Frequent subsequences, episodal rules

Jiří Kléma

Department of Cybernetics,
Czech Technical University in Prague

http://ida.felk.cvut.cz

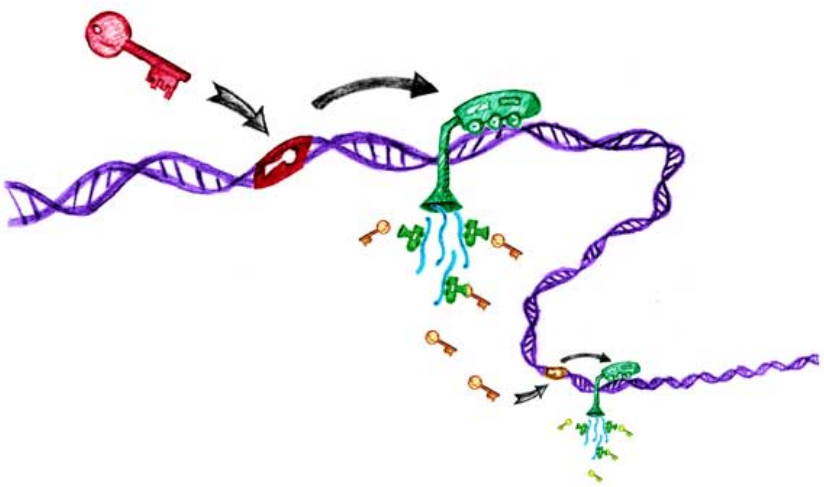
Outline

- Motivation for frequent subsequence search
  - applications, variance in needs and sequence definitions,
- what do we already know?
  - connection to itemsets, what changed?
  - directed sequences, without noise/gaps and time,
- why is it sometimes more difficult?
  - undirected sequences a their canonical form,
  - (complete) transactional representation and connected definitions,
  - GSP algorithm (Agrawal's APRIORI generalization),
  - other algorithms – FreeSpan, PrefixSpan,
- summary
  - categorization of methods according to sequence and pattern types,
- STULONG – a case study.

Frequent subsequences – example 1: DNA

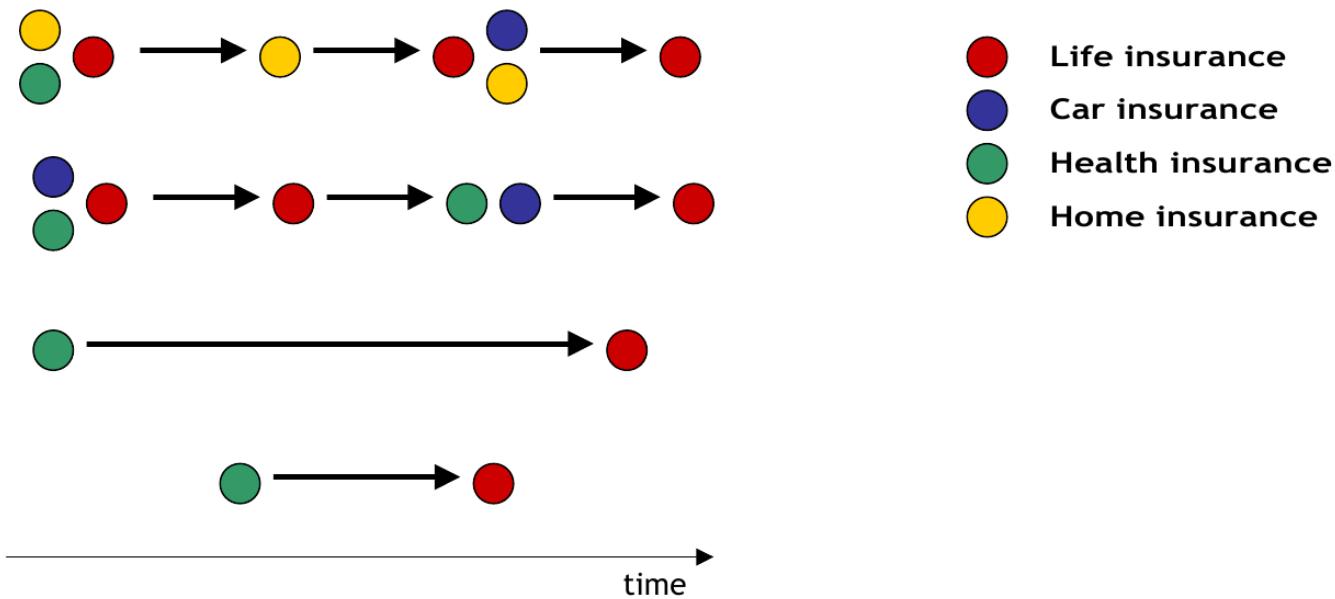
- **motif discovery**
  - searches for short sequential patterns in a file of unaligned DNA or protein sequences,
  - searches for **discriminative** patterns (characteristics)
    - typical for one sequence class, unusual in the other classes,
    - this pattern could relate with the biological/regulation function of the (protein) class,
  - transcription factor interacts with DNA through a particular motif,
  - frequent subsequence search is a subtask,
- event = nucleotide, string (no time), undirected DNA.

```
>Z3066
atgacgtgtctgtatattaataagctaacccgcattgagttaaccaataacggattccatacacaatacg
gccaacagagaaagatacctgtgctcacgccattgcttatattggctggttacaatgtgcactatcaatt
ttttaaataa
>Z3067
ttgactgtgaaaattacaggagctacaaaaatgaaccgattctcaaaaactcaaatttatttacattgga
taacgctgcttttcgttgcaataacctatgccgcgatggaactccgtggctggtttcctaaaggtagtag
tacttatctgctgatgcgagaaacacattacaatgcgggtatattcgtttgggtgttaatgttttcacgc
ctgattataaaacaccgttatagtgatccttctattgtgccaccgccacctgcctggcaaatgaaagcgg
cttcgctaatgcacatcatgctttatataaccttccttgcgcttcctctgctggggattgctttgatggc
ttacagtggaaaatcgtggagtttccttggtttcaatgtgtctccctttgttaccccaaacagcgaaatt
aaagcactgataaaaaatattcacgaaacctgggcaaatataggctactttttaatcgtagctcacgctg
gcgcagcactctttcatcactacattcagaaagataatactctgttacgaatgatgcctcgccgcaaata
a
>Z3069
gtggcggagagagggggatttgaacccccggtagagttgcccctactccggttttcgagacctatgctat
gggttaataaaatcaatatattatgtgttttatttggaataaatattctatatttaggattgaaaaatca
gatggttagcatcaaacaacctcagaatattccaagcaaacaggttaaaaataaaactgcacccgaacaa
ttgataacgacagaaaacgccttttccgagccaccaaaattacttcgcatcctatttattgctcacgtta
acgccttgtataactcgagctctccacggtatttaacctctcttctgtttaactataattccaataaatc
tcgtcactga
```



Frequent subsequences – example 2: insurance

- event is signing of a insurance contract of a certain type,
- several events may occur concurrently,
- sequence is a chronological series of events,
- analogy: directed acyclic graph, the edge length equals time between events,
- interesting patterns: typical contract sequences (signed in a reasonable time span).



Meyer: Sequence Mining in Marketing.

Frequent subsequences – similarity to frequent itemsets

- first of all, similarity in task **representation**,
- the process can be intrinsically identical, but we ask different questions
  - itemsets: which insurance contracts people arrange concurrently,
  - sequences: how people arrange insurance contracts in their course of life,
- transaction representation still formally possible and helpful (universal)
  - more factors must be concerned and stored.

Transaction	Items (insurance type)
t_1	home, life
t_2	car, home
t_3	pension, life
t_4	travel
t_5	pension, life
...	...

Customer	Date (time)	Items (insurance type)
c_1	5.10.2003	home, life
c_1	8.1.2005	travel
c_1	3.8.2010	car, pension
c_2	10.10.2003	car, home
c_2	20.11.2006	pension
...	...	...

Frequent subsequences – similarity to frequent itemsets

- secondly, similarity in terms of task **solution**,
- APRIORI property can easily be generalized for sequences:
  Each subsequence of a frequent sequence is frequent.
- the anti-monotone property can also be transformed to monotone one:
  No supersequence of an infrequent sequence can be frequent.
- the model APRIORI-like algorithm for sequential data
  - a direct analogy of the APRIORI algorithm for itemsets,
  - the basic operations (informal – a reminder only):
    1. search for trivial frequent sequences (typically of the length 0 or 1),
    2. generate candidate sequences with the length incremented by 1,
    3. check for their actual support in the transaction database,
    4. reduce the candidate sequence set
       - a subset of frequent sequences of the given length is created,
    5. until the frequent sequence set non-empty go to the step 2.

Frequent substrings – a trivial APRIORI application

- a string
  - a directed sequence, an equidistant step, exactly one item per transaction,
  - events given by a symbol alphabet, a pattern is an ordered list of **neighboring** events,
  - $\langle a_1 \dots a_m \rangle$ is a subsequence of $\langle b_1 \dots b_n \rangle$ iff $\exists i \; a_1 = b_i \wedge \dots \wedge a_m = b_{i+m}$.
- Example: DNA sequence ($n = 20$, $A = \{a, g, t\}$)

t t g a a a g g g g g t t g a a t g t t s>10%, s=f/(n-m+1)

t t g a a a g g g g g t t g a a t g t t s_{ggg}=3/18 , s_{ttgaa}=2/16

i	C_i	L_i
1	{a}, {g}, {t}	{a}, {g}, {t}
2	(9 patterns)	{aa}, {ga}, {gg}, {gt}, {tg}, {tt}
3	{aaa}, {gaa}, {gga}, ...(12 patterns)	{gaa}, {ggg}, {gtt}, {tga}, {ttg}
4	{gggg}, {gttg}, {tgaa}, {ttga}	{gggg}, {tgaa}, {ttga}
5	{ttgaa}	{ttgaa}

- how to check for support quickly, i.e. how to find all subsequence occurrences in a sequence?
  - algorithms Knuth-Morris-Pratt or Boyer-Moore.

Canonical form for sequences

- a **canonical** (standard) **code word**
  - a unique sequence representation, based on the symbol alphabet ordering,
  - a usual (not necessary) choice:
    - the lexicographical symbol alphabet ordering $a < b < c < \ldots$,
    - the lexicographically smallest (smaller) code word taken as canonical ($bac < cab$),
- a directed sequence
  - the only interpretation (way of reading), each (sub)sequence is a canonical code word,
- an undirected sequence
  - two possible ways of reading = two alternative code words,
  - the routine application of lexicographical ordering is not possible,
  - **prefix property** in a space of canonical code words does not hold:
    - every prefix of a canonical word is a canonical word itself,

sequence	canonical form	prefix	canonical form
bab	bab	ba	ab
$cabd$	$cabd$	cab	bac

  - we have to find a different way of forming code words.

Canonical form for undirected sequences

- The canonical code words with the prefix property will be formed as follows
  - even and odd length words will be handled separately,
  - code words are started in the middle of sequence,

	even length	odd length
sequence	$a_m\ a_{m-1}\ \ldots\ a_2\ a_1\ b_1\ b_2\ \ldots\ b_{m-1}\ b_m$	$a_m\ a_{m-1}\ \ldots\ a_2\ a_1\ a_0\ b_1\ b_2\ \ldots\ b_{m-1}\ b_m$
code word	$a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$	$a_0\ a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$
code word	$b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$	$a_0\ b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$

- canonical is the lexicographically smaller code word in the table,
- the sequence is **extended** by adding
  - a pair $a_{m+1}\ b_{m+1}$ or $b_{m+1}\ a_{m+1}$,
  - one item at the front and one item at the end.
- an example

even length			odd length		
sequence	code words		sequence	code words	
at	*at*	*ta*	*ule*	*lue*	*leu*
data	*atda*	*taad*	*rules*	*luers*	*leusr*

Canonical form for undirected sequences – the prefix property

- **Prefix property proof** for the new representation **by contradiction**

  1. suppose the prefix property does not hold,
  2. then there exists a canonical code word $w_m = a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$,
  3. whose prefix $w_{m-1} = a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}$ is not a canonical code word,
  4. as a consequence we have $w_m < v_m$, where $v_m = b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$,
  5. and $v_{m-1} < w_{m-1}$, where $v_{m-1} = b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}$,
  6. however $v_{m-1} < w_{m-1} \Rightarrow v_m < w_m$
     - because v_{m-1} is a prefix of v_m and w_{m-1} is a prefix of w_m,
  7. $v_m < w_m$ from the step 6 contradicts $w_m < v_m$ from the step 4 $\square$.

Canonical form for undirected sequences – efficiency

- two possible code words can be created and compared in $\mathcal{O}(m)$,
- an additional **symmetry flag** introduced for each sequence enables the same in $\mathcal{O}(1)$

$$s_m = \bigwedge_{i=1}^{m} (a_i = b_i)$$

- the symmetry flag is maintained in constant time with

$$s_{m+1} = s_m \ \wedge \ (a_{m+1} = b_{m+1})$$

- sequence extension is **permissible** when the flag:
  - if $s_m =$ true, it must be $a_{m+1} \leq b_{m+1}$,
  - if $s_m =$ false, any relation between a_{m+1} and b_{m+1} is possible.
- sequences and symmetry flags at the beginning
  - even length: an empty sequence, $s_0 = 1$,
  - odd length: all frequent alphabet symbols, $s_1 = 1$,
- the procedure guarantees exclusively the canonical sequence extensions.

Frequent subsequences – APRIORI application to undirected sequences

- consider undirected sequences, otherwise the formalization as yet
  - $\langle a_1 \dots a_m \rangle$ is a subsequence of $\langle b_1 \dots b_n \rangle$ if:
    $\exists i \; a_1 = b_i \wedge \dots \wedge a_m = b_{i+m}$, or
    $\exists i \; a_1 = b_{i+m} \wedge \dots \wedge a_m = b_i$.
- Example: DNA sequence ($n = 20$, $A = \{a, g, t\}$)

t t g a a a g g g g g t t g a a t g t t s>10%, s=f/[2(n-m+1)]

t t g a a a g g g g g t t g a a t g t t s_{gtt}=4/36 , s_{gggg}=4/34

i	C_i	L_i
0	{}	{}
1	{a}, {g}, {t}	{a}, {g}, {t}
2	**{aa}, {ag}, {at}, {gg}, {gt}, {tt}**	**{aa}, {gg}, {gt}, {tt}**
3	{aaa}, ~~{aag}~~, ~~{aat}~~, ~~{gag}~~, ~~{gat}~~, ~~{tat}~~, ~~{aga}~~, ~~{agg}~~, ~~{agt}~~, {ggg}, {ggt}, ~~{tgt}~~, ~~{ata}~~, ~~{atg}~~, ~~{att}~~, ~~{gtg}~~, {gtt}, {ttt}	{ggg}, {gtt}
4	**~~{aaaa}~~, ~~{aaag}~~, ~~{aaat}~~, ~~{gaag}~~, ~~{gaat}~~, ~~{taat}~~, ~~{agta}~~, ~~{agtg}~~, ~~{ggta}~~, ~~{agtt}~~, ~~{tgta}~~, ~~{ggtg}~~, ~~{ggtt}~~, ~~{tgtg}~~, . . . in total ~~27~~ (1) patterns**	**{gggg}**

A generalized subsequence definition in transactional representation

- Items: $I = \{i_1, i_2, \ldots, i_m\}$,
- itemsets: $(x_1, x_2, \ldots, x_k) \subseteq I$, $k \geq 1$, $x_i \in I$,
- sequences: $\langle s_1, \ldots, s_n \rangle$, $s_i = (x_1, x_2, \ldots, x_k) \subseteq I$, $s_i \neq \emptyset$, $x_1 < x_2 < \ldots < x_k$,
  - an ordered list of elements, elements = itemsets,
  - the canonical representation: lexicographical ordering of items in each itemset,
  - example: $\langle a(abc)(ac)d(cf) \rangle$, a simplification of the form: $(x_i) \sim x_i$,
- the sequence length l
  - given by the number of item instances (occurrences) in sequence,
  - l-sequence contains exactly l item instances,
  - ex.: $\langle a(abc)(ac)d(cf) \rangle$ is a 9-sequence,
- α is a subsequence of β, β is a supersequence of α: $\alpha \sqsubseteq \beta$
  - $\alpha = \langle a_1, \ldots, a_n \rangle$, $\beta = \langle b_1, \ldots, b_m \rangle$, $\exists 1 \leq j_1 < \ldots < j_n \leq m, \forall i = 1 \ldots n : a_i \subseteq b_{j_i}$,
  - example: $\langle a(bc)df \rangle \sqsubseteq \langle a(abc)(ac)d(cf) \rangle$, $\langle d(ab) \rangle \not\sqsubseteq \langle a(abc)(ac)d(cf) \rangle$
- a sequence database: $S = \{\langle sid_1, s_1 \rangle \ \ldots, \langle sid_k, s_k \rangle\}$
  - a set of ordered pairs a sequence identifier and a sequence.

Subsequence search in transaction representation

- Support of α sequence in the database S
  - the number of sequences $s \in S$ satisfying: $\alpha \sqsubseteq s$,
- Subsequence search in transaction representation, task definition
  - input: S a s_{min} – minimum support,
  - output: the complete set of frequent sequential patterns
    - $*$ all the subsequences with or above the threshold frequency.

Id	Sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Id	Time	Items
10	t_1	a
10	t_2	a, b, c
10	t_3	a, c
10	t_4	d
10	t_5	c, f

l	sequential pattern (s_{min}=2)
3	$\langle a(bc)\rangle$, $\langle aba\rangle$, $\langle abc\rangle$, $\langle (ab)c\rangle$, $\langle (ab)d\rangle$, $\langle (ab)f\rangle$, $\langle aca\rangle$, $\langle acb\rangle$, $\langle acc\rangle$, $\langle adc\rangle$, ...
4	$\langle a(bc)a\rangle$, $\langle (ab)dc\rangle$, ...

Pei, Han et al.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.

GSP: Generalized Sequential Patterns [Agrawal, Srikant, 1996]

- applies the core idea of APRIORI to sequential data,
- the key issue is generation of the candidate sequential patterns
  - divided into two steps
    1. **join**
       - l-sequence is created by joining of two (l-1)-sequences,
       - (l-1)-sequences can be joined when identical after removal of the first item in one and the last one in second,
    2. **prune**
       - skip each l-sequence which contains an infrequent (l-1)-subsequence.

L_3	C_4	
	after join	after prune
$\langle(ab)c\rangle$, $\langle(ab)d\rangle$, $\langle a(cd)\rangle$, $\langle(ac)e\rangle$, $\langle b(cd)\rangle$, $\langle bce\rangle$	$\langle(ab)(cd)\rangle$ $\langle(ab)ce\rangle$	$\langle(ab)(cd)\rangle$

Agrawal, Srikant: Mining Sequential Patterns: Generalizations and Performance.

Example: GSP, s_{min}=2

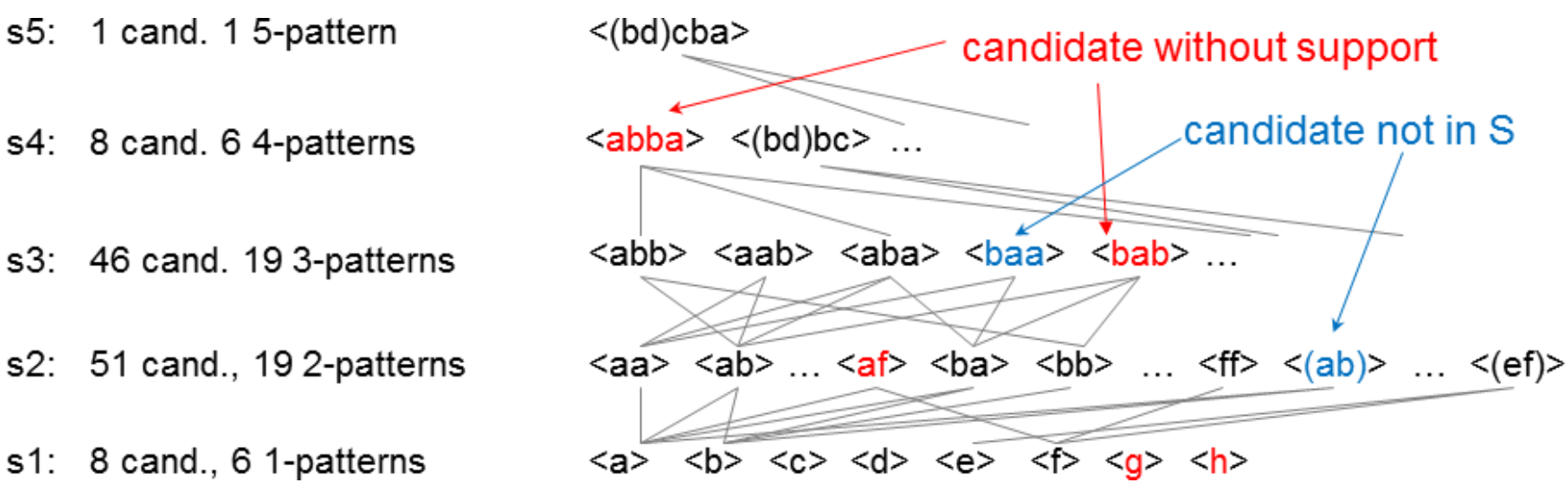
Id	Sequence
10	$\langle(bd)cb(ac)\rangle$
20	$\langle(bf)(ce)b(fg)\rangle$
30	$\langle(ah)(bf)abf\rangle$
40	$\langle(be)(ce)d\rangle$
50	$\langle a(bd)bcb(ade)\rangle$

- $s(\langle g\rangle) = s(\langle h\rangle) = 1 < s_{min}$

  (skips a large portion of 92 available 2-candidates),
- $\langle(bd)cba\rangle \sqsubseteq s_{10} \wedge \langle(bd)cba\rangle \sqsubseteq s_{50}$

  (created from $\langle(bd)cb\rangle$ a $\langle dcba\rangle$),

  (the patterns $\langle(bd)ba\rangle$, $\langle(bd)ca\rangle$ and $\langle bcba\rangle$ must also be frequent).



APRIORI algorithm for sequences – disadvantages

- the generate (join step) and test (prune step) method,
- the problems discussed in terms of frequent itemsets persist and intensify
  1. generates a large amount of candidate patterns
     - obvious even for 2-sequences: $m \times m + \frac{m(m-1)}{2} \rightarrow \mathcal{O}(m^2)$
       (for itemsets it was just the second fraction, one third or so of candidates),
  2. requires a lot of database scans
     - one scan per sequence length,
     - the number of scans given by the max pattern length
       the length $\leq max(|s|, s \in S)$ (typically $>> m$),
       (max itemset length is m and thus m scans at most),
  3. search for long sequential patterns is difficult
     - the total amount of candidate patterns is exponential with the pattern length,
       (the same growth as for itemsets, however the problem $max(|s|, s \in S) >> m$).
- the disadvantages addressed by alternative methods in FreeSpan and PrefixSpan algorithms.

FreeSpan [Han, Pei, Yin, 2000], $s_{min} = 2$

- takes the **recursive** divide and conquer approach
  - decides along the decreasingly sorted list of frequent items,
    - $f\text{-}list = \langle (a:4), (b:4), (c:4), (d:3), (e:3), (f:3) \rangle$,
    - $(g:1)$ is not frequent,
  - the sequential patterns splits into disjoint groups
    - the patterns containing the most frequent item only,
    - the patterns containing the second most frequent item and no less frequent items, etc.
  - creates a projection sequence database (one per each group)
    - from sequences it removes all the items disregarded by the group,
    - it removes sequences missing item which must be in the pattern.
- subproblems have fewer items (beginning), and contain fewer sequences (end of $f\text{-}list$).

Id	Sequence	a-projection	b-projection	...	f-projection
10	$\langle a(abc)(ac)d(cf) \rangle$	$\langle aaa \rangle$	$\langle a(ab)a \rangle$	...	$\langle a(abc)(ac)d(cf) \rangle$
20	$\langle (ad)c(bc)(ae) \rangle$	$\langle aa \rangle$	$\langle aba \rangle$	...	
30	$\langle (ef)(ab)(df)cb \rangle$	$\langle a \rangle$	$\langle (ab)b \rangle$	...	$\langle (ef)(ab)(df)cb \rangle$
40	$\langle eg(af)cbc \rangle$	$\langle a \rangle$	$\langle ab \rangle$	...	$\langle e(af)cbc \rangle$

Pei, Han et al.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.

PrefixSpan [Pei, Han et al., 2001]

- a similar idea, more efficient than its predecessor FreeSpan,
- the projection based on **prefix** subsequence occurrence (in FreeSpan arbitrary occurrence)
  - allows for a more efficient database decomposition,
- $\beta = \langle s'_1, \ldots, s'_m \rangle$ is a prefix of $\alpha = \langle s_1, \ldots, s_n \rangle$ if:

  (1) $m \leq n$, (2) $\forall i \leq m-1 \; s'_i = s_i$, (3) $s'_m \subseteq s_m$,
  (4) $\forall$ items from $(s_m - s'_m) > \forall$ items from s'_m,
  - ex.: $\langle a \rangle$, $\langle aa \rangle$, $\langle a(ab) \rangle$ and $\langle a(abc) \rangle$ are the prefixes $\langle a(abc)(ac)d(cf) \rangle$,
  - ex.: $\langle ab \rangle$, $\langle a(bc) \rangle$ are not the prefixes $\langle a(abc)(ac)d(cf) \rangle$,
- informally: postfix is a prefix complement
  - ex.: prefix $\langle a \rangle$ has wrt $\langle a(abc)(ac)d(cf) \rangle$ the postfix $\langle (abc)(ac)d(cf) \rangle$,
  - ex.: prefix $\langle aa \rangle$ has wrt $\langle a(abc)(ac)d(cf) \rangle$ the postfix $\langle (_bc)(ac)d(cf) \rangle$,
- $\alpha' \sqsubseteq \alpha$ is a projection α wrt the prefix $\beta \sqsubseteq \alpha$, if:

  (1) α' has a prefix β,
  (2) there is no α'' that is (i) a supersequence of α' (i.e. $\alpha' \sqsubset \alpha''$),
  (ii) subsequence of α and (iii) has the prefix β,
  - ex.: the projection of $\langle a(abc)(ac)d(cf) \rangle$ wrt the prefix $\langle (ac)d \rangle$ is $\langle (ac)d(cf) \rangle$.

PrefixSpan – algorithm, example ($s_{min} = 2$)

- PrefixSpan: the input is S and s_{min}
  1. $i = 1$, the init projection prefix database $S|_{\alpha_0} = S|_{\emptyset} = S$,
  2. repeat for all the projection prefix databases $S|_{\alpha_{i-1}}$
     (a) find frequent i-patterns (sufficient support in $\alpha_{i-1} \cdot S|_{\alpha_{i-1}}$),
     (b) until the set of i-patterns is not empty
        i. split the state space having the i-patterns (α_i) as prefixes
           a projection database set originates $S|_{\alpha_i} = (\alpha_{i-1} \cdot S|_{\alpha_{i-1}})|_{\alpha_i}$,
        ii. i=i+1 and go to the step (2).

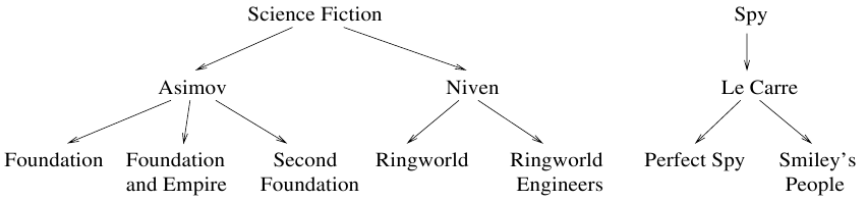
Id	Sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Prefix	Projection database (postfixes) or patterns
$\langle a\rangle$	$\langle (abc)(ac)d(cf)\rangle$, $\langle (_d)c(bc)(ae)\rangle$, $\langle (_b)(df)cb\rangle$, $\langle (_f)cbc\rangle$
	2-patterns: $\langle aa\rangle$:2, $\langle ab\rangle$:4, $\langle ac\rangle$:4, $\langle ad\rangle$:2, $\langle af\rangle$:2, $\langle (ab)\rangle$:2
$\langle b\rangle$	$\langle (_c)(ac)d(cf)\rangle$, $\langle (_c)(ae)\rangle$, $\langle (df)cb\rangle$, $\langle c\rangle$
	2-patterns: $\langle ba\rangle$:2, $\langle bc\rangle$:3, $\langle (bc)\rangle$:2, $\langle bd\rangle$:2, $\langle bf\rangle$:2
$\langle aa\rangle$	$\langle (_bc)(ac)d(cf)\rangle$, $\langle (_e)\rangle$
	STOP (no 3-patterns)
$\langle (ab)\rangle$	$\langle (_c)(ac)d(cf)\rangle$, $\langle (df)cb\rangle$
	3-patterns: $\langle (ab)c\rangle$:2, $\langle (ab)d\rangle$:2, $\langle (ab)f\rangle$:2

Frequent sequence search – problem generalization

- subsequence definition (slide 13) is still not general enough for certain practical problems,
- ex.: book stores

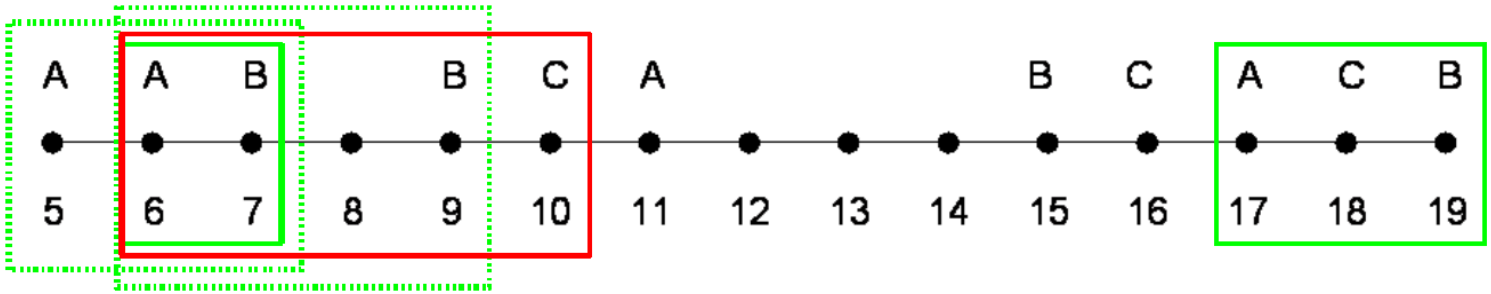
ID	Time	Items
C1	1	Ringworld
C1	2	Foundation
C1	15	Ringworld Engineers, Second Foundation
C2	1	Foundation, Ringworld
C2	20	Foundation and Empire
C2	50	Ringworld Engineers



- the GSP algorithm contributed in several additional aspects
  1. introduction of time constraints
     - adjoining sequence elements must not be too far (MaxGap) nor close (MinGap),
  2. extended transaction definition
     - the items with near time stamps included in the same transaction,
     - sliding window, the parameter WinSize gives its size,
  3. a taxonomy of items
     - a directed acyclic graph defines a concept hierarchy built upon items.

Episodal rules

- association rule analogy,
- predict the further development of sequence with the aid of patterns,
- ex.: the only sequence, the only item per position, MaxGap=3



- S is a sequence, $\alpha = \langle AB \rangle$ and $\beta = \langle ABC \rangle$ its subsequences,
- α is a prefix of β,
- **episodal rule** is a probabilistic implication
  - $\alpha \Rightarrow postfix(\beta, \alpha)$, i.e. $\langle AB \rangle \Rightarrow \langle C \rangle$
- likewise for association rules, min support parameter is usually completed by min confidence
  - $conf(\alpha \Rightarrow postfix(\beta, \alpha)) = \frac{s(\beta,S,3)}{s(\alpha,S,3)} = \frac{1}{2}$.

Frequent subsequences – summary, categorization

- Different problem types distinguished according to the sequence types
  - single vs more sequences in a database
    - asks for different support definition,
  - directed vs undirected sequences
    - asks for different treatment of canonical representation,
  - one item vs more items per single sequence position
    - influences complexity of the solution,
  - constraints may be needed
    - e.g. the window size for transaction definition,
    - MinGap and MaxGap for sequence definition,
    - constraints extend practical applicability, slightly increase complexity,
  - a taxonomy of items may exist
    - similar to constraints – extends applicability, but it may increase complexity,
  - items or (labeled) intervals
    - an interval: $I = (start, end, label)$.
- the next lecture: from sequences towards structural patterns (trees/graphs).

Recommended reading, lecture resources

:: Reading

- Agrawal, Srikant: **Mining Sequential Patterns.**
- Agrawal, Srikant: **MSPs: Generalizations and Performance.**
  - from APRIORI towards its sequential versions AprioriAll and GSP,
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2818&rep=rep1&type=pdf,
- Pei, Han et al.: **PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.**
  - FreeSpan (idea) and PrefixSpan algorithms, efficiency comparison with GSP,
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.7211,
- Mannila et al.: **Discovery of Frequent Episodes in Event Sequences.**
  - episodal rules,
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.3594,
- Borgelt: **Frequent Pattern Mining.**
  - undirected sequences,
  - http://www.borgelt.net/teach/fpm/slides.html.