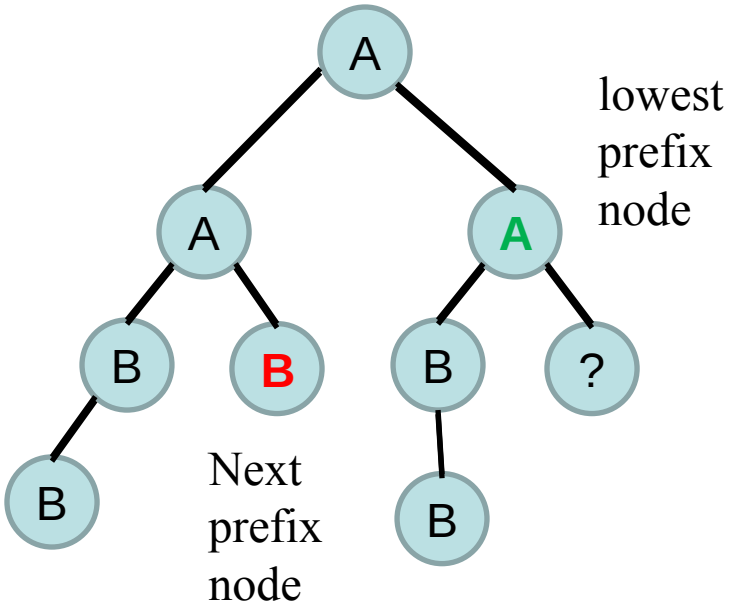
    T1 < T2 => T1 ist in NF





Definitionen

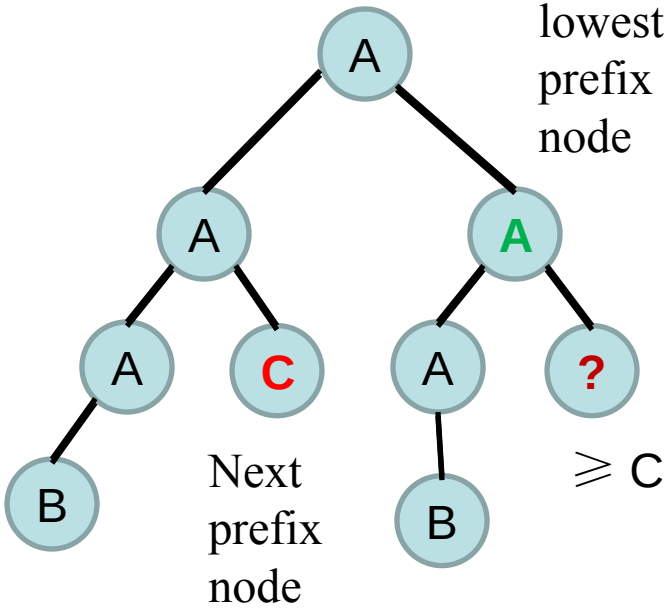
- lowest prefix node υ :
  - an dem RMB(T)
  - subtree(υ) hat maximale Größe
  - Subtree(υ) ist Prefix von Subtree(prevsibling(υ))
  - oder nicht definiert
- next prefix node:
  - Knoten in subtree (prevsibling(υ)), der als nächster von subtree(υ) kopiert wird



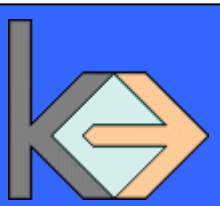
Gültige Expansion

- eine rightmost Expansion mit (d,l) bietet ein Tree in NF gdw.
  - (i) (d,l) $\geqslant$ p(d), wobei
    p(d) ist Label vom Knoten mit Tiefe d am RMB von T
  - (ii) (d,l) $\geqslant$ (d′,l′), wobei
    (d′,l′) ist **next prefix node**

* (ii) gilt nur, wenn **lowest prefix node** ist definiert.



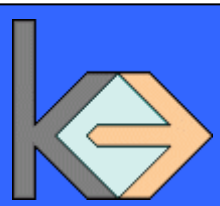
uFreqt - Enumeration



1. print(T)
2. **if** t is defined **then**
3. Increase t
4. Let (d, l) be the tuple at position t in $l(T)$.
5. Enumerate (T expanded with (d, l), $l(T) \cdot (d, l)$, t).
6. **for each** rightmost expansion $(d', l') > (d, l)$, $l' \geq l(p(d'))$ **do**
7. **if** $l' = l(p(d'))$ **then**
8. Enumerate (T expanded with (d', l'), $l(T) \cdot (d', l')$, position of $p(d')$ in $l(T)$);
9. **else**
10. Enumerate (T expanded with (d', l'), $l(T) \cdot (d', l')$, undefined);
11. Decrease t
12. **else**
13. **for each** rightmost expansion (d', l'), $l' \geq l(p(d'))$ **do**
14. **if** $l' = l(p(d'))$ **then**
15. Enumerate (T expanded with (d', l'), $l(T) \cdot (d', l')$, position of $p(d')$ in $l(T)$);
16. **else**
17. Enumerate (T expanded with (d', l'), $l(T) \cdot (d', l')$, undefined);

Fig. 3. An algorithm for enumerating all trees in normal form.

Enumeration



Input:

T: ein Tree in NF

L(T): pre-order-String von T

t: ein Index, entweder als nicht definiert oder als eine Position in l(T)

(es hilft bei Suchen von Nextprefix node)

Output:

Eine Menge von Kinderbäume durch eine Expansion

- Case I :

i) Code line 3-5

- Linke Bruder Baum von Lowest Prefix node wird defniert.
- Next Prefix node wird von t+1 verwiesen.
- Tupel (d,l) gleich next Prefix node wird als neuen Tupel expandiert.

Enumeration

ii) Code line 6-8

- Für alle Expandieren mit (d', l'), die für folgende Kriterien gelten:
  a) $(d', l') \geqslant p(d')$,
  b) $(d', l') \geqslant (d, l)$

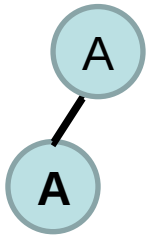
- T wird als neu definiert, oder nicht definiert.

- Case II: code line 13 -17
- Für alle Expandieren mit (d', l'), die für folgende Kriterien gelten:
  a) $(d', l') \geqslant p(d')$,

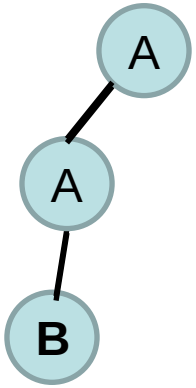
- T wird als neu definiert, oder nicht definiert.

Enumeration: Beispiel

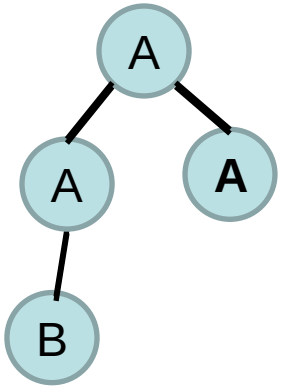
l(T)={0A}
t = undefined



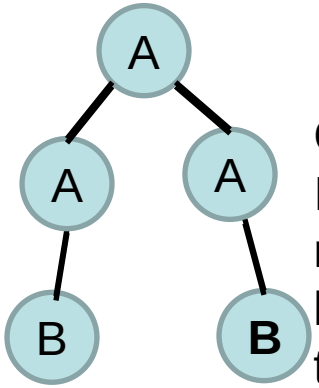
Case 2
l(T)={0A, **1A**}
t = undefined



Case 2
l(T)={0A, 1A, **2B**}
t = undefined

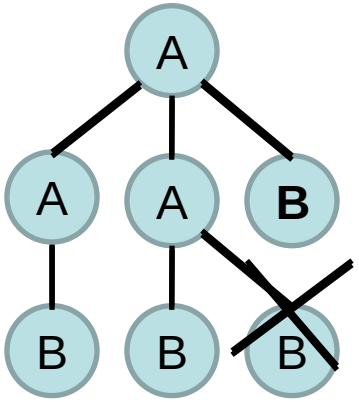


Case 2
l(T)={0A, 1A, 2A, **1A**}
t = 2, wegen l´ =l(P(d´))



Case 1 i)
Increase t, t = 3 Next Prefix node wird gefunden
l(T)={0A, 1A, 2B, 1A, **2B**}
t = 3, Next prefix node wird kopiert

Beispiel: Go on

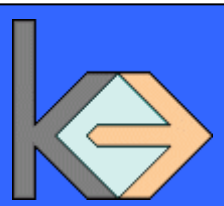


Case 1 ii)

l(T)={0A, 1A, 2B, 1A, 2B, **1B**}
t= undefined, wegen l´ $\geqslant$ (P(d´))
(t hat sich nicht reduziert,
deswegen von Loop verlassen)

3B ist nicht erlaubt, weil Kopie
ist schon vollständig.

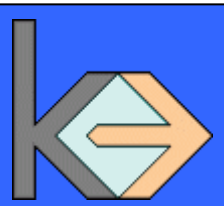
Tree Counting



- Unendlich viele ungeordnete Trees möglich
- Threshold für frequency

- Tree Mapping
- Maximum Bipartite Matching Problem

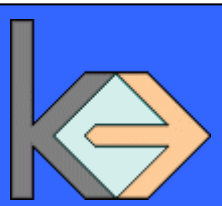
Tree Counting



Algorithm Update
Input: a tree T in normal form, its associated mappings, and an expansion (d, l).
Output: an expanded tree T in which the mappings have been updated.

1. Let v be the node at depth $d-1$ on the rightmost path of the pattern tree.
2. Add a node v' at depth d with label l
3. **for all** $m \in \text{Map}(v)$ **do**
4. Let w be the node in the data tree to which m maps
5. k_1, k_2 := number of children of w (respectively v) with label l
6. **if** $k_1 - k_2 \geq 0$ **then**
7. **for all** children w' of w with label l **do**
8. Add to $\text{Map}(v')$ a mapping from v' to w'
9. **else** Remove Mappings(m)

Tree Counting



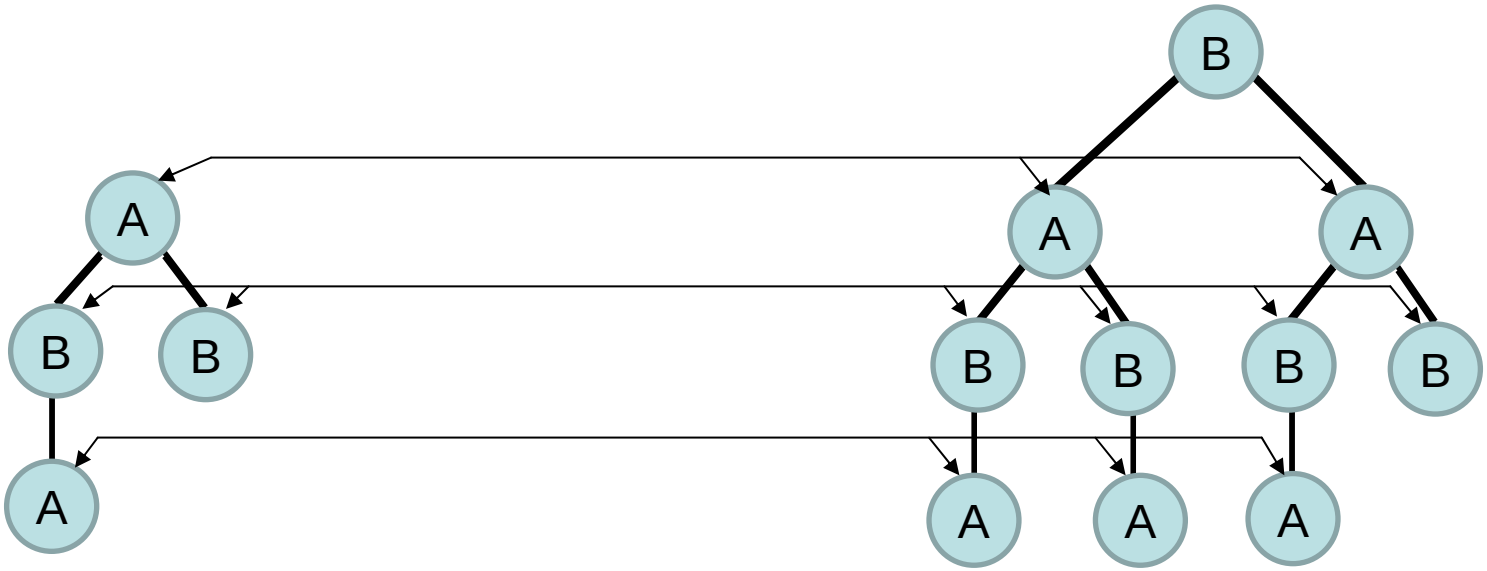
Procedure Remove Mappings
Input: a mapping m from a node v in T to a node w in a data tree
Output: a tree in which m is removed and the mappings of all nodes which are a child of the rightmost path are updated accordingly.

10. Let m' be parent(m), if v is not the root
11. Remove Mappings Below(m)
12. **if** v is not the root of T **then**
13. Let G be the (new) bipartite graph matching problem associated with m'
14. **if** G has no bipartite matching **then** Remove Mappings(m')

Procedure Remove Mappings Below
Input: a mapping m from a node v in T to a node w in a data tree
Output: a tree in which m is removed and the mappings of all nodes which are below v and are child of the rightmost path are updated accordingly.

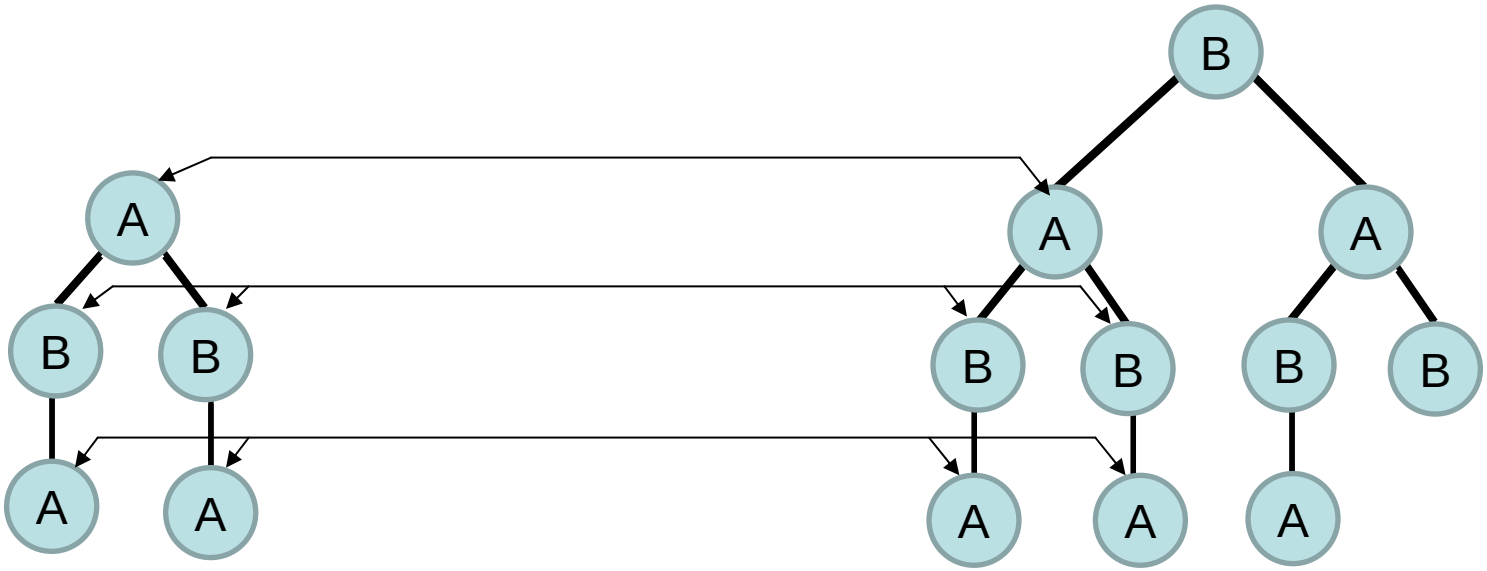
15. **for all** children v' of v not on the rightmost path of T **do**
16. **for all** $m' \in$ Map(v', m) **do** Remove m' from Map(v')
17. Let v' be the child of v on the rightmost path of T
18. **for all** $m' \in$ Map(v', m) **do** Remove Mappings Below (m')
19. Remove m from Map(v)

Tree Counting: Beispiel

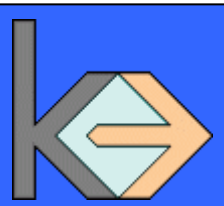


Tree Counting: Beispiel

Update nach hinfügen von (2, A):



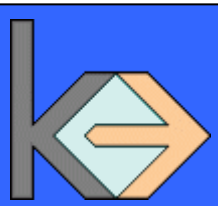
Conclusions



- Freqt, Unot, uFreqt untersuchen induced trees with DFS Travesierung

- Unot und uFreqt suchen für unordered trees, Freqt für ordered

- Freqt sucht in ein Wald, uFreqt in ein einziges Datenbaum



Danke für Ihre Aufmerksamkeit!