Non-linearity in Davenport-Schinzel Sequences

Seth Pettie

University of Michigan

Isomorphism and Subsequences

- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**

Isomorphism and Subsequences

- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**

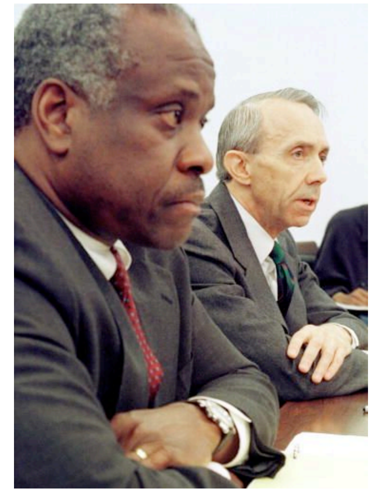
Isomorphism and Subsequences

- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**
  - **THOMAS** is isomorphic to **SOUTER**

Isomorphism and Subsequences

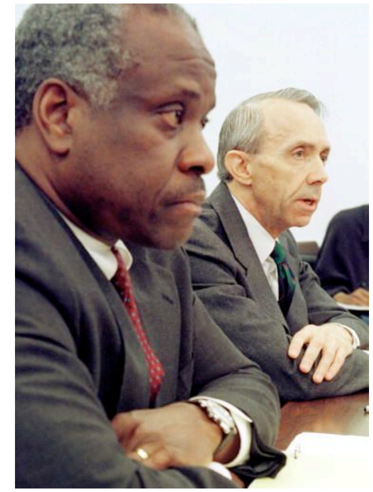
- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**
  - **THOMAS** is isomorphic to **SOUTER**
  - **CIA,NSA,DOD** is *not* isomorphic to **NSF,EPA,NIH**

Isomorphism and Subsequences



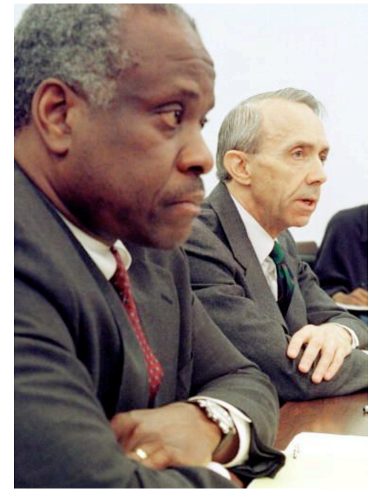
- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**
  - **THOMAS** is isomorphic to **SOUTER**
  - **CIA,NSA,DOD** is *not* isomorphic to **NSF,EPA,NIH**
- Happiness via Subsequences
  - **WITH_WHOM_WOULD_I_RATHER_HAVE_A_BEER?**

Isomorphism and Subsequences



- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**
  - **THOMAS** is isomorphic to **SOUTER**
  - **CIA,NSA,DOD** is *not* isomorphic to **NSF,EPA,NIH**
- Happiness via Subsequences
  - **WITH_WHOM_WOULD_I_RATHER_HAVE_A_BEER?**
  - **TH_WHO LD_ R_ VE ?**

Isomorphism and Subsequences



- Political Isomorphism
  - **BUSH** is isomorphic to **GORE**
  - **C** is isomorphic to **A**
  - **THOMAS** is isomorphic to **SOUTER**
  - **CIA,NSA,DOD** is *not* isomorphic to **NSF,EPA,NIH**
- Happiness via Subsequences
  - **WITH_WHOM_WOULD_I_RATHER_HAVE_A_BEER?**
  - **TH_WHO LD_ R_ VE ?**
  - **TARJAN FOR PR EZ ?**

Seth Pettie

Definitions

- $x \subset y$: x is ***isomorphic*** to a ***subsequence*** of y

- $Ex(\sigma,n) = \max |S|$:

  $S \in \{1,\ldots,n\}^*$

  $\sigma \not\subset S$

  S is $|\sigma|$-regular (technical condition)

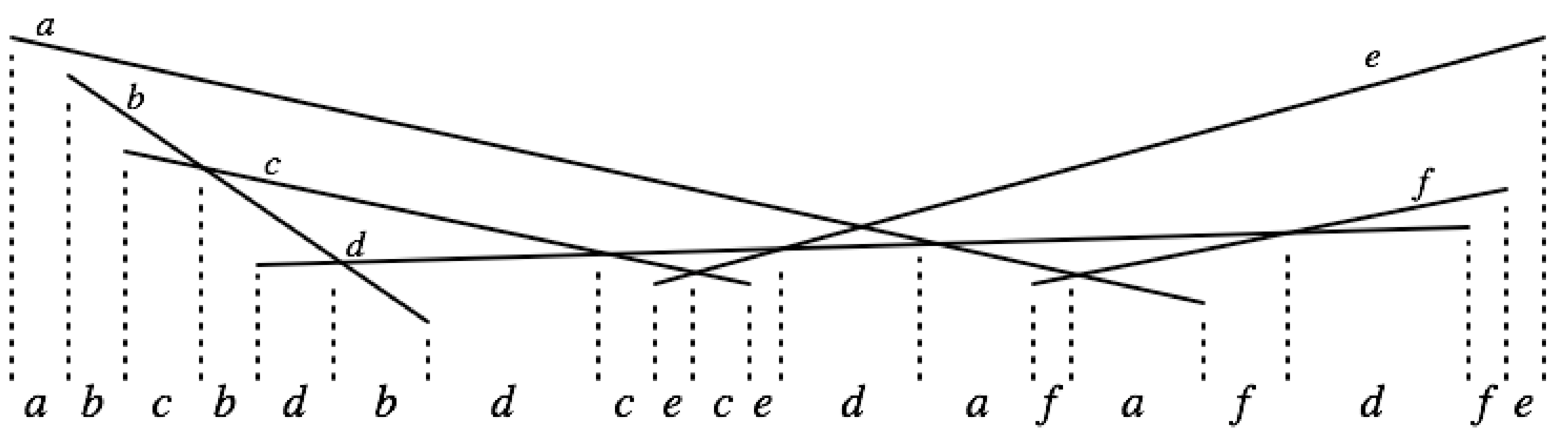
- How fast does $Ex(\sigma,n)$ grow as a function of n?

Original application: lower envelopes

(1) Give each object (line segment, quadratic, etc.) a symbol

(2) Map the lower envelope to a sequence |S|

(3) Show |S| ≤ Ex(σ,n) for some ***forbidden subseq.*** σ



this sequence does not contain **ababa**

Original motivation: lower envelopes

(1) Give each object (line segment, quadratic, etc.) a symbol

(2) Map the lower envelope to a sequence |S|

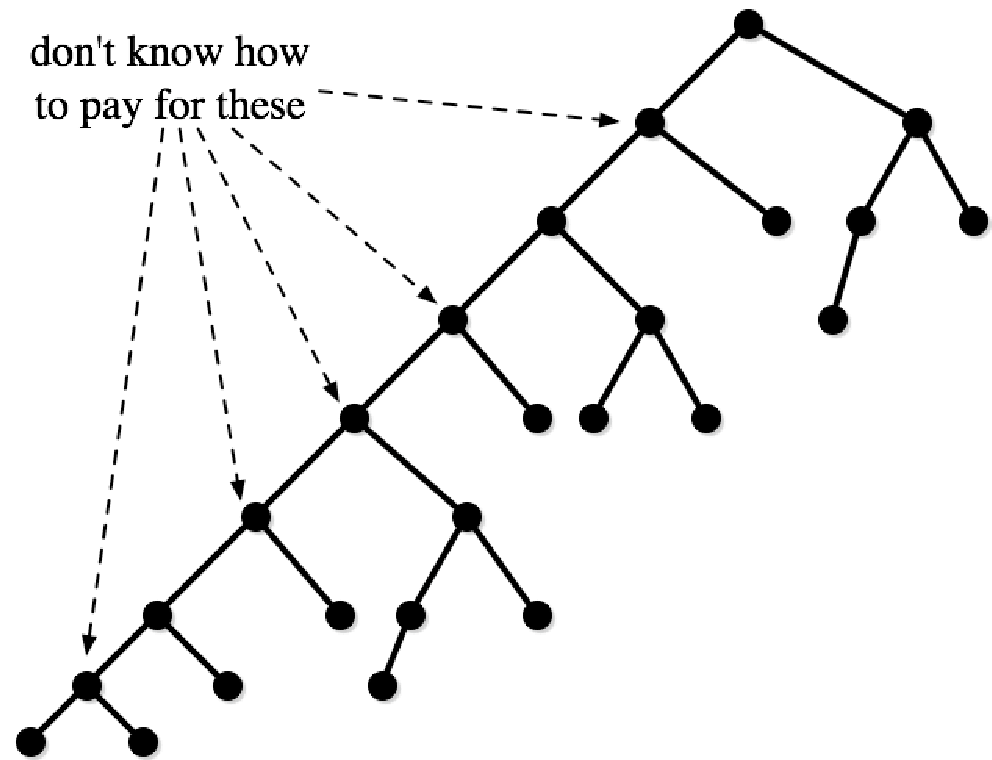
(3) Show $|S| \le Ex(\sigma,n)$ for some ***forbidden subseq.*** σ

standard case: σ = ababab…a ← length k+2

"order k Davenport-Schinzel sequence"

Splay trees and Davenport-Schinzel sequences

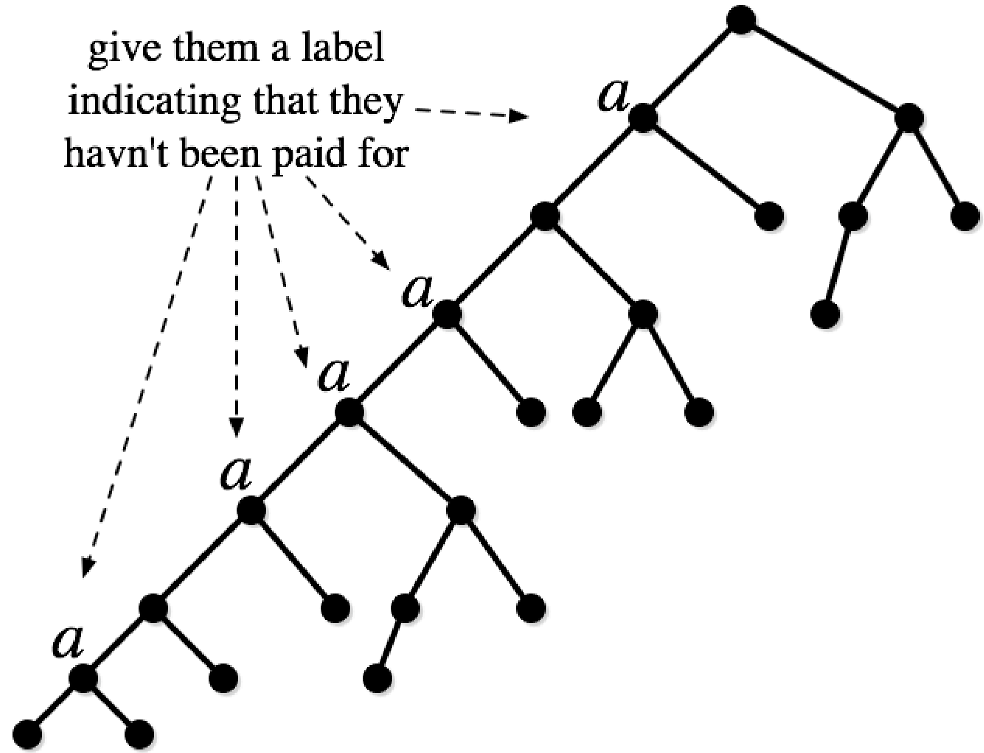
- Amortized analysis: Normally pay for time consuming ops with a reduction in potential



Splay trees and Davenport-Schinzel sequences

- New kind of amortized analysis:
- ***Label nodes*** that cannot be paid for by other means
- ***Transcribe the labels*** as a sequence S:

  $|S| \leq Ex(\sigma,n)$



- In [SODA'08] σ = abaabba or ababab a

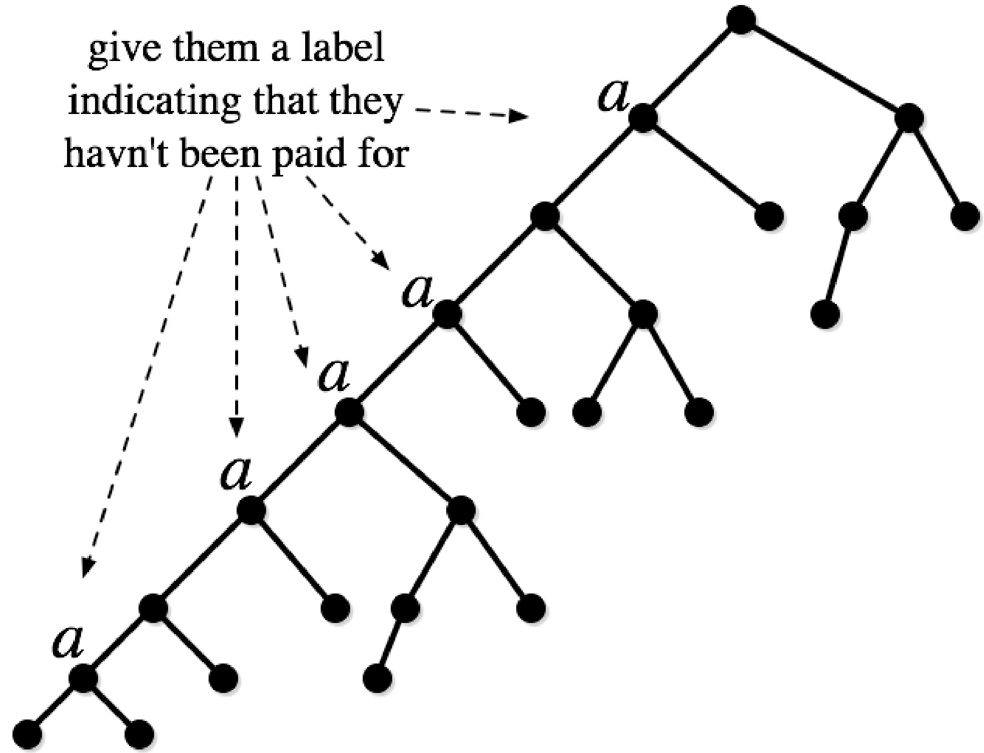
  Thm. n deque operations take $O(n\alpha^*(n))$ time

Splay trees and Davenport-Schinzel sequences

- New kind of amortized analysis:
- ***Label nodes*** that cannot be paid for by other means
- ***Transcribe the labels*** as a sequence S:

  $|S| \leq Ex(\sigma,n)$

  …



A much better way to end the proof:

… where $Ex(\sigma,n) = O(n)$

Standard Davenport-Schinzel seqs.

- $\alpha = \alpha(n)$ *α is the inverse-Ackermann function*

trivial →

Ex(**aba**, n)	n
Ex(**abab**, n)	2n-1

Standard Davenport-Schinzel seqs.

- $\alpha = \alpha(n)$ *α is the inverse-Ackermann function*

trivial	Ex(**aba**, n)	n
trivial	Ex(**abab**, n)	2n-1
Hart-Sharir	Ex(**ababa**, n)	$\Theta(n\alpha)$

Standard Davenport-Schinzel seqs.

- $\alpha = \alpha(n)$ *α is the inverse-Ackermann function*

trivial →	Ex(**aba**, n)	n
trivial →	Ex(**abab**, n)	2n-1
Hart-Sharir →	Ex(**ababa**, n)	$\Theta(n\alpha)$
Agarwal-Sharir-Shor	Ex(**ababab**, n)	$\Theta(n2^{\alpha})$

Standard Davenport-Schinzel seqs.

- $\alpha = \alpha(n)$ *α is the inverse-Ackermann function*

trivial	Ex(**aba**, n)	n
trivial	Ex(**abab**, n)	2n-1
Hart-Sharir	Ex(**ababa**, n)	$\Theta(n\alpha)$
Agarwal-Sharir-Shor	Ex(**ababab**, n)	$\Theta(n2^{\alpha})$
	Ex(**abababa**, n)	$n \exp(O(\alpha \log \alpha))$
	Ex(**abababab**, n)	$n \exp(\Theta(\alpha^2))$
	Ex(**ababababa**, n)	$n \exp(O(\alpha^2 \log \alpha))$
	Ex(**ababababab**, n)	$n \exp(\Theta(\alpha^3))$

Standard Davenport-Schinzel seqs.

- $\alpha = \alpha(n)$ *α is the inverse-Ackermann function*

trivial	Ex(**aba**, n)	n
trivial	Ex(**abab**, n)	2n-1
Hart-Sharir	Ex(**ababa**, n)	$\Theta(n\alpha)$
Agarwal-Sharir-Shor	Ex(**ababab**, n)	$\Theta(n2^{\alpha})$
	Ex(**abababa**, n)	n $\exp(O(\alpha \log \alpha))$
	Ex(**abababab**, n)	n $\exp(\Theta(\alpha^2))$
	Ex(**ababababa**, n)	n $\exp(O(\alpha^2 \log \alpha))$
	Ex(**ababababab**, n)	n $\exp(\Theta(\alpha^3))$
Klazar	Ex($\boldsymbol{\sigma}$, n)	n $\exp(O(\alpha^{\lvert\sigma\rvert}))$

Two-Letter Forbidden Subsequences

[Adamec-Klazar-Valtr]

$$\mathrm{Ex}(abbaab,n) = O(n)$$

The Two-Letter Theorem:

For any $\sigma \in \{a,b\}^*$

$\mathrm{Ex}(\sigma,n) = \omega(n)$ *if and only if* $ababa \subset \sigma$

(i.e., there is only one “cause” of superlinearity over two symbols)

The *Three-Letter Theorem*

- [Klazar-Valtr]

For $\sigma \in \{a,b,c\}^*$

$Ex(\sigma,n) = O(n)$

unless…

$ababa \subset \sigma$ or ← *non-linear*

$abcacbc \subset \sigma$ or ← *status still open*

$abcbcac \subset \sigma$ ← *status still open*

or their reversals

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i, n) = O(n)$

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

uw and **v** have disjoint alphabets

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

uw and **v** have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$Ex(\mathbf{uvw},n) = O(n)$

uw and **v** have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$

aaaa

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

uw and v have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

aa**bb**aa**bbb**

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

uw and **v** have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

aabbaab**ccccc**bb**cc**

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

uw and **v** have disjoint alphabets

aabbaabc**ddd**ccccbbcc**dd**

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(uw,n) = O(n)$ and $Ex(v,n) = O(n)$

$$Ex(uvw,n) = O(n)$$

uw and v have disjoint alphabets

(3) If $Ex(uawa,n) = O(n)$

$$Ex(uab^iwab^i) = O(n)$$

aabbaabcdddccccbbccdd

ee

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $\mathrm{Ex}(a^i,n) = O(n)$

(2) If $\mathrm{Ex}(\mathbf{uw},n) = O(n)$ and $\mathrm{Ex}(\mathbf{v},n) = O(n)$

$$\mathrm{Ex}(\mathbf{uvw},n) = O(n)$$

uw and **v** have disjoint alphabets

(3) If $\mathrm{Ex}(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$\mathrm{Ex}(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

aabbaabcdddccccbbccdd

effef

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$Ex(\mathbf{uvw},n) = O(n)$

uw and **v** have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$

aabbaabcdddccccbbccdd

ef**g**fef**g**

Seth Pettie

Recipe for linear forbidden sequences

[Klazar-Valtr]

(1) $Ex(a^i,n) = O(n)$

(2) If $Ex(\mathbf{uw},n) = O(n)$ and $Ex(\mathbf{v},n) = O(n)$

$$Ex(\mathbf{uvw},n) = O(n)$$

uw and **v** have disjoint alphabets

(3) If $Ex(\mathbf{u}a\mathbf{w}a,n) = O(n)$

$$Ex(\mathbf{u}ab^i\mathbf{w}ab^i) = O(n)$$

aabbaabcdddc**efgfefg**cccbbccdd

efgfefg

Seth Pettie

More than one cause of non-linearity

- [Klazar]
  - σ is a sequence without repetitions
  - (x,y) is in $G(\sigma)$ iff $xyyx \subset \sigma$ or $yxyx \subset \sigma$

  - If $G(\sigma)$ is ***strongly connected*** then

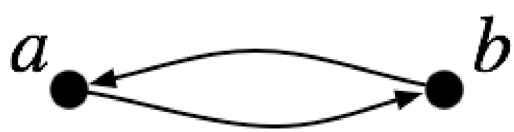
$$Ex(\sigma,n) = \Omega(n\alpha(n))$$

More than one cause of non-linearity

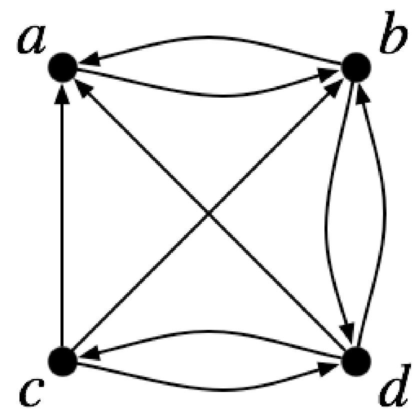
- [Klazar]
  - σ is a sequence without repetitions
  - (x,y) is in G(σ) iff xyyx ⊂ σ or yxyx ⊂ σ

  - If G(σ) is ***strongly connected*** then

$$Ex(\sigma,n) = \Omega(n\alpha(n))$$



G(ababa)



G(abcbadadbcd)

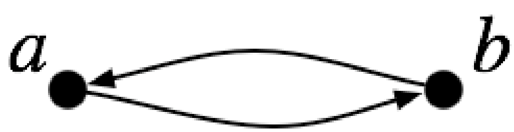
only two examples known

Seth Pettie

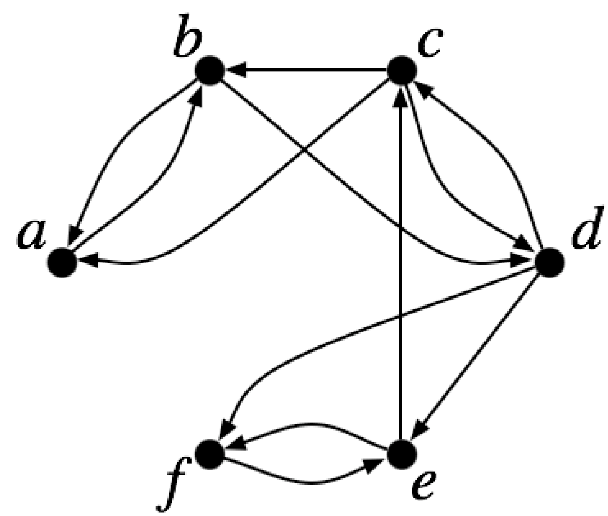
Another cause of non-linearity

- [Klazar]
  - σ is a sequence without repetitions
  - (x,y) is in G′(σ) iff xyyx ⊂ σ or ~~yxyx ⊂ σ~~
  - If G′(σ) is ***strongly connected*** then

$$\mathrm{Ex}(\sigma,n) = \sout{\Omega(n\alpha(n))} \quad \Omega(n2^{\alpha(n)})$$



G′(ababab)



G′(abcbadadbecfcfedef)

only two examples known

Seth Pettie

- Defn. Φ = minimal non-linear forbidden seqs.
- What we know about Φ:
  - ababa $\in \Phi$
  - $|\Phi| \geq 2$ (the other a subseq of abcbadadbcd)

- Defn. Φ = minimal non-linear forbidden seqs.
- What we know about Φ:
  - ababa $\in \Phi$
  - $|\Phi| \geq 2$ (the other a subseq of abcbadadbcd)
- ***Q***: Is $|\Phi|$ infinite?
- ***A***: Still Open. But we have a candidate!

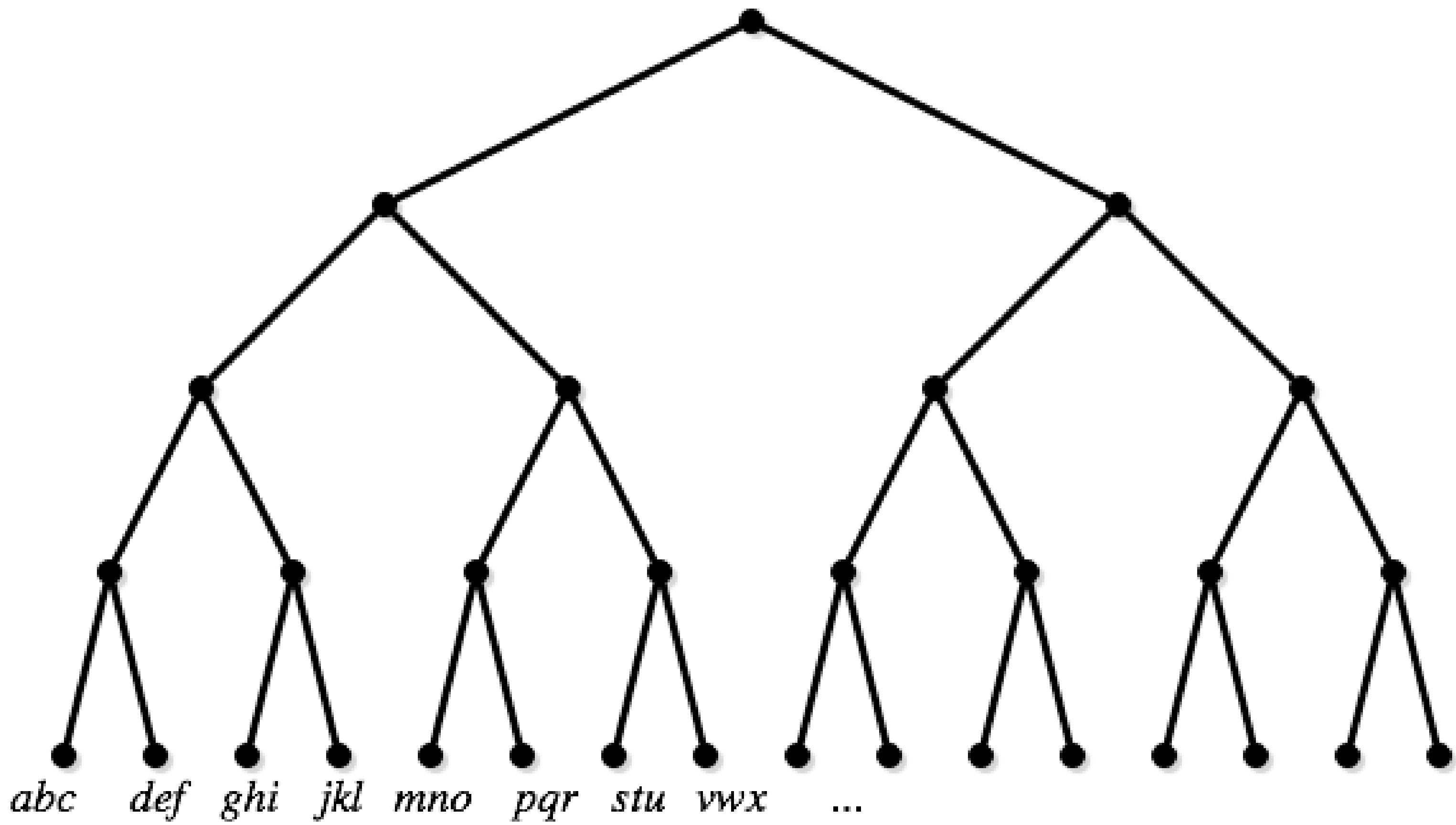
- Defn. Φ = minimal non-linear forbidden seqs.

- What we know about Φ:
  - ababa $\in \Phi$
  - $|\Phi| \geq 2$ (the other a subseq of abcbadadbcd)

- ***Q***: Is $|\Phi|$ infinite?
- ***A***: Still Open. But we have a candidate!
- ***Q***: How big is it Φ?
- ***A***: New result: $|\Phi| \geq 5$

Constructing Sequences

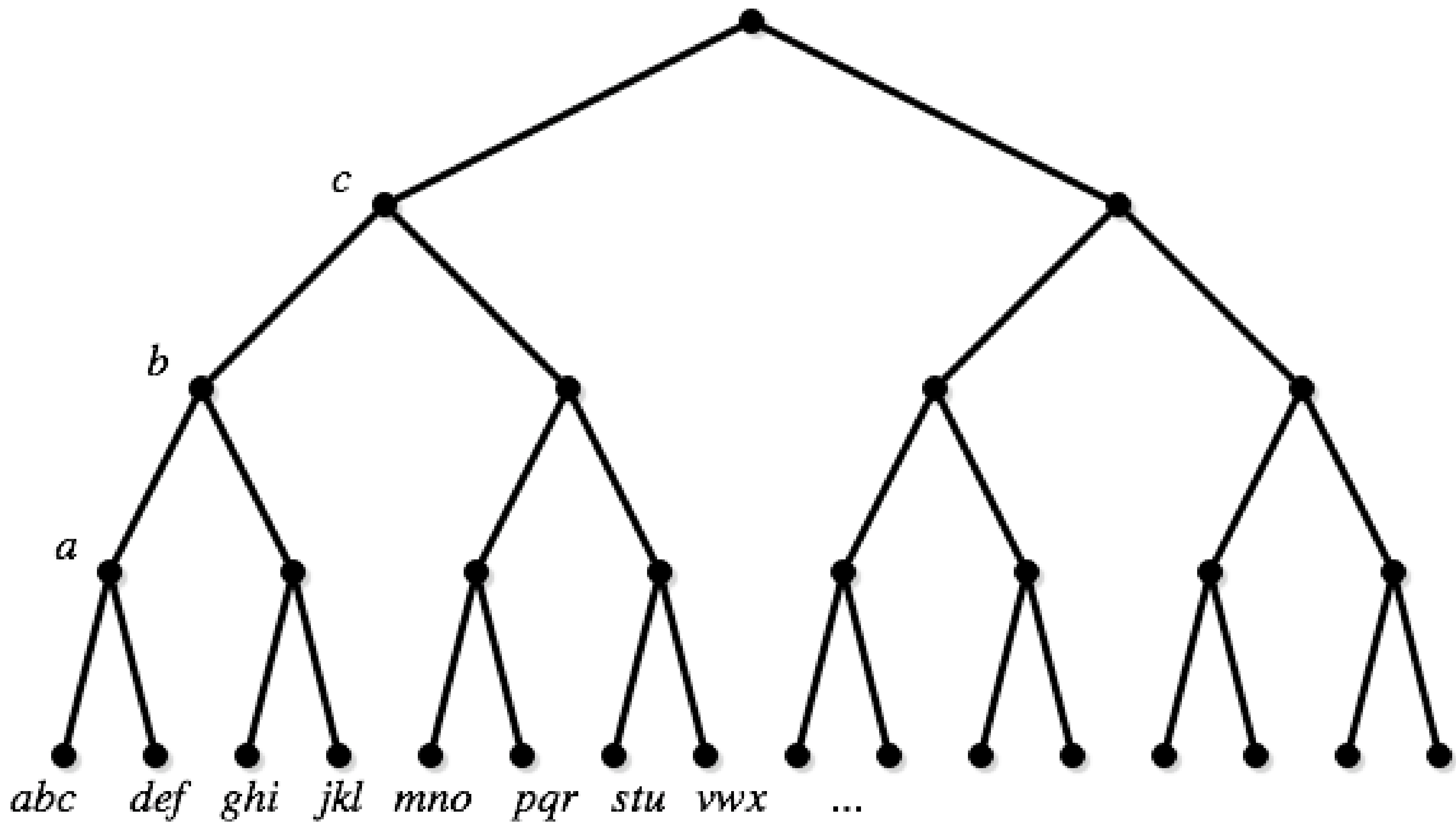
- T(1,j) : a binary tree with height j+1
  j distinct letters at each leaf



Seth Pettie

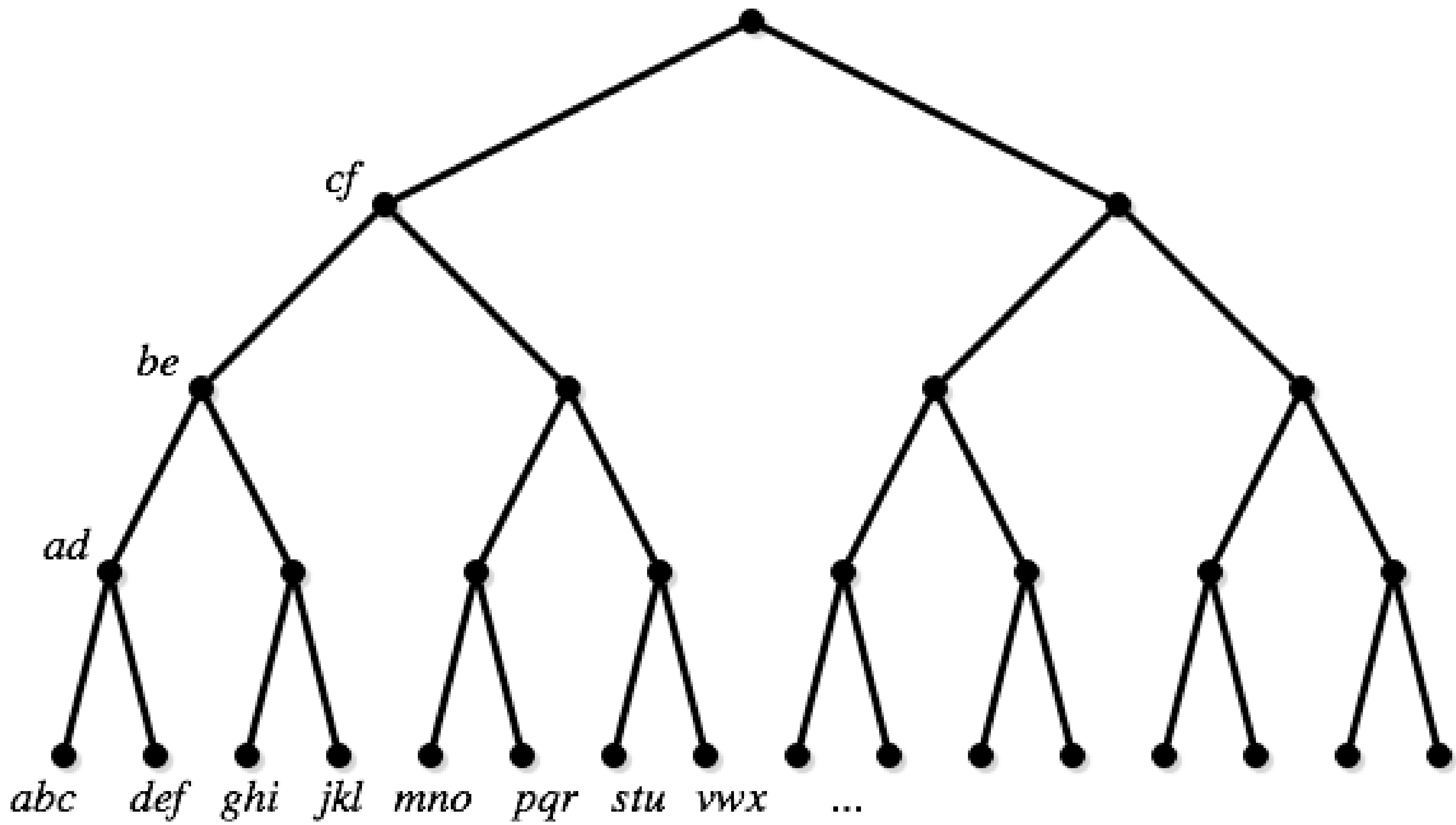
Constructing Sequences

- T(1,j) : a bin. tree w/height j+1,   j letters at each leaf
- i^{th} letter at a leaf added to label of i^{th} ancestor



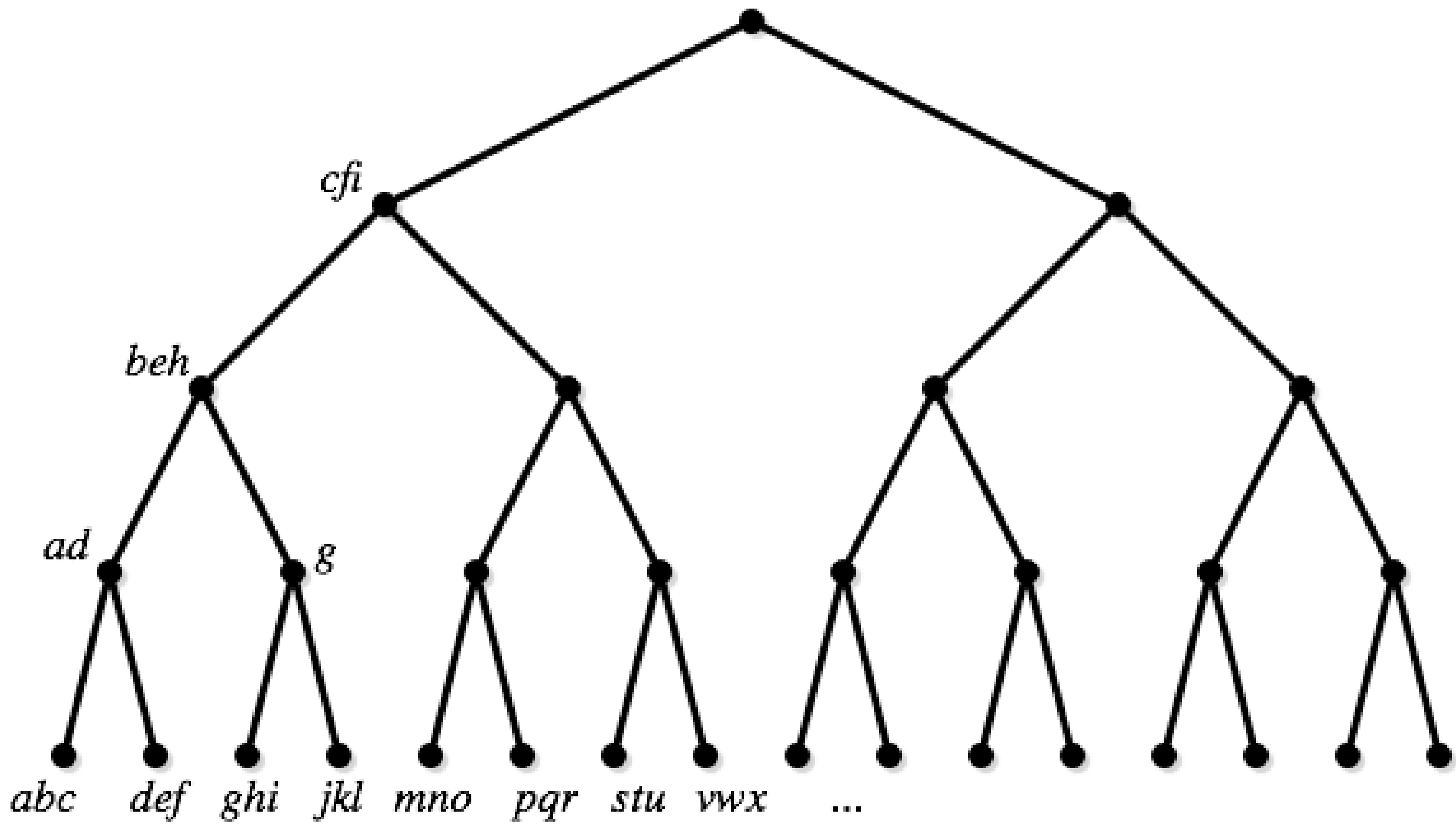
Constructing Sequences

- T(1,j) : a bin. tree w/height j+1,   j letters at each leaf
- ith letter at a leaf added to label of ith ancestor



Constructing Sequences

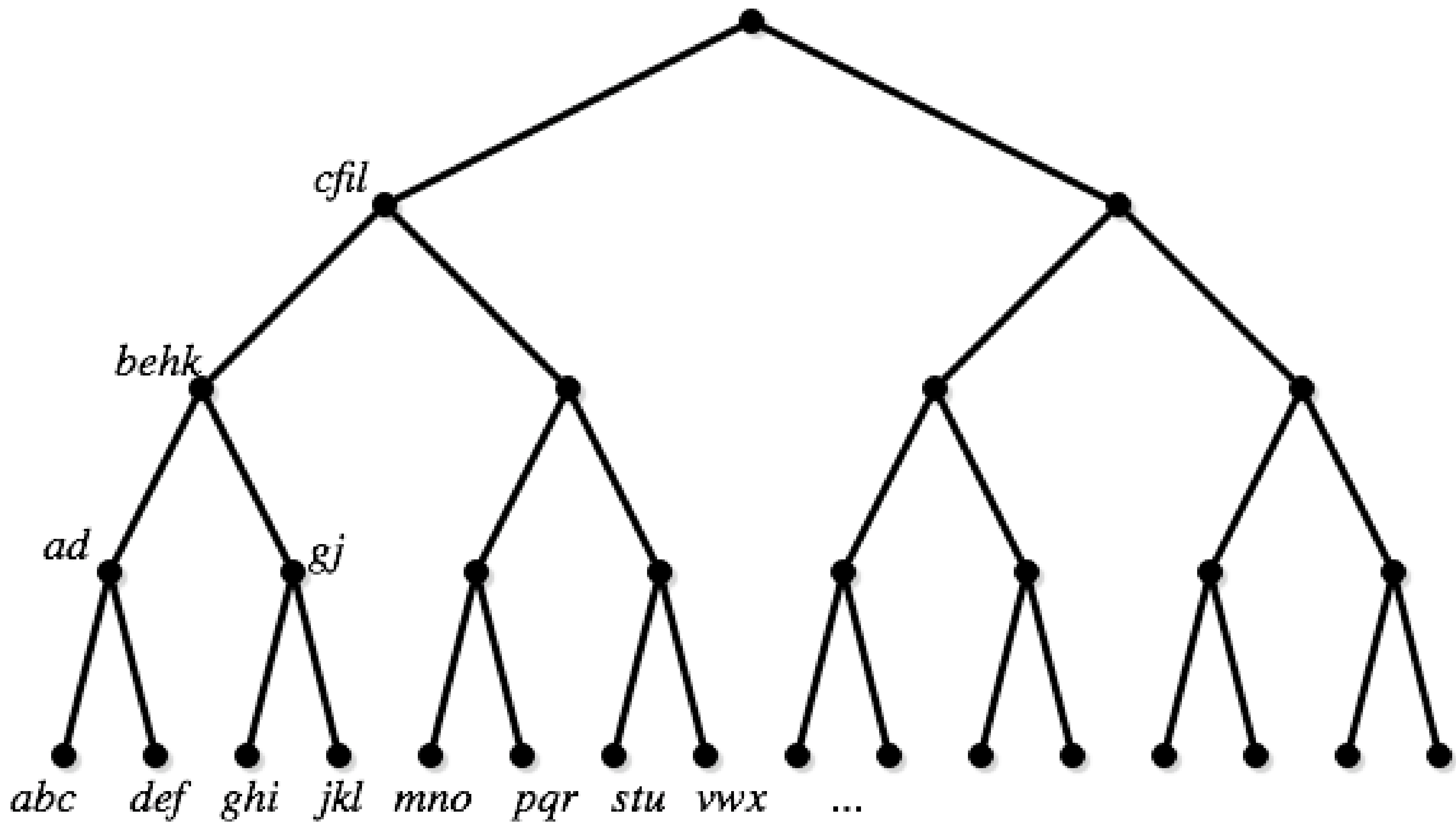
- T(1,j) : a bin. tree w/height j+1,   j letters at each leaf
- i^{th} letter at a leaf added to label of i^{th} ancestor



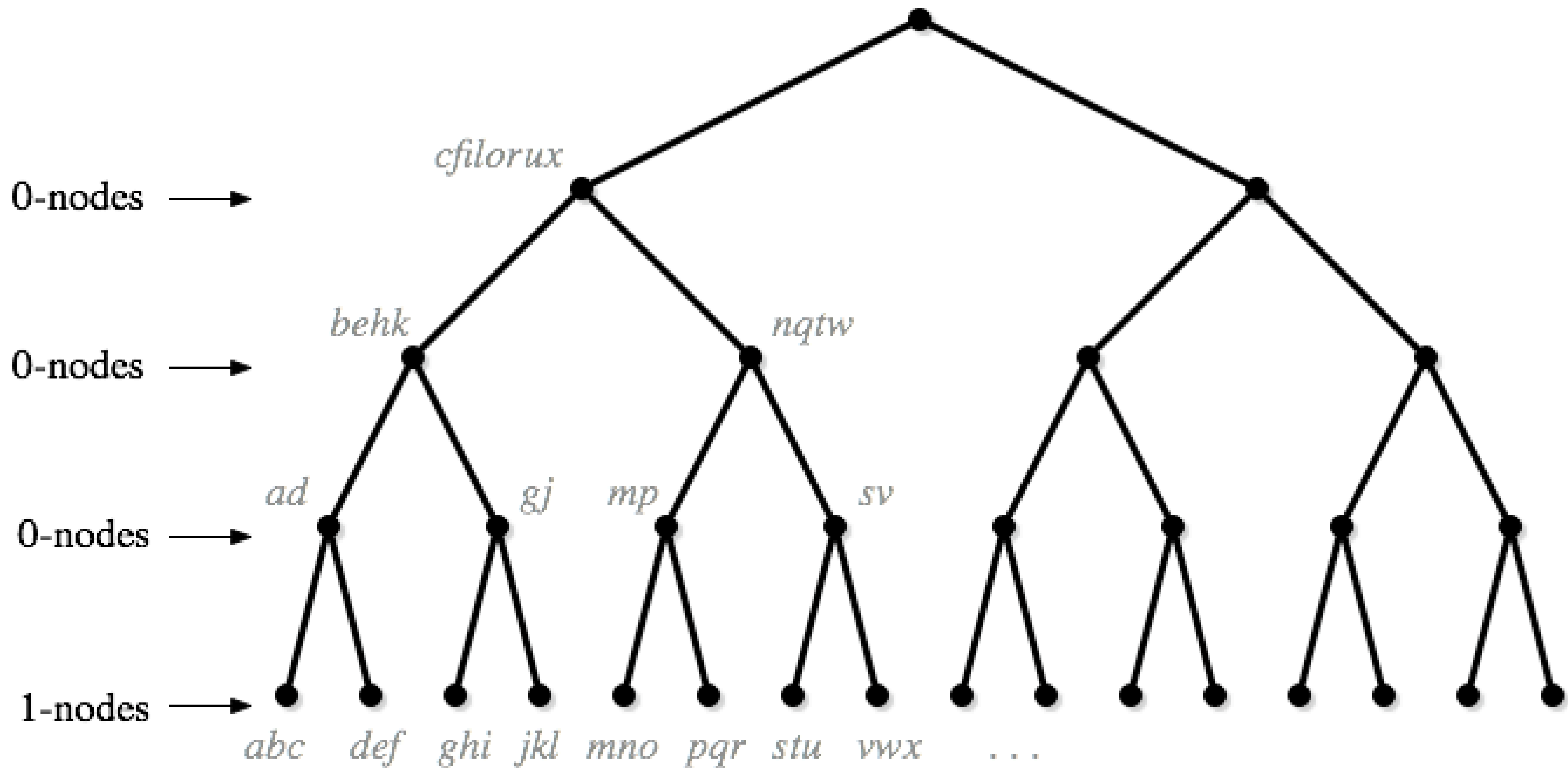
Seth Pettie

Constructing Sequences

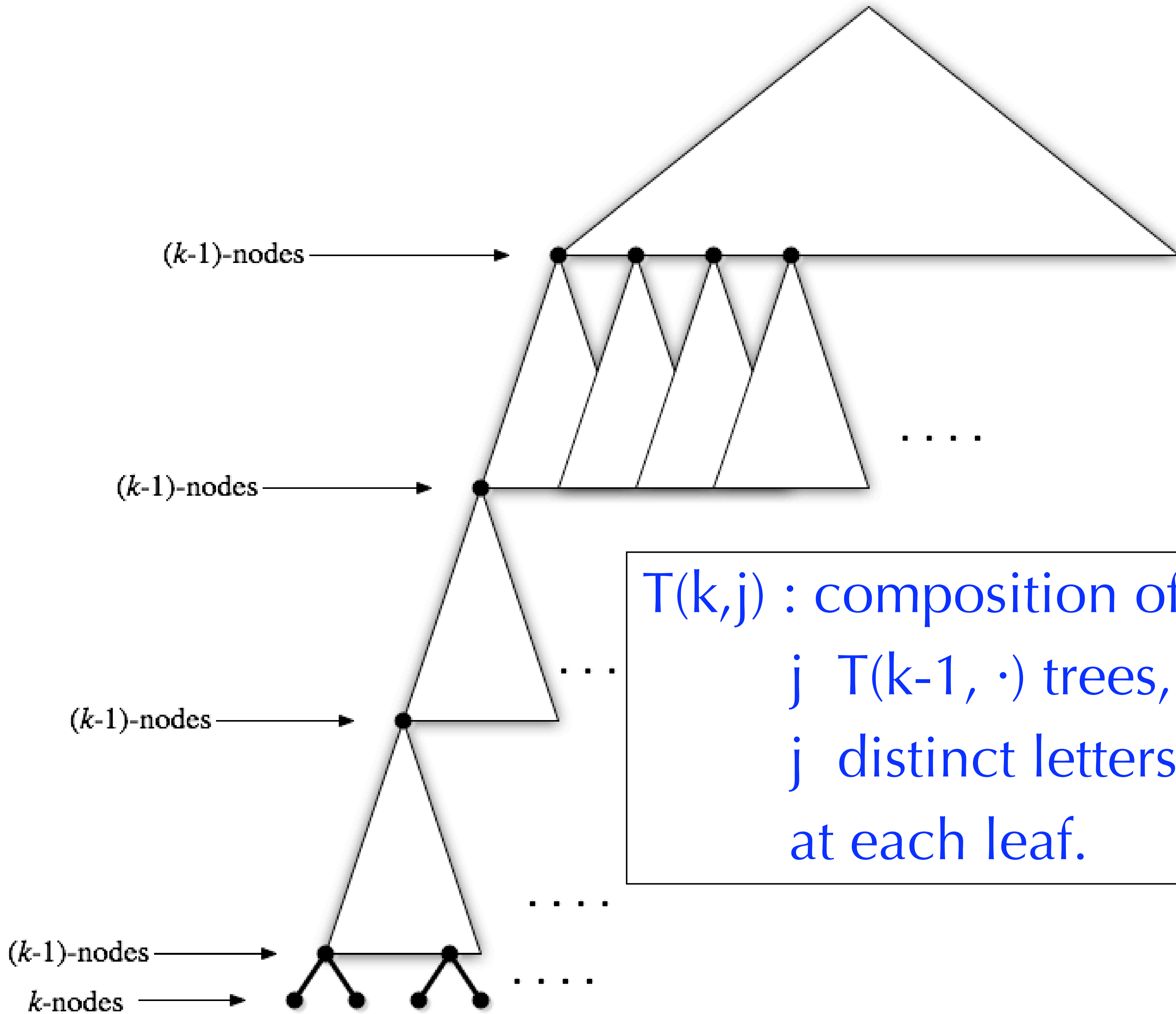
- T(1,j) : a bin. tree w/height j+1,   j letters at each leaf
- i^{th} letter at a leaf added to label of i^{th} ancestor



Constructing Sequences



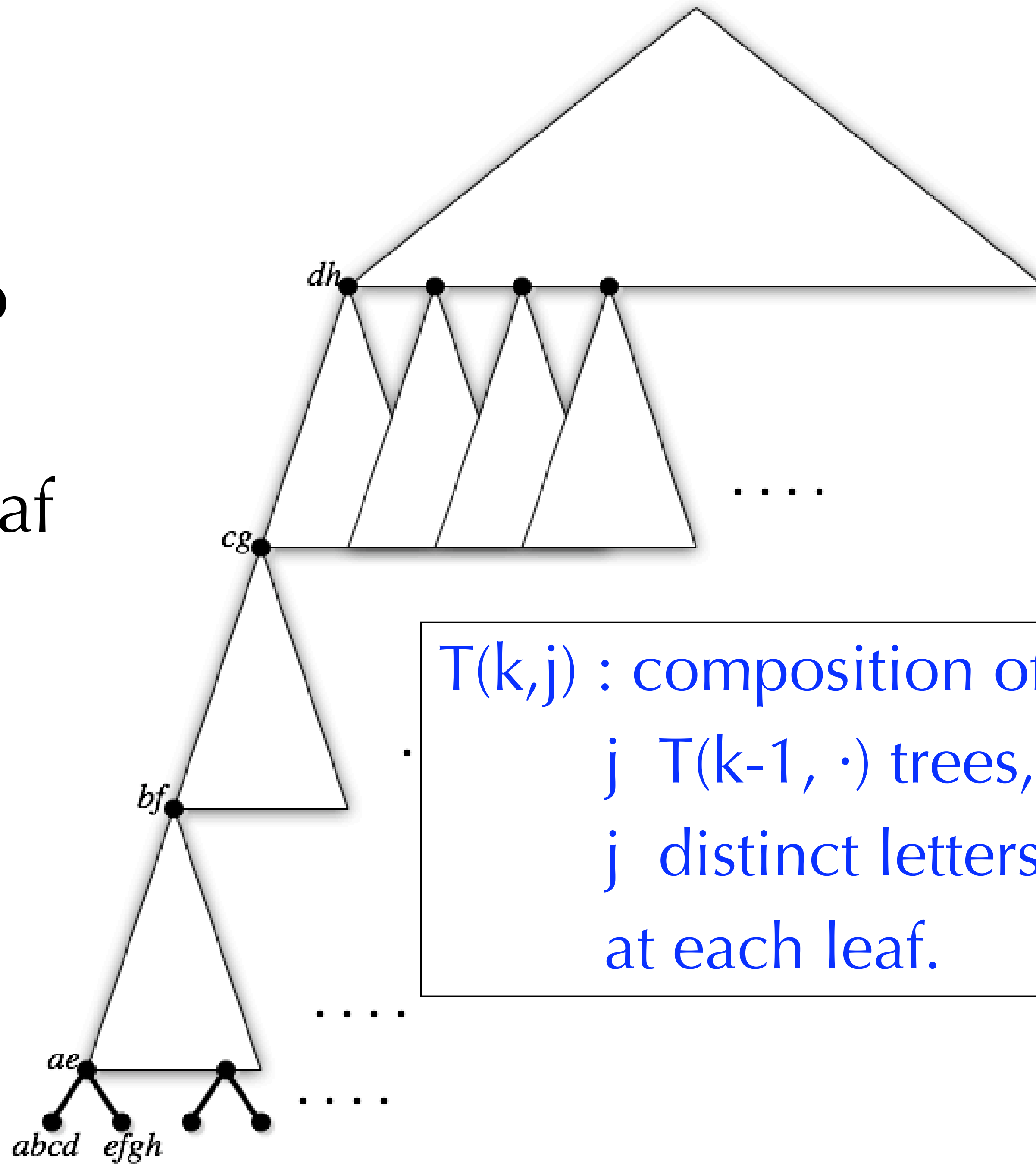
Seth Pettie



T(k,j) : composition of
j T(k-1, ·) trees,
j distinct letters
at each leaf.

Seth Pettie

the ith letter at a leaf is assigned to the ith (k-1)-node ancestor of the leaf

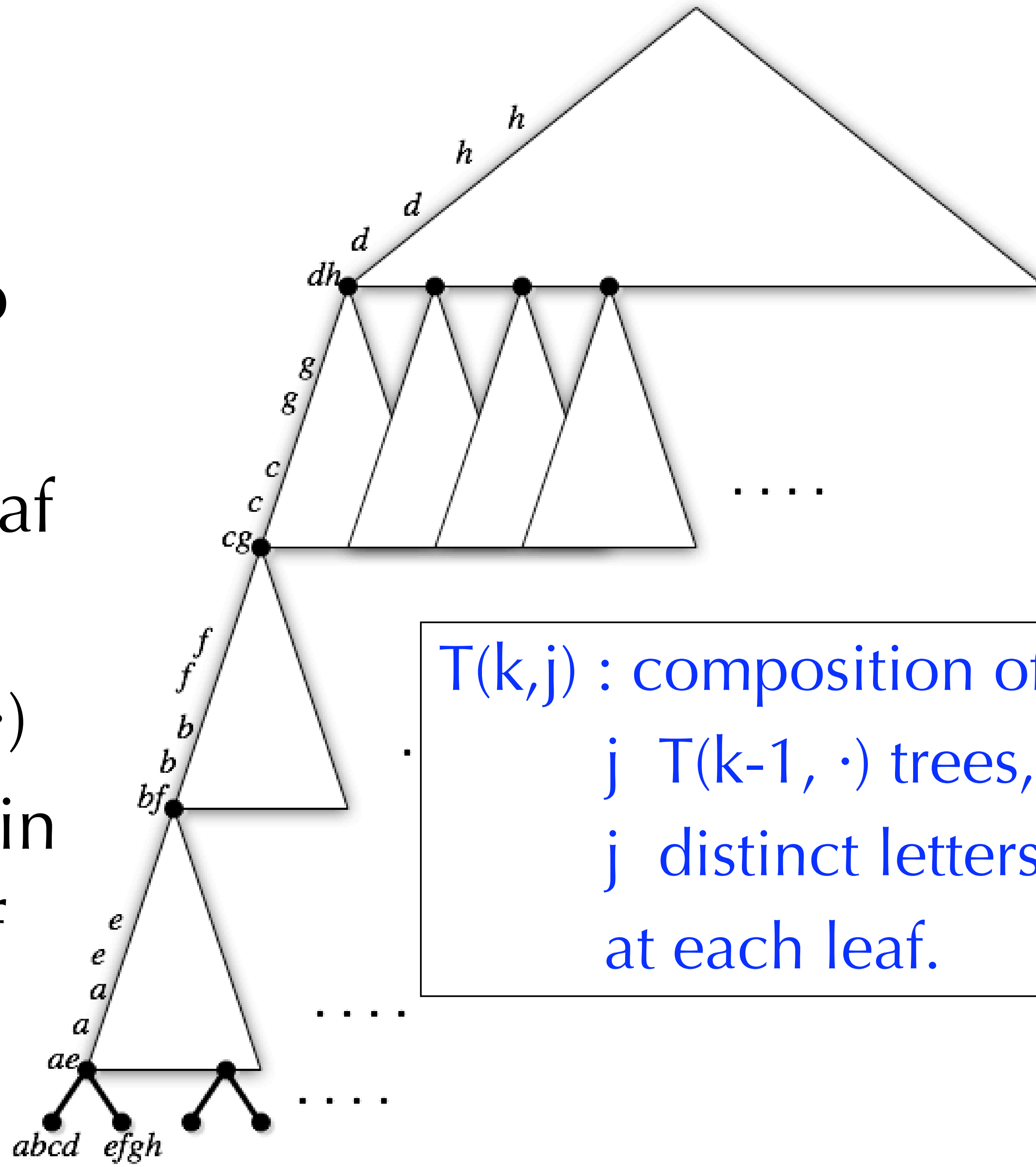


T(k,j) : composition of j T(k-1, ·) trees, j distinct letters at each leaf.

Seth Pettie

the i^{th} letter at a leaf is assigned to the i^{th} (k-1)-node ancestor of the leaf

…and the T(k-1, ·) trees are defined in terms of their leaf labels…



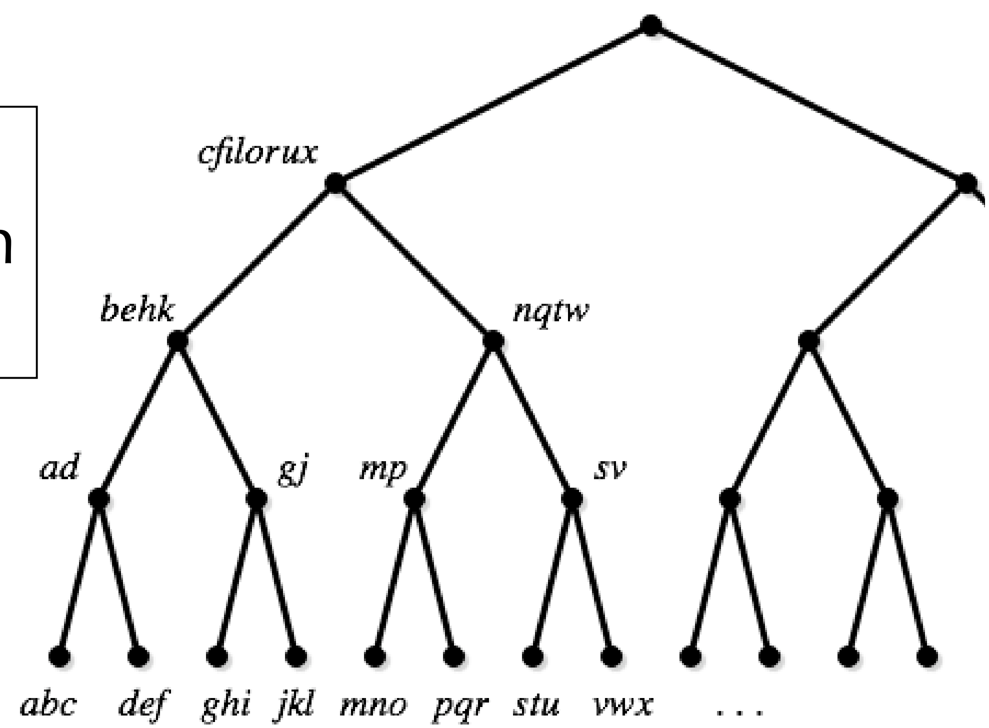
T(k,j) : composition of j T(k-1, ·) trees, j distinct letters at each leaf.

Constructing Sequences

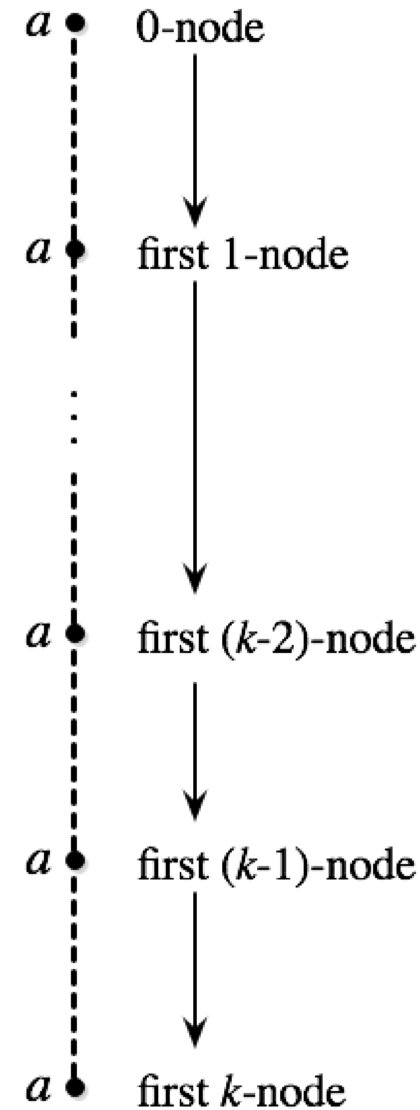
- $v_1, v_2, \ldots, v_n$: nodes listed in ***postorder***
- L(v) : the label of v in ***reverse order***
- The final sequence: $\Sigma = L(v_1), L(v_2), \ldots, L(v_n)$

The sequence for T(1,4) :

cba fed da ihg lkj jg kheb onm rqp pm uts xwv vs wtqn xurolifx …

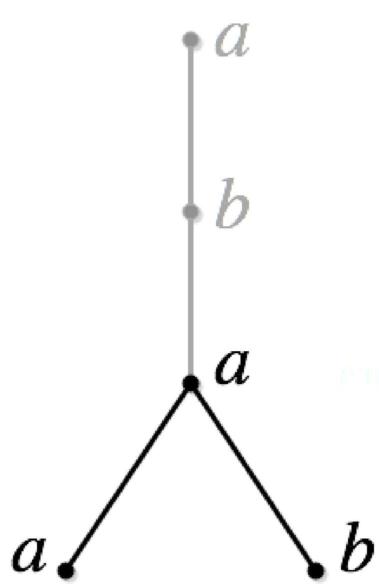


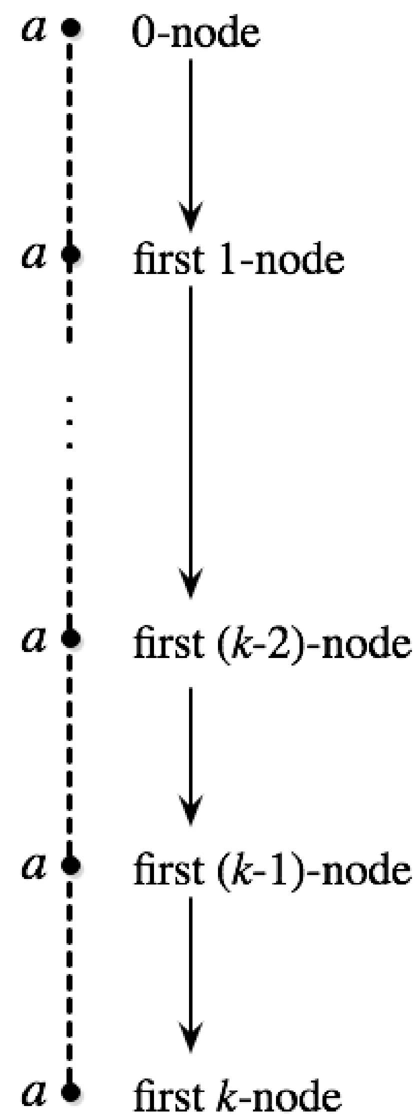
Seth Pettie


a
0-node
a
first 1-node
a
first (k-2)-node
a
first (k-1)-node
a
first k-node

Forbidden subseq: ***ababa***

- Σ is (ababa)-free:

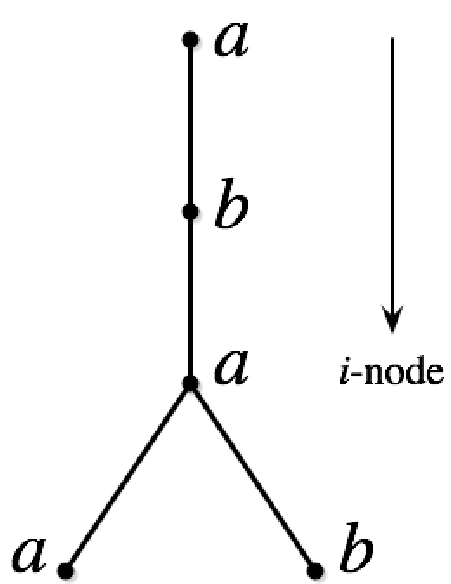


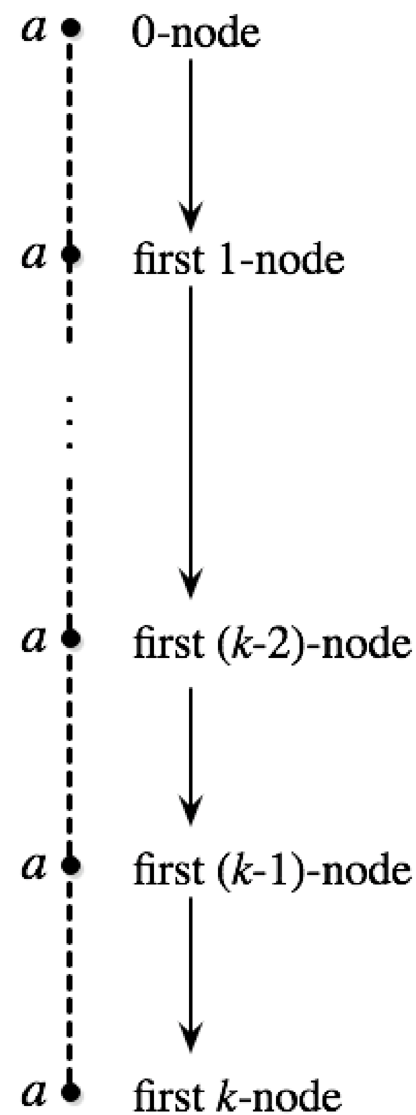


Seth Pettie

Forbidden subseq: ***ababa***

- Σ is (ababa)-free:

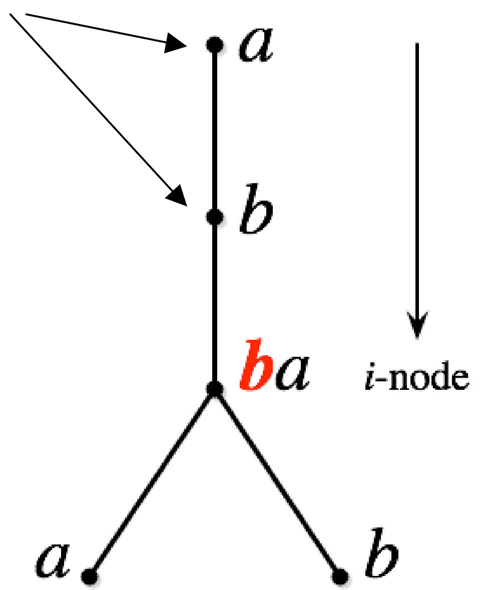


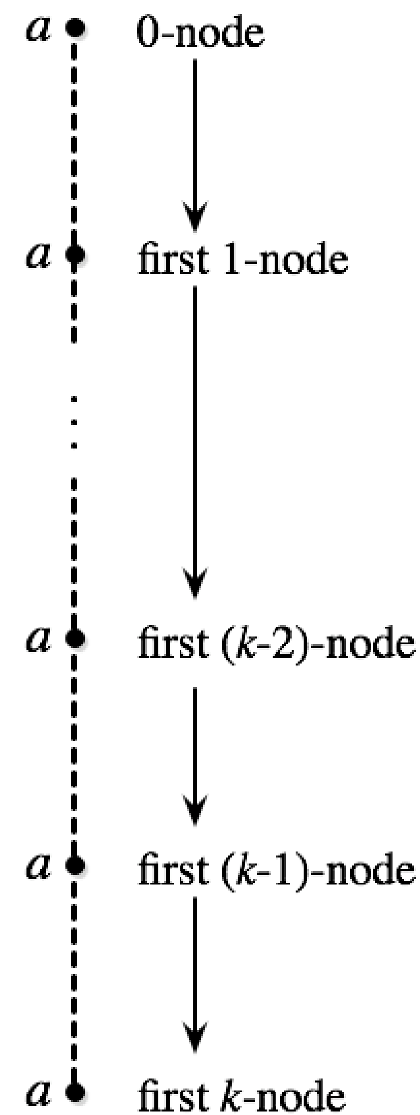


Forbidden subseq: ***ababa***

- Σ is (ababa)-free:

these are in the wrong order!

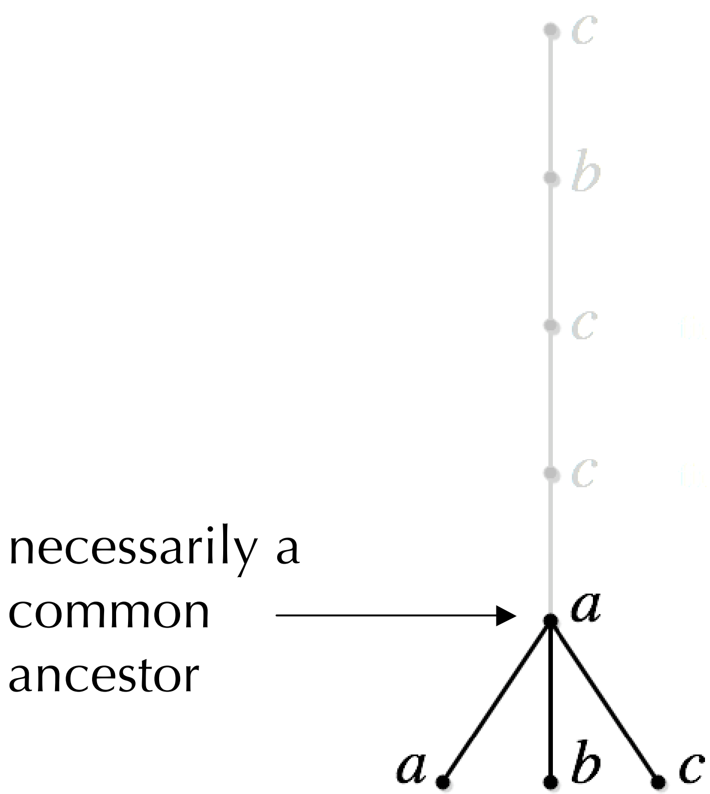




Seth Pettie

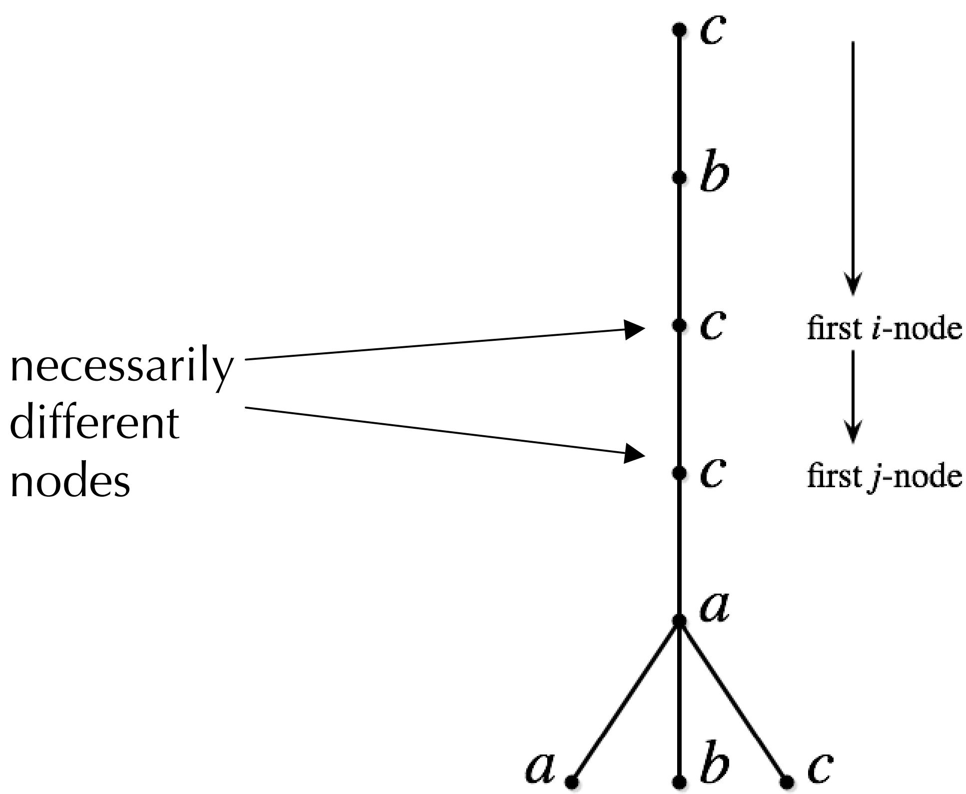
Forbidden subseq: ***abcaccbc***

- Σ is (abcaccbc)-free:



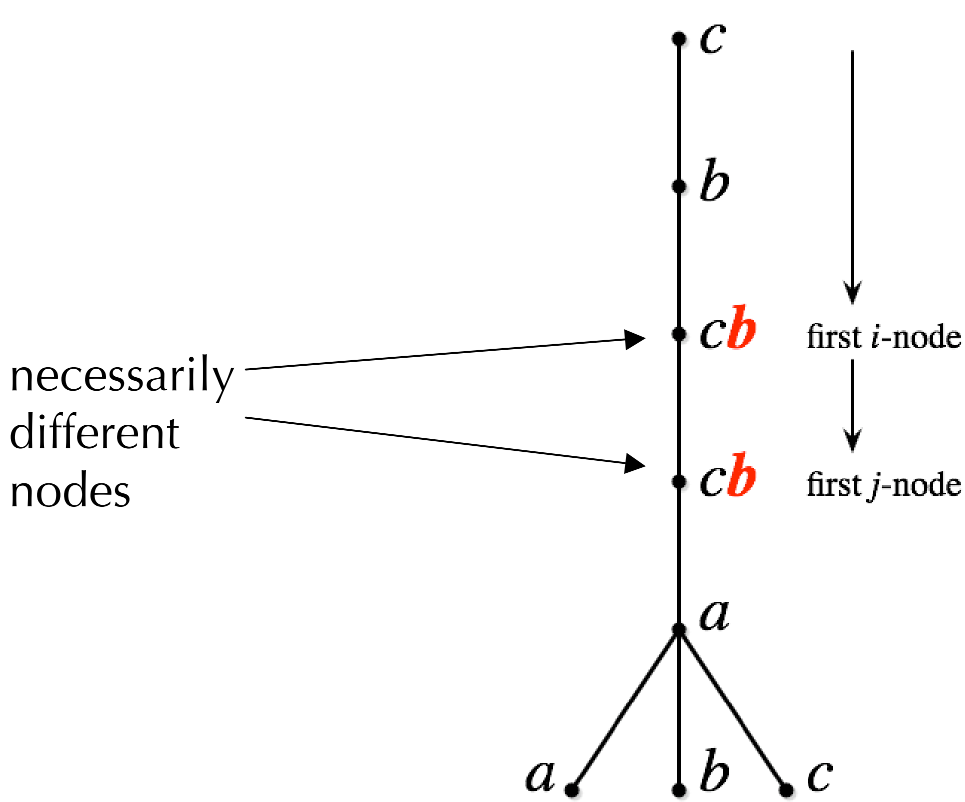
Forbidden subseq: ***abcaccbc***

- Σ is (abcaccbc)-free:



Forbidden subseq: ***abcaccbc***

- Σ is (abcaccbc)-free:

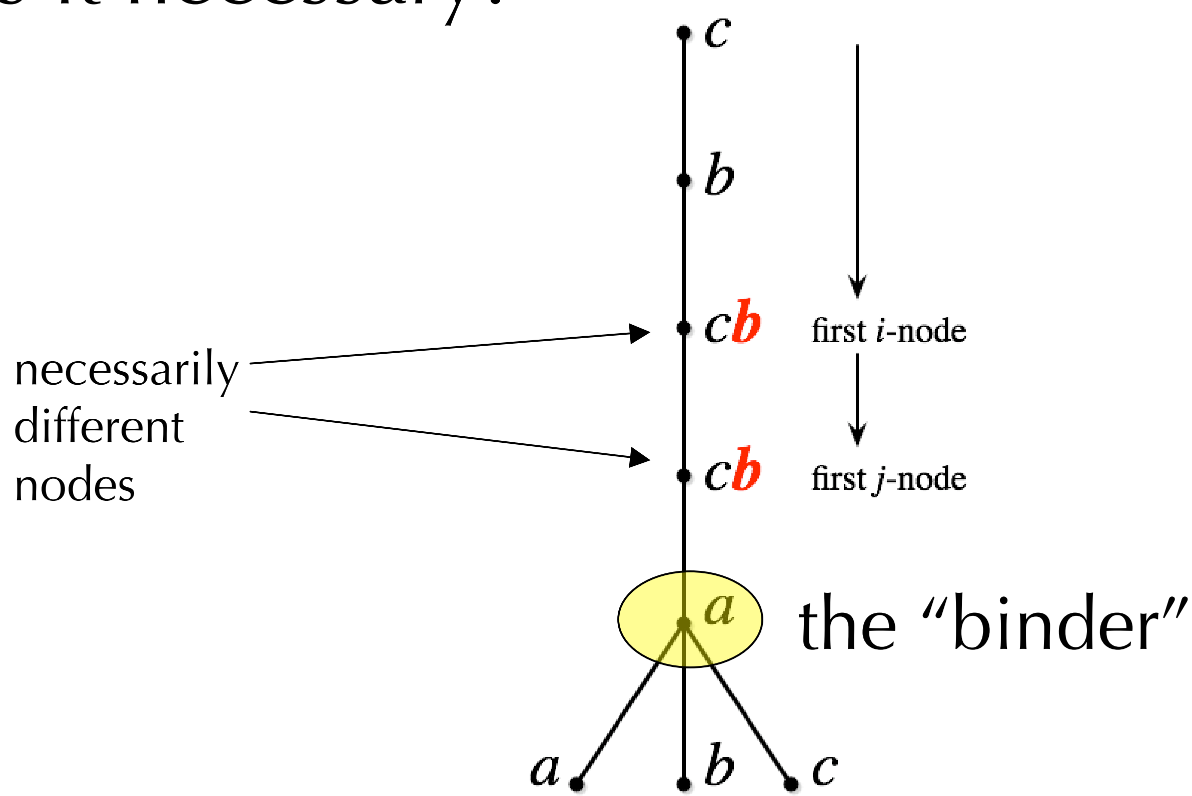


Forbidden subseq: ***abcaccbc***

"a" does not appear in the final contradiction

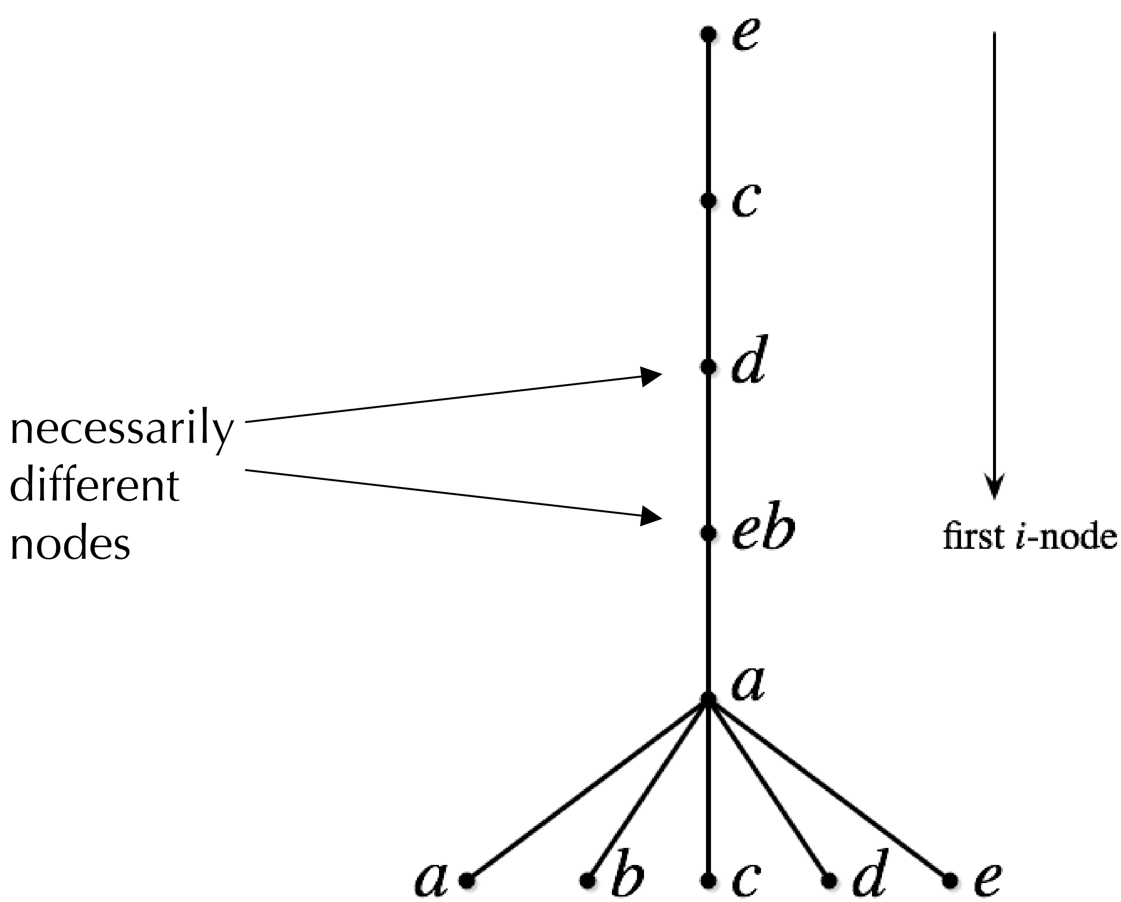
(an implied occurrence of ***bcbcbc***)

Why is it necessary?



Forbidden subseq: ***abcdeaebdce***

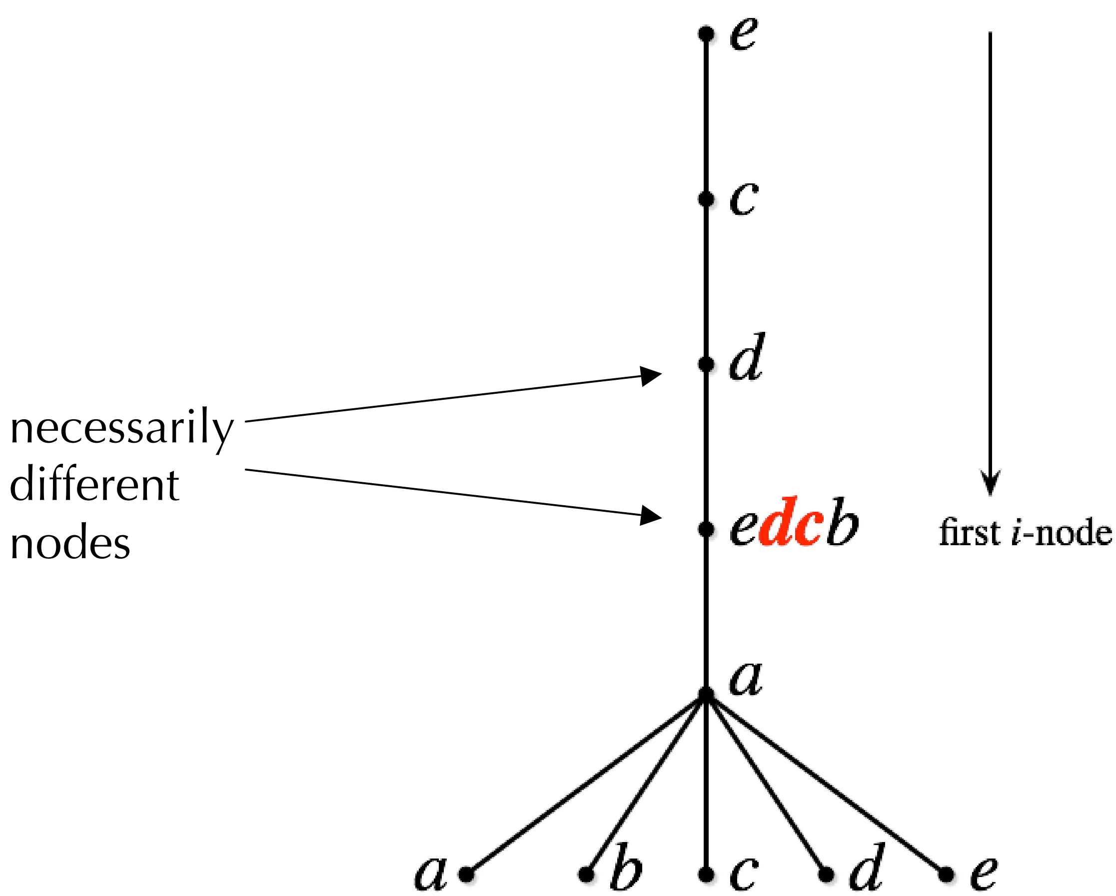
- Σ is (abcdeaebdce)-free:



Seth Pettie

Forbidden subseq: ***abcdeaebdce***

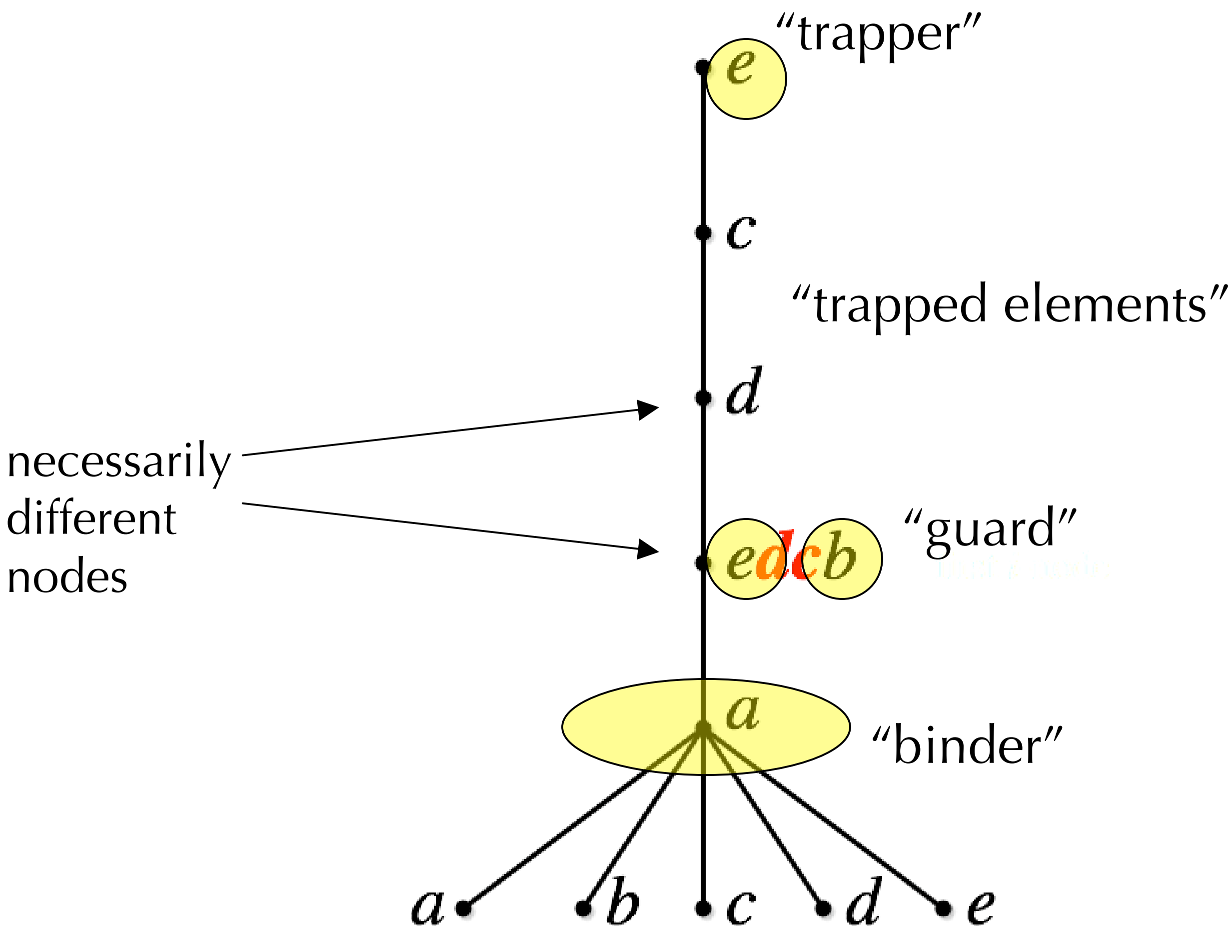
- Σ is (abcdeaebdce)-free:



Seth Pettie

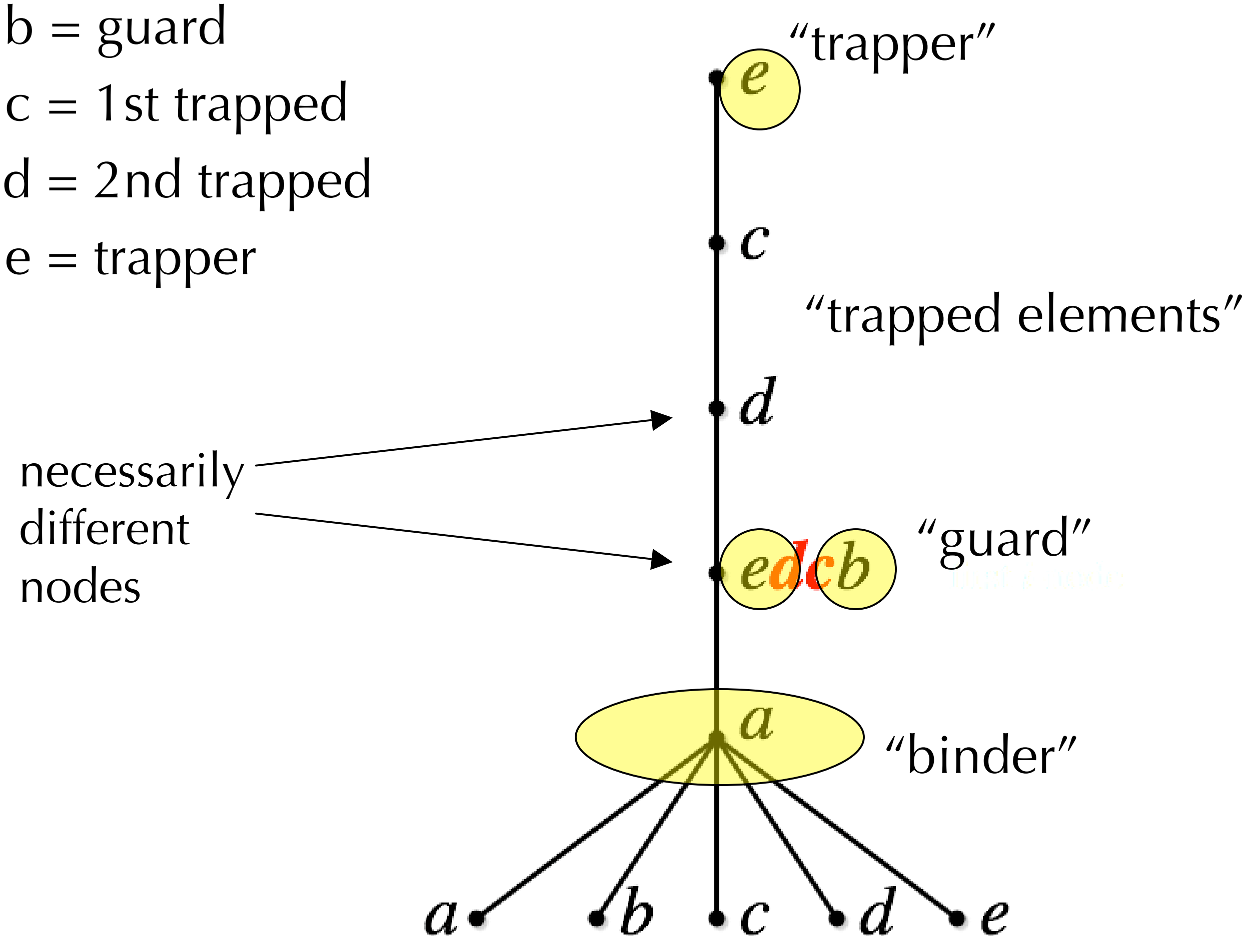
Forbidden subseq: ***abcdeaebdce***

- Σ is (abcdeaebdce)-free:



Forbidden subseq: ***abcdeaebdce***

- Succinct Encoding: ♥♦♠♠♣
  - ♥ : a = binder
  - ♦ : b = guard
  - ♠ : c = 1st trapped
  - ♠ : d = 2nd trapped
  - ♣ : e = trapper



All of these encodings make sense & work:

♥♦♠♠♣

♦♥♠♠♣

♥♠♦♠♣

These don’t:

♦♠♠♥♣ ⟵ the binder doesn’t bind (but this can be fixed!)

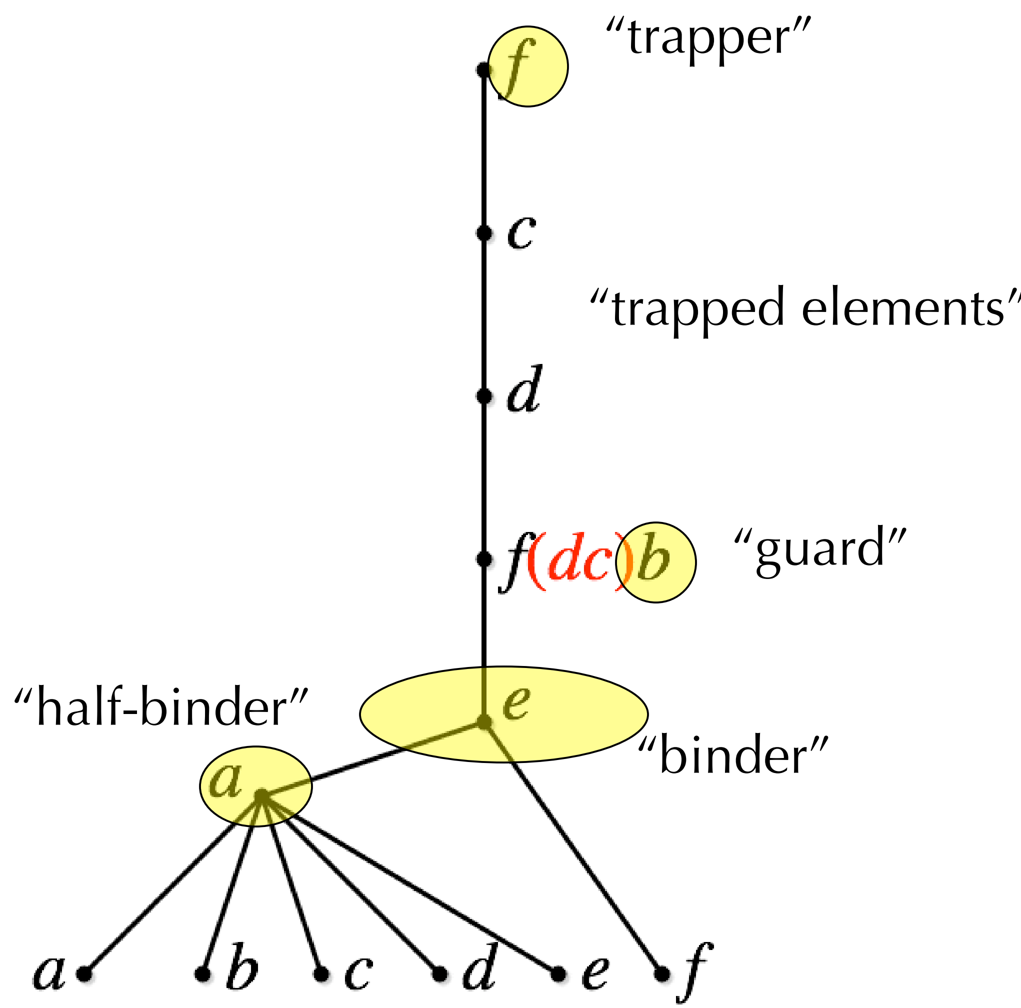
♥♠♠♦♣ ⟵ the guard doesn’t guard

♥♣♦♠♣ ⟵ this doesn’t make any sense

Seth Pettie

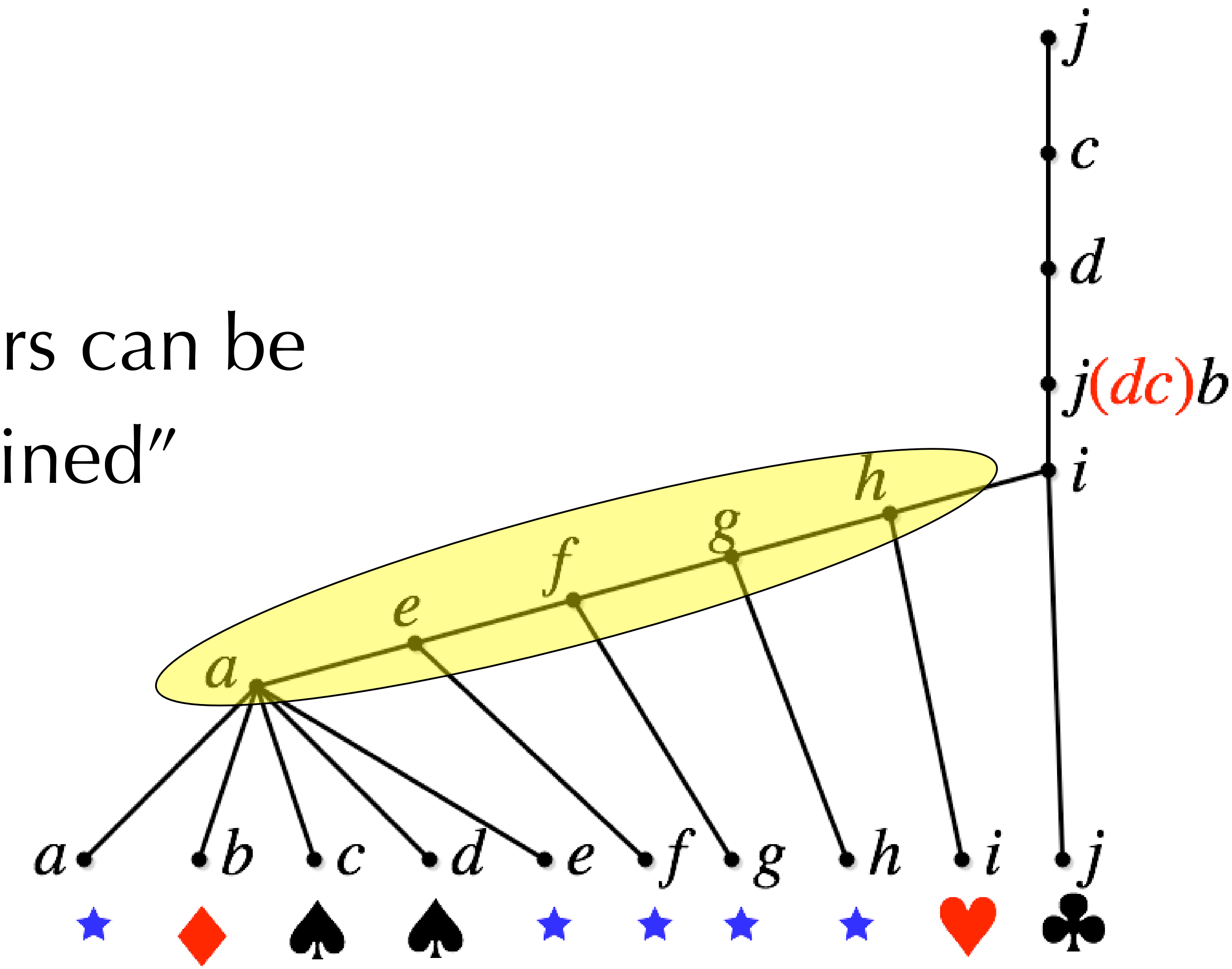
Forbidden subseq: ***abcdeafefbdcf***

- Encoding: ★♦♠♠♥♣
  - ★ : a = half-binder
  - ♦ : b = guard
  - ♠ : c = 1st trapped
  - ♠ : d = 2nd trapped
  - ♥ : e = binder
  - ♣ : f = trapper



Forbidden subseq: ***abcdeafegfhgihjijbdcj***

Half-binders can be
"daisy-chained"



Seventeen legal encodings

♥♠(♦♠♣)

★♣♠♥♣

★♠♥(♦♠♣)

★♦♠♠♥♣

♦♥♠♠♣

★♠♦♥♠♣

★♠♥(♦♠)♣

♥♣♠♣

♦★♠♥♠♣

♥♠♦♠♣

★♠♦♠♥♣

★♠♥♦♠♣

♥♠(♦♠)♣

♦★♠♠♥♣

♥♦♠♠♣

★♦♠♥♠♣

★♠(♦♠)♥

Seth Pettie

- Some open problems
  - Are there infinitely many “causes” of non-linearity?
  - Are there any more linear seqs. to be discovered?
  - For each c, is there an (ababa)-free σ such that:

$$Ex(\sigma,n) = n \exp(\alpha^c(n))$$