MaMuX

March, 2012

David Rizo

Univ. Alicante

Metrical trees

music comparison, classification and composition

ircam Centre Pompidou

PRAIg

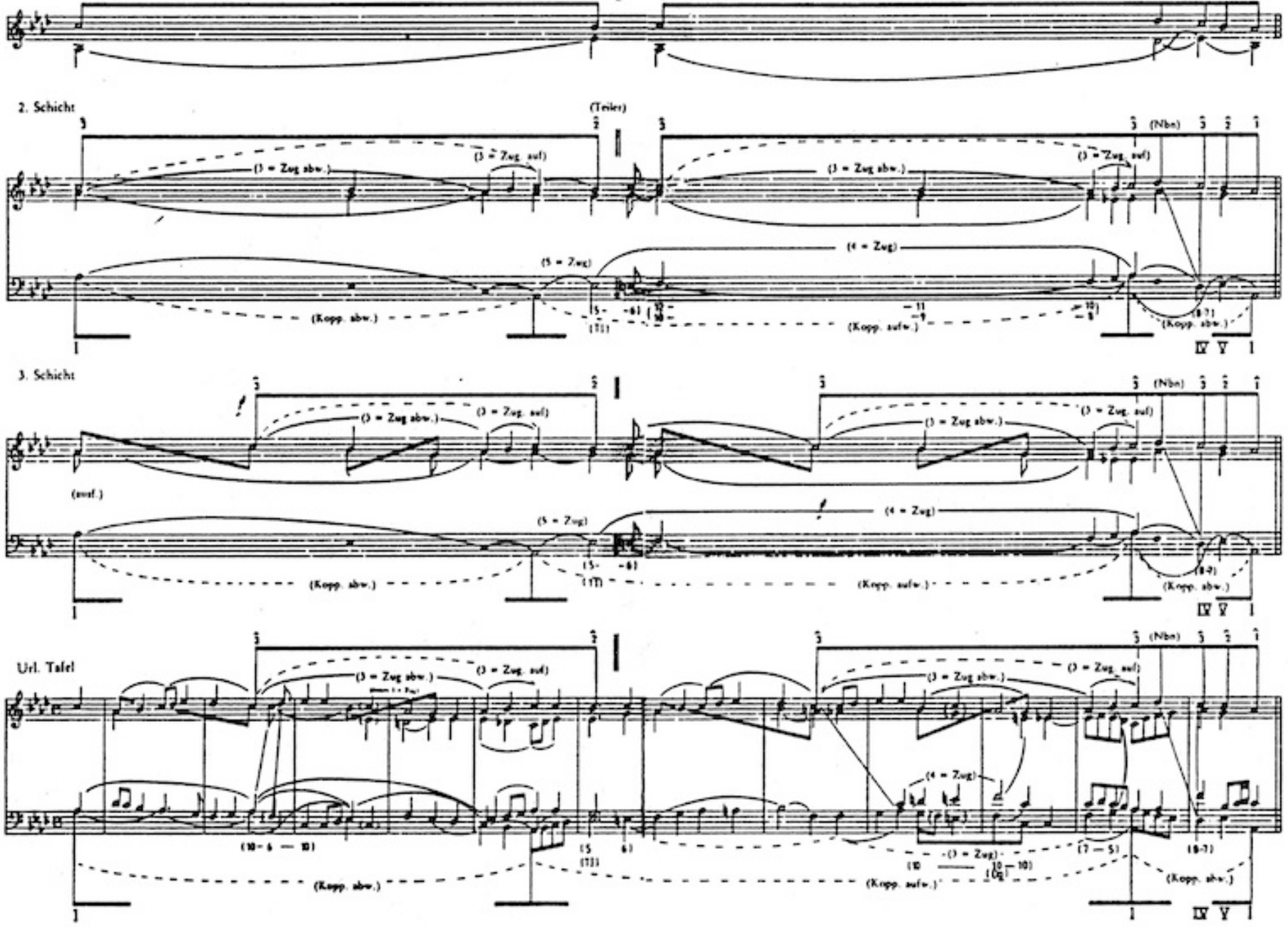
IUII

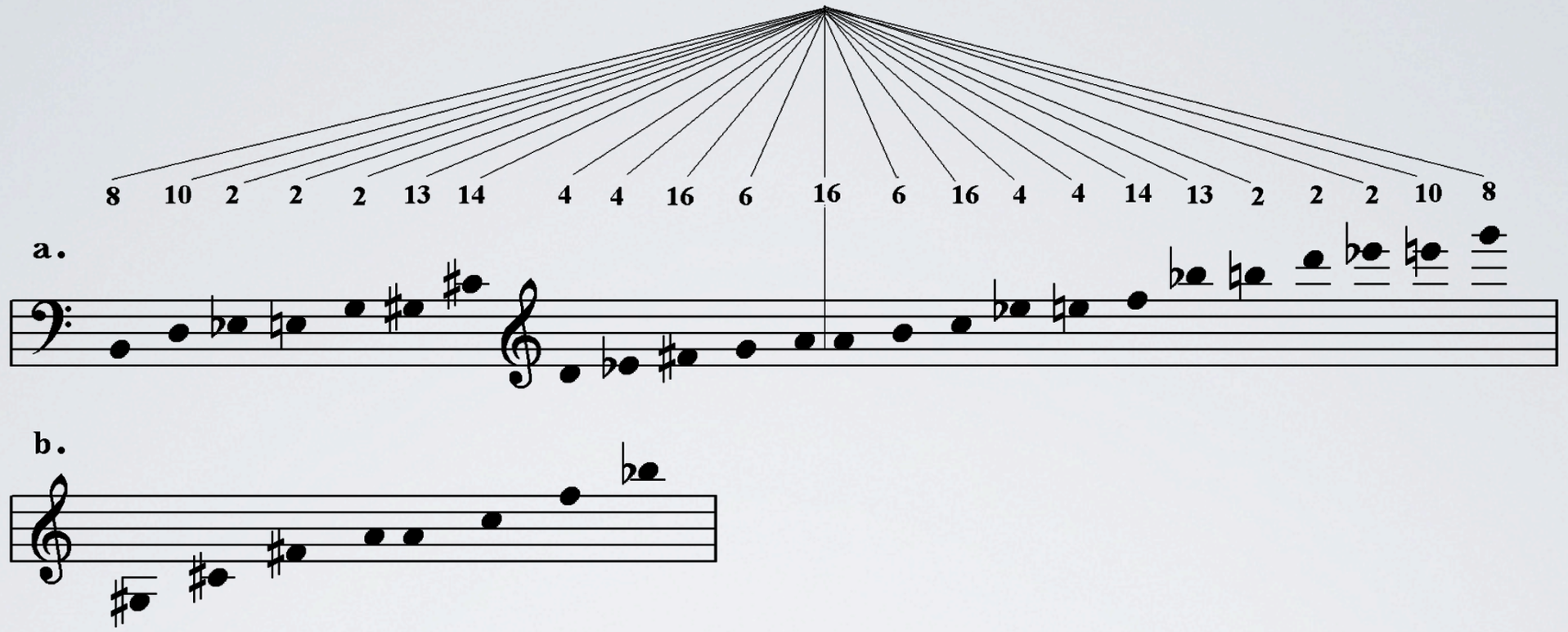
Universitat d'Alacant
Universidad de Alicante

Trees
&
Science

Newton Tree

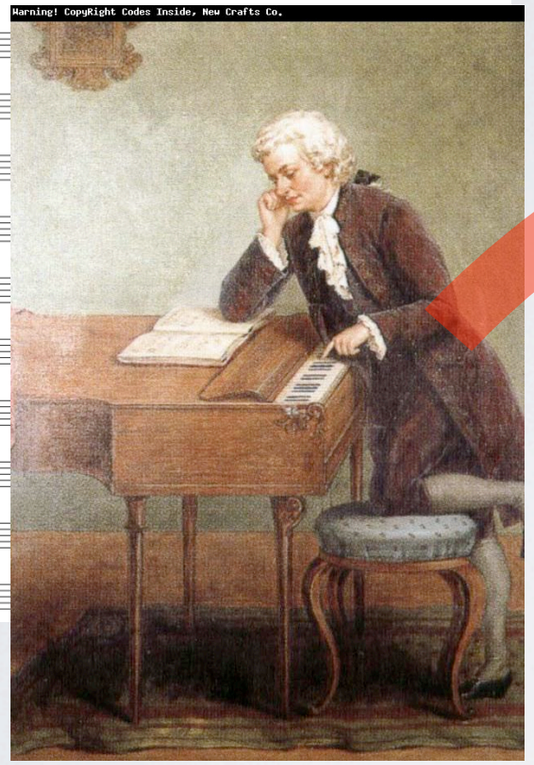
Trees & Music


2. Schicht
(Teiler)
(Nbn)
(3 = Zug abw.)
(3 = Zug auf)
(4 = Zug)
(5 = Zug)
(Kopp. abw.)
(Kopp. auf.)
3. Schicht
(auf.)
Url. Tafel



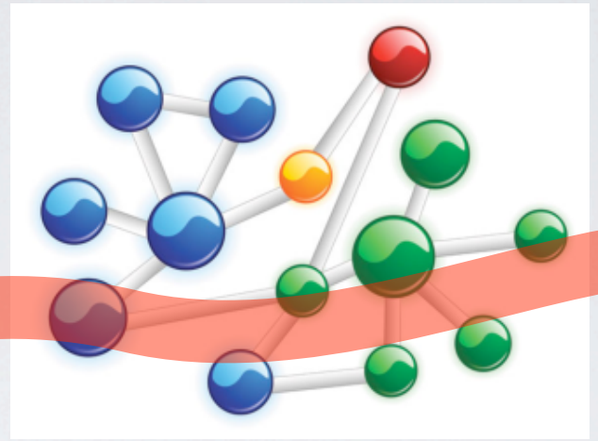
This talk is about the
mathematical structure
TREE
and
MUSIC

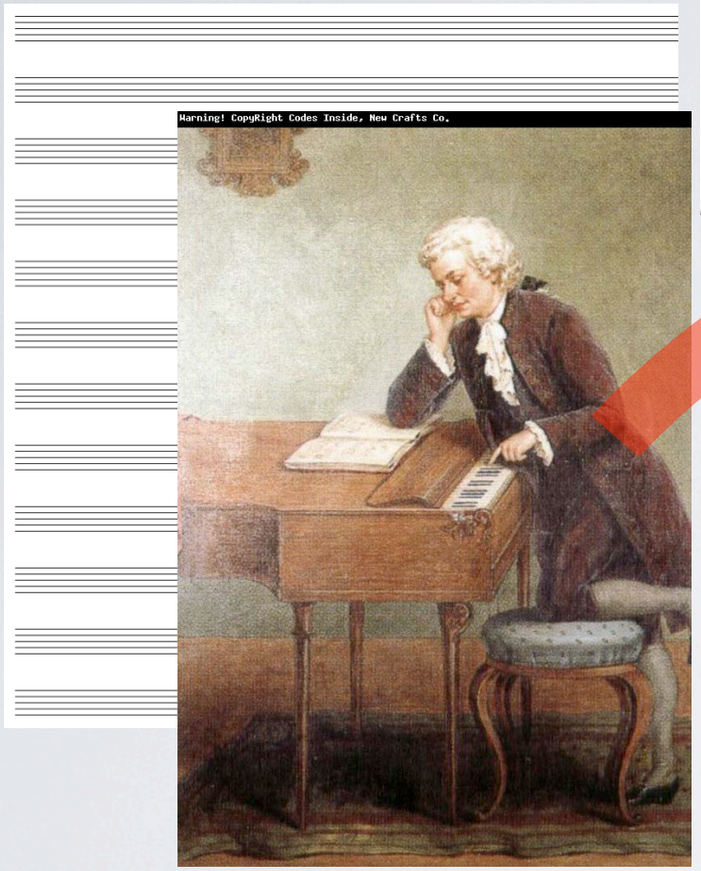
HUMAN MUSICAL PROCESS



Creation

HUMAN MUSICAL PROCESS

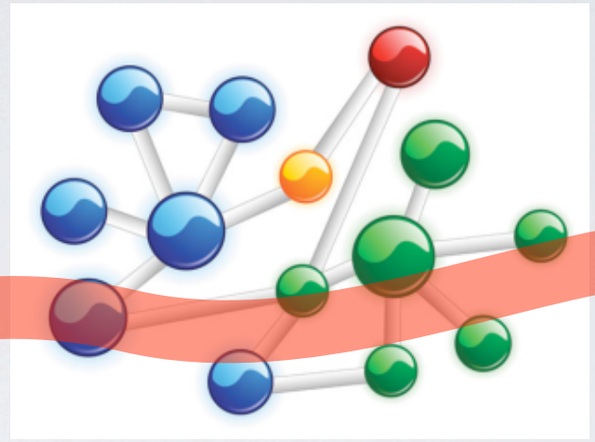




Creation

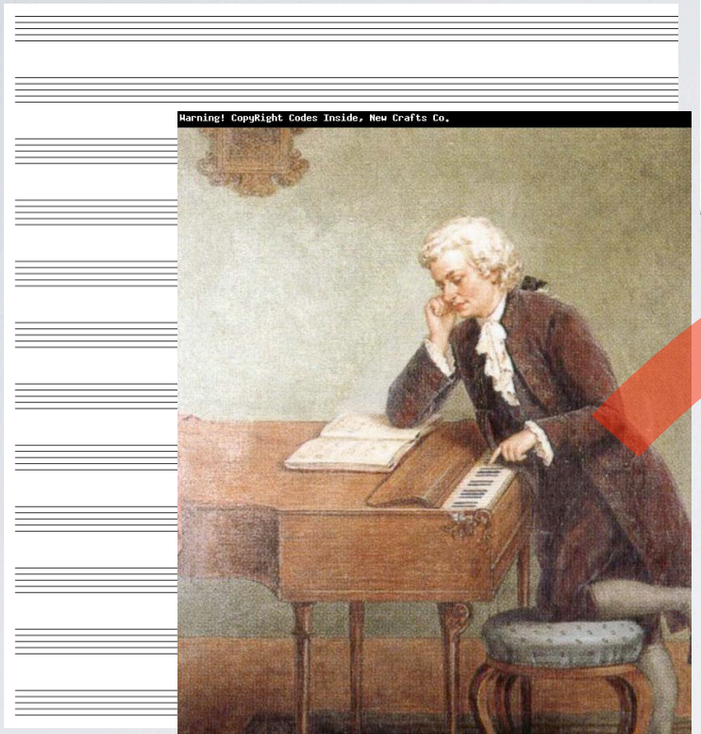
Based on an abstraction / structure

HUMAN MUSICAL PROCESS



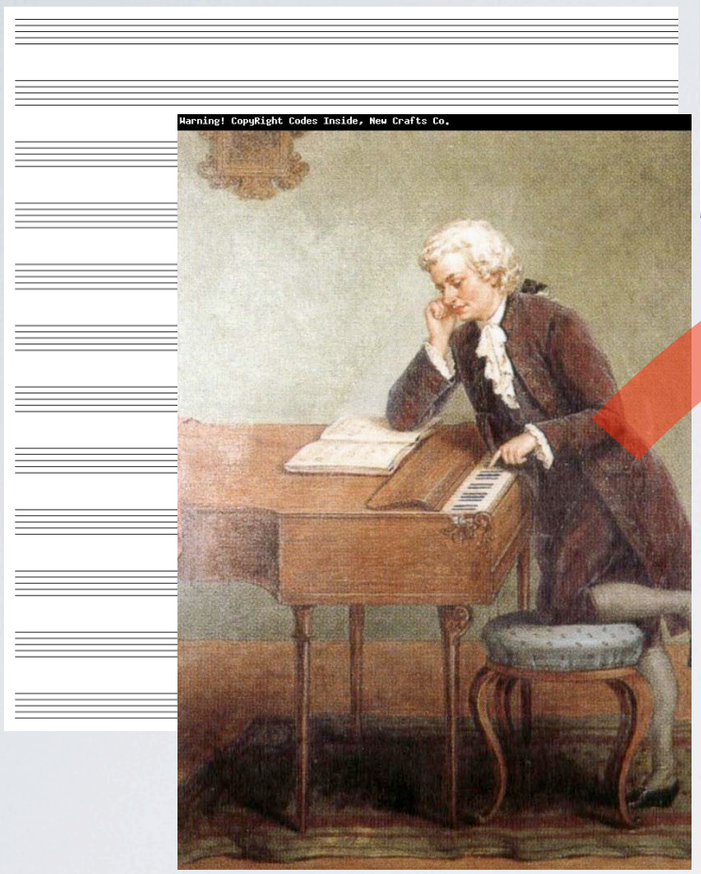
Based on an abstraction / structure

Symbols: score

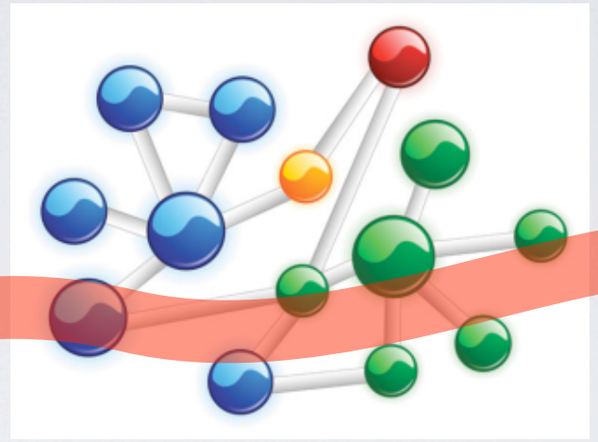


Creation

HUMAN MUSICAL PROCESS



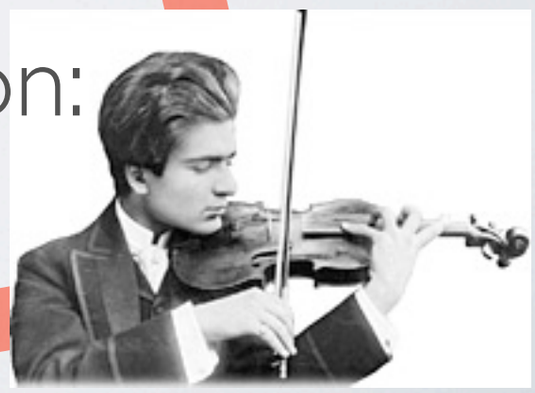
Creation



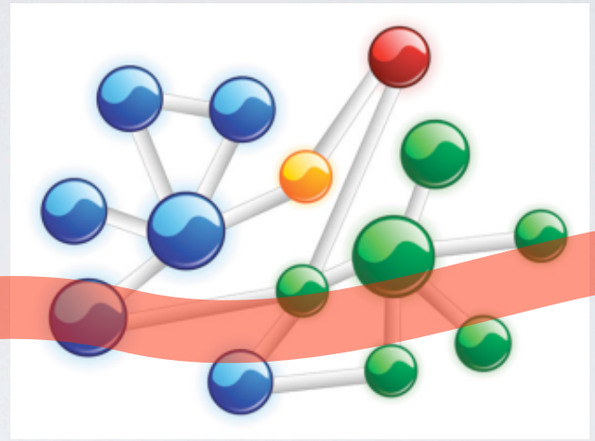
Based on an abstraction / structure

Symbols: score

Interpretation: produces

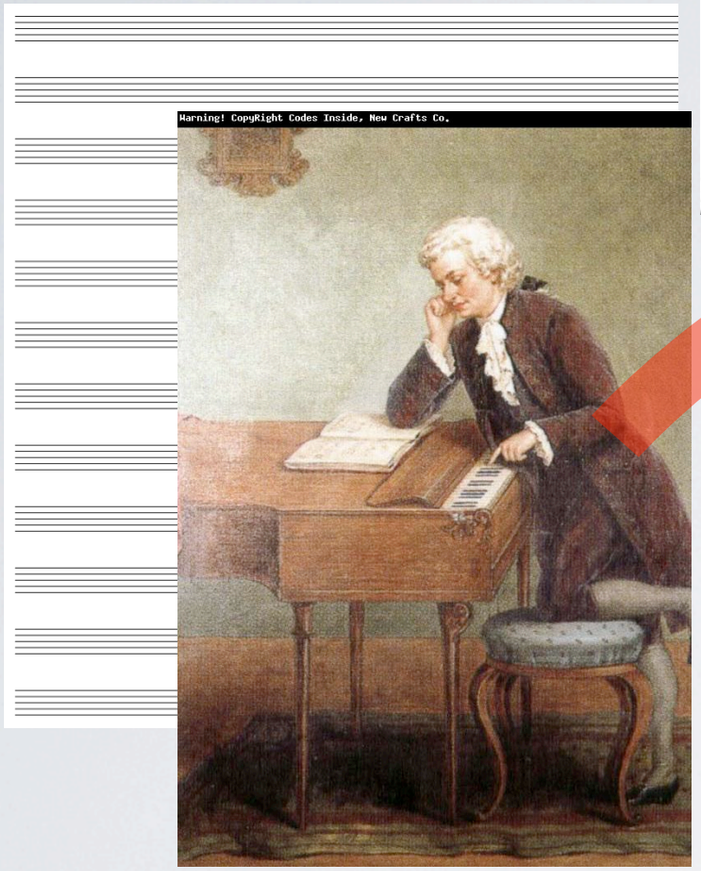


HUMAN MUSICAL PROCESS



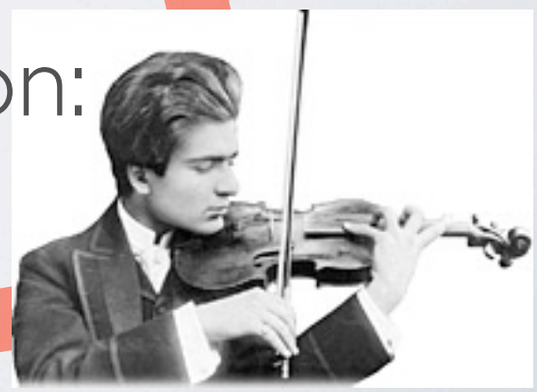
Based on an abstraction / structure

Symbols: score

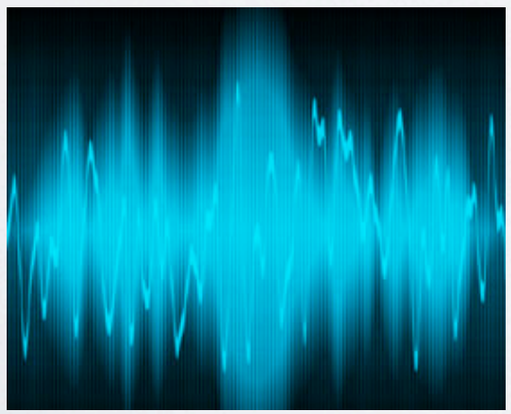


Creation

Interpretation: produces



Music Sound

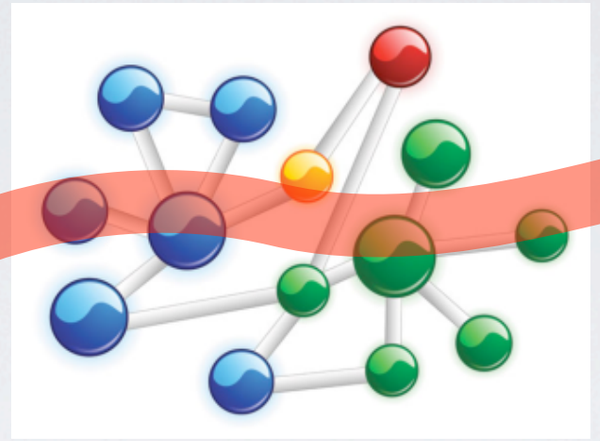


COMPUTER MUSICAL PROCESS

Creation

COMPUTER MUSICAL PROCESS

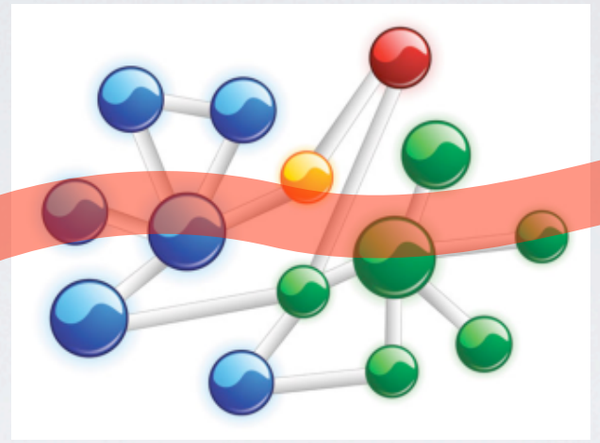
Creation



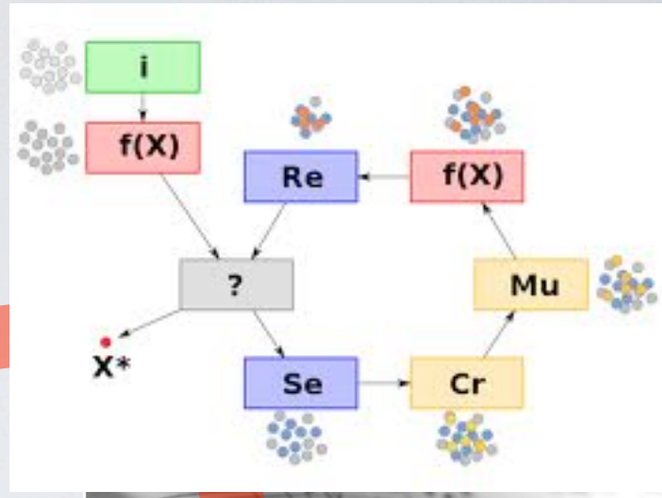
Based on an abstraction / structure

COMPUTER MUSICAL PROCESS

Creation



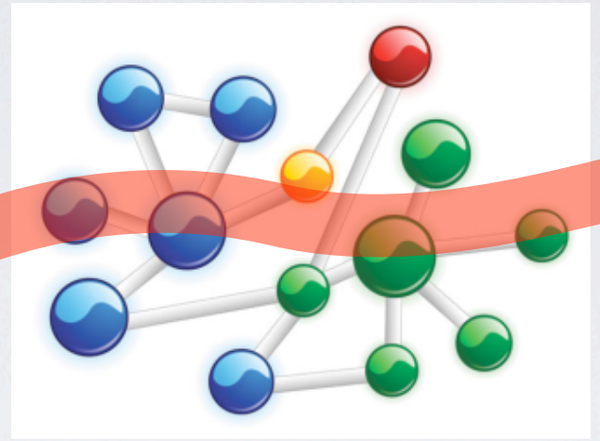
Based on an abstraction / structure



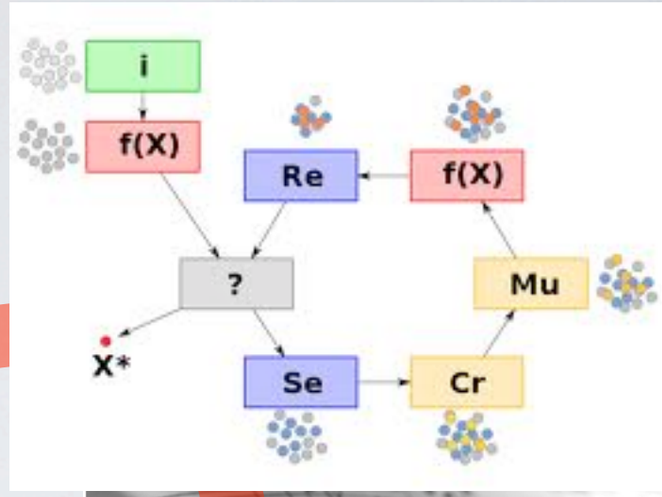
Symbols: score

COMPUTER MUSICAL PROCESS

Creation



Based on an abstraction / structure

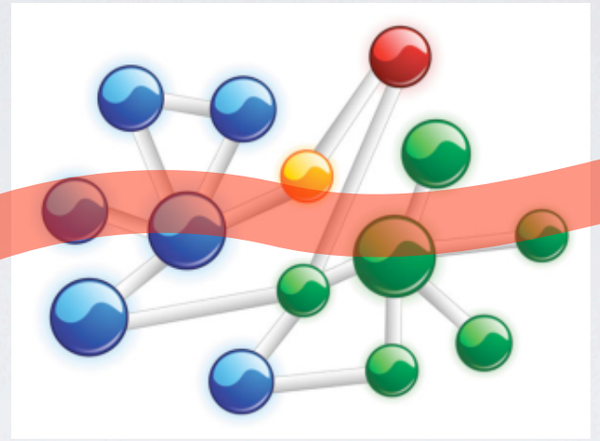


Symbols: score

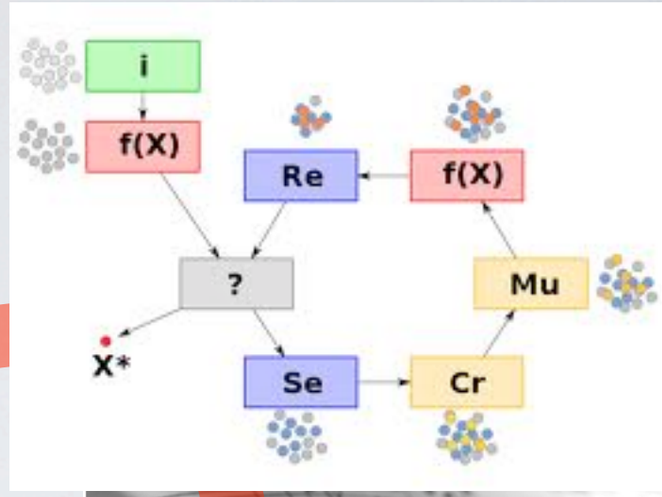
Interpretation: produces

COMPUTER MUSICAL PROCESS

Creation



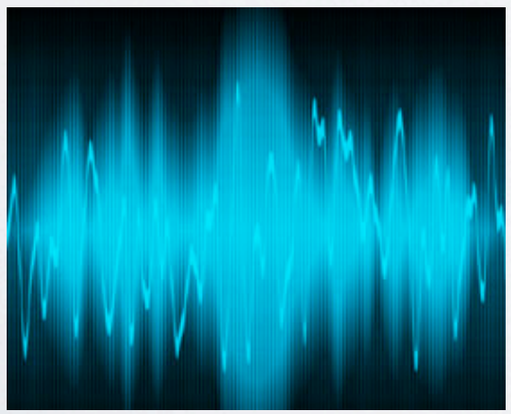
Based on an abstraction / structure



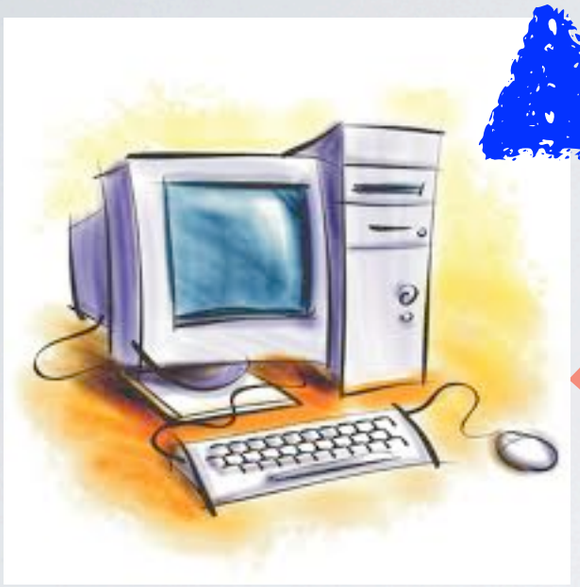
Symbols: score

Interpretation: produces

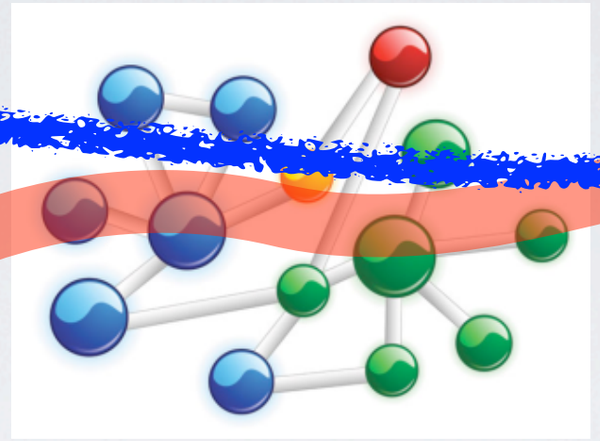
Music Sound



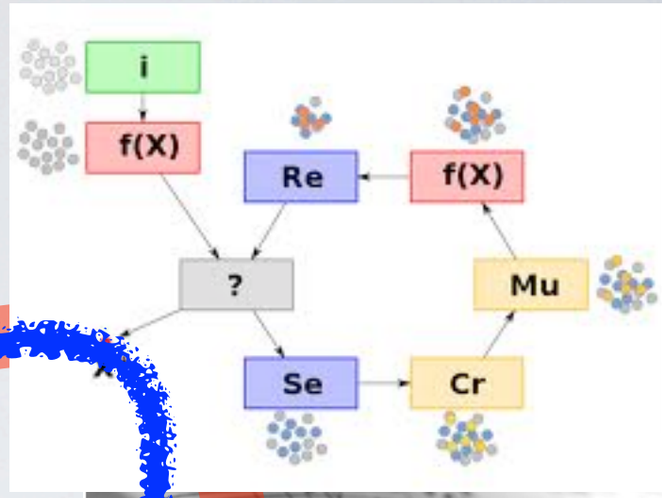
COMPUTER MUSICAL PROCESS



Analysis

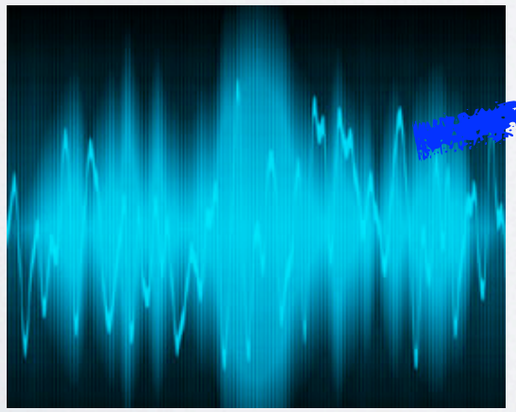


Based on an abstraction / structure



Symbols: score

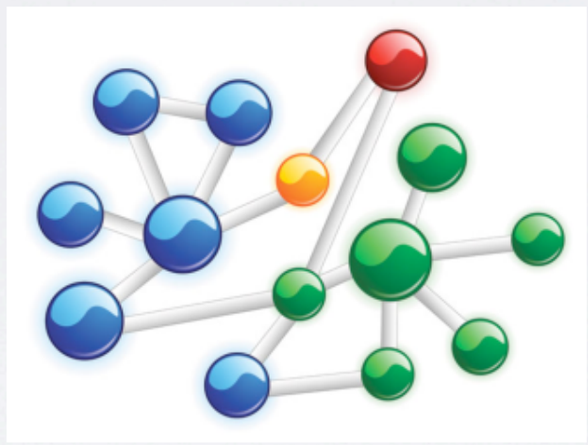
Interpretation: produces



Music
Sound

What role can trees play in

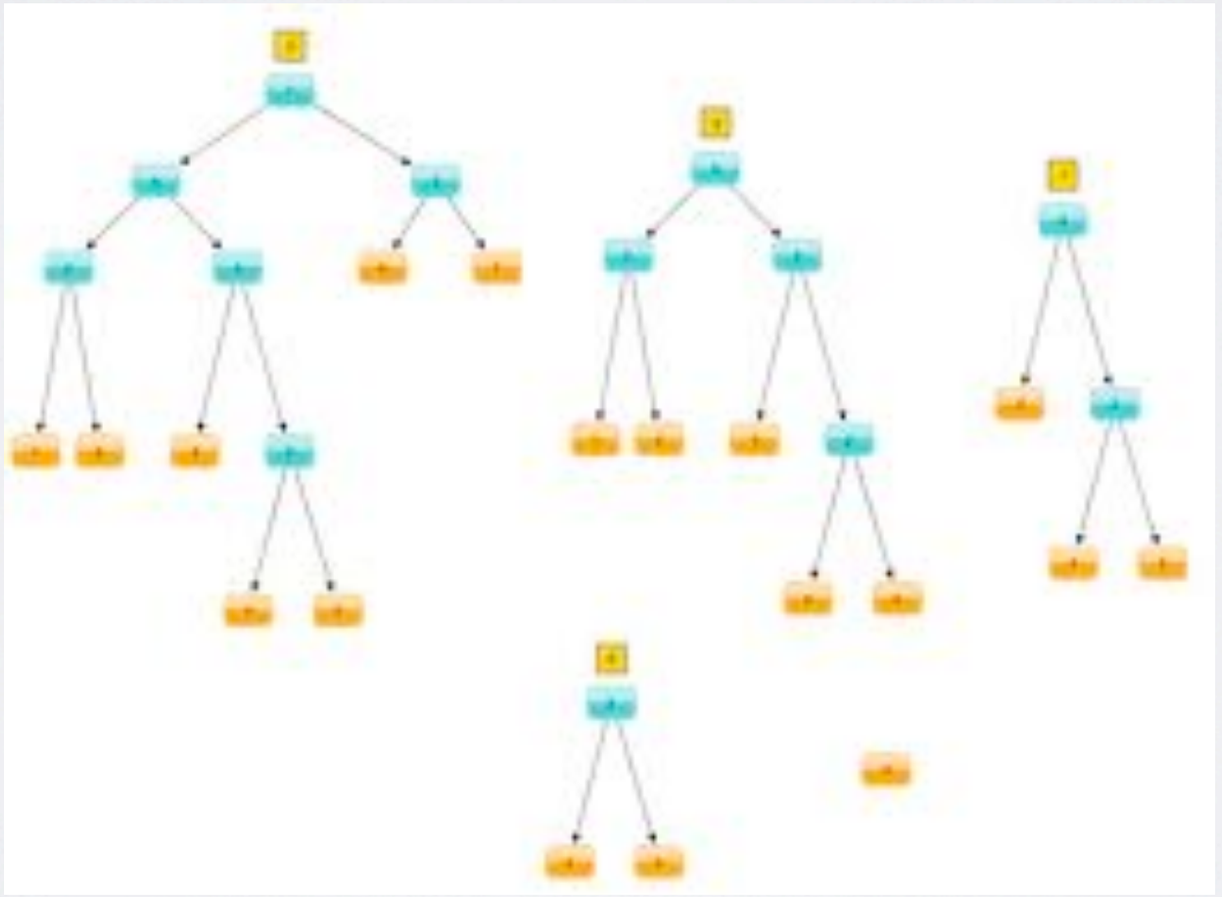
music ?



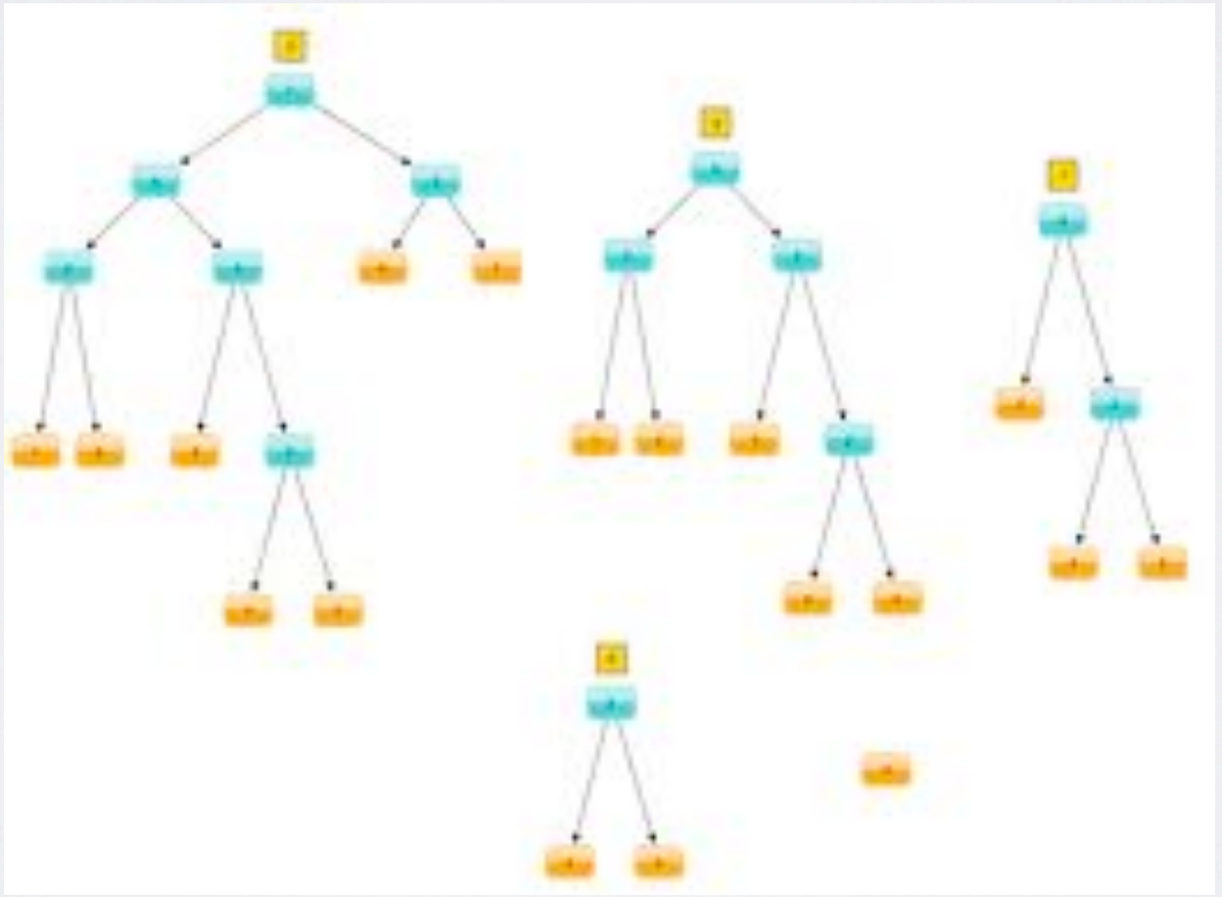
Based on an abstraction / structure

What role can trees play in

music ?



What role can trees play in computer music ?



REPRESENTATION

- Internal intermediate representation dramatically conditions the application

REPRESENTATION

- Internal intermediate representation dramatically conditions the application
  - e.g. Which is better for printing a score?

REPRESENTATION

- Internal intermediate representation dramatically conditions the application
  - e.g. Which is better for printing a score?

a) Sequence of frequency and onset time in milliseconds

REPRESENTATION

- Internal intermediate representation dramatically conditions the application
  - e.g. Which is better for printing a score?

a) Sequence of frequency and onset time in milliseconds

b) Sequence of note names and figures

REPRESENTATION

- Internal intermediate representation dramatically conditions the application
  - e.g. Which is better for printing a score?

a) Sequence of frequency and onset time in milliseconds

b) Sequence of note names and figures

REPRESENTATION

REPRESENTATION

- Which is best

REPRESENTATION

- Which is best
  - depends on the application

REPRESENTATION

- Which is best
  - depends on the application
- **Composition**

REPRESENTATION

- Which is best
  - depends on the application
- **Composition**
  - easily generate new content

REPRESENTATION

- Which is best
  - depends on the application
- **Composition**
  - easily generate new content
- **Analysis and retrieval**

REPRESENTATION

- Which is best
  - depends on the application
- **Composition**
  - easily generate new content
- **Analysis and retrieval**
  - easily capture important features

SYMBOLIC REPRESENTATIONS

- **Strings**
- **n-grams**
- Graphs
- Point sets
- Geometric
- Hidden Markov Models
- Spiral arrays, etc....
- **Trees**

ABSTRACT DATA TYPE **TREE**

- **Suitable intermediate representation** for:
  - **music information retrieval** (MIR)
  - **composing**

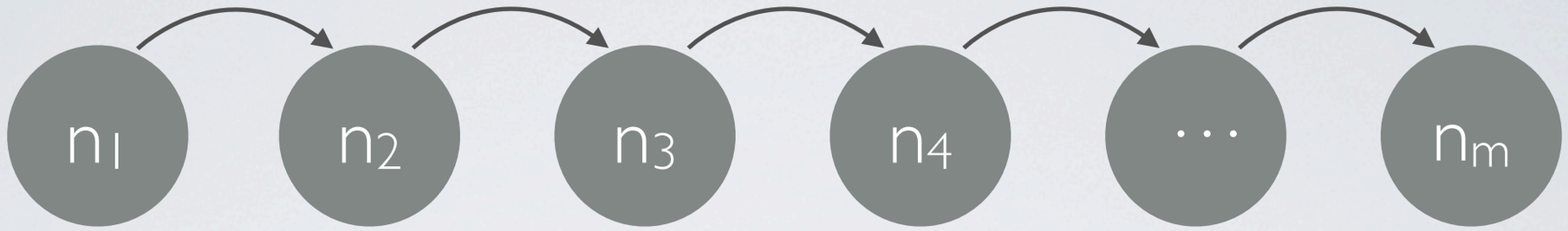
OVERVIEW

- Background
  - Intermediate representations
- The origin: strings
- Previous uses of trees in computer music

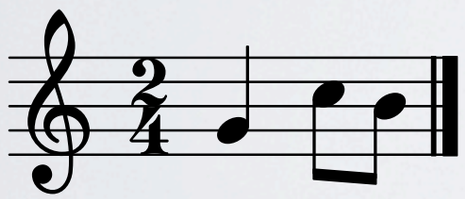
- Our tree approach
  - metric tree construction
    - probabilistic k-testables
- Applications
  - classification and generation
  - composition

String = sequence of symbols

STRINGS



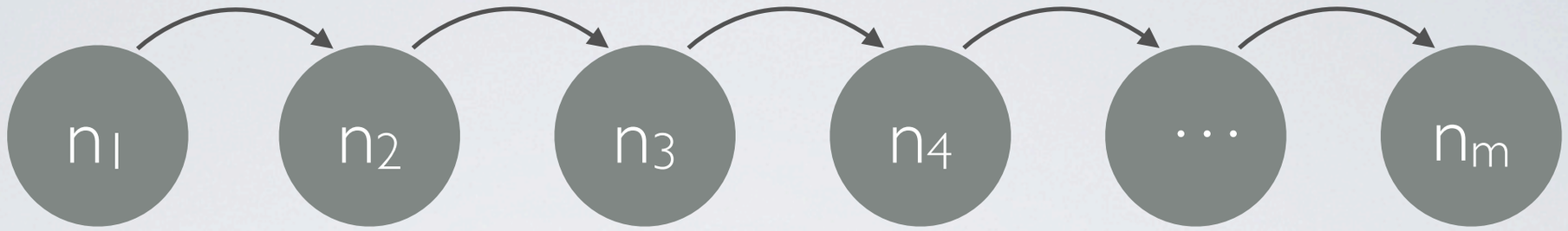
- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**



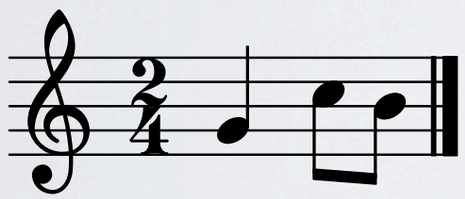
Pitch

Rhythm

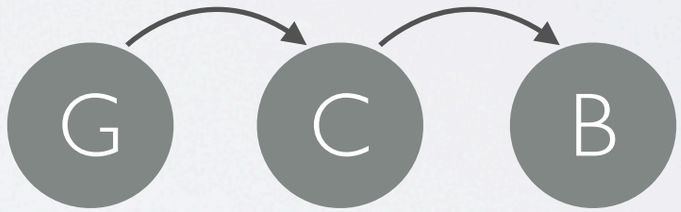
STRINGS



- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**



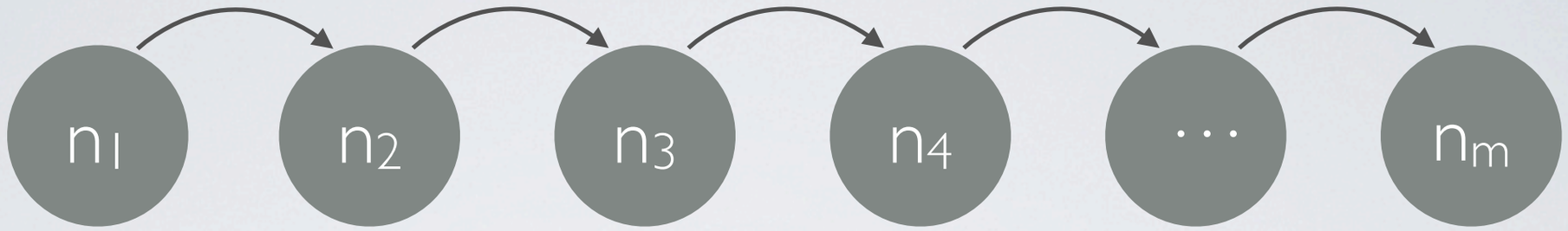
Pitch



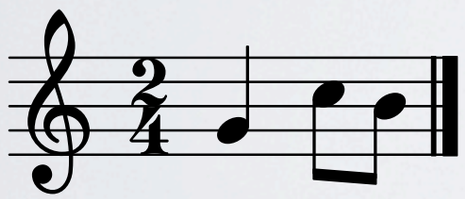
Note name

Rhythm

STRINGS



- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**

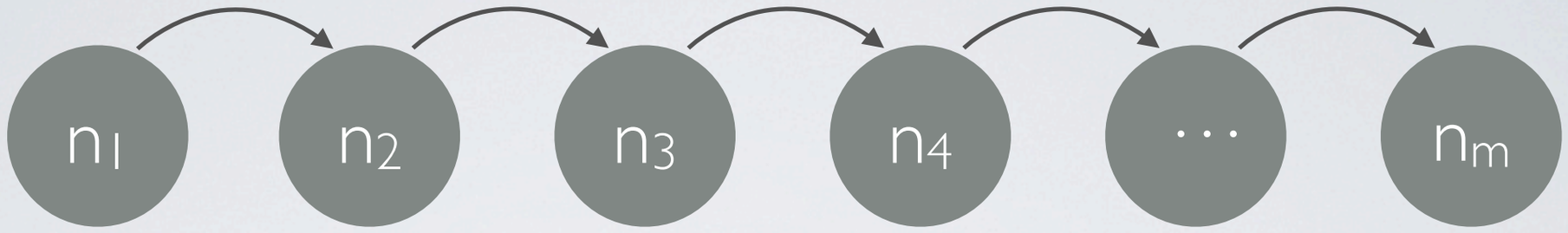


Pitch

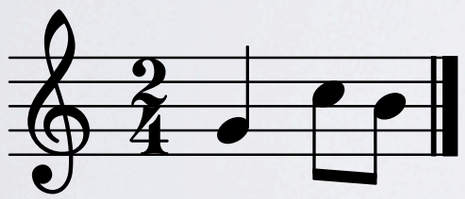
MIDI pitch

Rhythm

STRINGS



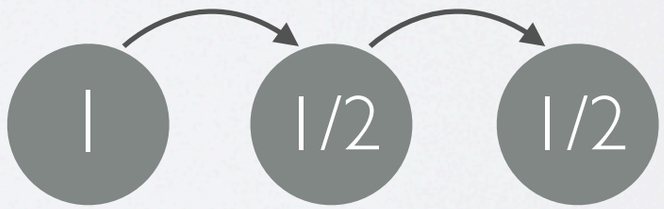
- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**



Pitch

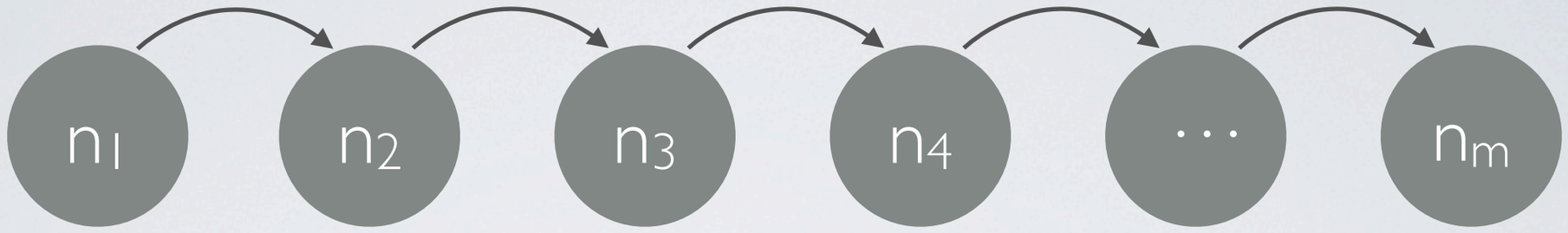
MIDI pitch

Rhythm

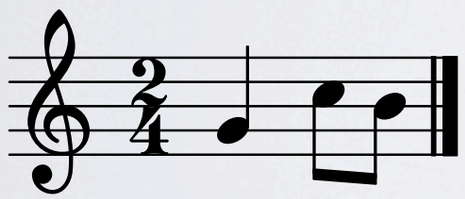


Duration

STRINGS



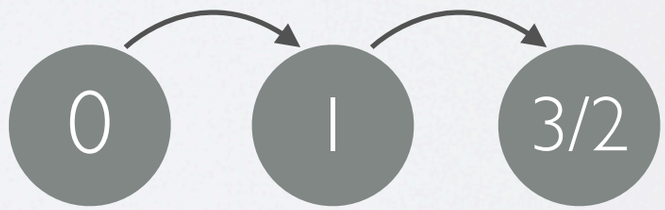
- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**



Pitch

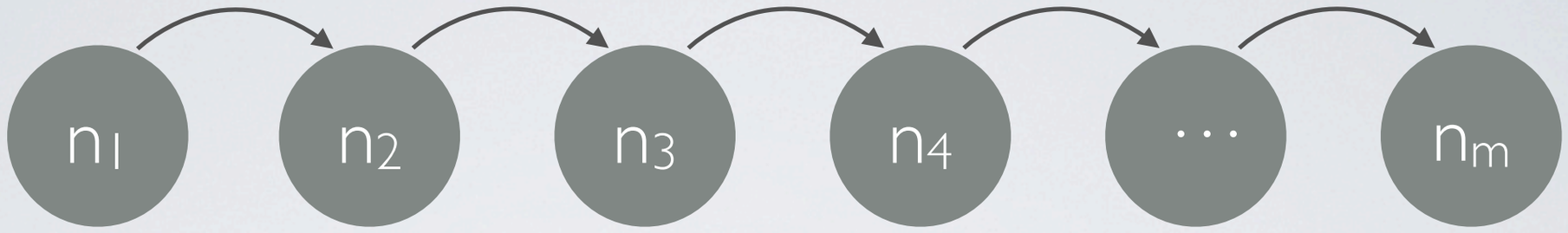
MIDI pitch

Rhythm

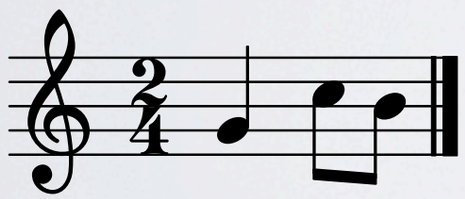


Onset

STRINGS



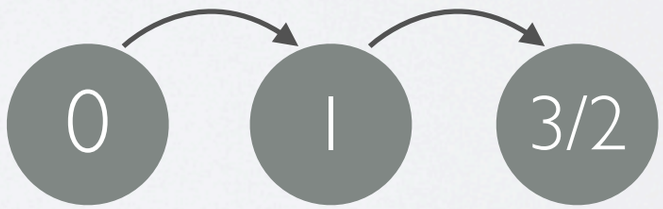
- A **melody** is represented as a **sequence** of symbols $\mathbf{n_i}$
  - each symbol $\mathbf{n_i}$ may represent **a note**



Pitch

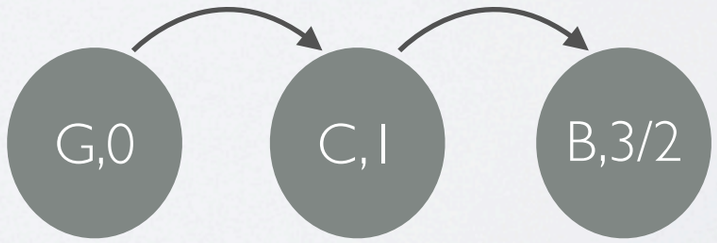
MIDI pitch

Rhythm



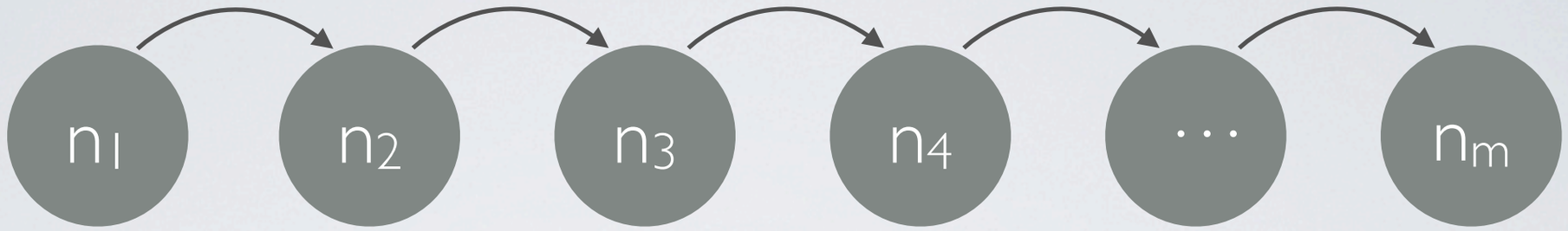
Onset

A combination

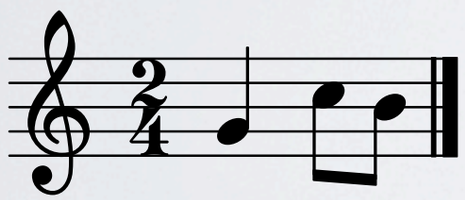


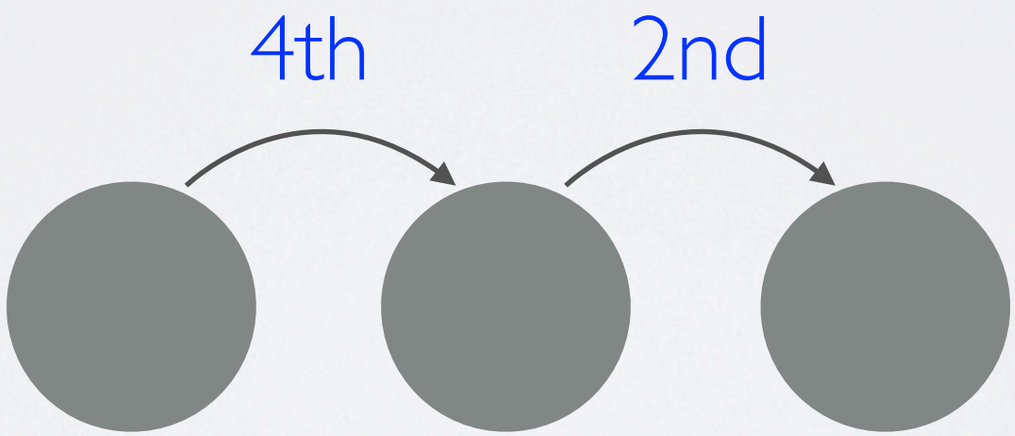
[Pitch, Onset]

STRINGS



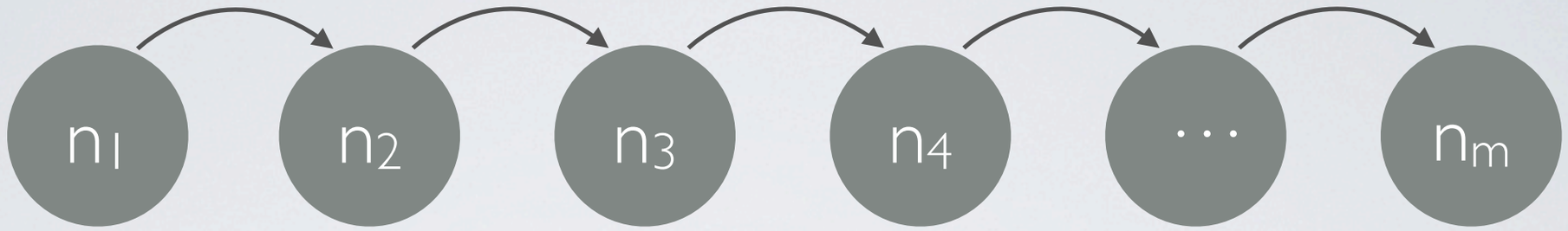
- A melody is represented as a sequence of symbols n_i
  - each symbol $\mathbf{n_i}$ may represent a note **or a relation**



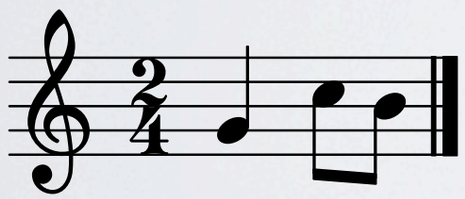


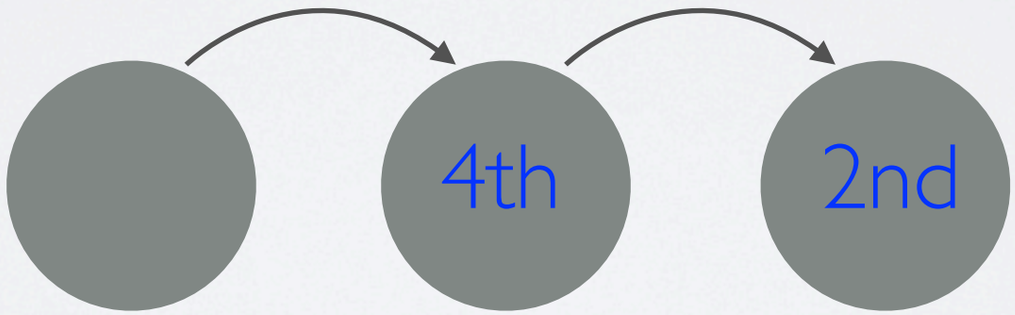
interval

STRINGS



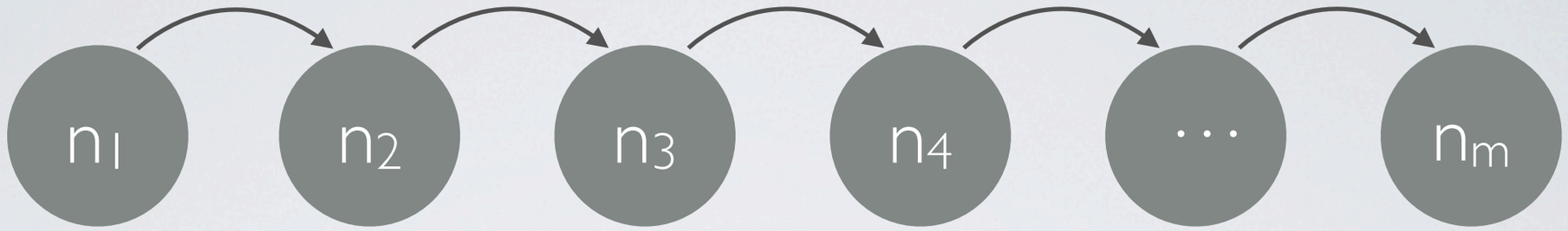
- A melody is represented as a sequence of symbols n_i
  - each symbol $\mathbf{n_i}$ may represent a note **or a relation**



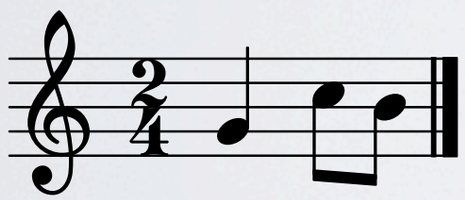


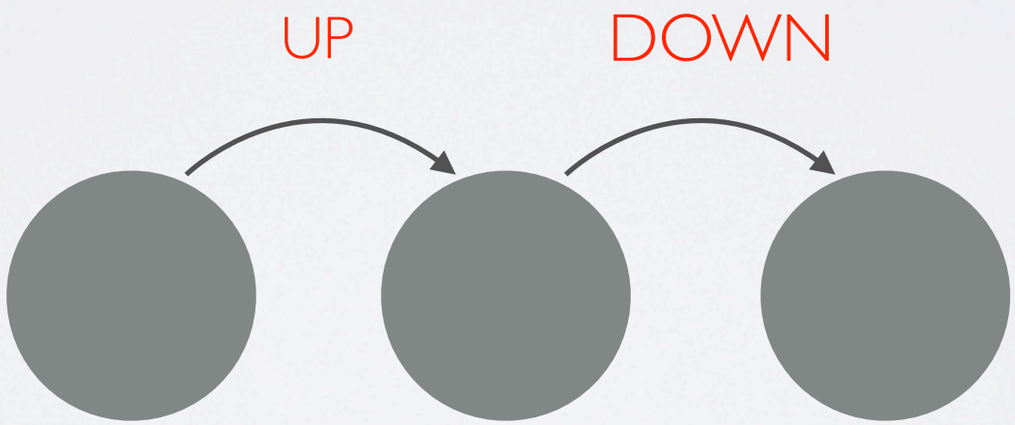
interval

STRINGS



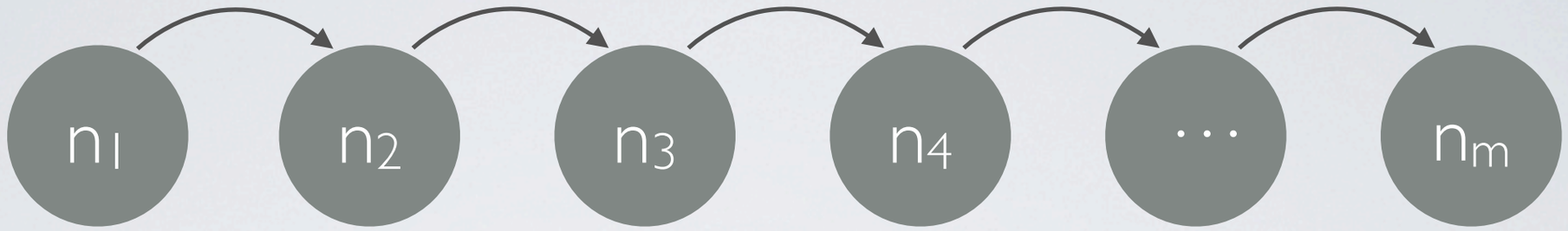
- A melody is represented as a sequence of symbols n_i
  - each symbol $\mathbf{n_i}$ may represent a note **or a relation**



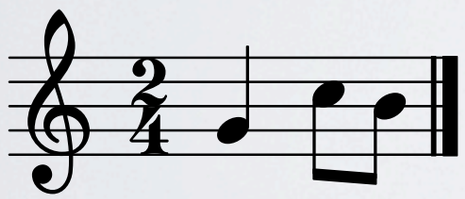


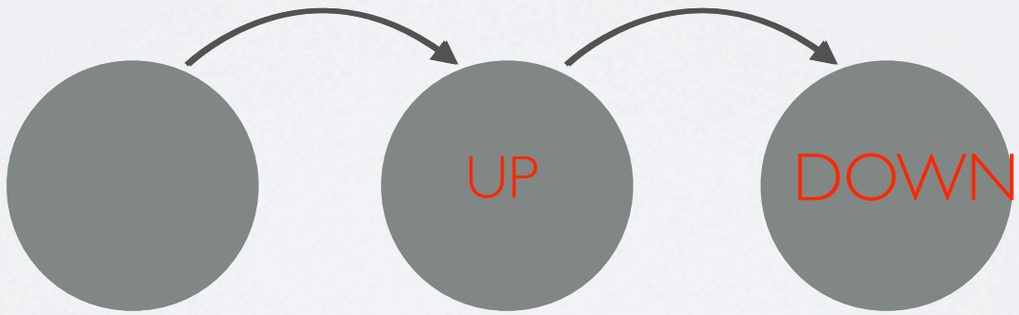
contour

STRINGS



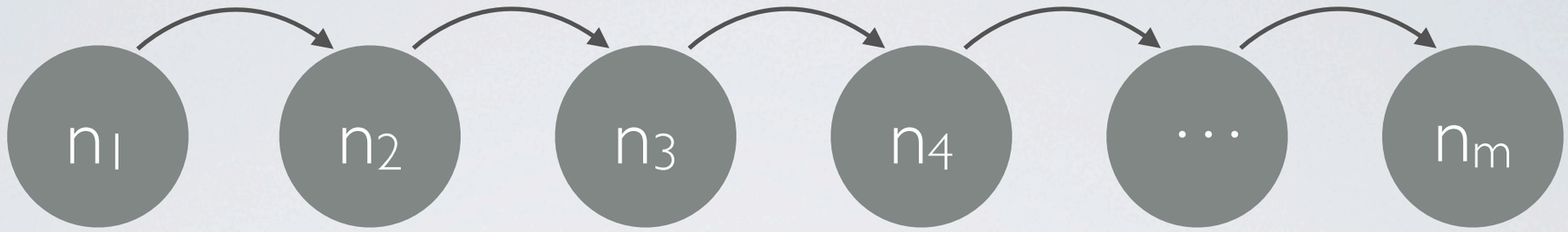
- A melody is represented as a sequence of symbols n_i
  - each symbol $\mathbf{n_i}$ may represent a note **or a relation**





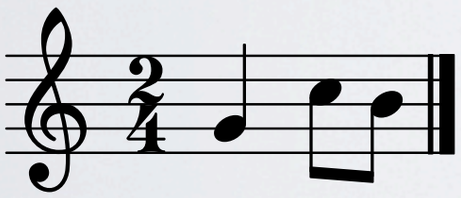
contour

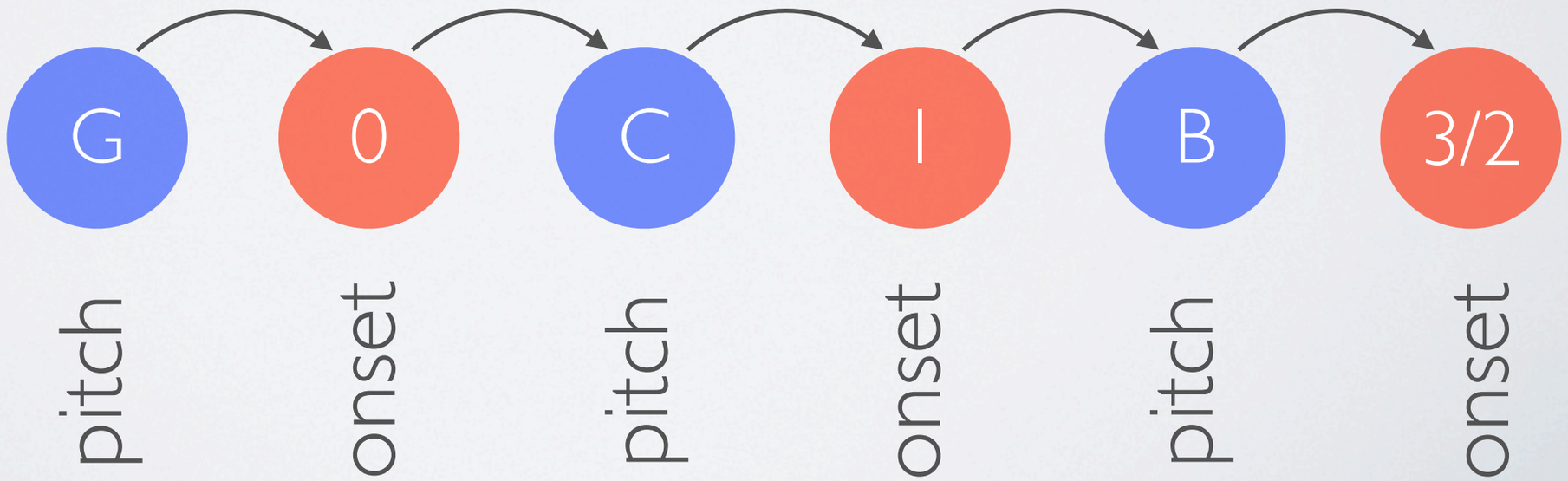
DECOUPLED STRINGS



- A melody is represented as a sequence of symbols n_i
  - each symbol $\mathbf{n_i}$ may represent a note or a relation

any of these **dimensions**





STRING LANGUAGES

STRING LANGUAGES

- **String** = sequence of **symbols**

STRING LANGUAGES

- **String** = sequence of **symbols**
- Each **symbol** belongs to an **alphabet** Σ

STRING LANGUAGES

- **String** = sequence of **symbols**
- Each **symbol** belongs to an **alphabet** Σ
- String = **sequence of** Σ

STRING LANGUAGES

- **String** = sequence of **symbols**
- Each **symbol** belongs to an **alphabet** Σ
- String = **sequence of** Σ
  - denoted by Σ^* (meaning: 0 or more symbols)

STRING LANGUAGES

- **String** = sequence of **symbols**
- Each **symbol** belongs to an **alphabet** Σ
- String = **sequence of** Σ
  - denoted by Σ^* (meaning: 0 or more symbols)
  - Σ^+ denotes 1 or more symbols

STRING LANGUAGES

STRING LANGUAGES

- Language = set of strings

STRING LANGUAGES

- Language = set of strings
- An **automaton** can **generate** and **recognize** this kind of language following states and transitions

STRING LANGUAGES

- Language = set of strings
- An **automaton** can **generate** and **recognize** this kind of language following states and transitions
- e.g. Let $\sum$ = {A, B, C},

STRING LANGUAGES

- Language = set of strings

- An **automaton** can **generate** and **recognize** this kind of language following states and transitions

- e.g. Let $\sum$ = {A, B, C},

  the language $\mathbf{AB^{+}C}$ is the **set of strings**

STRING LANGUAGES

- Language = set of strings
- An **automaton** can **generate** and **recognize** this kind of language following states and transitions
- e.g. Let $\sum$ = {A, B, C},

  the language **AB^+C** is the **set of strings**

starting with A

STRING LANGUAGES

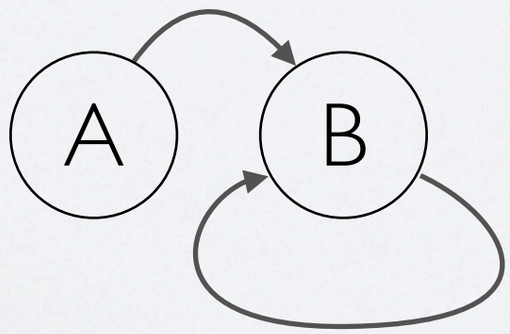
- Language = set of strings

- An **automaton** can **generate** and **recognize** this kind of language following states and transitions

- e.g. Let $\sum$ = {A, B, C},

the language **$AB^{+}C$** is the **set of strings**

starting with A containing 1 or more B



STRING LANGUAGES

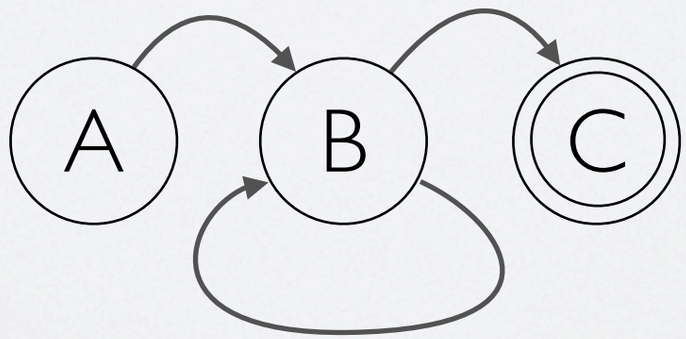
- Language = set of strings

- An **automaton** can **generate** and **recognize** this kind of language following states and transitions

- e.g. Let $\sum$ = {A, B, C},

the language **$AB^{+}C$** is the **set of strings**

starting with A containing 1 or more B ending with C



STRING LANGUAGES

STRING LANGUAGES

- A **music language** can be built on strings

STRING LANGUAGES

- A **music language** can be built on strings
  - if **only notes** are represented,

STRING LANGUAGES

- A **music language** can be built on strings
  - if **only notes** are represented,
  - alphabet $\sum$ represents a combination pitch ($\sum_p$) and rhythm ($\sum_r$)

$$\sum = \sum_p \otimes \sum_r$$

STRING LANGUAGES

- A **music language** can be built on strings
  - if **only notes** are represented,
    - alphabet $\sum$ represents a combination pitch ($\sum_p$) and rhythm ($\sum_r$)

$$\sum = \sum_p \otimes \sum_r$$

    - Coupled $\sum = \sum_p \times \sum_r$

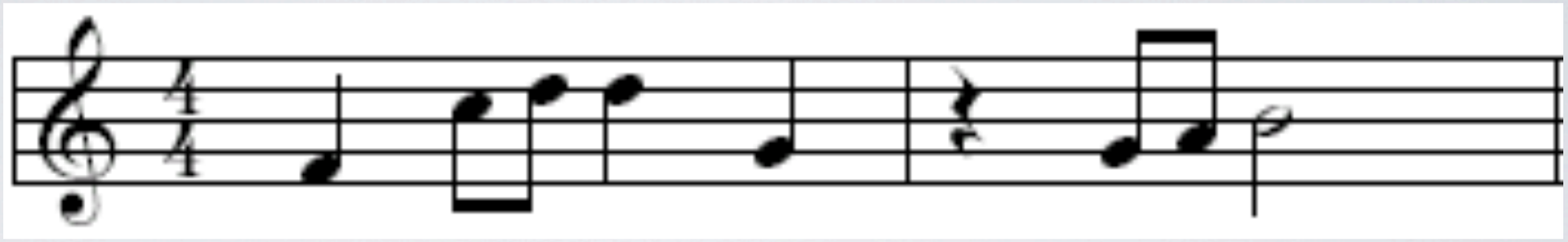
STRING LANGUAGES

- A **music language** can be built on strings
  - if **only notes** are represented,
    - alphabet Σ represents a combination pitch (Σ_p) and rhythm (Σ_r)

$$\Sigma = \Sigma_p \otimes \Sigma_r$$

      - Coupled $\Sigma = \Sigma_p \times \Sigma_r$
      - Decoupled $\Sigma = \Sigma_p \cup \Sigma_r$

PITCH ALPHABETS Σ_p



Absolute magnitudes

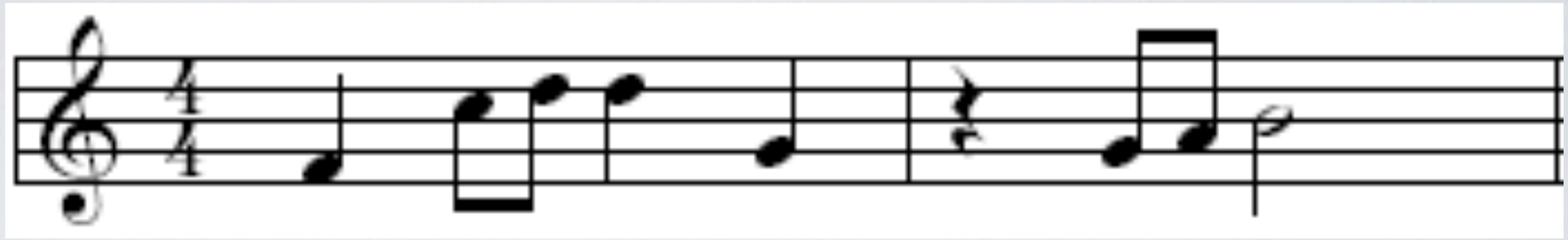
Scientific pitch:	F_4	C_5	D_5	D_5	G_4	R	G_4	A_4	B_4	$\lvert\Sigma_p\rvert = 89$
Pitch class:	5	0	2	2	7	R	7	9	11	$\lvert\Sigma_p\rvert = 13$

Relative magnitudes

Contour:	x	+	+	=	-		=	+	+	$\lvert\Sigma_p\rvert = 3$
High Def. Contour	x	+2	+1	0	-2		0	+1	+1	$\lvert\Sigma_p\rvert = 5$
Intervals:	x	+7	+2	0	-7		0	+2	+2	$\lvert\Sigma_p\rvert = 49$
Relative to tonic:	5	<u>0</u>	2	2	7	R	7	9	11	$\lvert\Sigma_p\rvert = 13$

etc.....

RHYTHM ALPHABETS $\sum_r$



Absolute magnitudes

- Durations: n c c n n n c c b
- *Inter-onset intervals* (IOIs): 1 1/2 1/2 1 2 1/2 1/2 (2)

$$IOI_n = o_{n+1} - o_n$$

Relative magnitudes

- Duration relative to previous: x 1/2 1 2 1 1 1/2 1 4
- Duration relative to most frequent: 2 1 1 2 2 2 1 1 4
- Relation between IOIs (IORs): 1/2 1 2 2 1/4 1 4 x

$$IOR_n = IOI_{n+1}/IOI_n$$

- Rhythmic contour: x - = + = = - = +

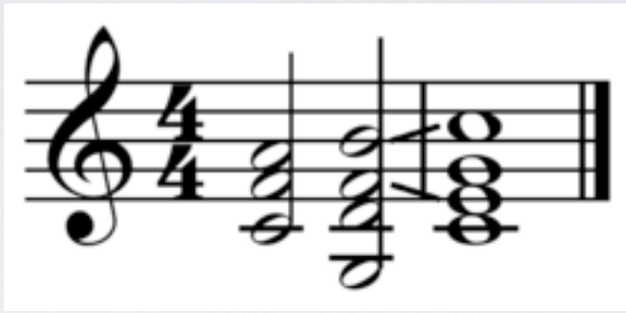
etc.....

MONOPHONIC

POLYPHONIC

STRINGS - POLYPHONY

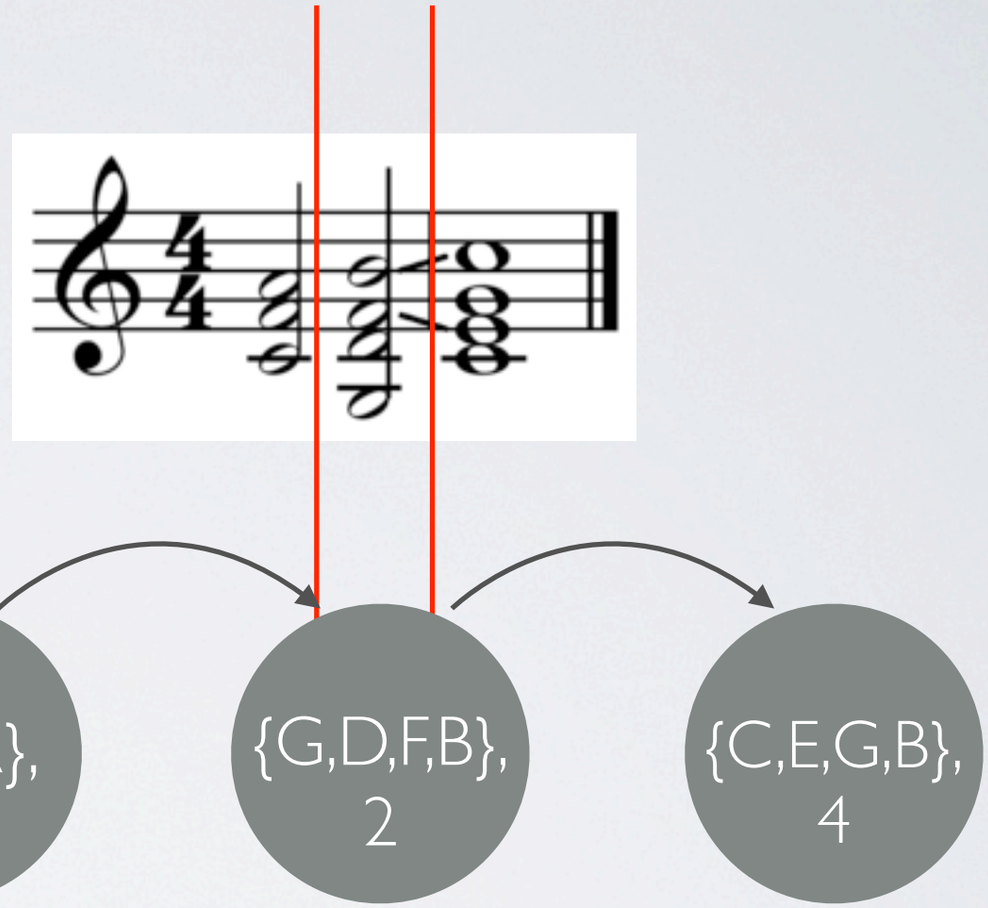
- **Chord** sequence approach:



Onset group based

STRINGS - POLYPHONY

- **Chord** sequence approach:



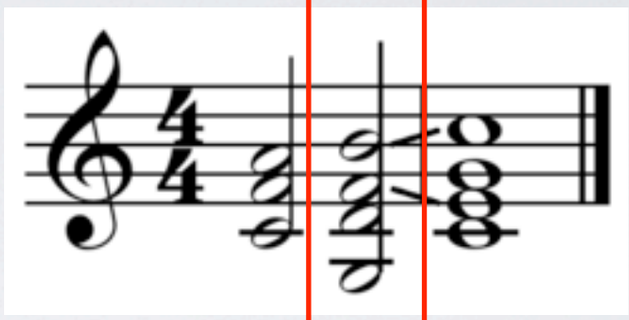
Onset group based

requires segmentation

$$\Sigma = (\Sigma_r \times \Sigma^{+}_p)$$

STRINGS - POLYPHONY

- **Chord** sequence approach:



Onset group based

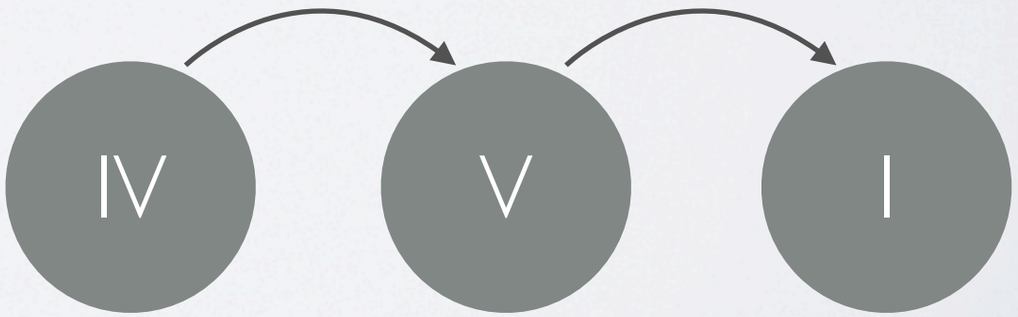
requires segmentation

$$\Sigma = (\Sigma_r \times \Sigma^{+}_p)$$

Named chord:

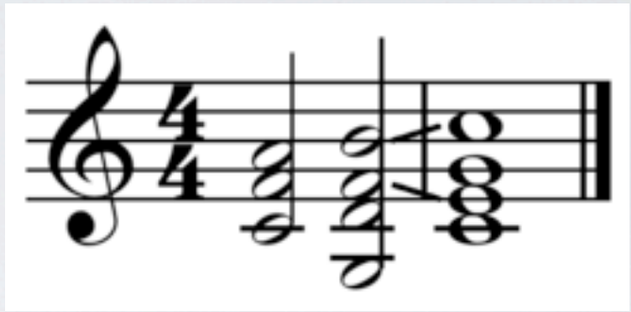
requires chord analysis

Σ = *set of chord names*



STRINGS - POLYPHONY

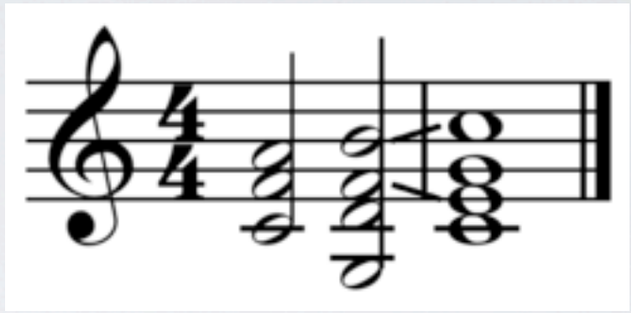
- **Individual note** approach



Not interleaved:
requires voice analysis, deal with divisi, etc...

STRINGS - POLYPHONY

- **Individual note** approach

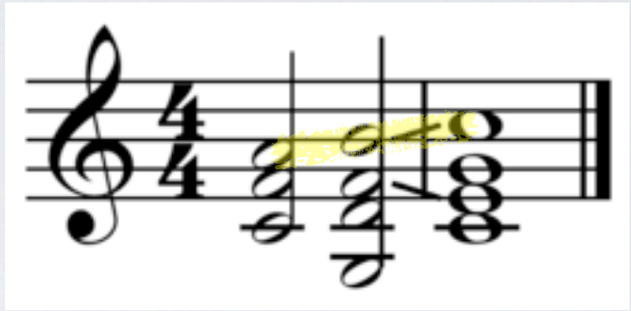


Not interleaved:
requires voice analysis, deal with divisi, etc...

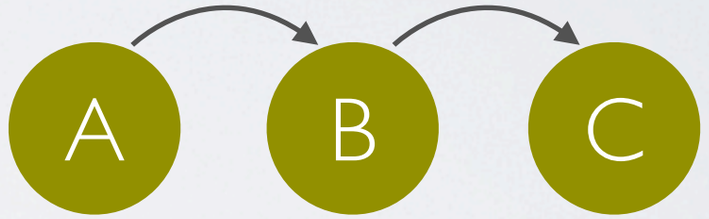
sequential arrangement of all monophonic lines

STRINGS - POLYPHONY

- **Individual note** approach



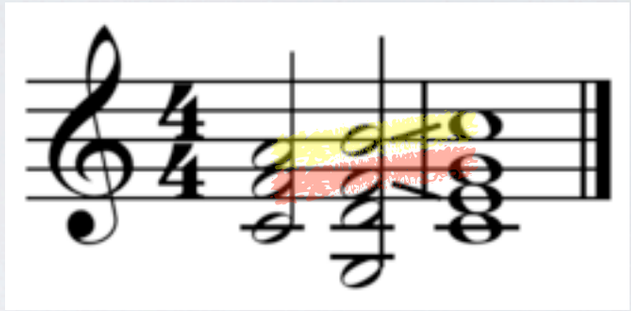
Not interleaved:
requires voice analysis, deal with divisi, etc...



sequential arrangement of all monophonic lines

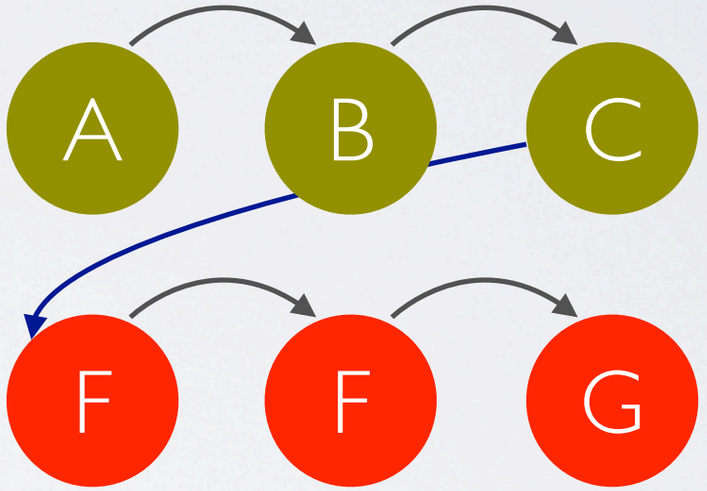
STRINGS - POLYPHONY

- **Individual note** approach



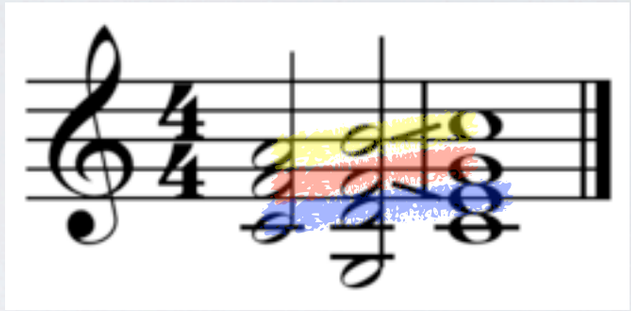
Not interleaved:
requires voice analysis, deal with divisi, etc...

sequential arrangement of all monophonic lines



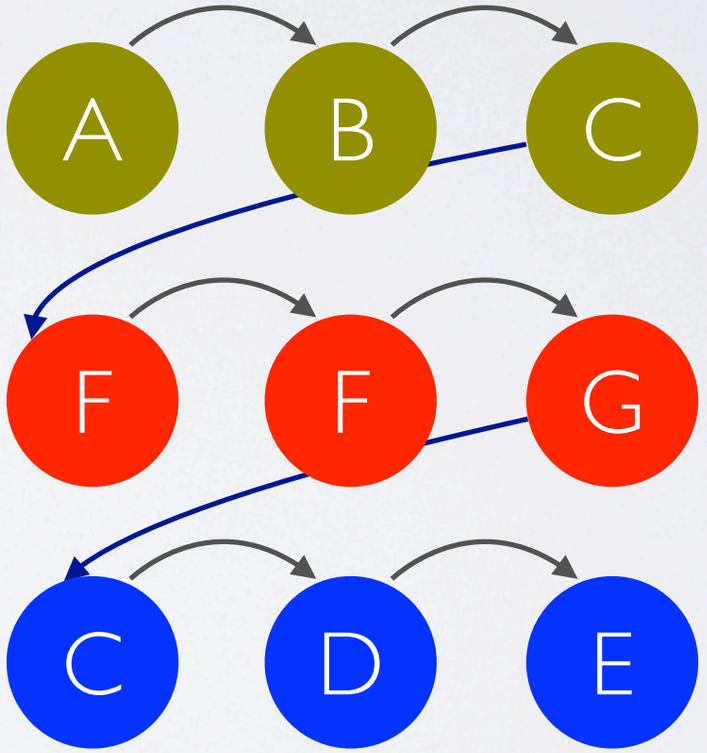
STRINGS - POLYPHONY

- **Individual note** approach



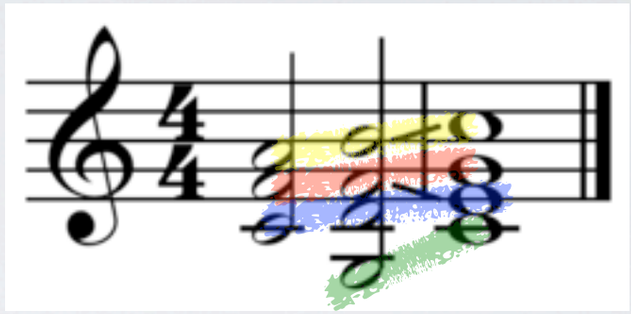
Not interleaved:
requires voice analysis, deal with divisi, etc...

sequential arrangement of all monophonic lines



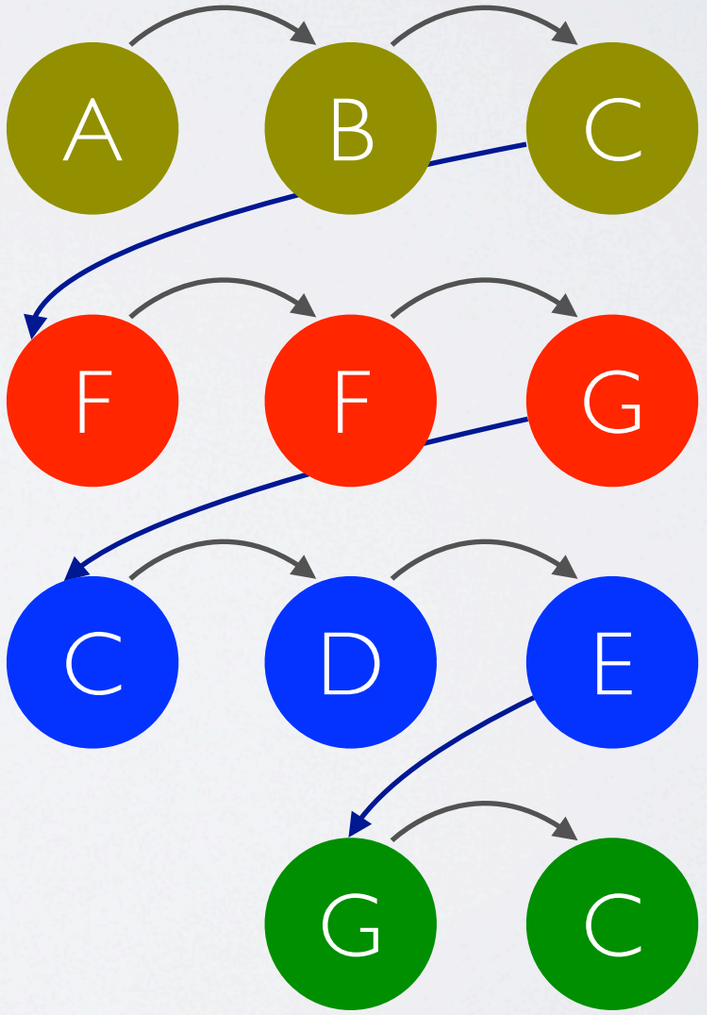
STRINGS - POLYPHONY

- **Individual note** approach



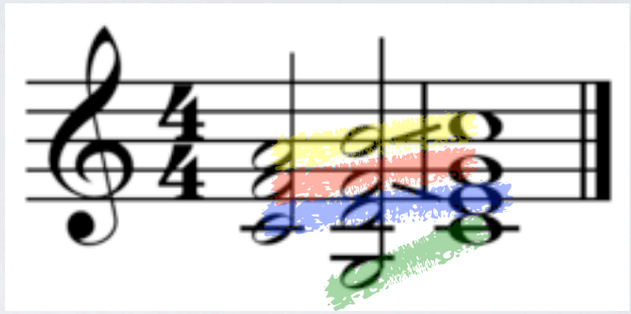
Not interleaved:
requires voice analysis, deal with divisi, etc...

sequential arrangement of all monophonic lines



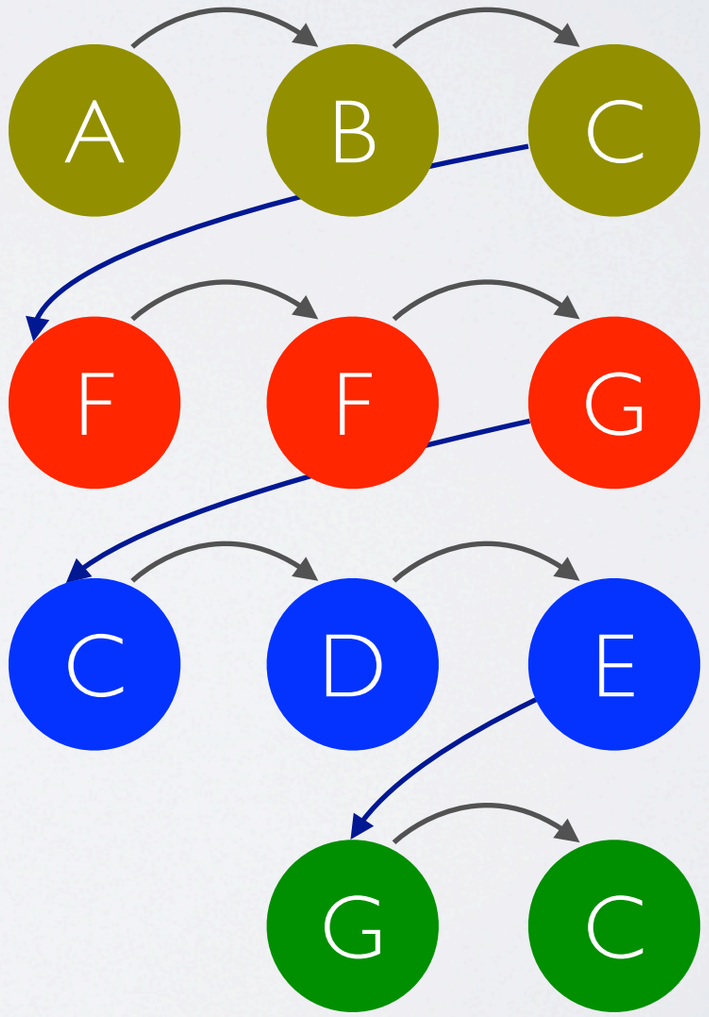
STRINGS - POLYPHONY

- **Individual note** approach



Not interleaved:
requires voice analysis, deal with divisi, etc...

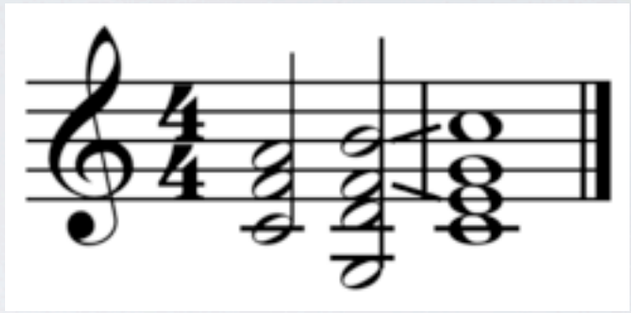
sequential arrangement of all monophonic lines



$\sum$ = note alphabet + voice

STRINGS - POLYPHONY

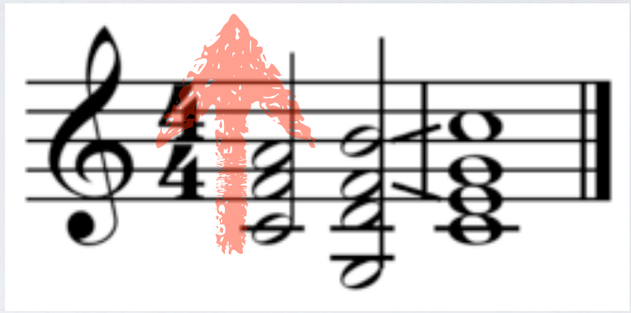
- **Individual note** approach



Interleaved

STRINGS - POLYPHONY

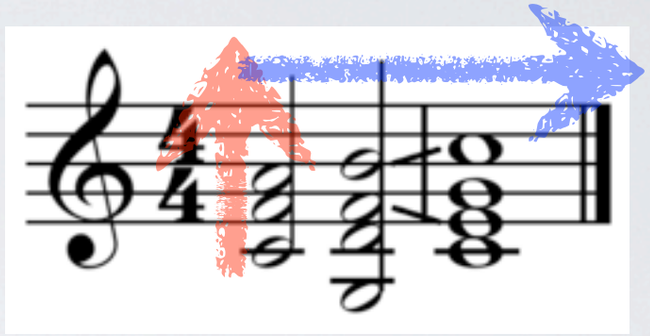
- **Individual note** approach



Interleaved

STRINGS - POLYPHONY

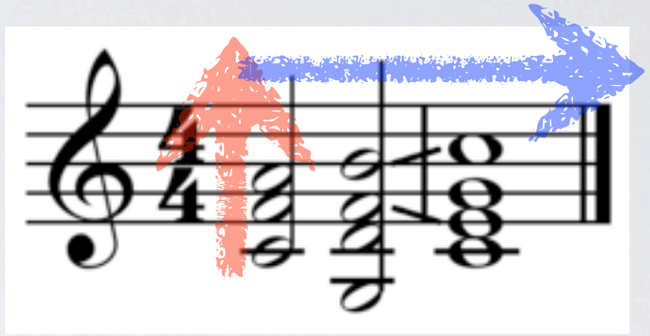
- **Individual note** approach



Interleaved

STRINGS - POLYPHONY

- **Individual note** approach

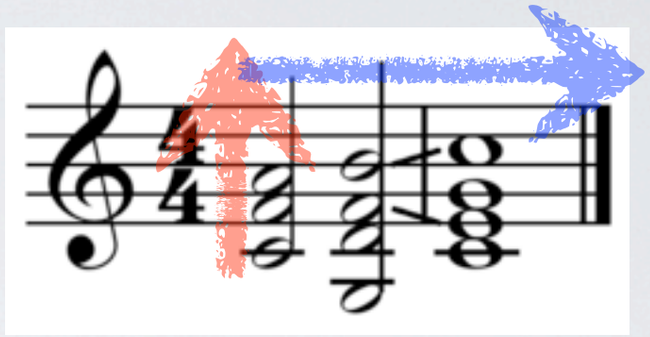


Interleaved

notes sorted lexicographically (e.g. first order by onset, them by pitch)

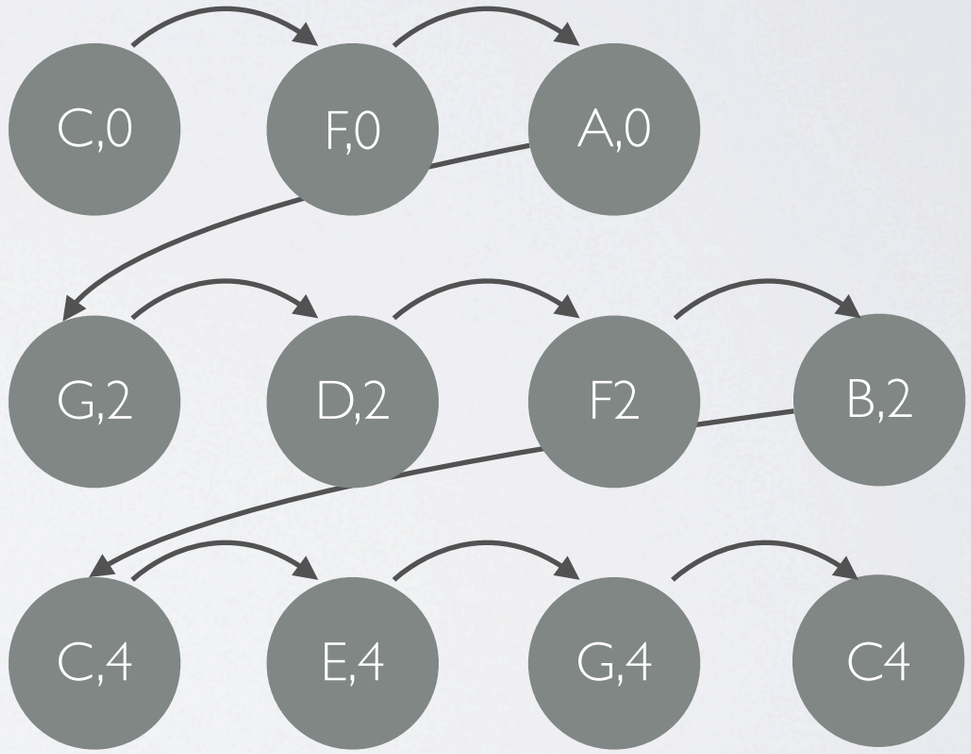
STRINGS - POLYPHONY

- **Individual note** approach



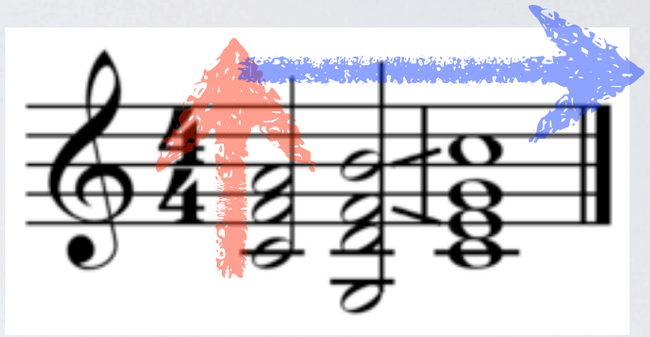
Interleaved

notes sorted lexicographically (e.g. first order by onset, them by pitch)

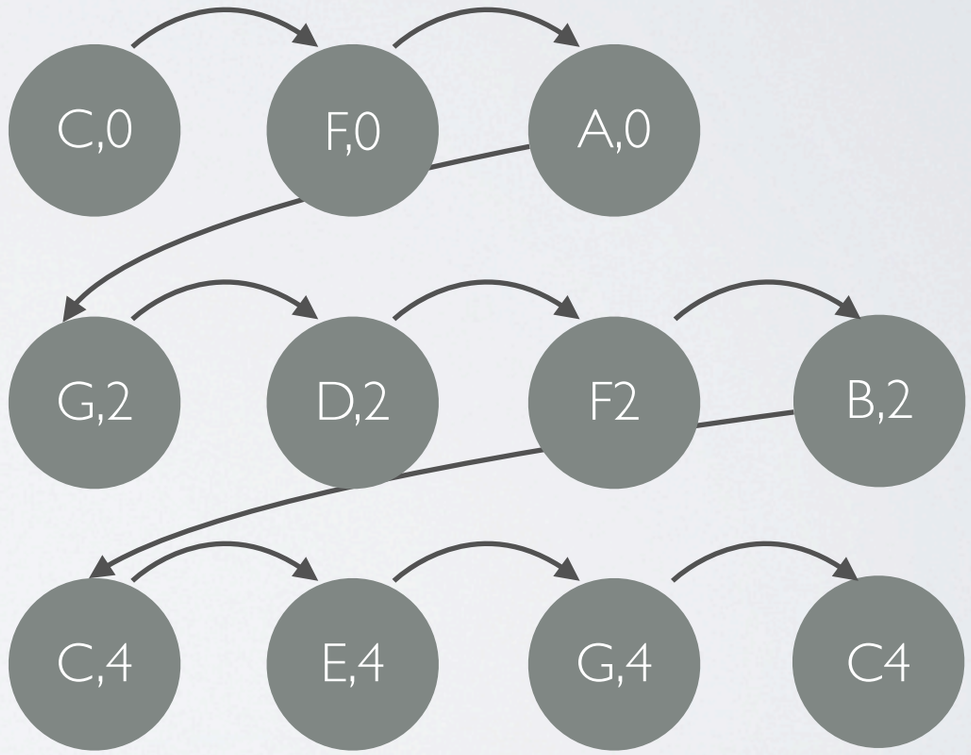


STRINGS - POLYPHONY

- **Individual note** approach



Interleaved



notes sorted lexicographically (e.g. first order by onset, them by pitch)

$$\sum = \sum_r \times \sum_p$$

Evolve Strings to Trees

Evolve Strings to Trees

- Many possible note representations (many alphabets Σ)

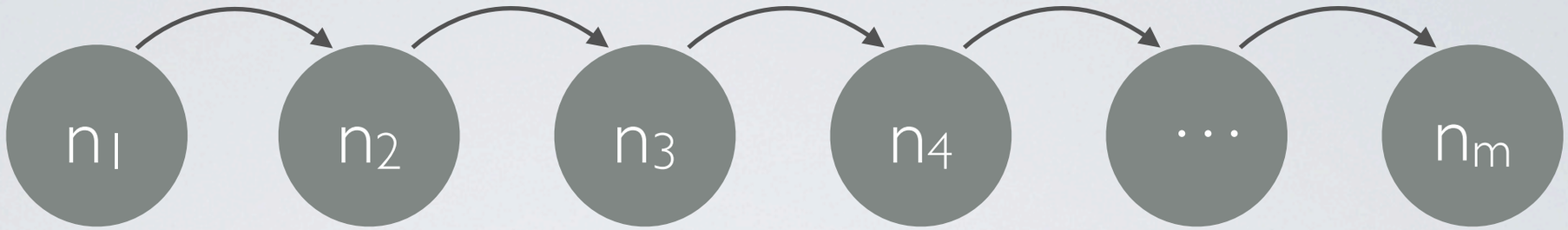
Evolve Strings to Trees

- Many possible note representations (many alphabets Σ)

- But...

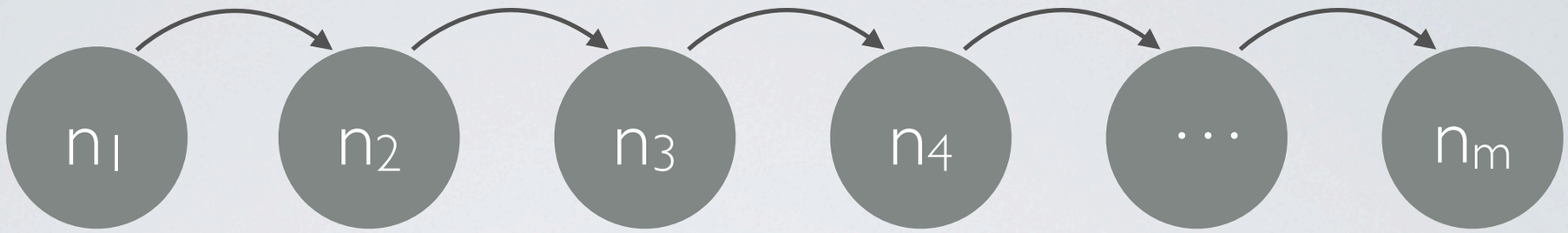
  difficult to represent structure

String



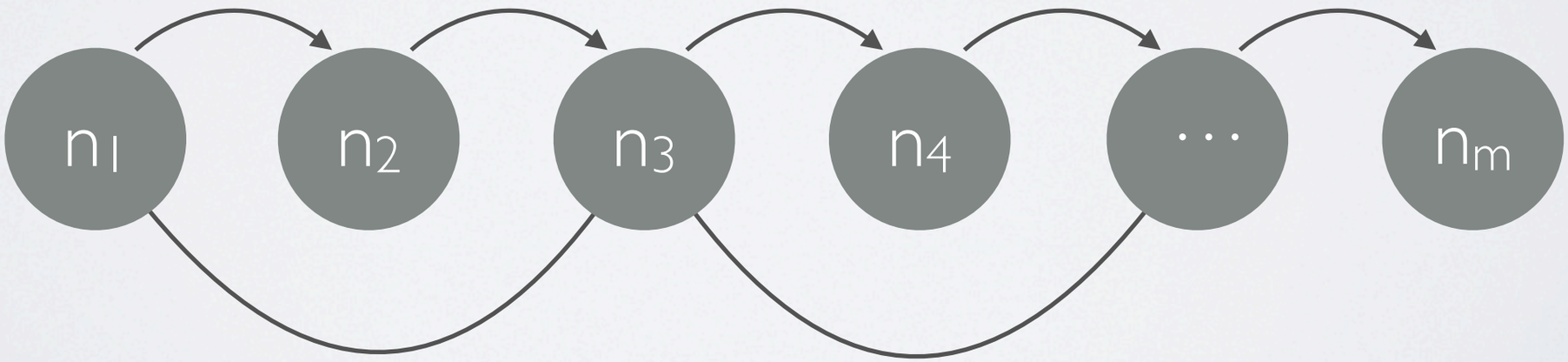
In strings
each note /symbol is linked to | from
just other note/symbol

String



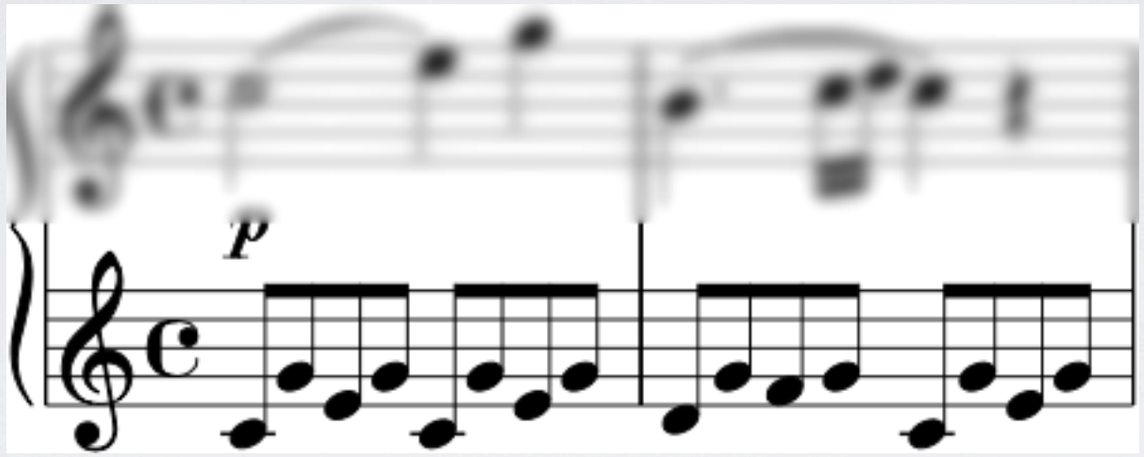
In strings
each note /symbol is linked to | from
just other note/symbol

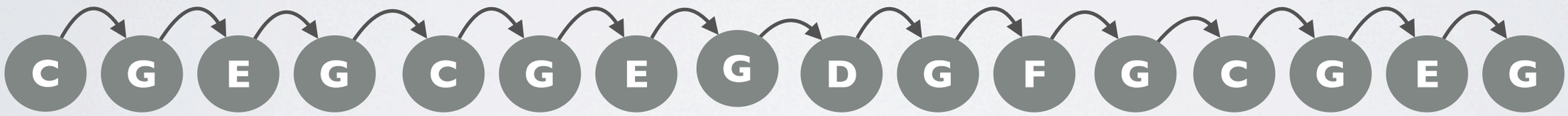
If some structure is required? e.g.



STRUCTURE

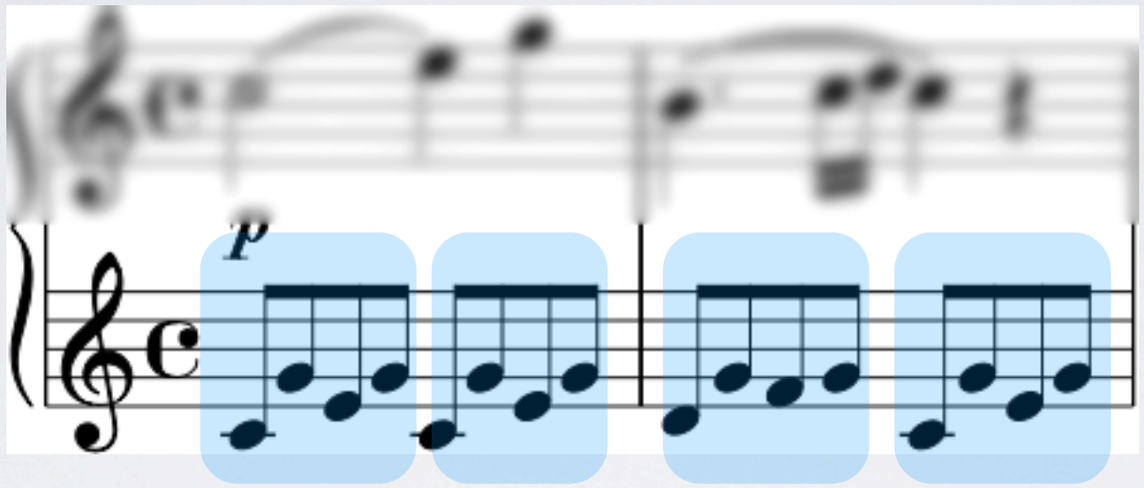
- e.g. "Alberti Bass" like piano left hand accompaniment

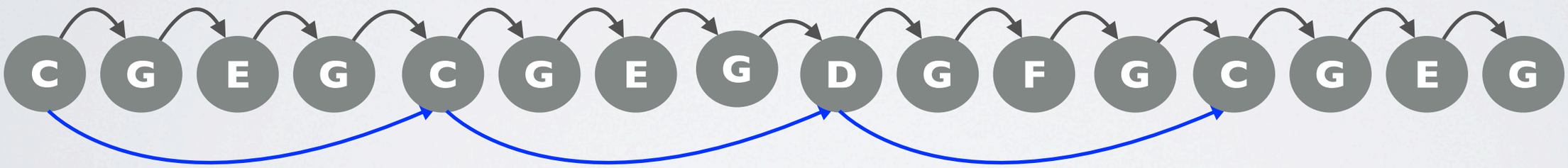




STRUCTURE

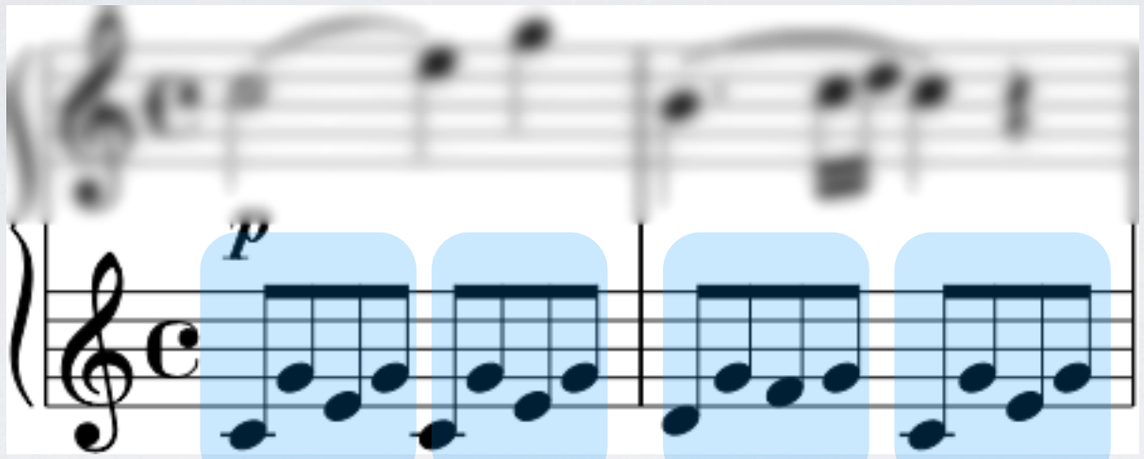
- e.g. "Alberti Bass" like piano left hand accompaniment

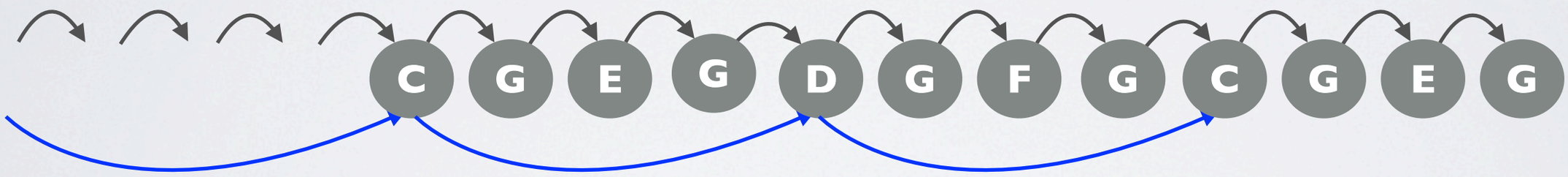


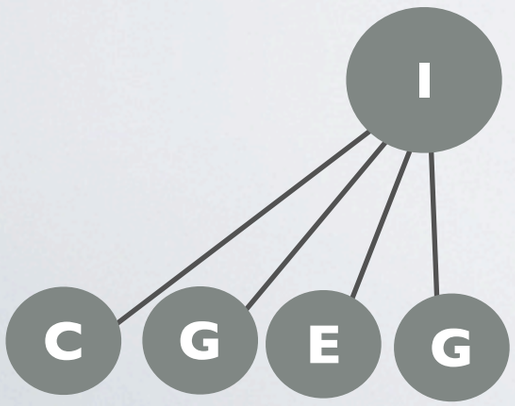


STRUCTURE

- e.g. "Alberti Bass" like piano left hand accompaniment

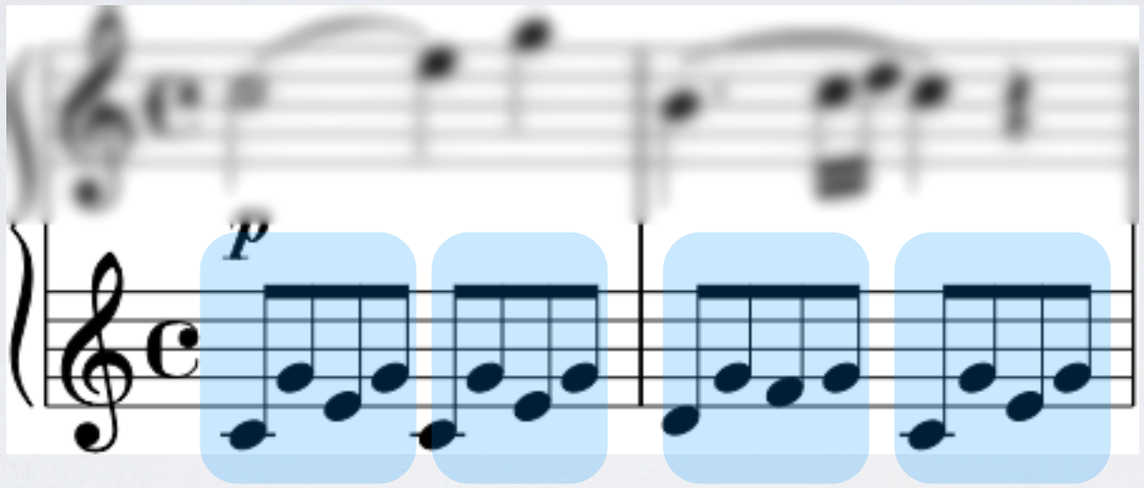


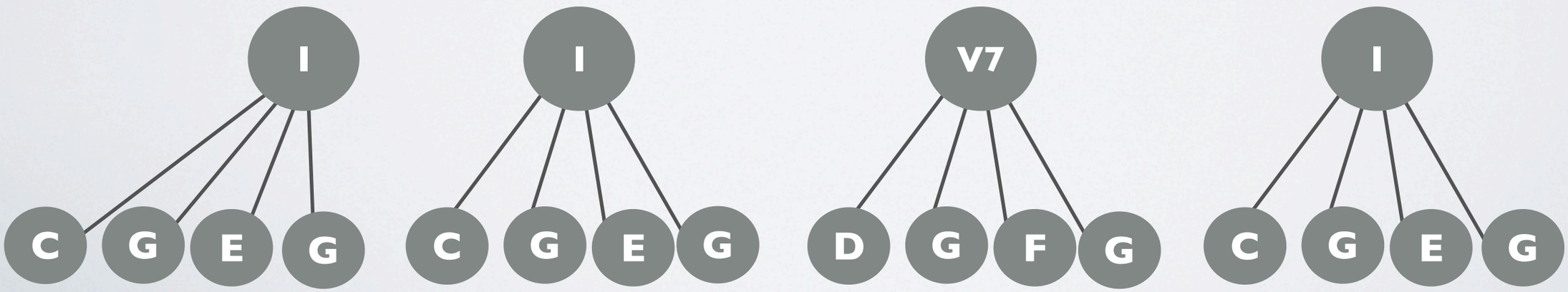




STRUCTURE

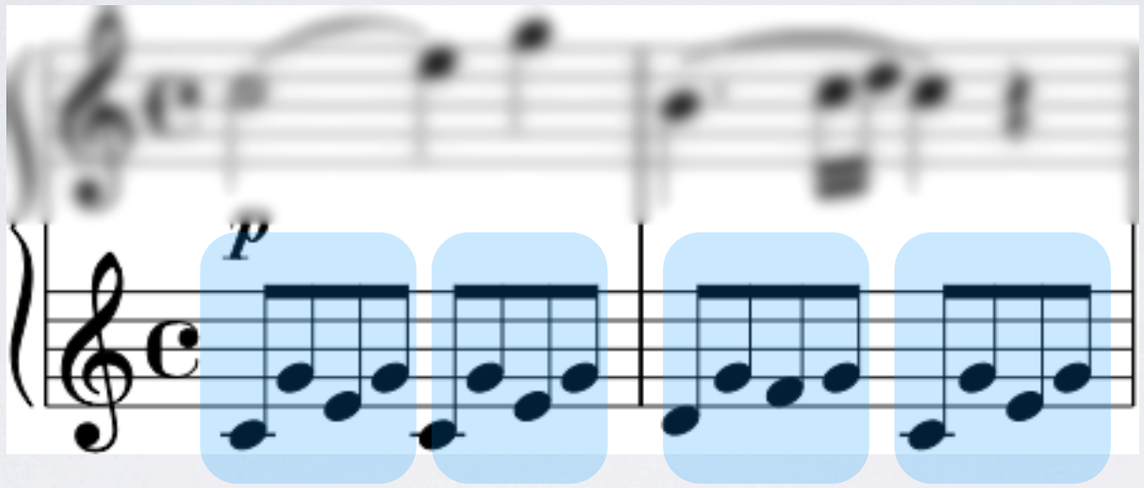
- e.g. "Alberti Bass" like piano left hand accompaniment

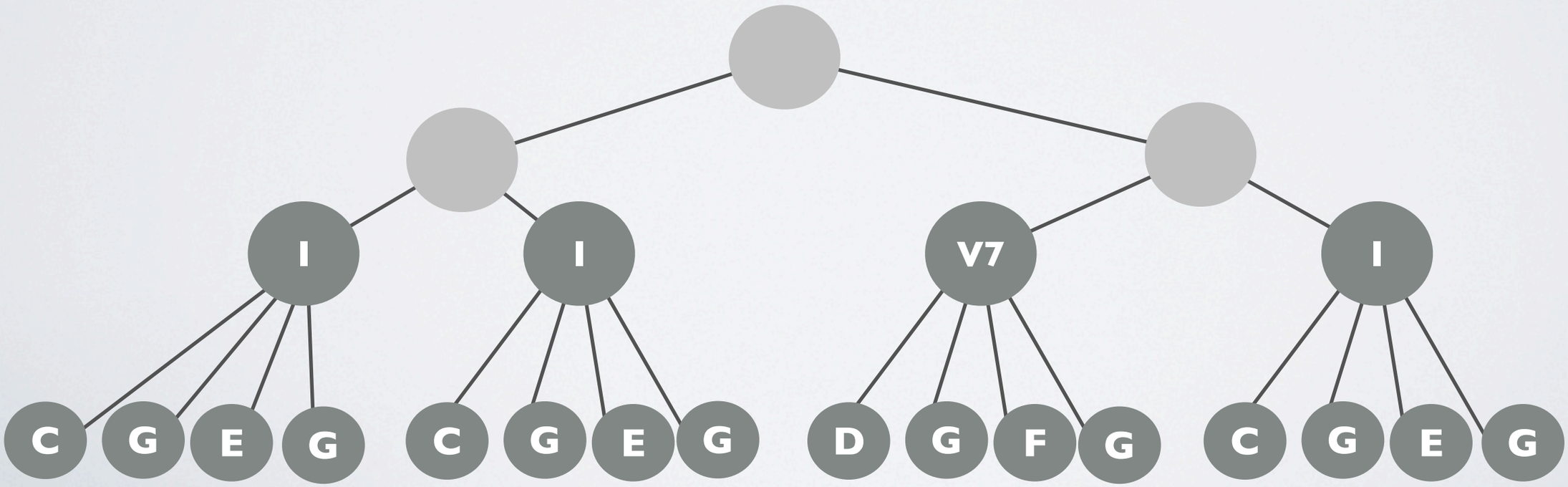




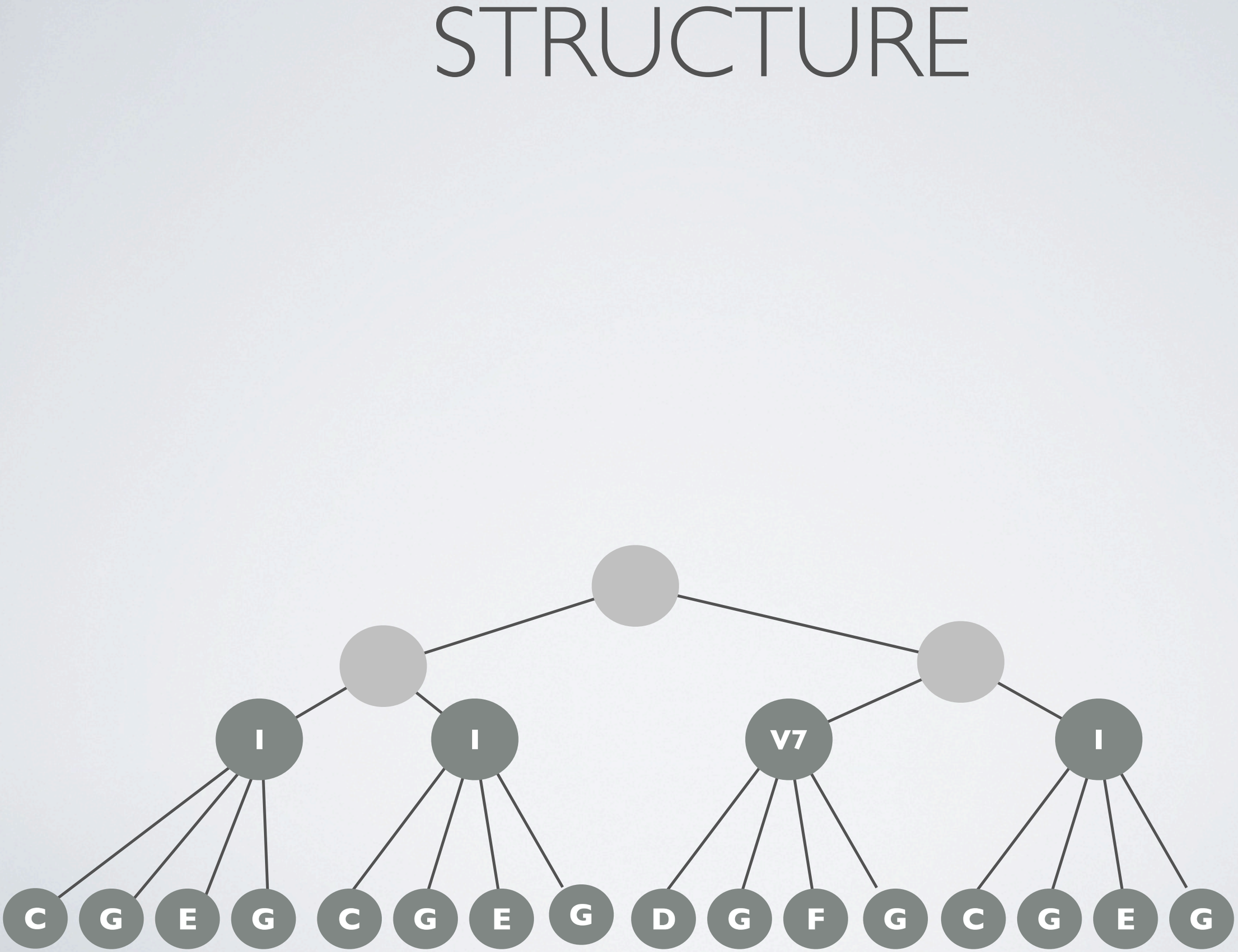
STRUCTURE

- e.g. "Alberti Bass" like piano left hand accompaniment





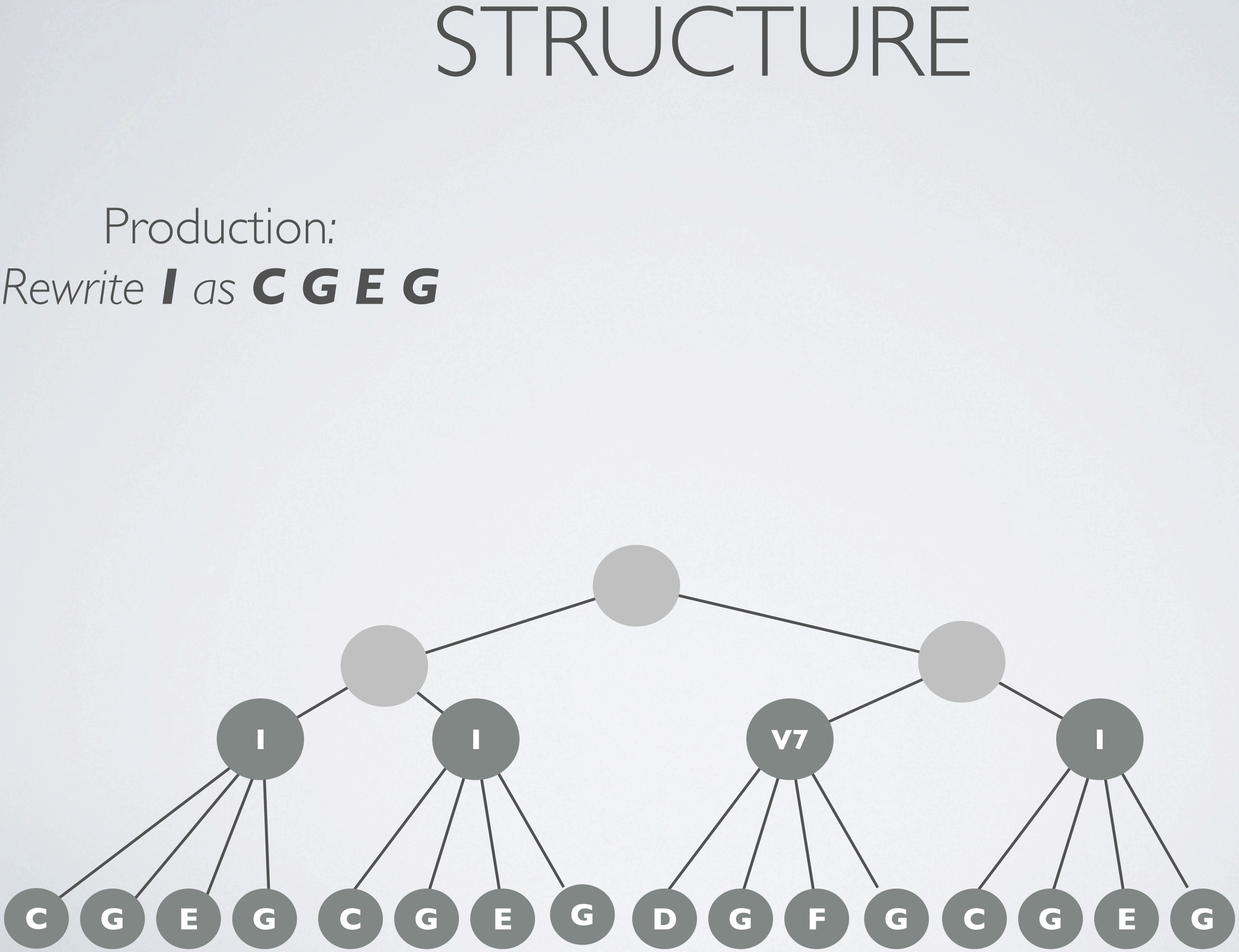
STRUCTURE



STRUCTURE

Production:

Rewrite ***I*** *as* ***C G E G***

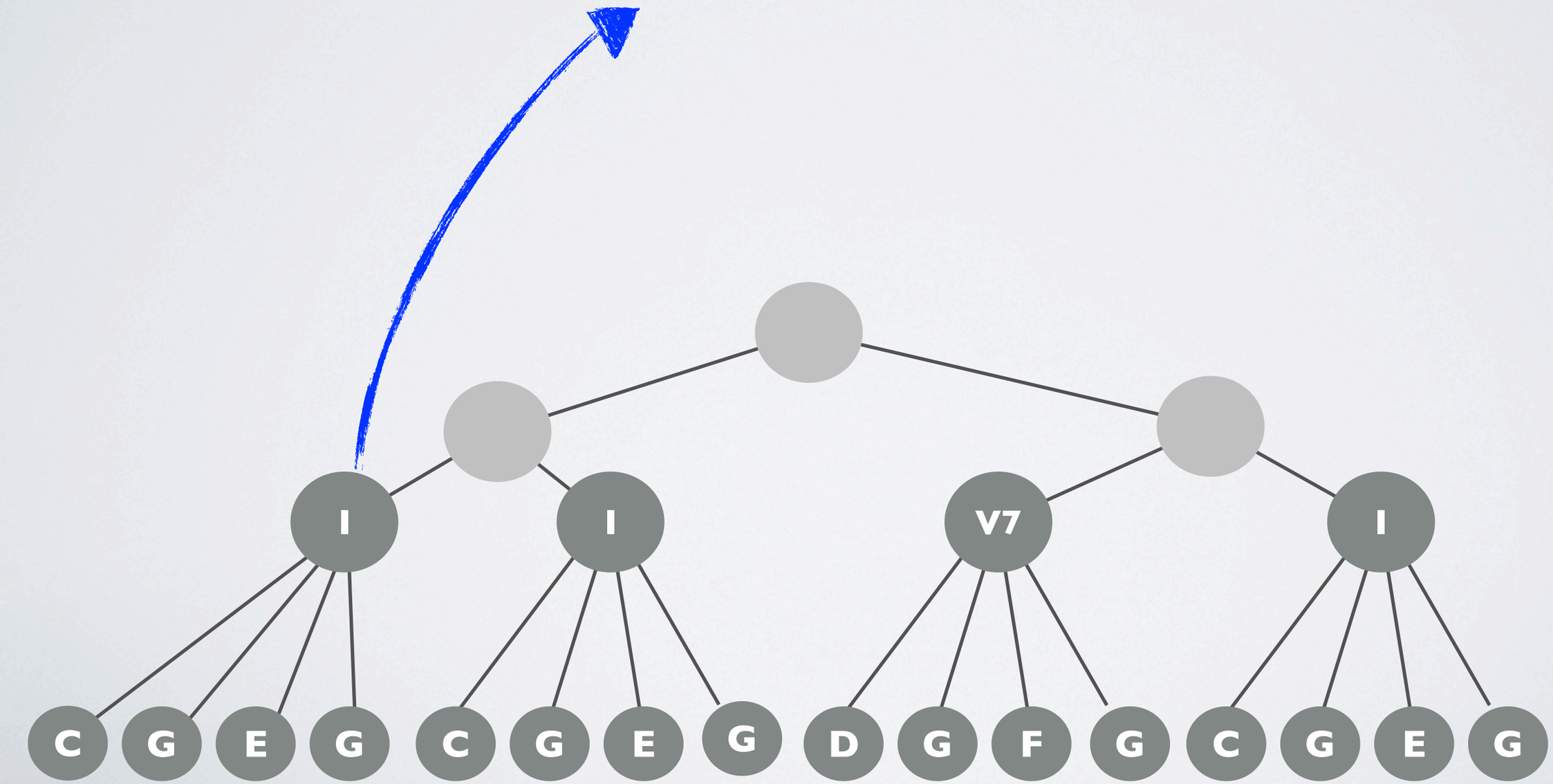


STRUCTURE

Production:

*Rewrite **I** as **C G E G***

I →C G E G

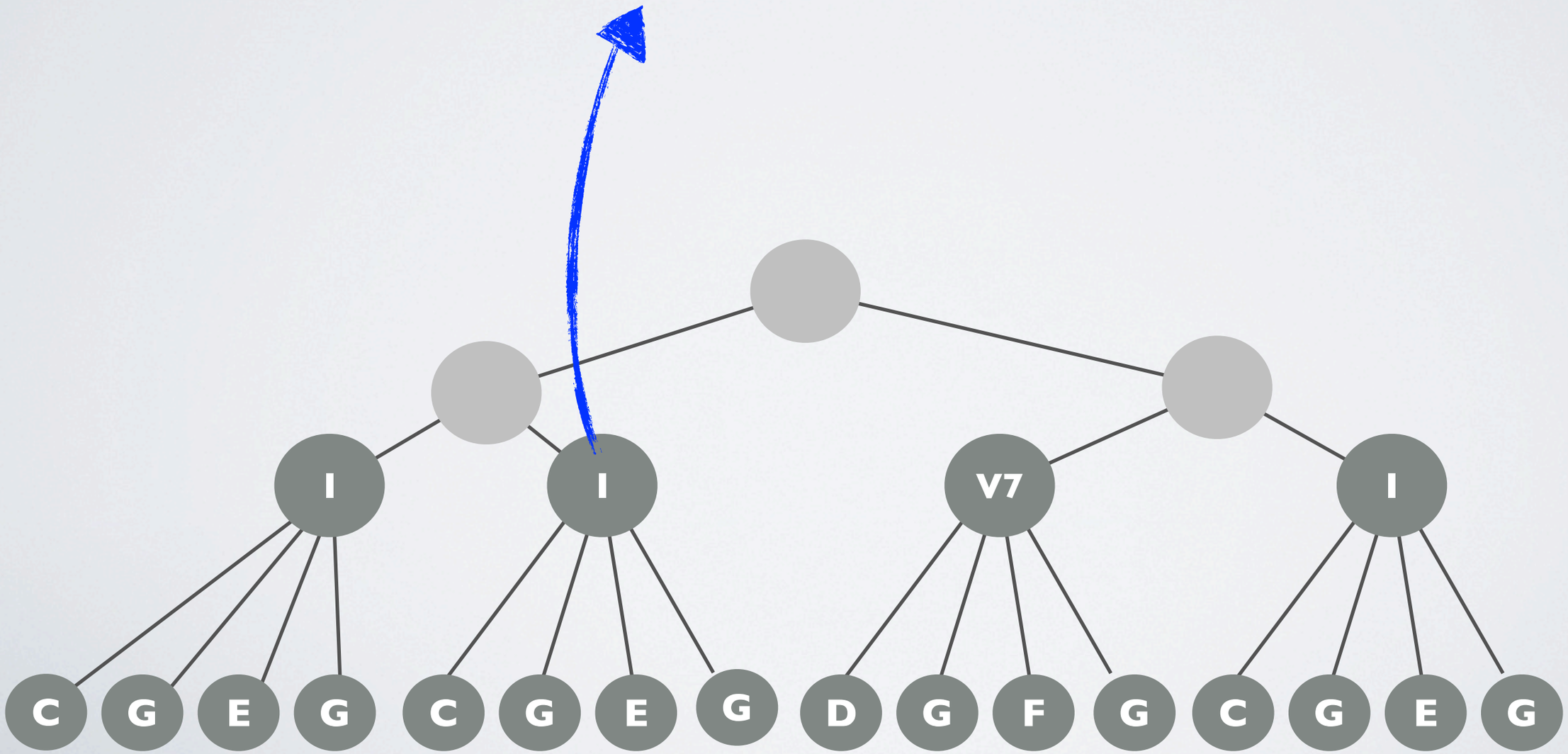


STRUCTURE

Production:

*Rewrite **I** as **C G E G***

I → C G E G



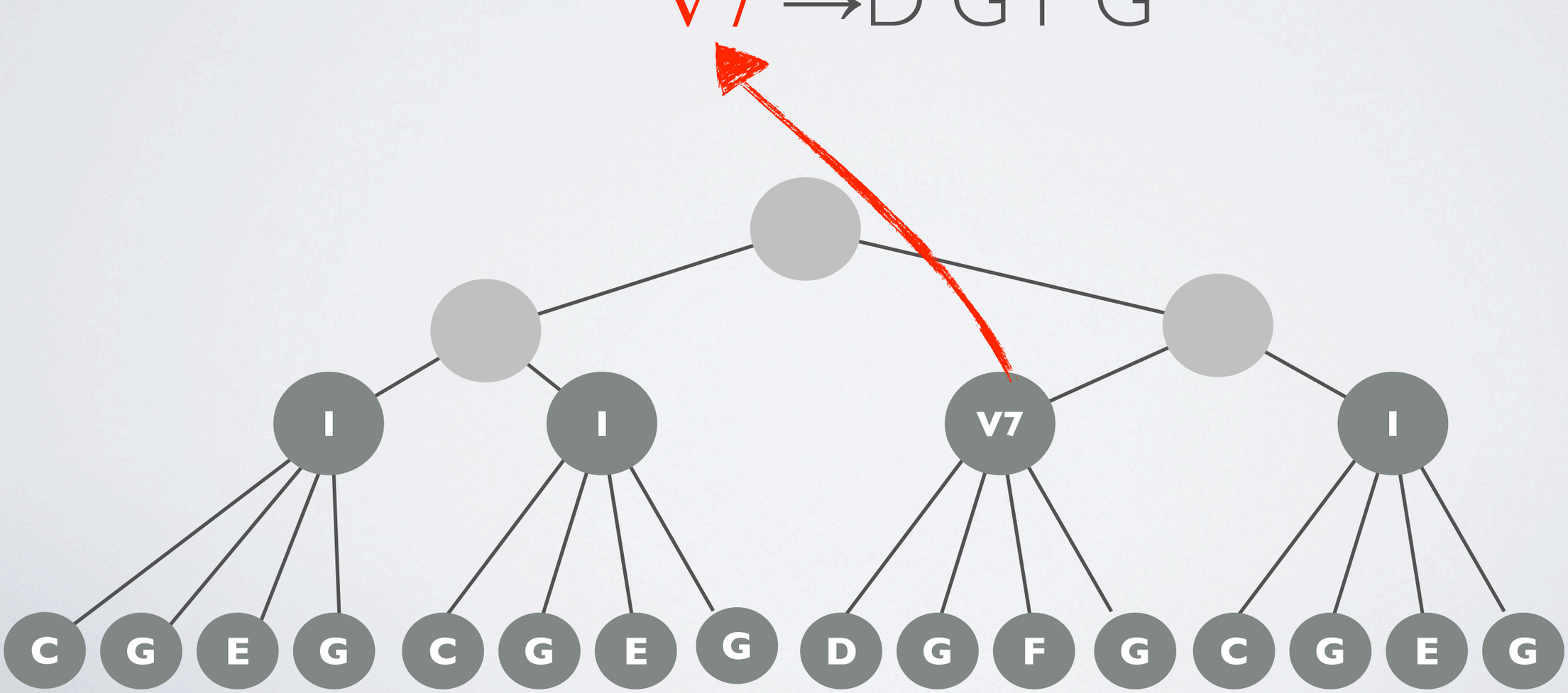
STRUCTURE

Production:

*Rewrite **I** as **C G E G***

I →C G E G

V7 →D G F G



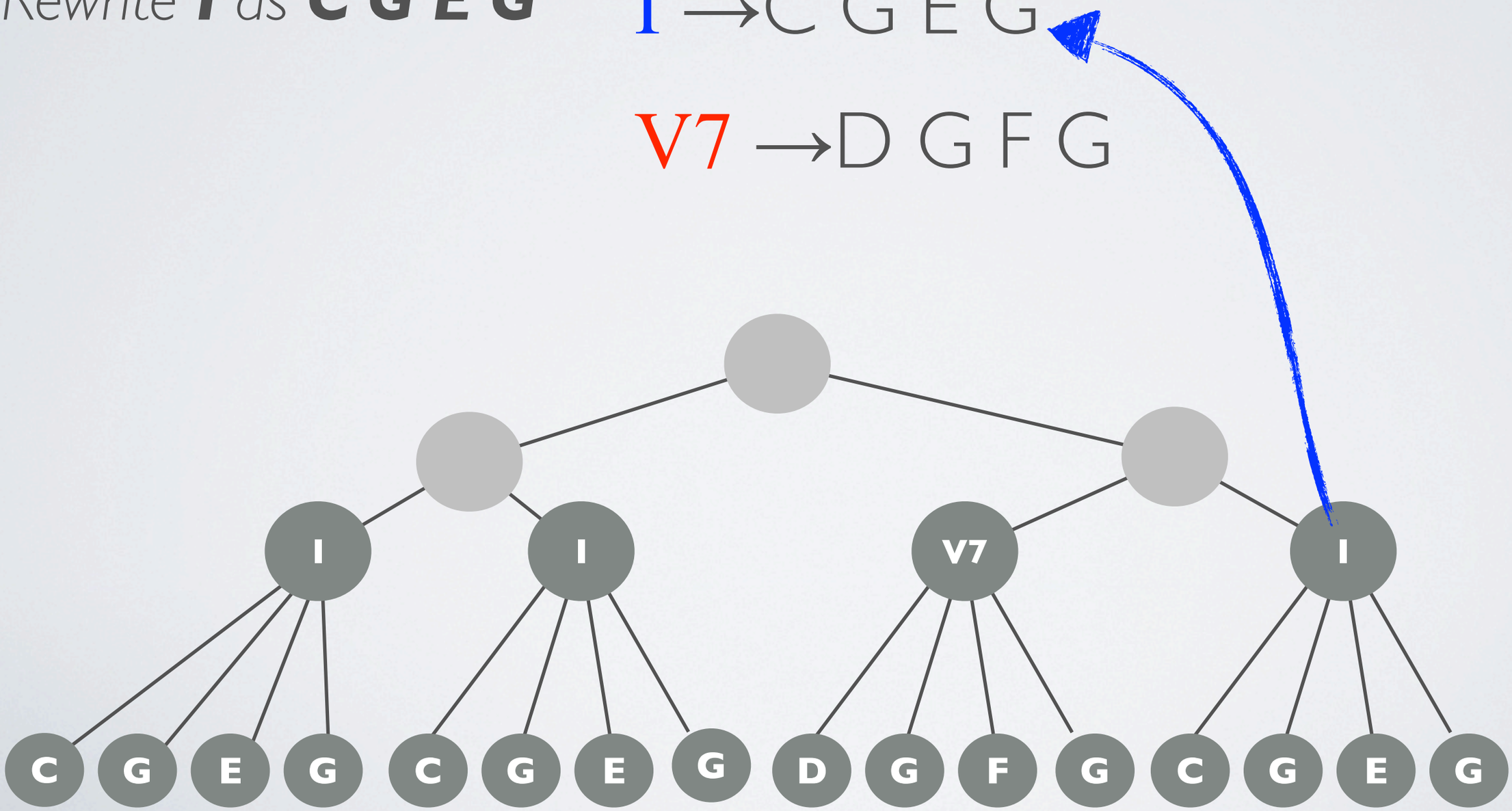
STRUCTURE

Production:

*Rewrite **I** as **C G E G***

I →C G E G

V7 →D G F G



STRUCTURE

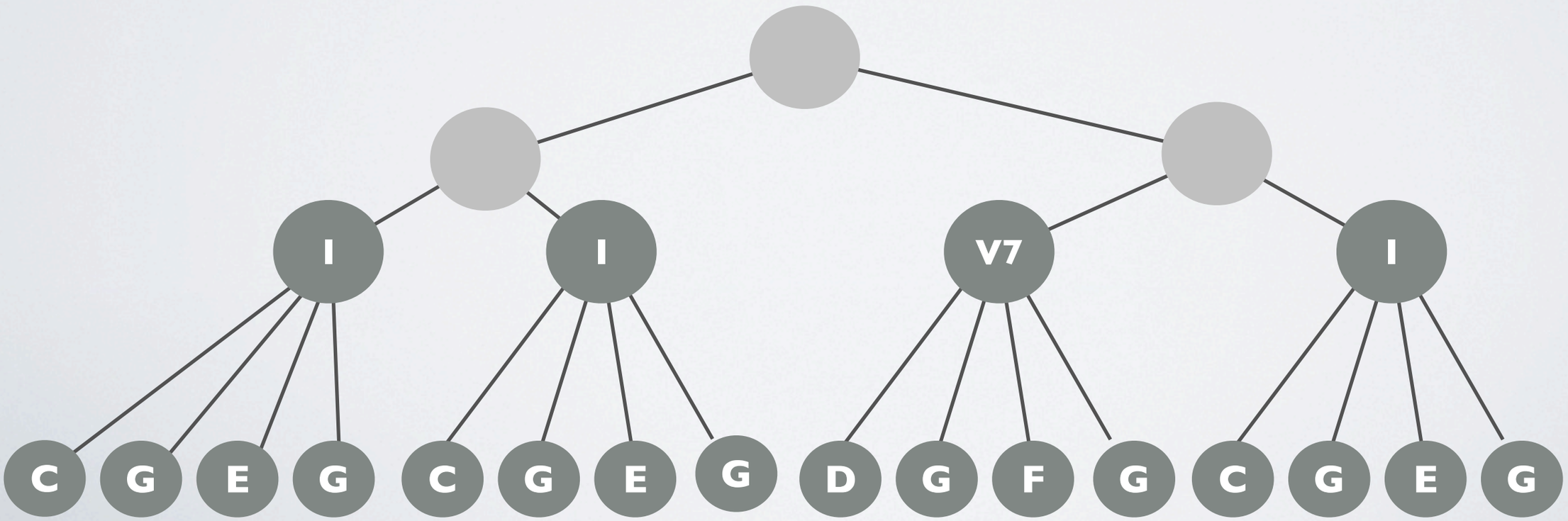
Production:

*Rewrite **I** as **C G E G***

Melody → I I V7 I

I →C G E G

V7 →D G F G



STRUCTURE

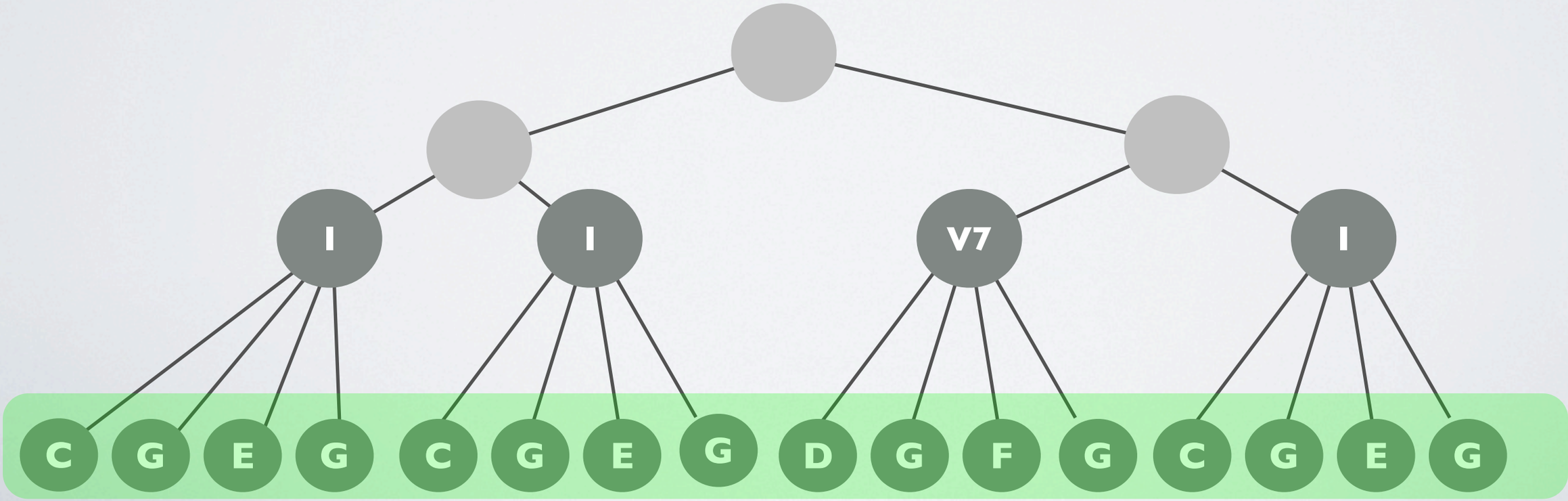
Production:
*Rewrite **I** as **C G E G***

Melody → I I V7 I

I → C G E G

V7 → D G F G

Terminals



STRUCTURE

Production:

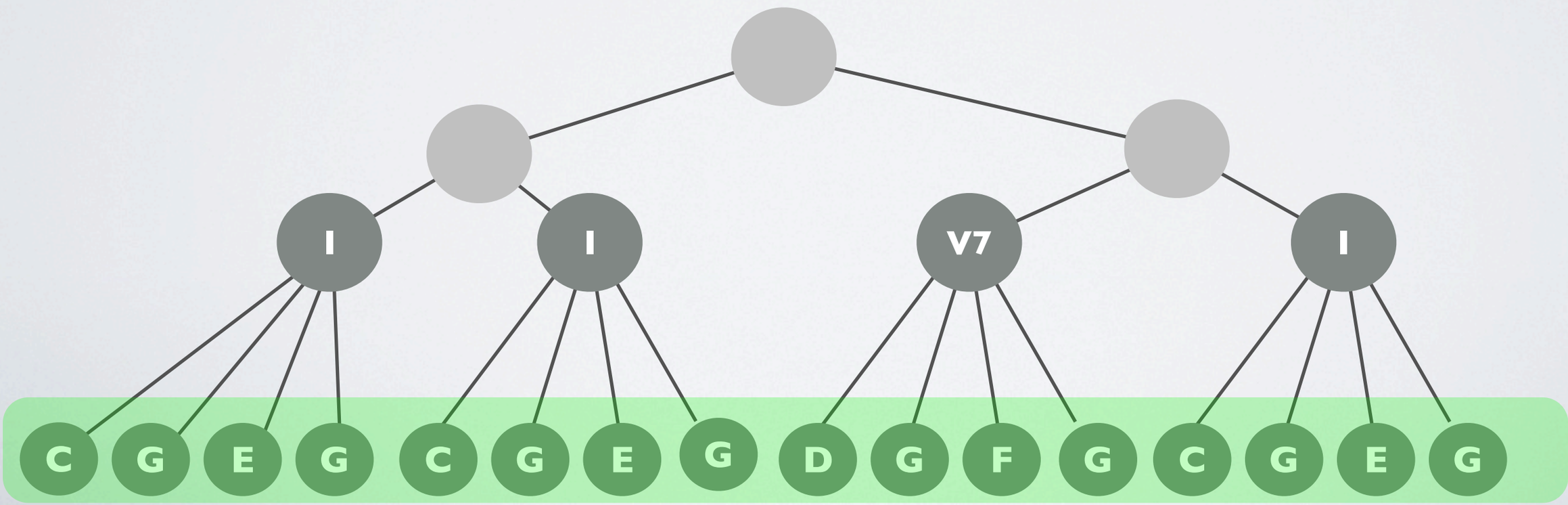
*Rewrite **I** as **C G E G***

Melody → I I V7 I

I → C G E G

V7 → D G F G

Terminals ∈ ∑



GRAMMAR

Production:

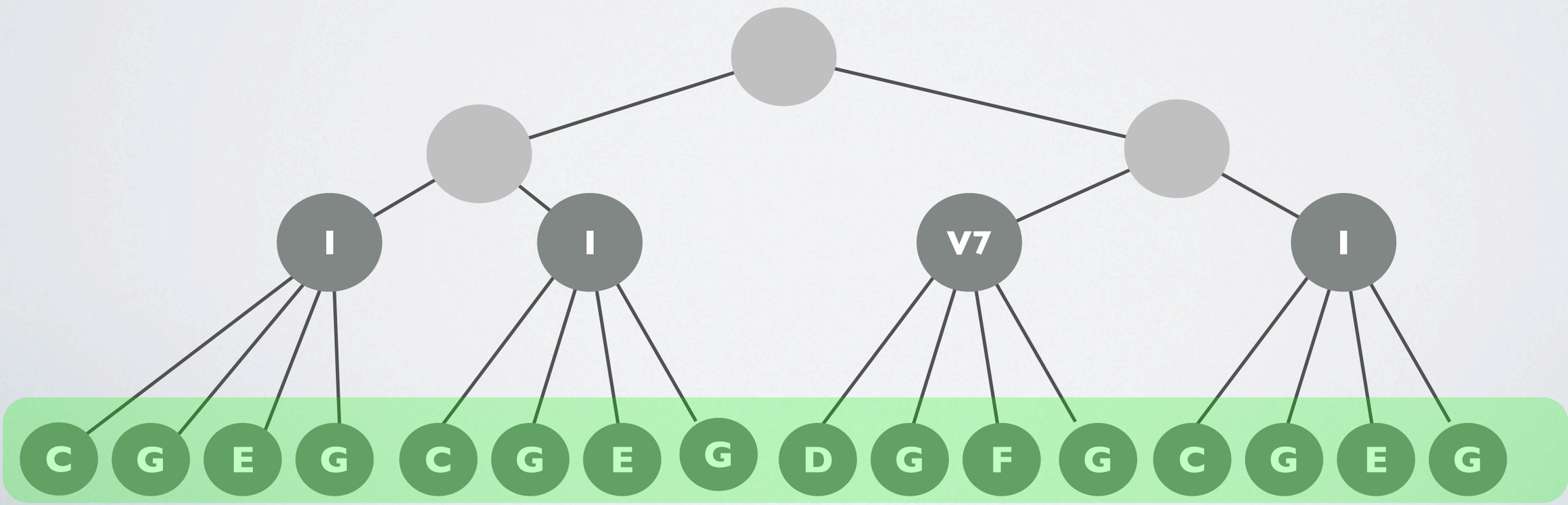
*Rewrite **I** as **C G E G***

Melody → I I V7 I

I → C G E G

V7 → D G F G

Terminals ∈ ∑



GRAMMAR

Production:

*Rewrite **I** as **C G E G***

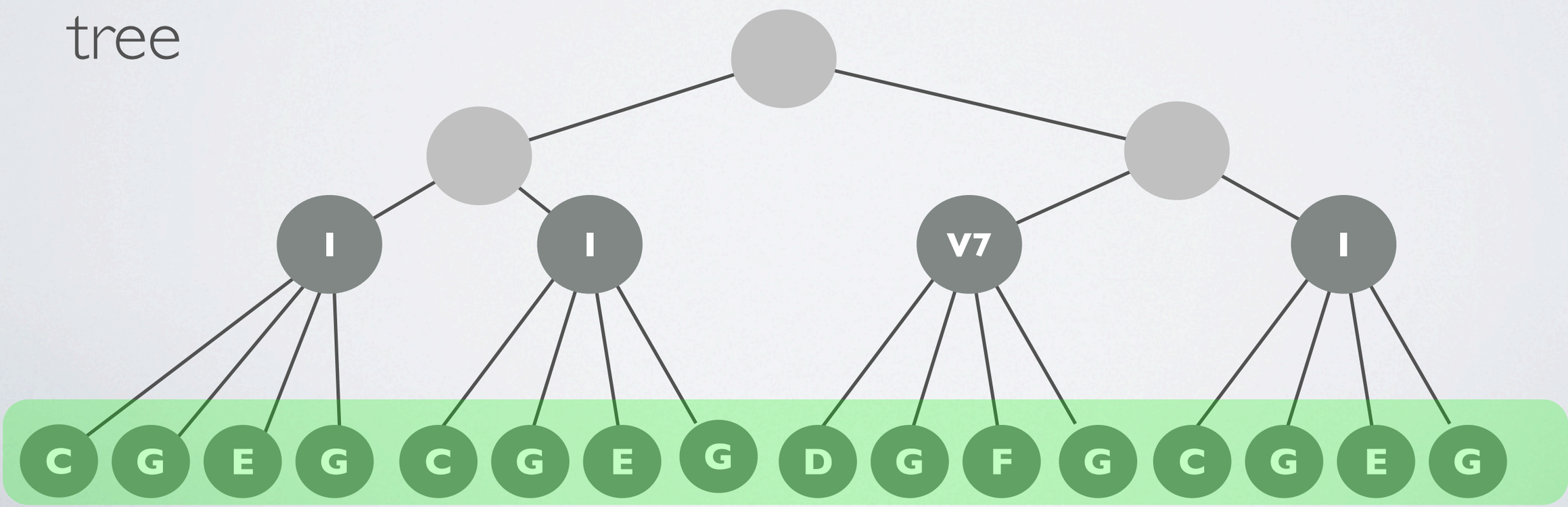
Melody → I I V7 I

I → C G E G

V7 → D G F G

Terminals ∈ ∑

Parsing tree



GRAMMAR

- Generalizing the grammar

  Melody ⟶ ChordRealization+

  ChordRealization ⟶ Note Note Note Note

  Note ∈ any Σ

TREE PROPERTY

Trees capture more ***easily* structure** than strings

MUSIC TREE REPRESENTATION

previous uses

Formal architecture

Grammars

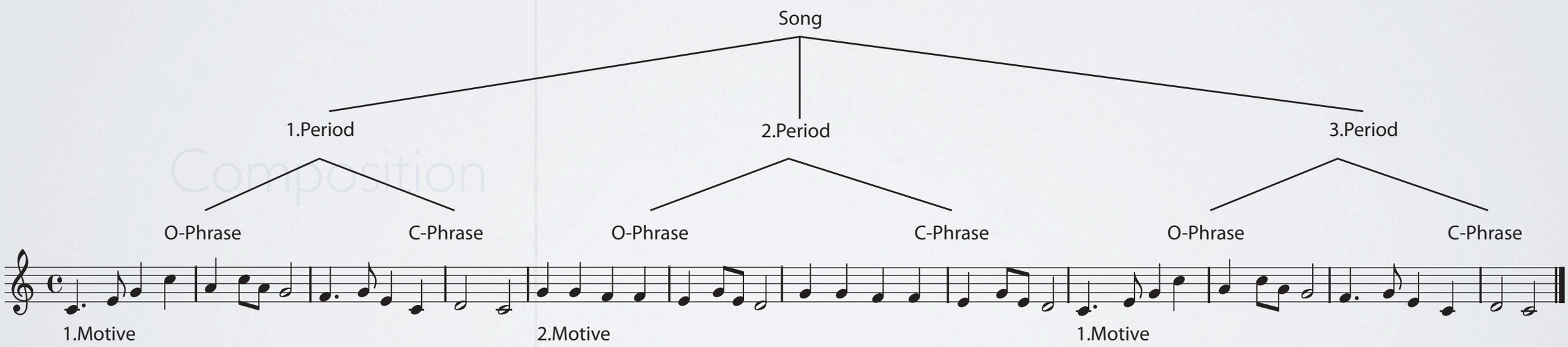
Composition

Analysis

MUSIC TREE REPRESENTATION

Formal architecture —

Musical form



MUSIC TREE REPRESENTATION

Lee'85 grammars

Formal architecture

Grammars

Composition

Analysis

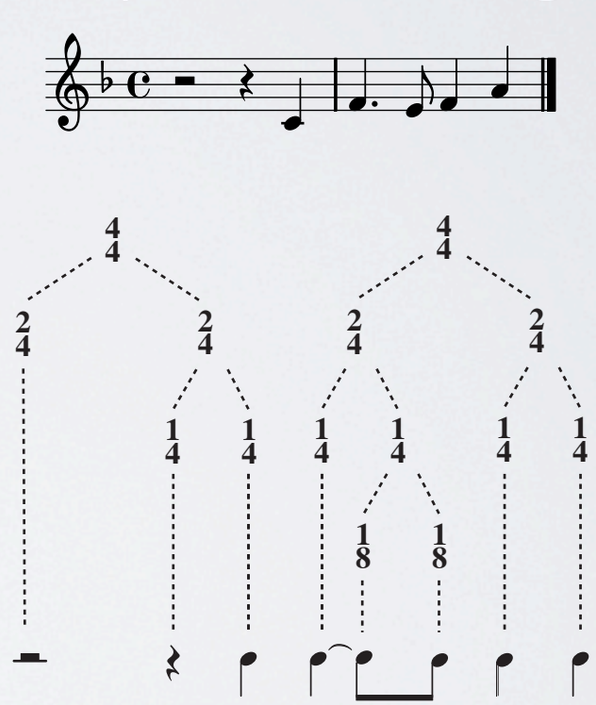
Explain rhythm perception with grammars

$$\mathbf{C} \rightarrow \mathbf{o} \mid \mathbf{-} \mid \tfrac{2}{4} + \tfrac{2}{4}$$

$$\tfrac{2}{4} \rightarrow \mathbf{d} \mid \mathbf{-} \mid \tfrac{1}{4} + \tfrac{1}{4}$$

$$\tfrac{1}{4} \rightarrow \mathbf{q} \mid \mathbf{\mathcal{z}} \mid \tfrac{1}{8} + \tfrac{1}{8}$$

$$\tfrac{1}{8} \rightarrow \mathbf{e} \mid \mathbf{7} \mid \cdots$$



MUSIC TREE REPRESENTATION

Formal architecture

Grammars

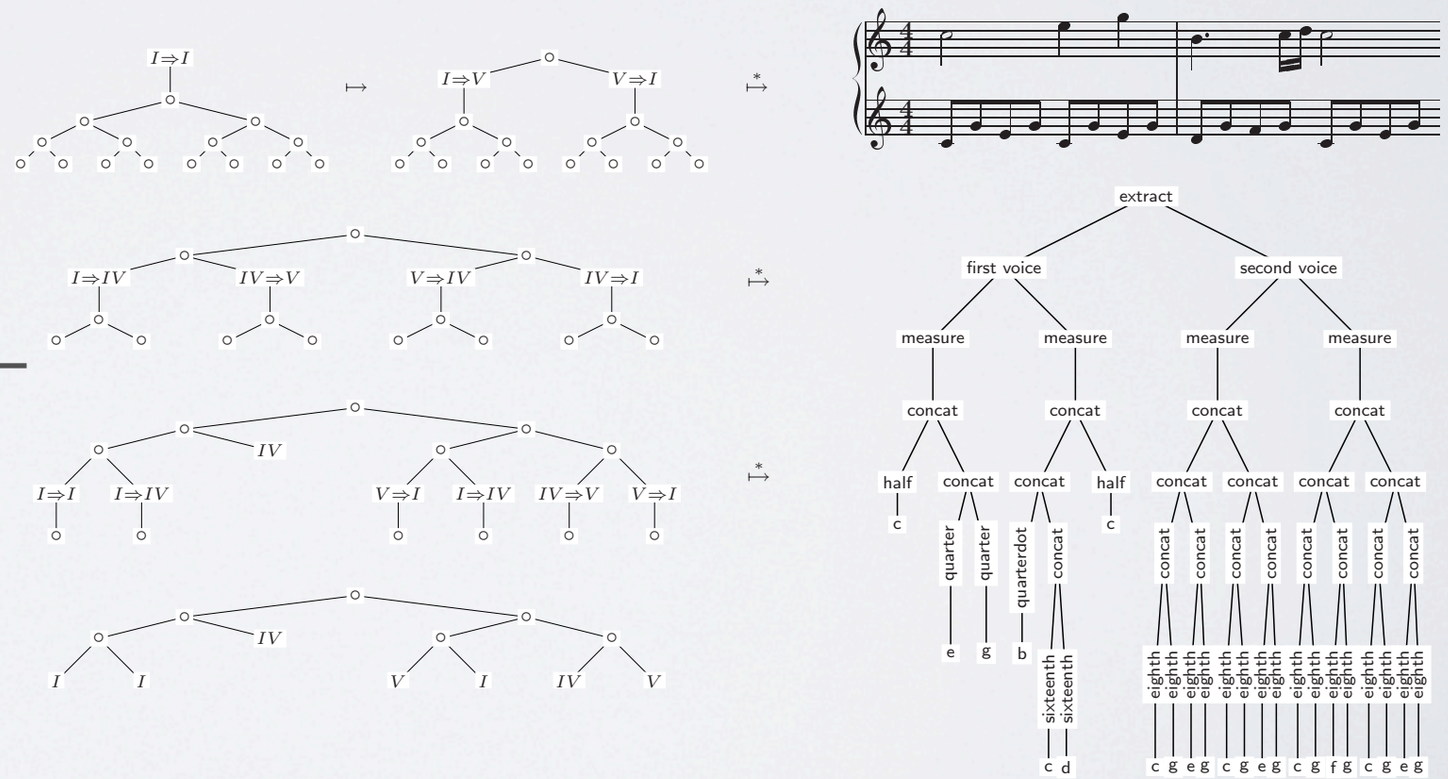
Composition

Analysis

Wind in the Willows

Högberg' 05

Tree transducers to compose



MUSIC TREE REPRESENTATION

Formal architecture

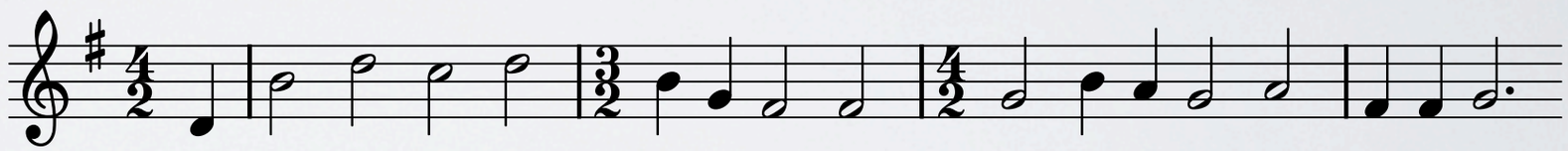
Grammars

Composition

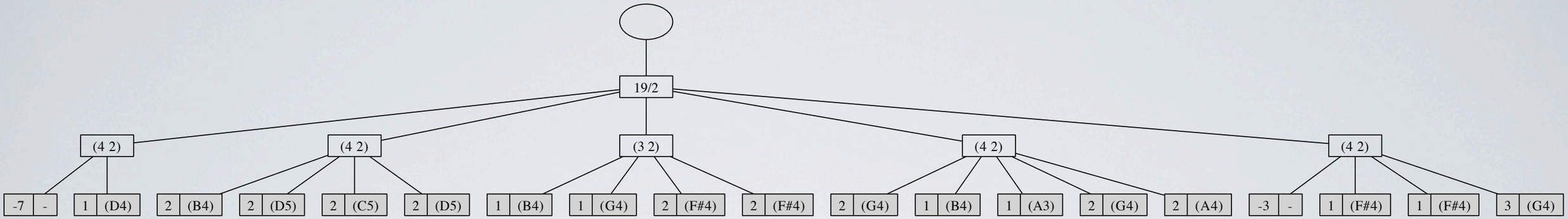
Analysis

OpenMusic

Trees (list of lists)
to encode rhythm patterns



```
(19/2
   (((4 2) (-7 1))
((4 2)(2 2 2 2)) ((3 2) (1 1 2 2))
   ((4 2) (2 1 1 2 2))
   ((4 2)  (-3 1 1 3)))
   )
```



OpenMusic

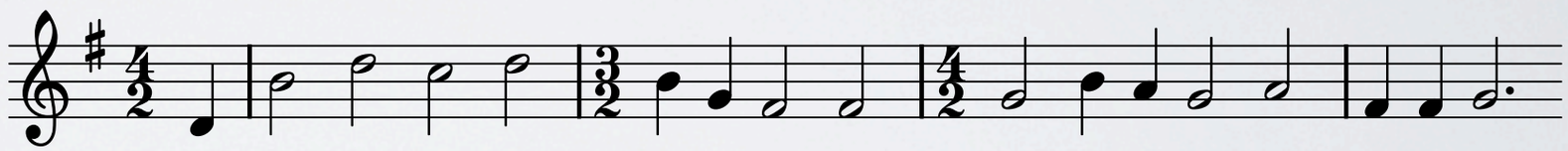
Formal architecture

Grammars

Composition

Analysis

Trees (list of lists) to encode rhythm patterns



```
(19/2
   (((4 2) (-7 1))
((4 2)(2 2 2 2)) ((3 2) (1 1 2 2))
      ((4 2) (2 1 1 2 2))
     ((4 2)  (-3 1 1 3)))
)
```

MUSIC TREE REPRESENTATION

Formal architecture

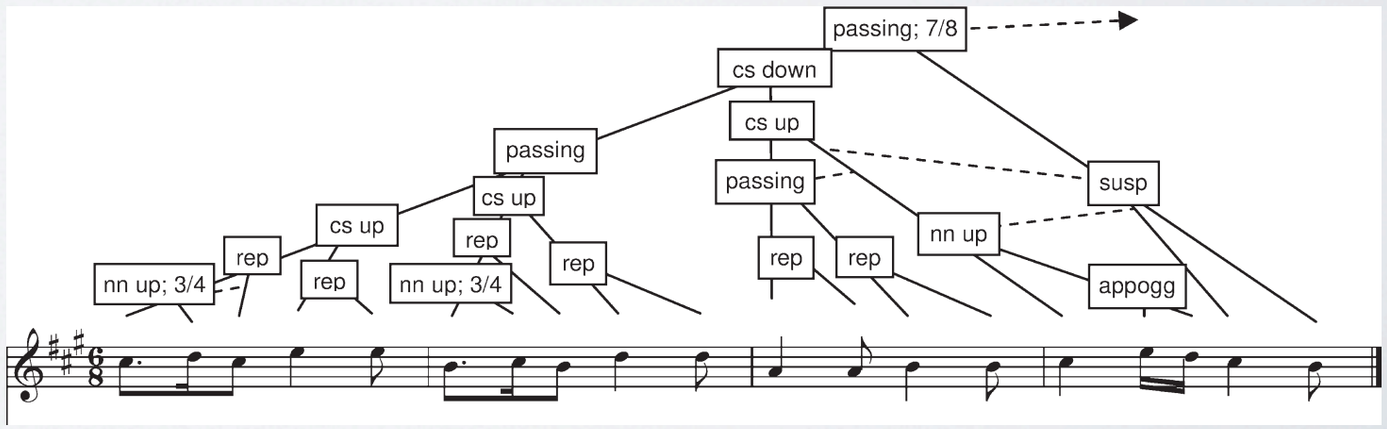
Grammars

Composition

Analysis

Use trees to represent the result of analyses

Schenkerian analysis



MUSIC TREE REPRESENTATION

Formal architecture

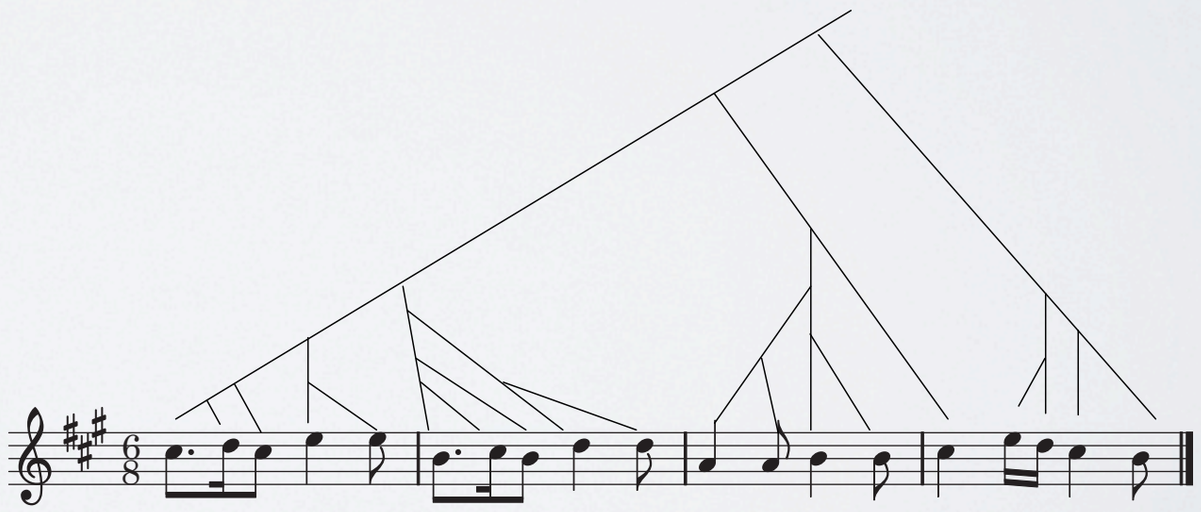
Grammars

Composition

Analysis

Use trees to represent the result of analyses

Generative theory of tonal music (GTTM)



OUR APPROACH

OUR APPROACH

- We found trees a suitable representation for

OUR APPROACH

- We found trees a suitable representation for
  - first: **similarity computation**

OUR APPROACH

- We found trees a suitable representation for
  - first: **similarity computation**
  - then:

OUR APPROACH

- We found trees a suitable representation for
  - first: **similarity computation**
  - then:
    - classification

OUR APPROACH

- We found trees a suitable representation for
  - first: **similarity computation**
  - then:
    - classification
    - composition

OUR APPROACH

- We found trees a suitable representation for
  - first: **similarity computation**
  - then:
    - classification
    - composition
- We developed a tree representation fitted for that goal

OUR HYPOTHESIS

OUR HYPOTHESIS

"One of the most obvious facts about the experience of **listening** to practically any piece of music is that one **perceives** not merely an arbitrary sequence of note durations, but some **sort of temporal structure** in which the notes are **grouped** into various kinds of units." (Lee, 1985)

OUR HYPOTHESIS

"One of the most obvious facts about the experience of **listening** to practically any piece of music is that one **perceives** not merely an arbitrary sequence of note durations, but some **sort of temporal structure** in which the notes are **grouped** into various kinds of units." (Lee, 1985)

"**Reductions** offer a tempting solution to fuzzy matching because they **suppress** the **differences** of surface **detail**. But do they suppress the most appropriate surface details?" (Selfridge-Field, 1998)

OUR HYPOTHESIS

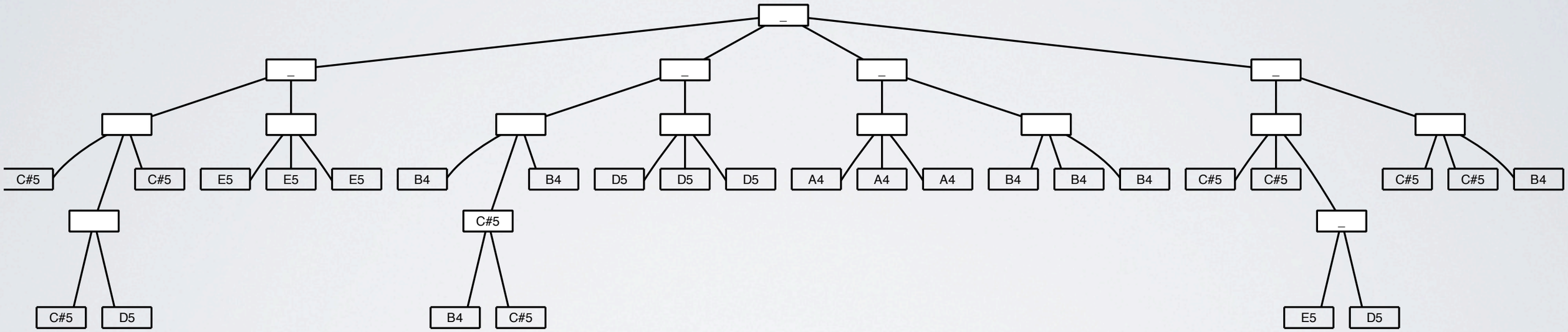
sort of temporal structure in which the notes are **grouped** into various kinds of units." (Lee, 1985)

"Reductions

suppress the **differences** of surface **detail**.

OUR HYPOTHESIS

sort of temporal structure in which the notes are **grouped** into various kinds of units." (Lee, 1985)

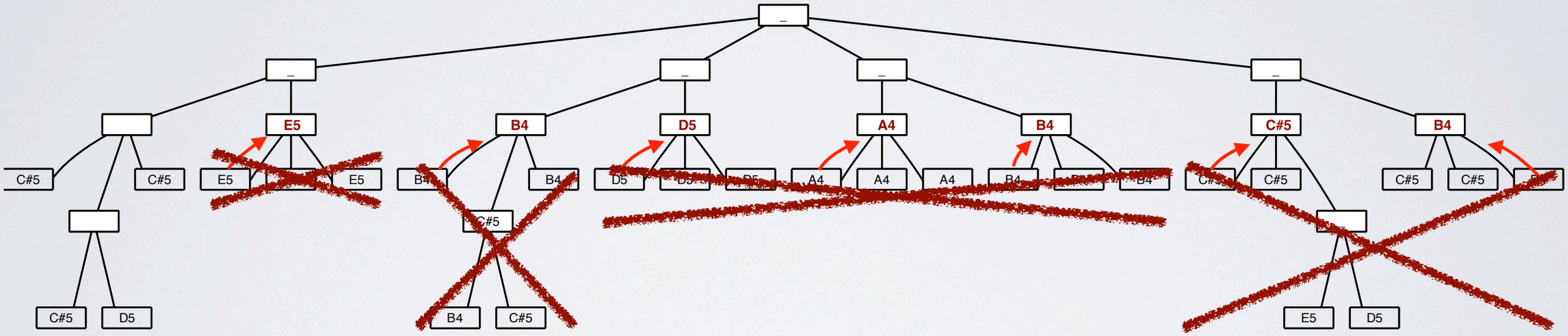


"Reductions

suppress the differences of surface detail.

OUR HYPOTHESIS

sort of temporal structure in which the notes are **grouped** into various kinds of units." (Lee, 1985)

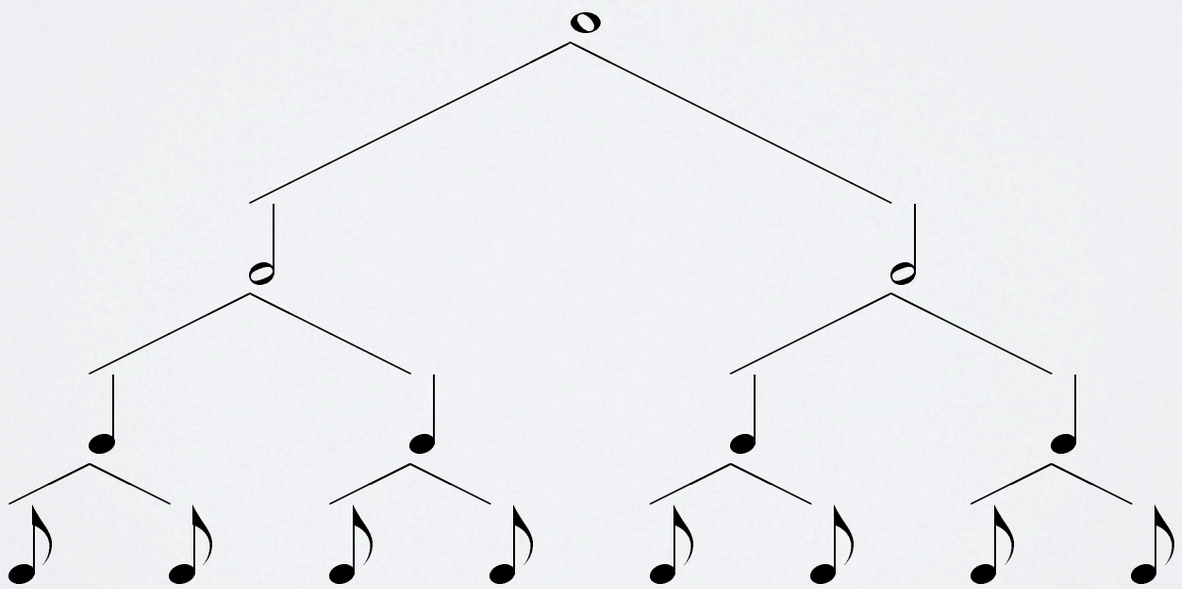


"Reductions

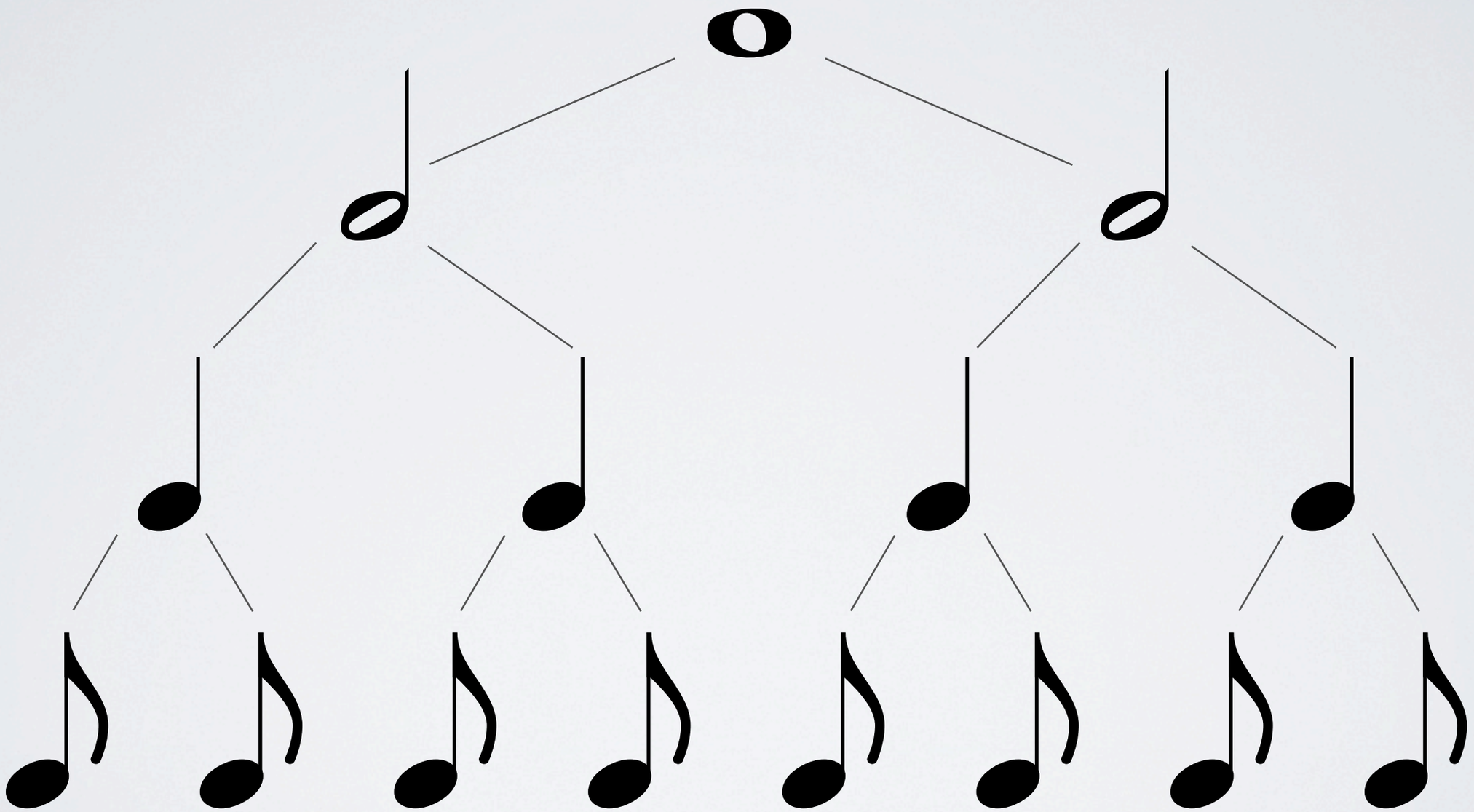
suppress the **differences** of surface **detail**.

MONOPHONIC METRIC TREES CONSTRUCTION

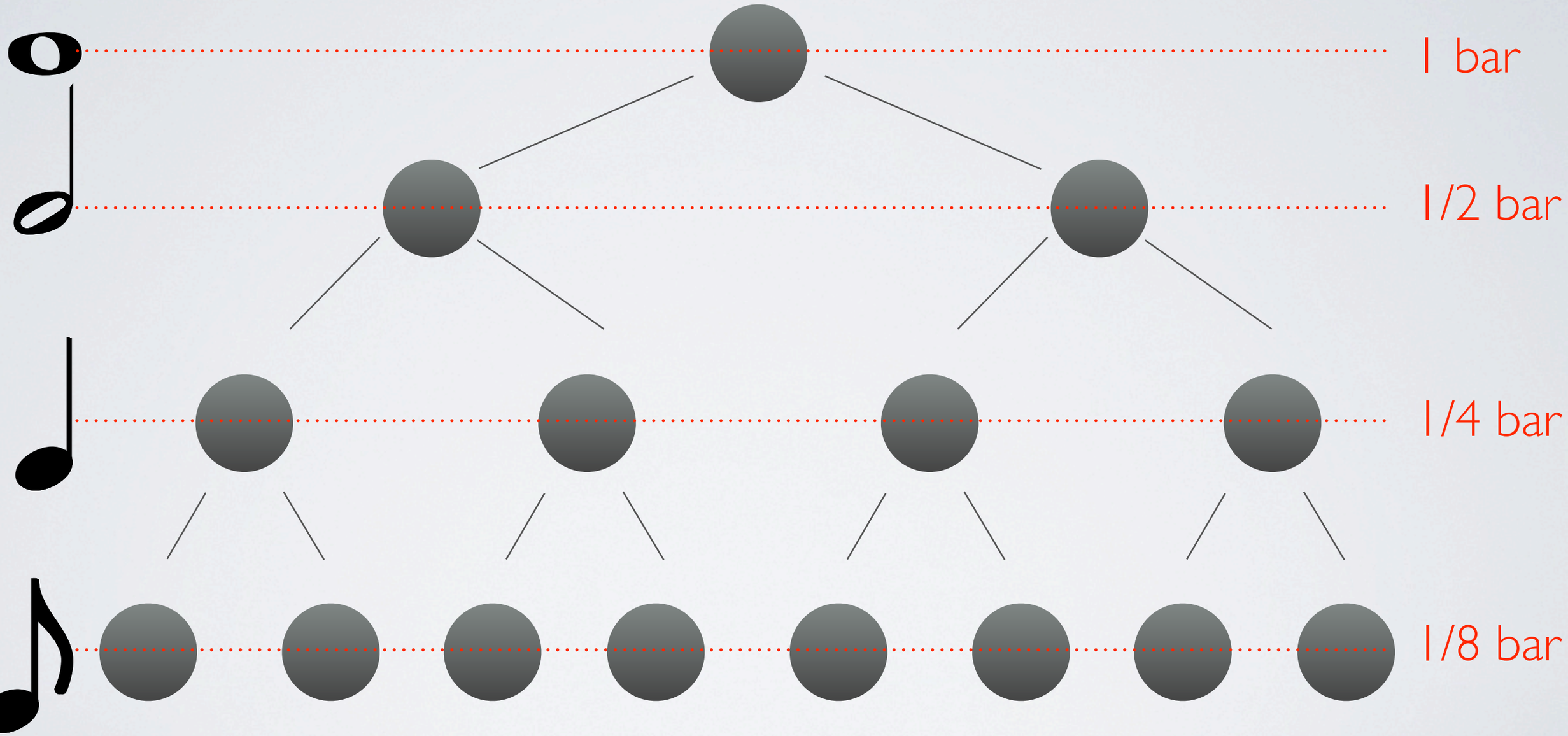
Based on metrical structure



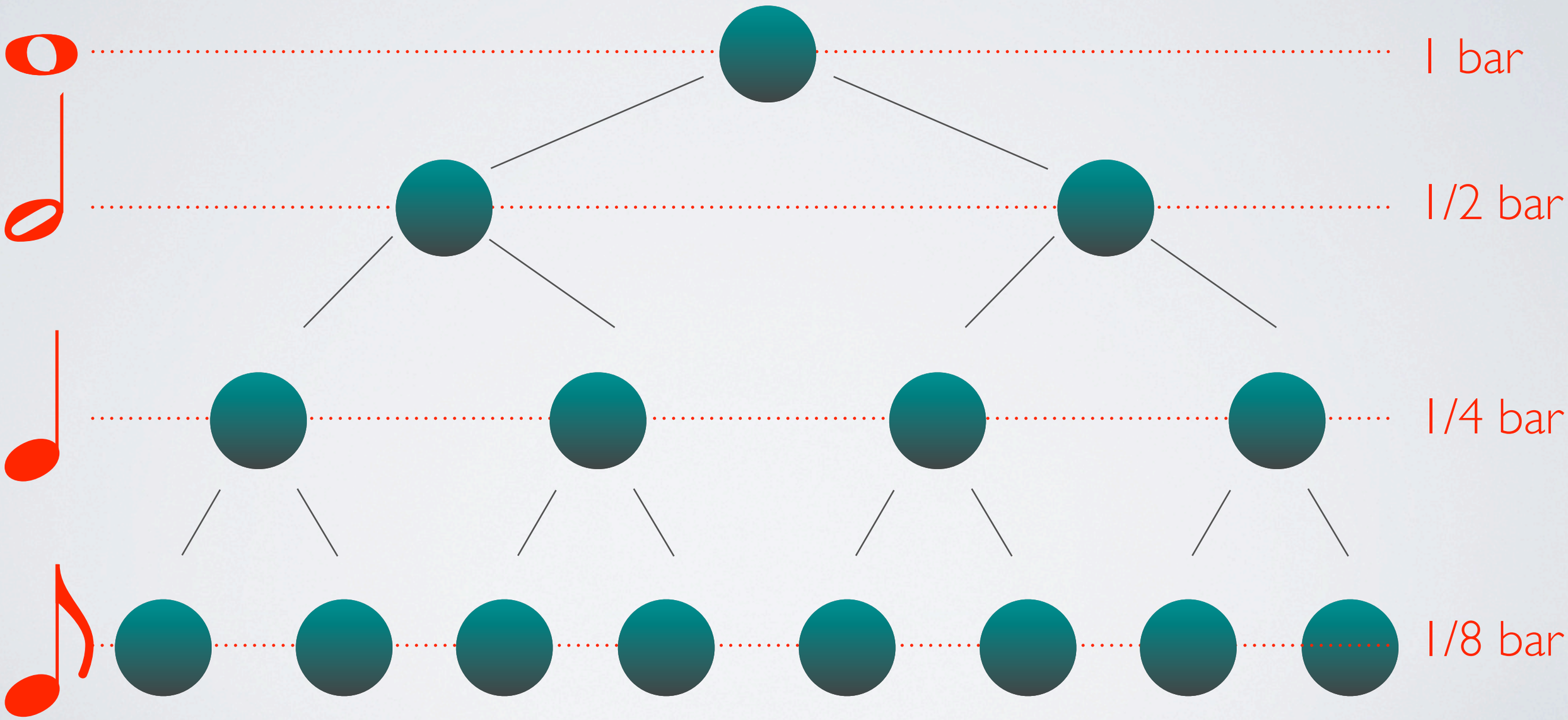
The level of the tree determines the duration



The level of the tree determines the duration



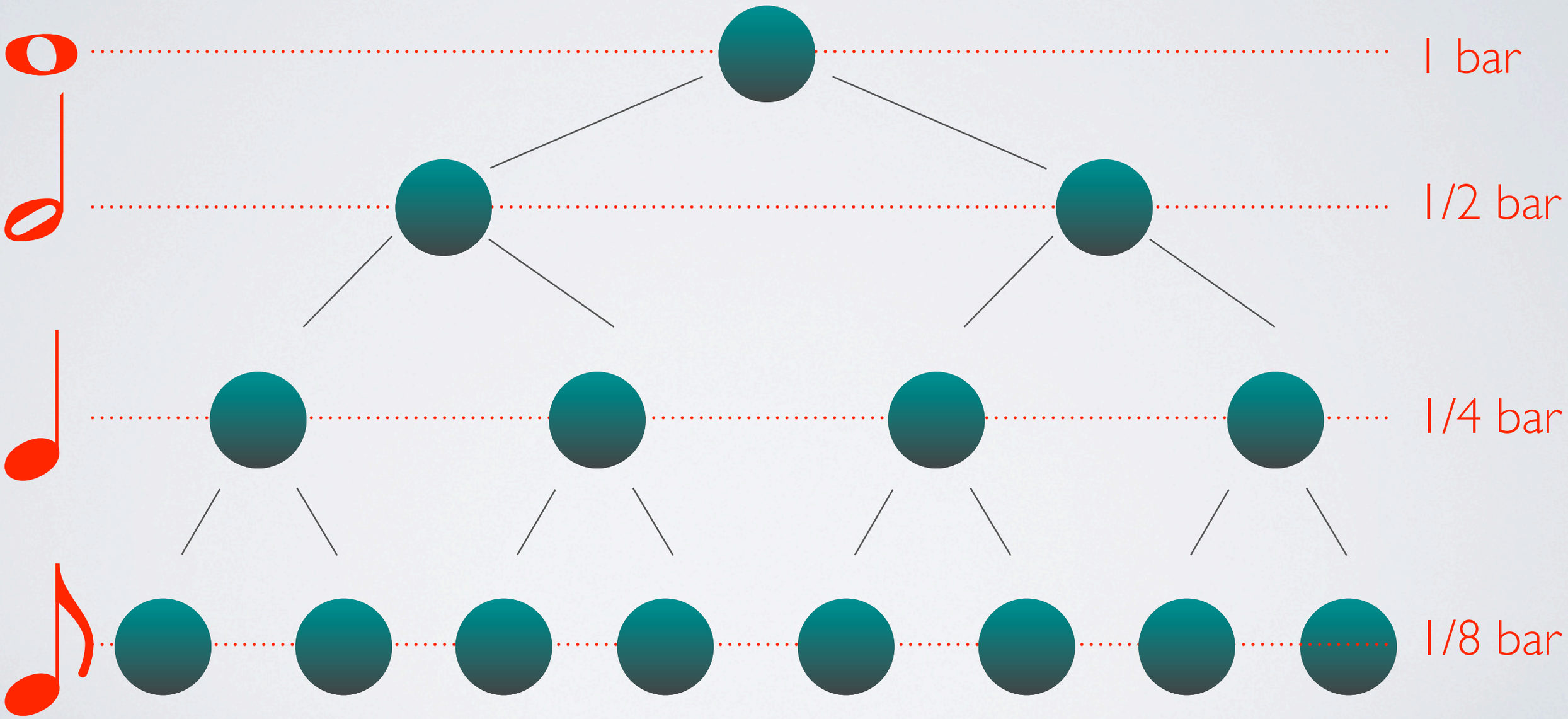
The level of the tree determines the duration

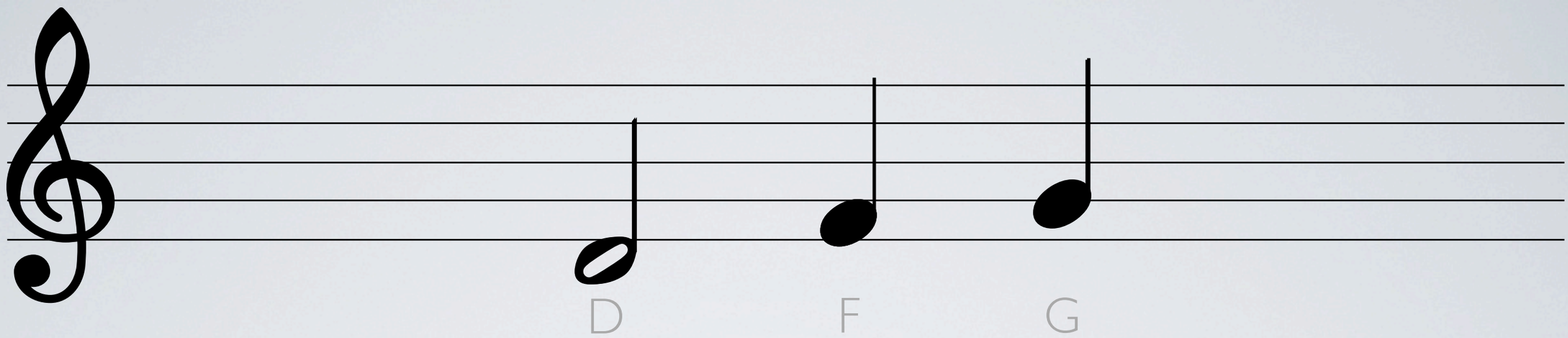


The level of the tree determines the duration

Leaves are labeled using the pitch

Nodes are created top-down left-right until reaching the figure duration



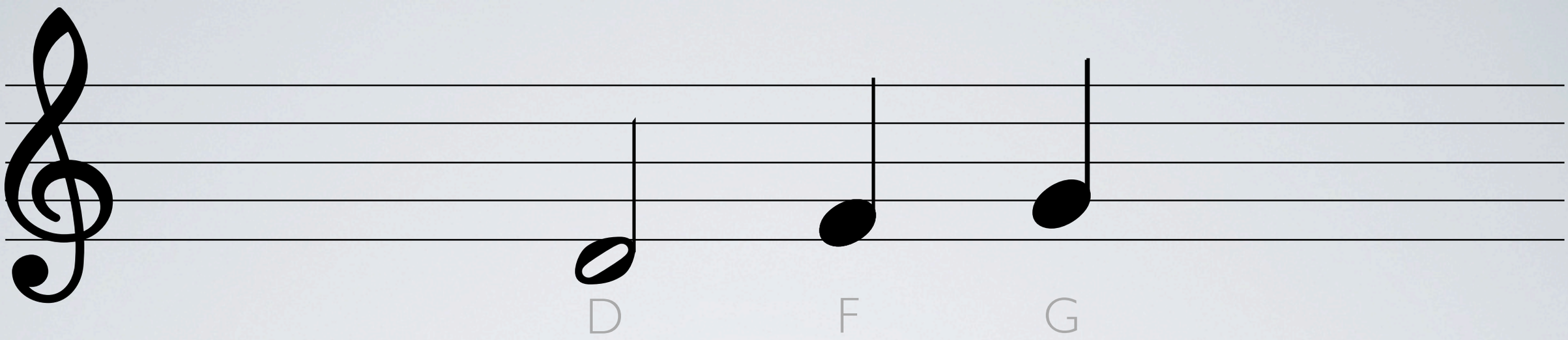


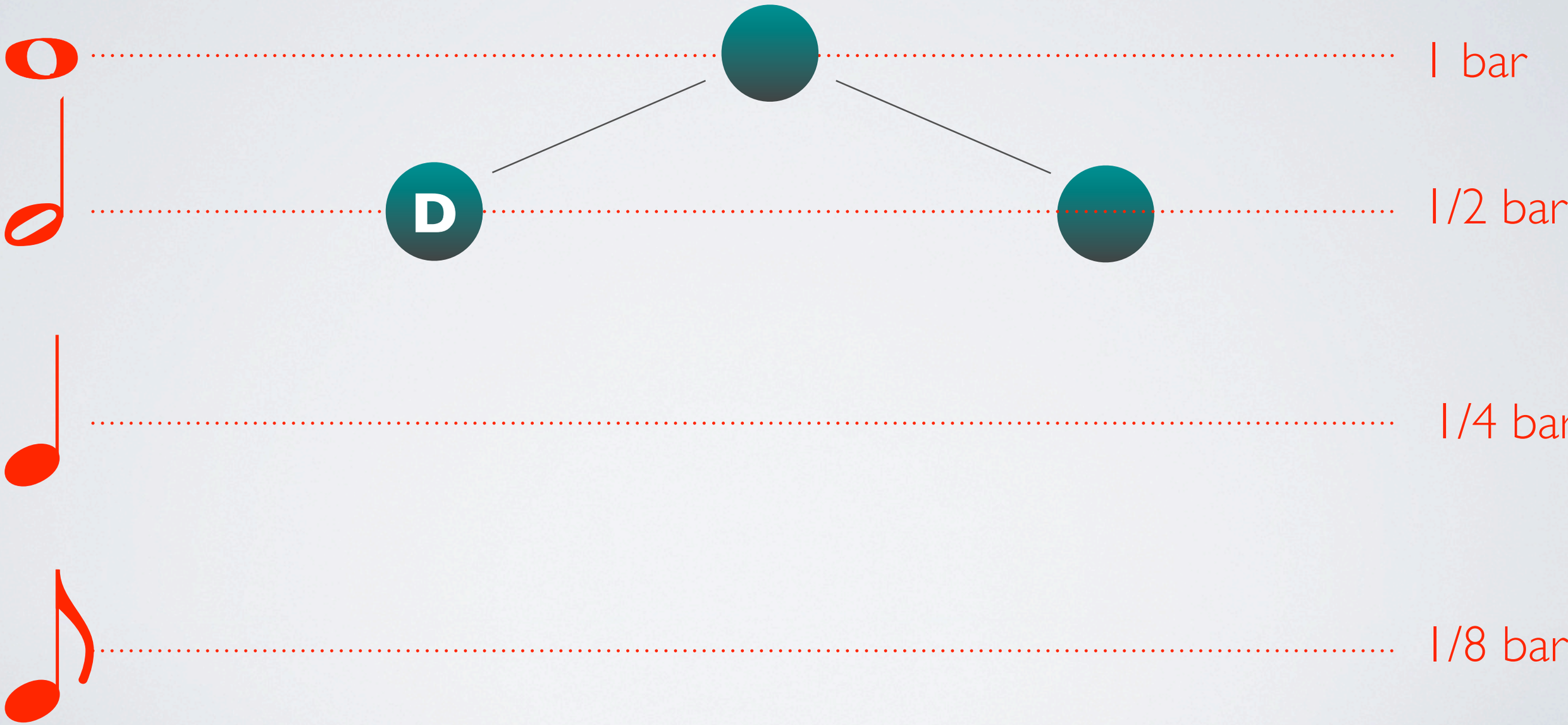
1 bar

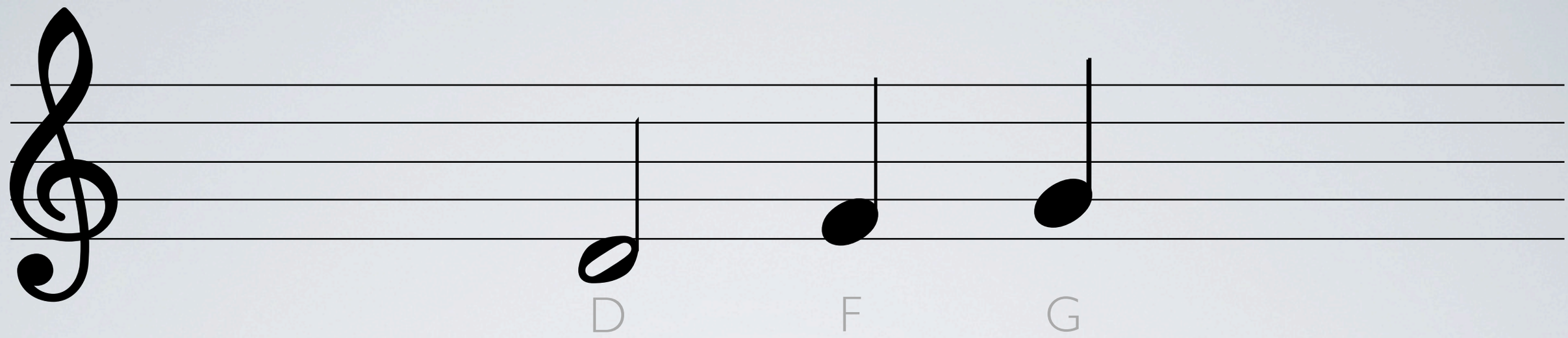
1/2 bar

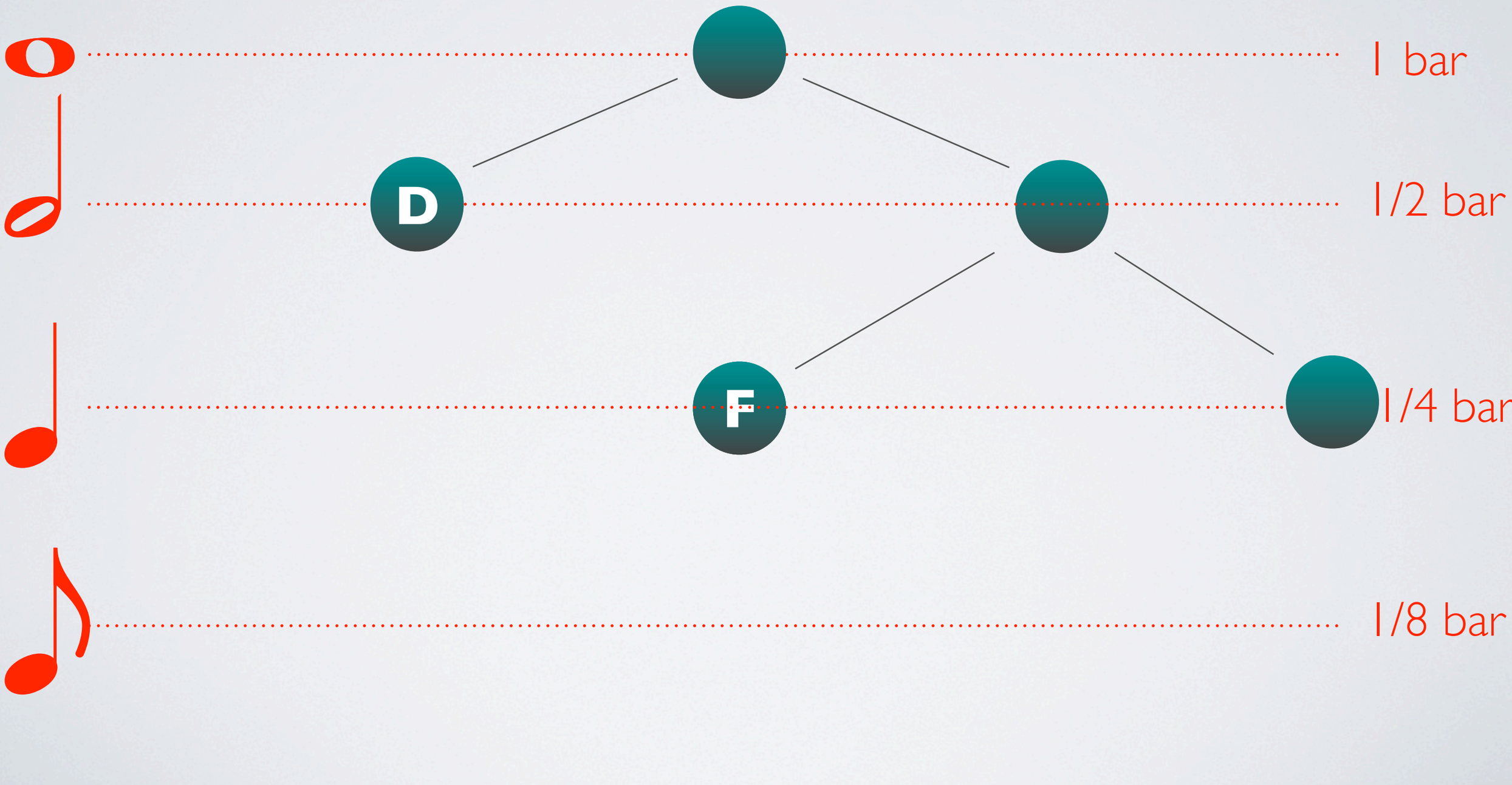
1/4 bar

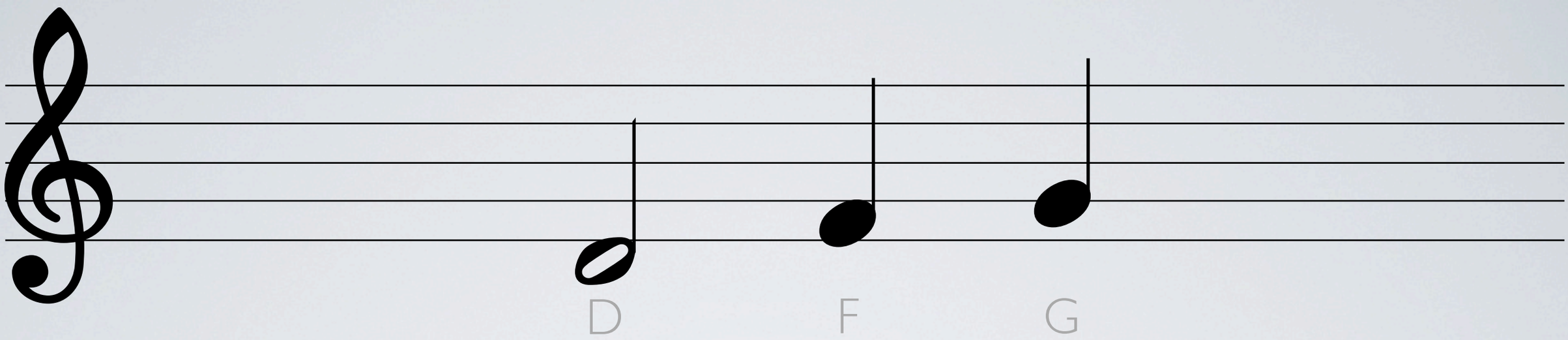
1/8 bar

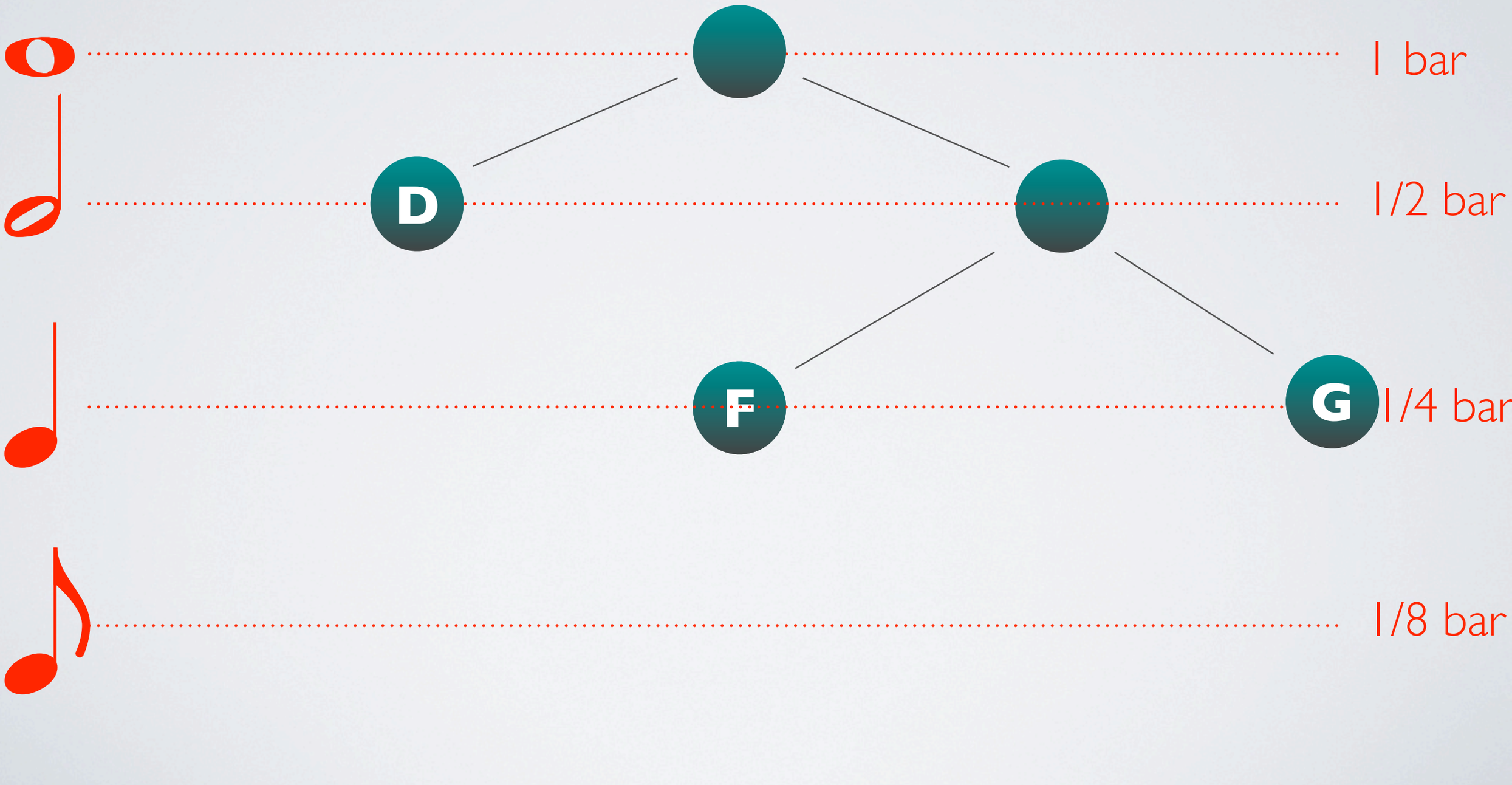

D
F
G


1 bar
D
1/2 bar
1/4 bar
1/8 bar


D
F
G


D
F
1 bar
1/2 bar
1/4 bar
1/8 bar


D
F
G


1 bar
1/2 bar
1/4 bar
1/8 bar
D
F
G

TREE CONSTRUCTION

Any pitch encoding Σ_p can be used

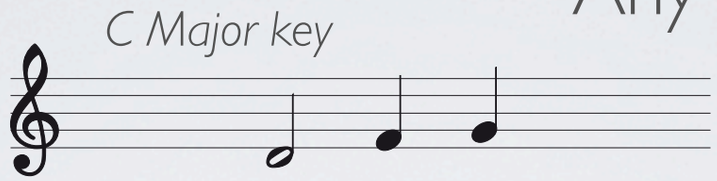
Pitches

Rests

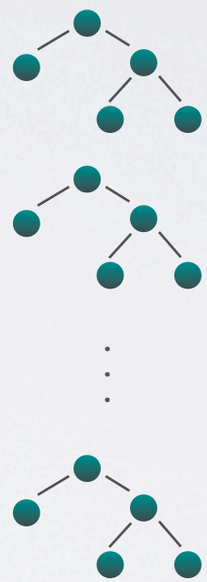
Dots and ties

Meters

Whole melodies



p_7	D	F	G
p_{abs}	62	65	67
p_{ift}	2	5	7
p_{itv}	0	3	2
p_c	-	Up	Up
p_{hdc}	-	Up	Up
p_{21}	4	10	13
p_{40}	9	20	26



TREE CONSTRUCTION

Any pitch encoding Σ_p can be used

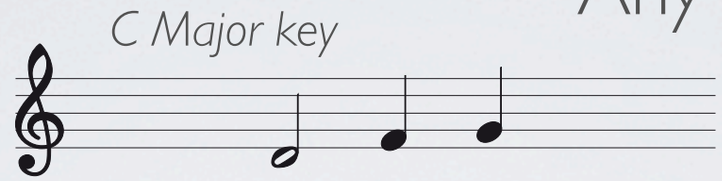
Pitches

Rests

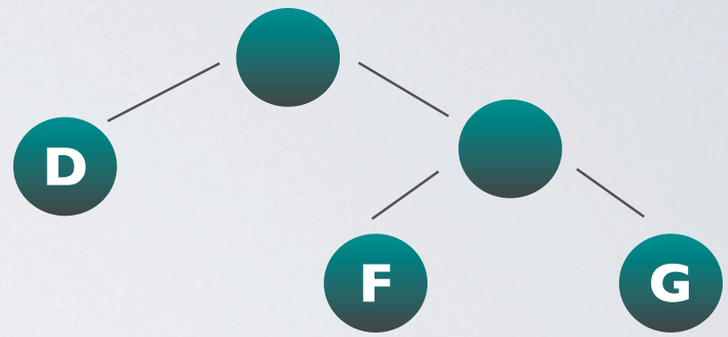
Dots and ties

Meters

Whole melodies



p_7	D	F	G
p_{abs}	62	65	67
p_{ift}	2	5	7
p_{itv}	0	3	2
p_c	-	Up	Up
p_{hdc}	-	Up	Up
p_{21}	4	10	13
p_{40}	9	20	26



TREE CONSTRUCTION

Any pitch encoding Σ_p can be used

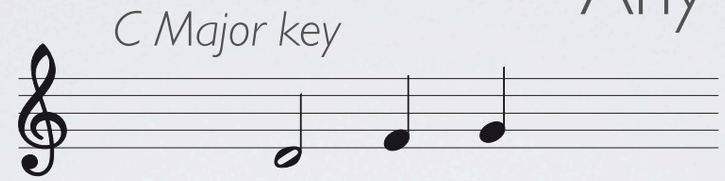
Pitches

Rests

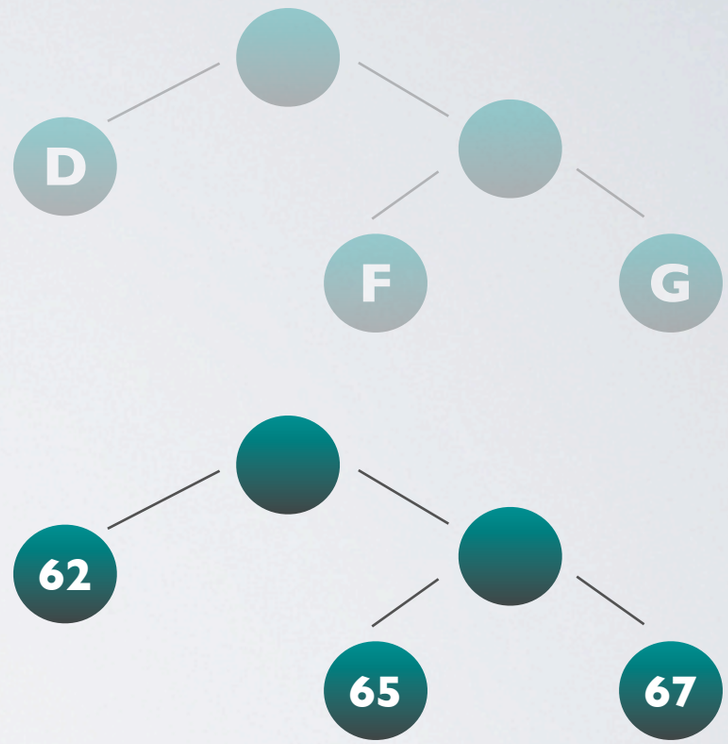
Dots and ties

Meters

Whole melodies



p_7	D	F	G
p_{abs}	62	65	67
p_{ift}	2	5	7
p_{itv}	0	3	2
p_c	-	Up	Up
p_{hdc}	-	Up	Up
p_{21}	4	10	13
p_{40}	9	20	26



TREE CONSTRUCTION

Any pitch encoding Σ_p can be used

Pitches

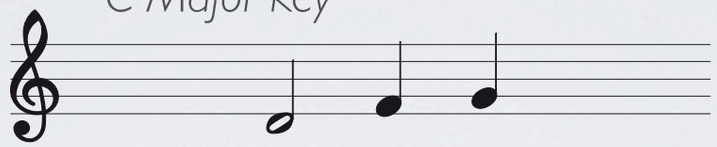
Rests

Dots and ties

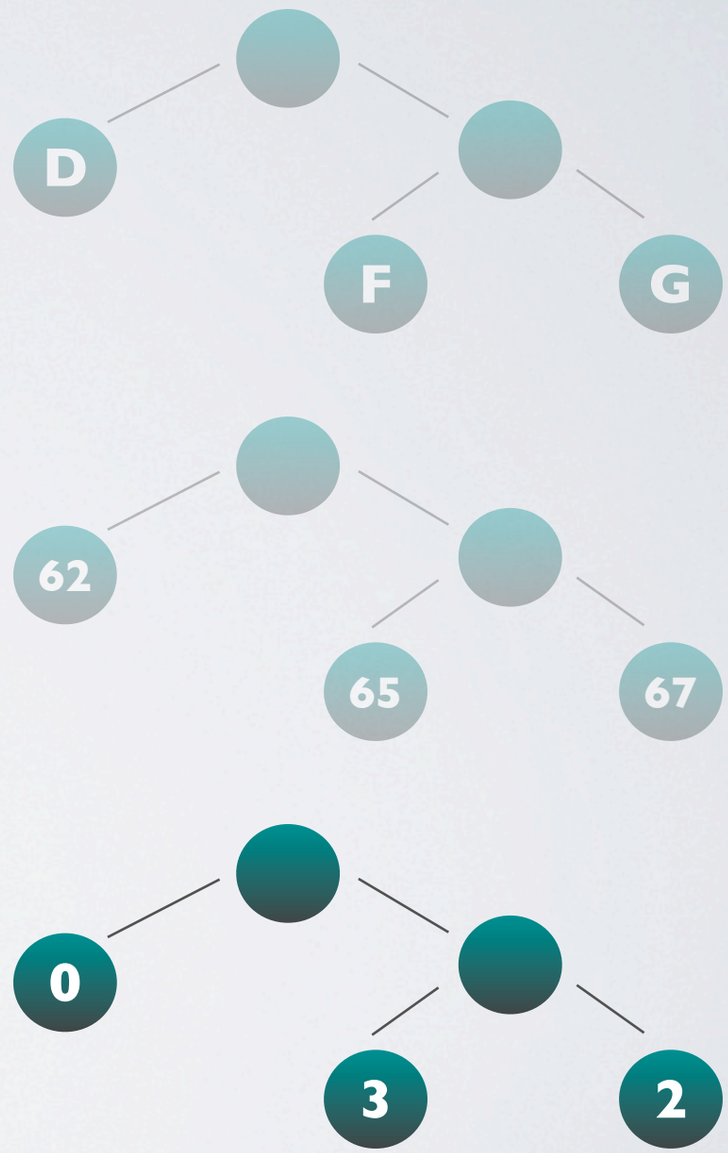
Meters

Whole melodies

C Major key



p_7	D	F	G	
p_{abs}	62	65	67	
p_{ift}	2	5	7	⋮
p_{itv}	0	3	2	
p_c	–	Up	Up	⋮
p_{hdc}	–	Up	Up	
p_{21}	4	10	13	
p_{40}	9	20	26	



TREE CONSTRUCTION

Pitches

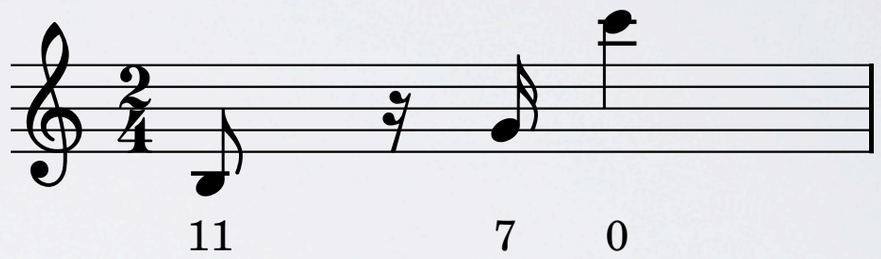
Rests

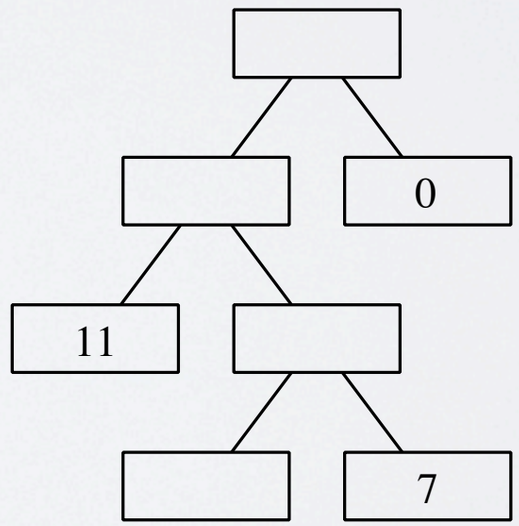
Dots and ties

Meters

Whole melodies

Encoded as a special code





TREE CONSTRUCTION

Pitches

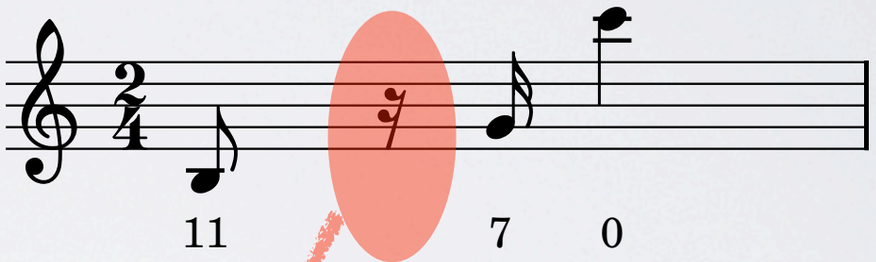
Rests

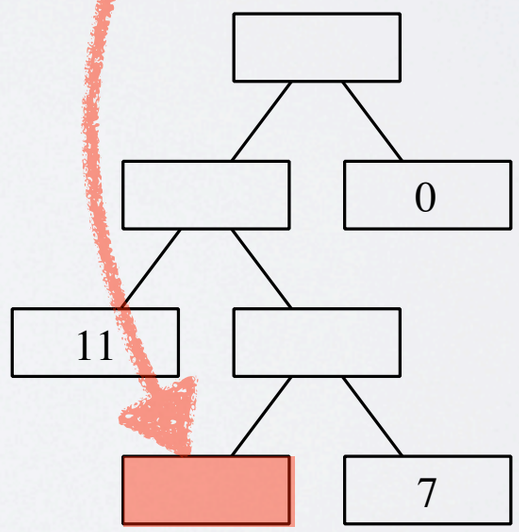
Dots and ties

Meters

Whole melodies

Encoded as a special code





TREE CONSTRUCTION

Pitches

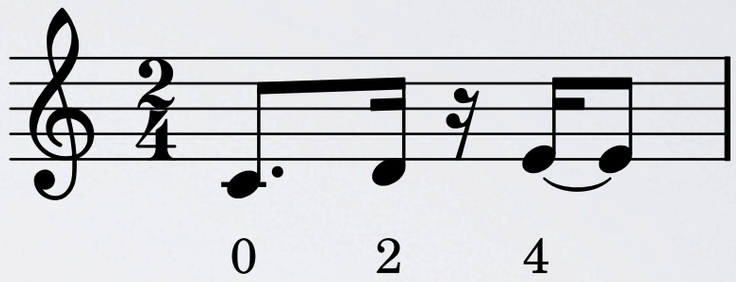
Rests

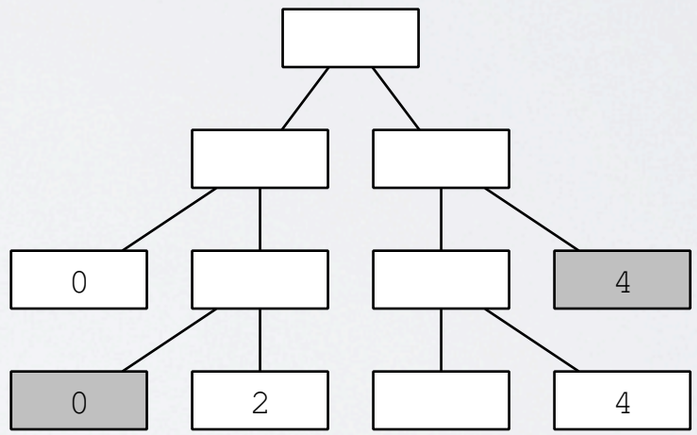
Dots and ties

Meters

Whole melodies

Ties and dots are broken





TREE CONSTRUCTION

Pitches

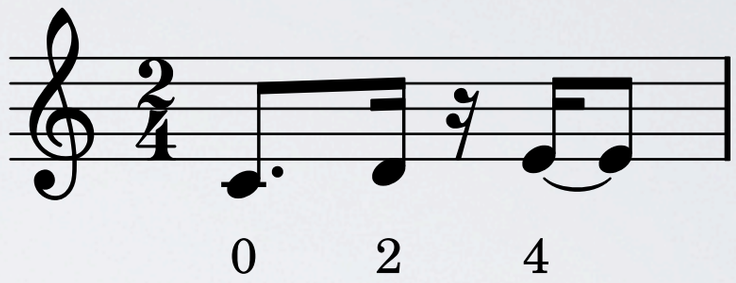
Rests

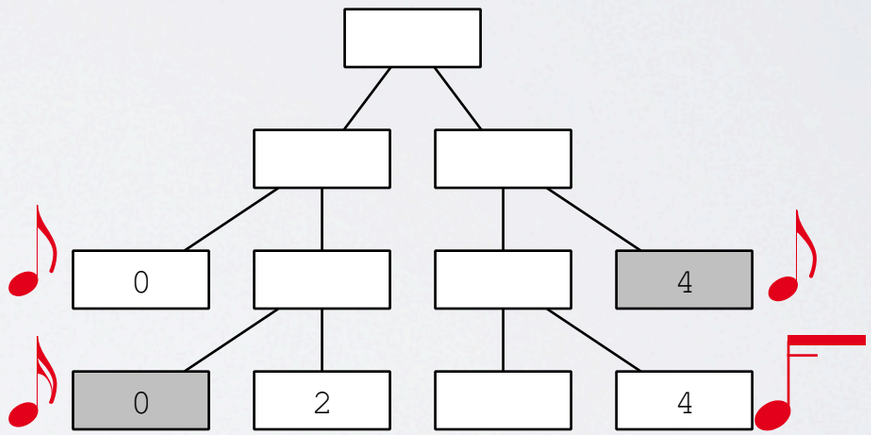
Dots and ties

Meters

Whole melodies

Ties and dots are broken





Continuation nodes are shaded

TREE CONSTRUCTION

Pitches

Rests

Dots and ties

Meters

Whole melodies

Meter numerator determines the tree first levels structure

Simple meters

$\frac{2}{4}$

$\frac{3}{4}$

$\frac{4}{4}$

TREE CONSTRUCTION

Meter numerator determines the tree first levels structure

Pitches

Rests

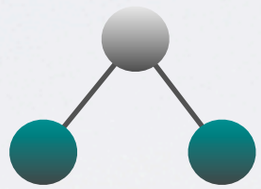
Dots and ties

Meters

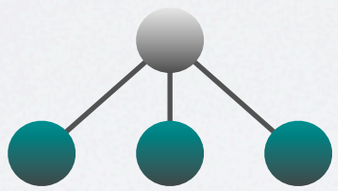
Whole melodies

Simple meters

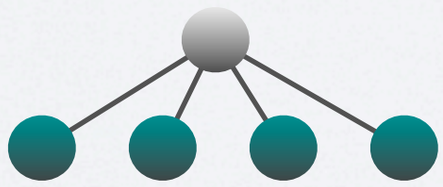
$\frac{2}{4}$



$\frac{3}{4}$



$\frac{4}{4}$



Beats

TREE CONSTRUCTION

Pitches

Rests

Dots and ties

Meters

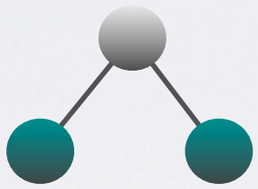
Whole melodies

Meter numerator determines the tree first levels structure

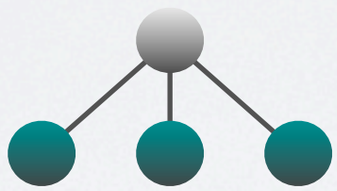
Simple meters

Compound meters

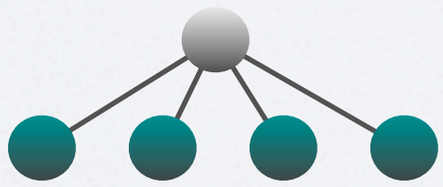
$\frac{2}{4}$ $\frac{6}{8}$



$\frac{3}{4}$ $\frac{9}{8}$



$\frac{4}{4}$



Beats

TREE CONSTRUCTION

Pitches

Rests

Dots and ties

Meters

Whole melodies

Meter numerator determines the tree first levels structure

Simple meters

Compound meters

TREE CONSTRUCTION

Pitches

Rests

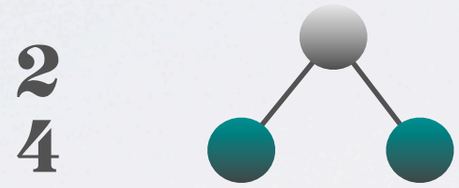
Dots and ties

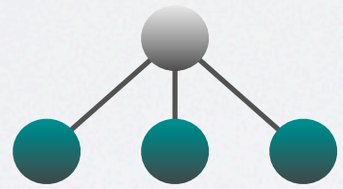
Meters

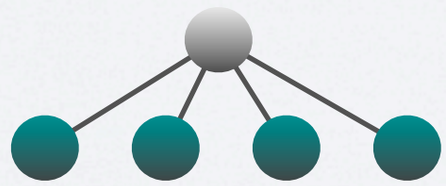
Whole melodies

Meter numerator determines the tree first levels structure

Simple meters

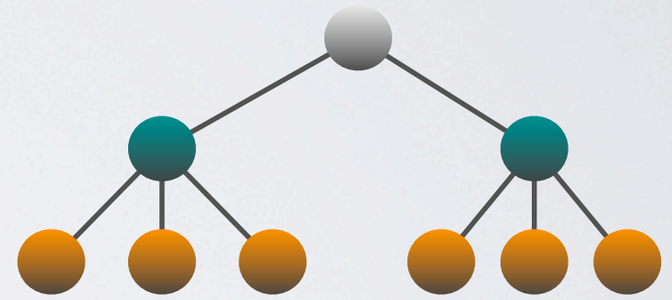


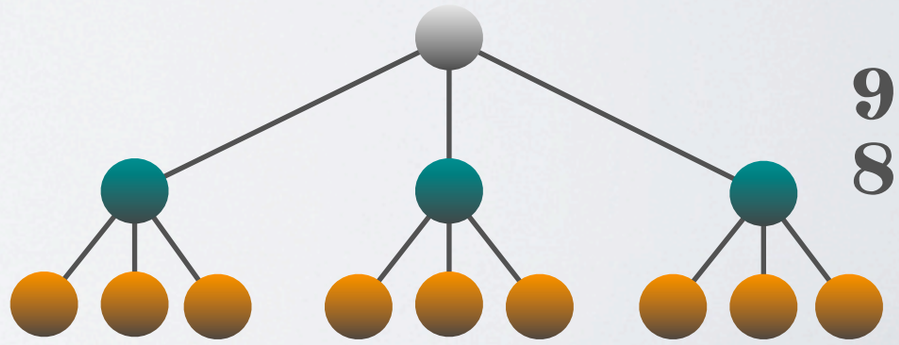




Beats

Compound meters





Beats and parts

TREE CONSTRUCTION

Pitches

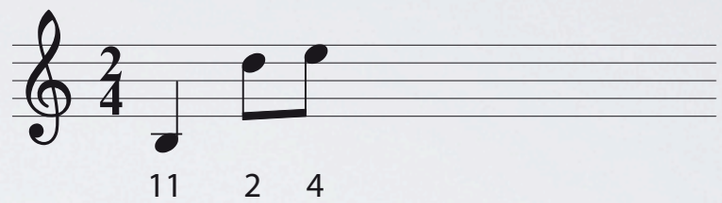
Rests

Dots and ties

Meters

Whole melodies

Each bar is a tree



TREE CONSTRUCTION

Pitches

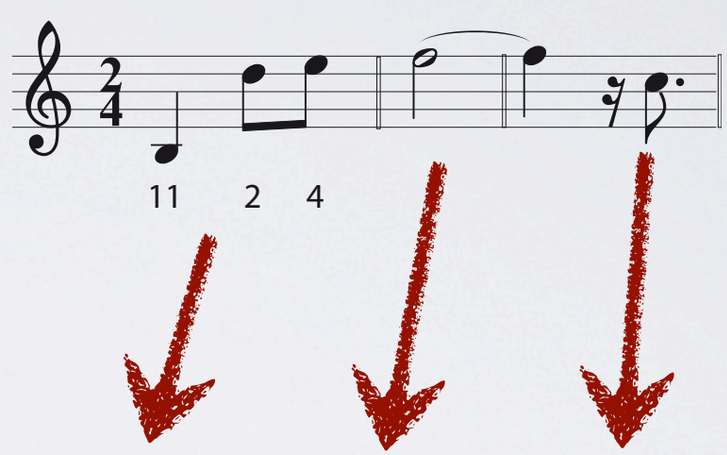
Rests

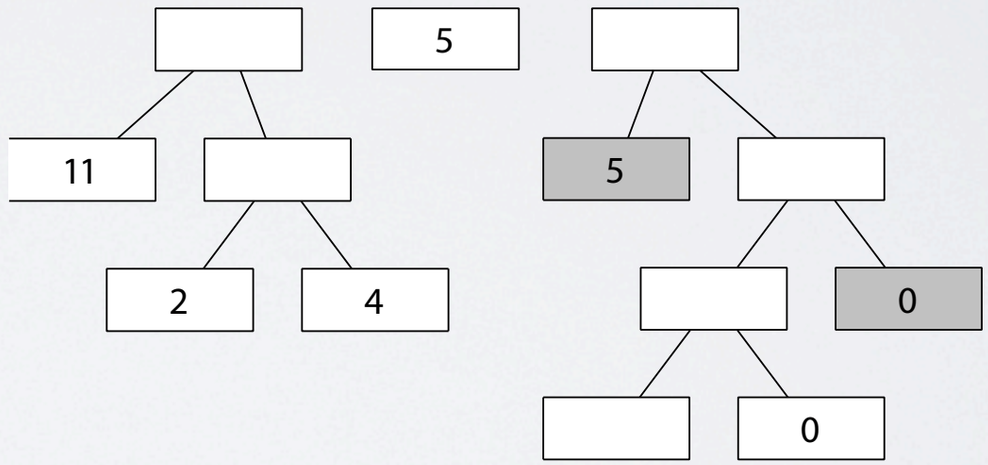
Dots and ties

Meters

Whole melodies

Each bar is a tree





TREE CONSTRUCTION

Pitches

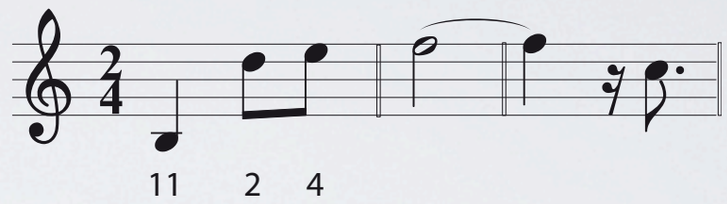
Rests

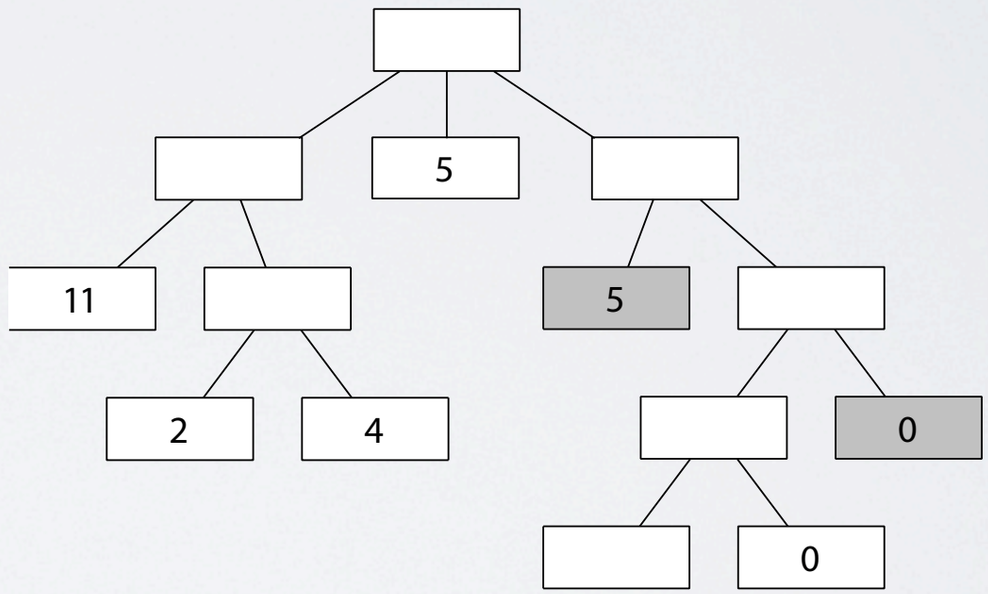
Dots and ties

Meters

Whole melodies

Each bar is a tree





Linked to a common root

SOME FACTS

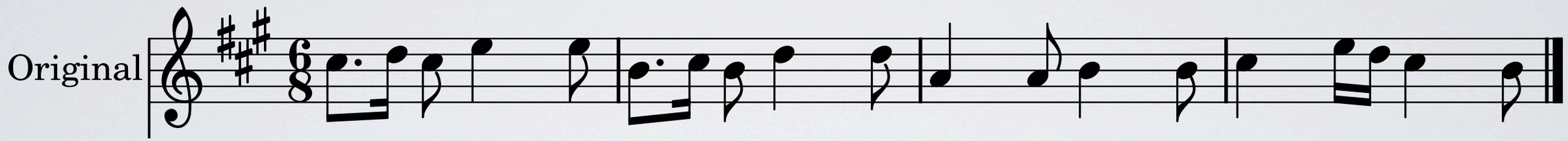
- Initially only leaves contain note information $\sum_p$
- This is not a parsing tree
  - actually it does not represent a grammar
    .... but could represent it
- Note no rhythm alphabet $\sum_p$ is required
  - rhythm encoded implicitly in tree structure

MUSIC SUMMARIZATION

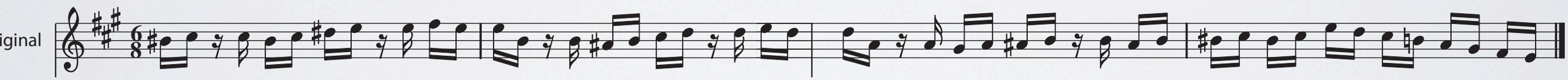
Mozart's K331 variations

Manual summarization

Theme



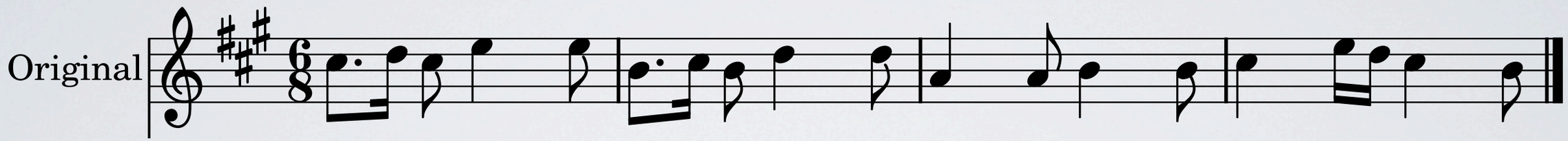
Variation 1



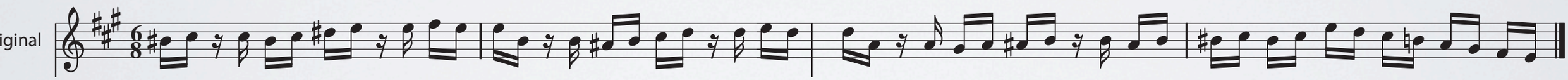
Mozart's K331 variations

Manual summarization

Theme



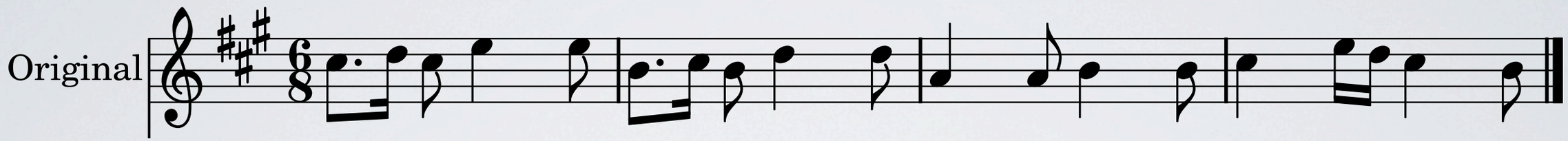
Variation 1



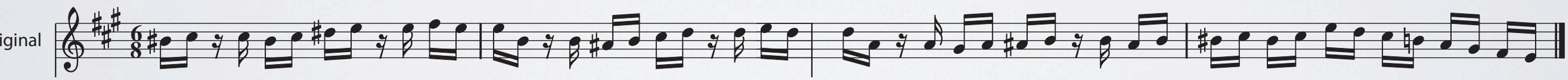
Mozart's K331 variations

Manual summarization

Theme



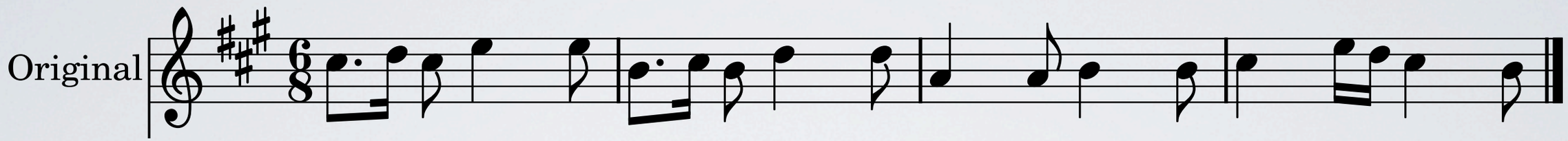
Variation 1



Mozart's K331 variations

Manual summarization

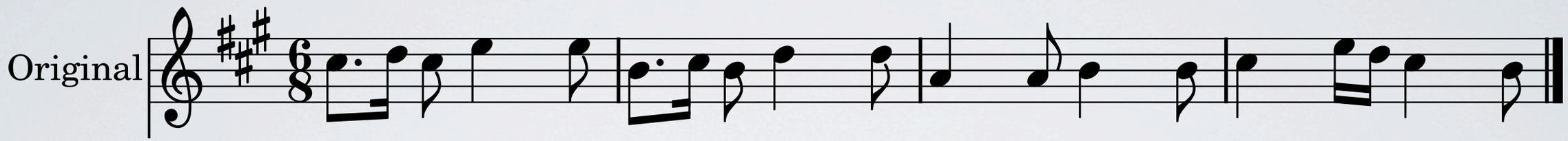
Theme



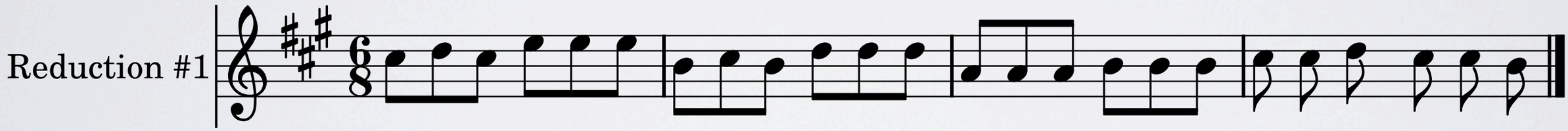
Mozart's K331 variations

Manual summarization

Theme



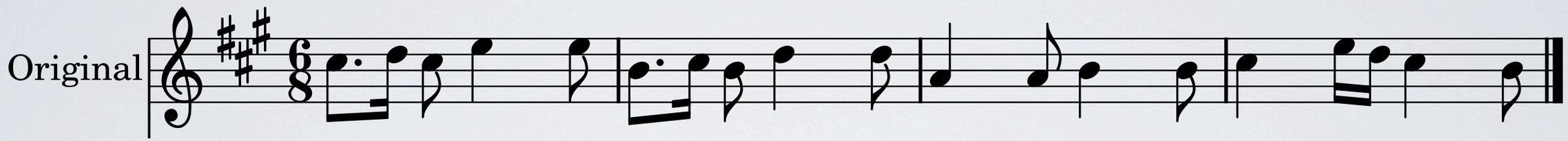
Reduce to 8th notes



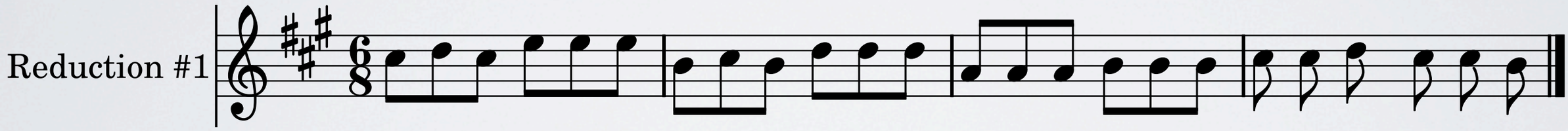
Mozart's K331 variations

Manual summarization

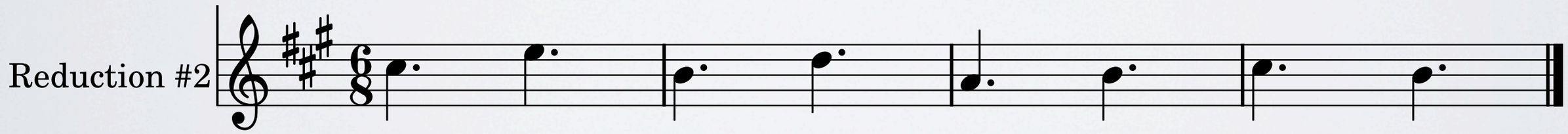
Theme



Reduce to 8th notes



Reduce to dotted quarter notes

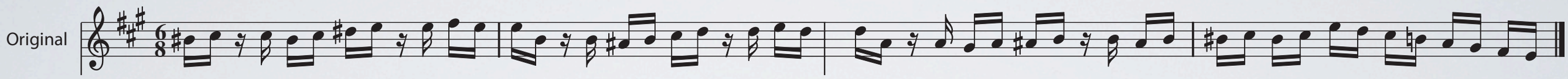


Mozart's K331 variations

Manual summarization

Variation 1

Original

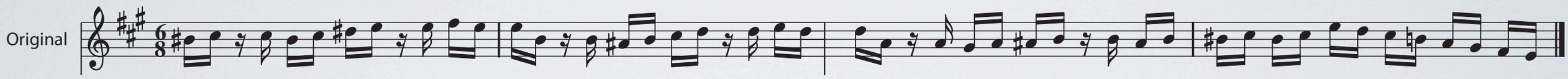


Mozart's K331 variations

Manual summarization

Variation 1

Original

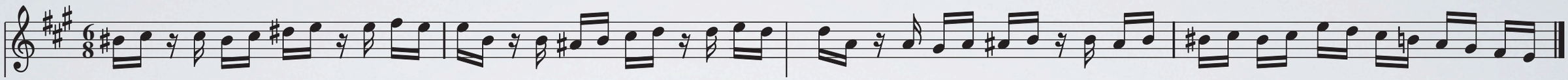


Mozart's K331 variations

Manual summarization

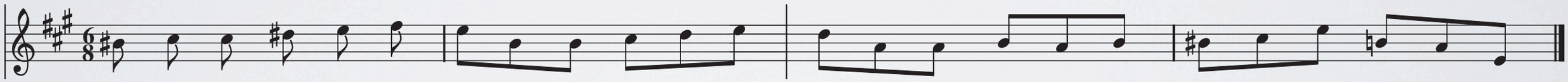
Variation 1

Original



Reduction to 8th notes

uction #1

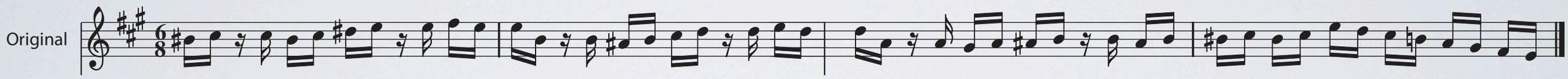


Mozart's K331 variations

Manual summarization

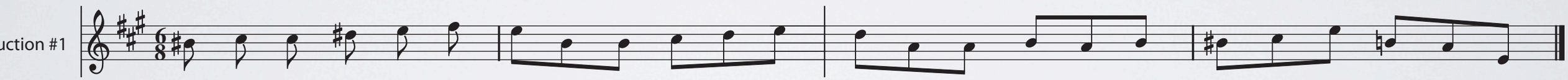
Variation 1

Original



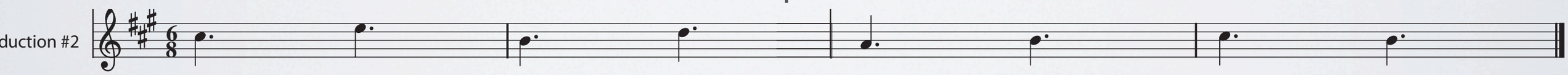
Reduction to 8th notes

uction #1

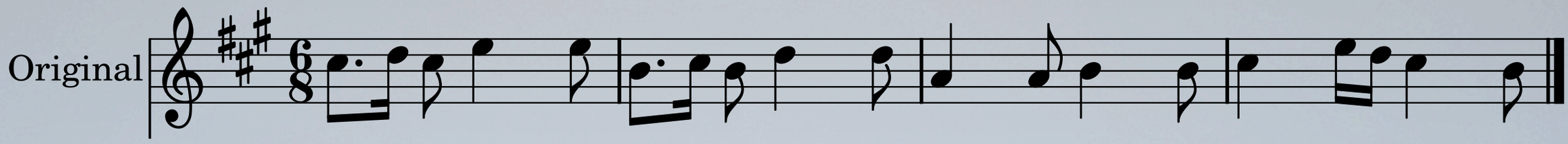


Reduction to quarter notes

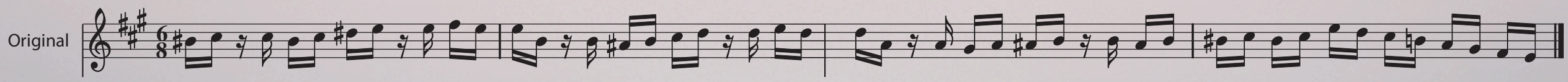
duction #2



Theme

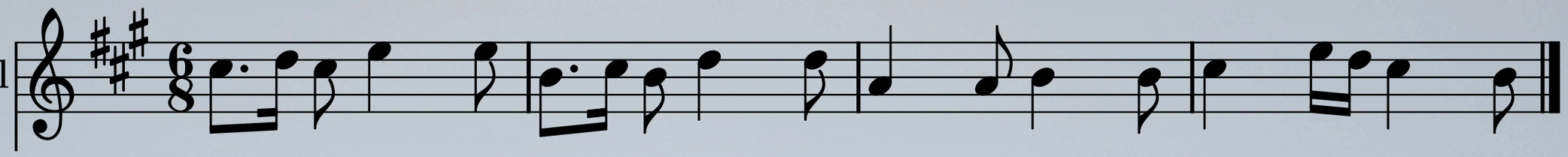


Variation 1



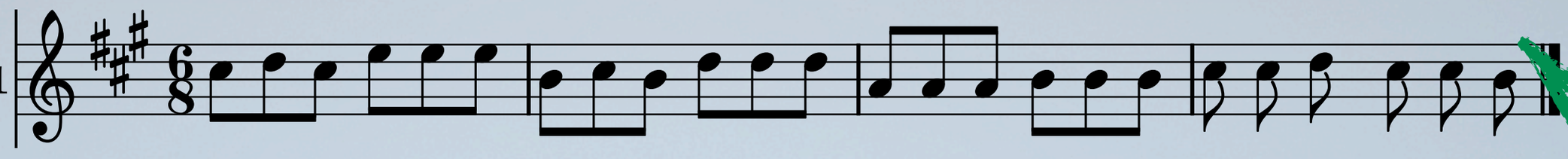
Theme

Original



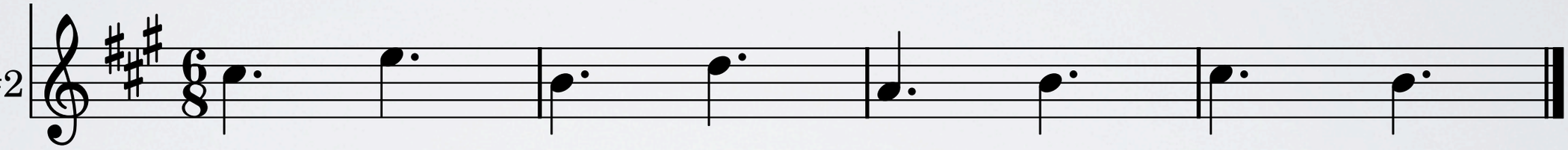
Reduce to 8th notes

Reduction #1



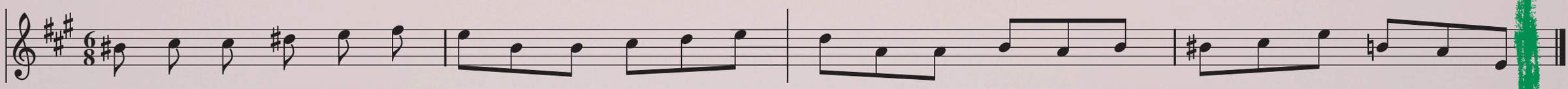
Same reduction to dotted quarter notes for both melodies

Reduction #2



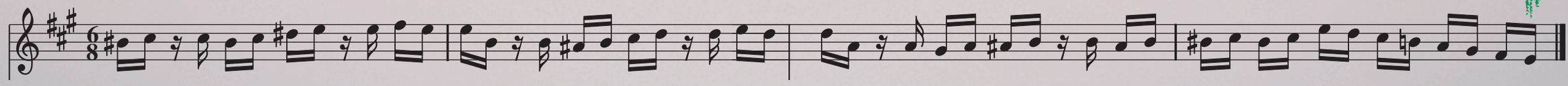
Reduction to 8th notes

duction #1



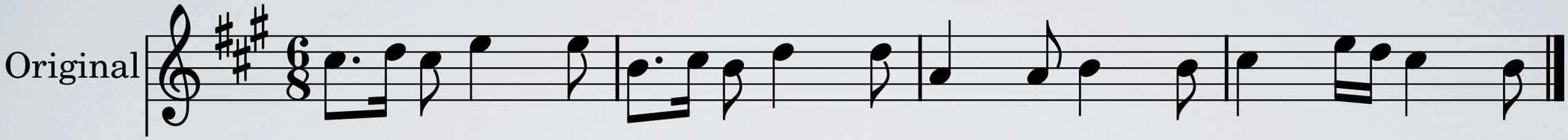
Variation 1

Original



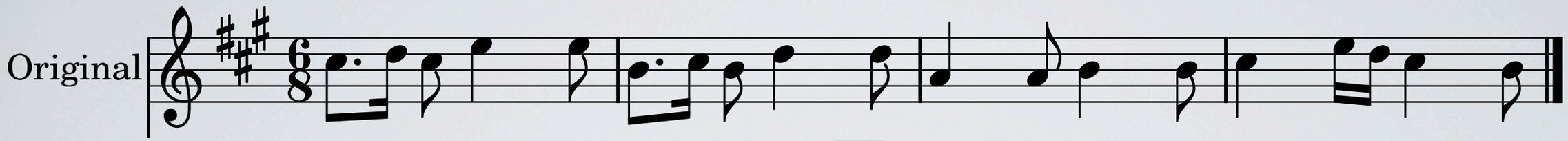
Theme

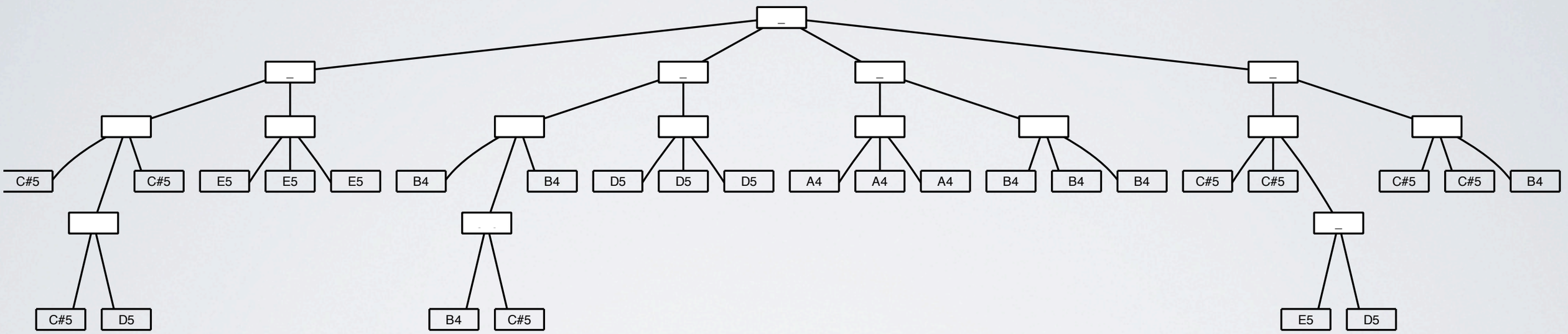
Original



Variation 1

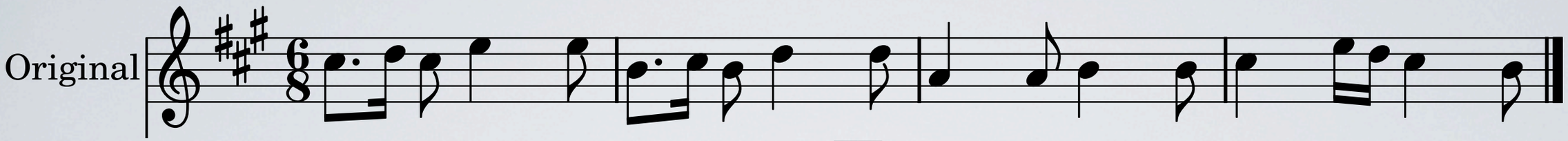
Theme

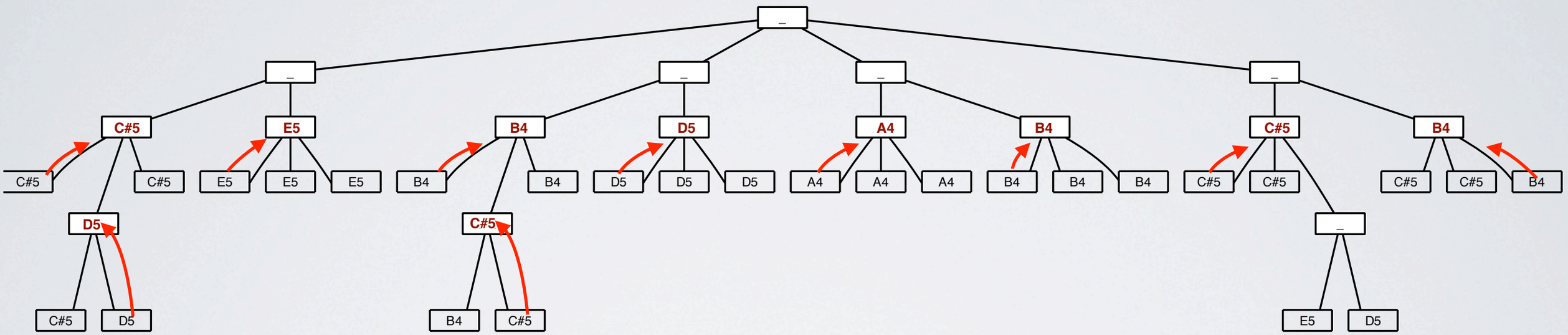




Variation 1

Theme



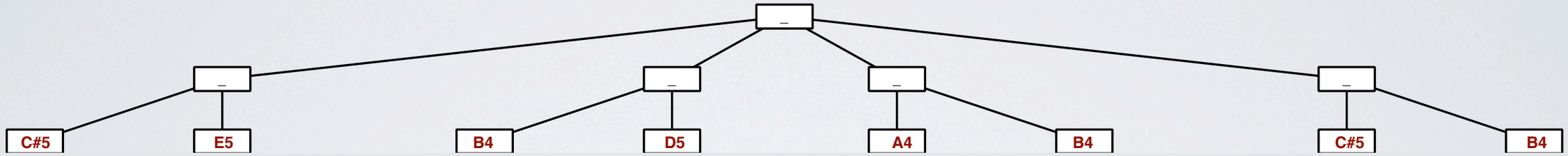


Propagate bottom-up important notes

Variation 1

Theme

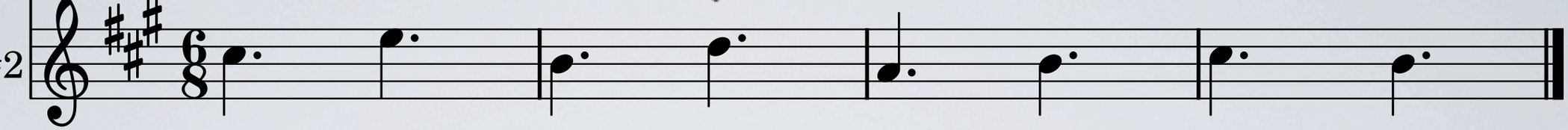
Original

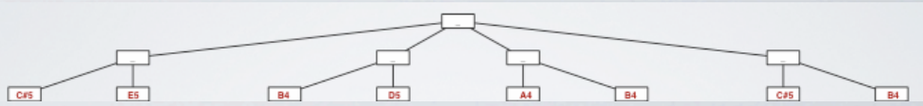


Reduction #2

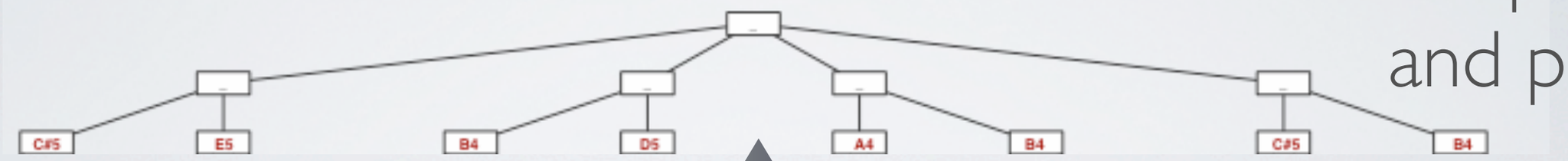
Variation 1

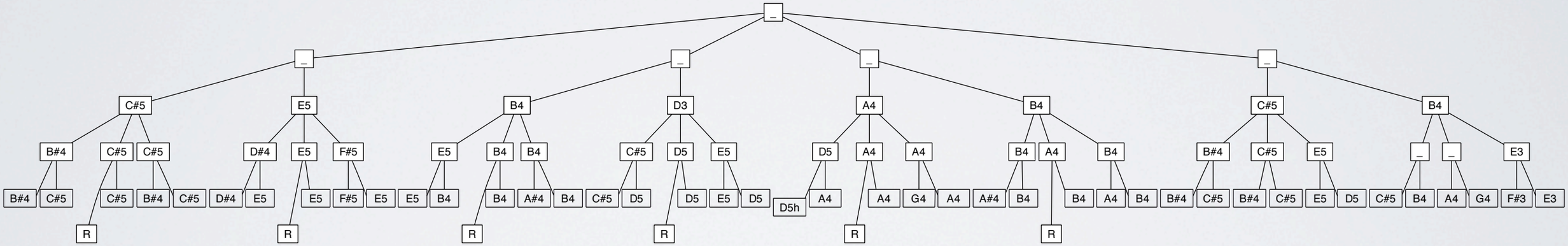
Reduction #2



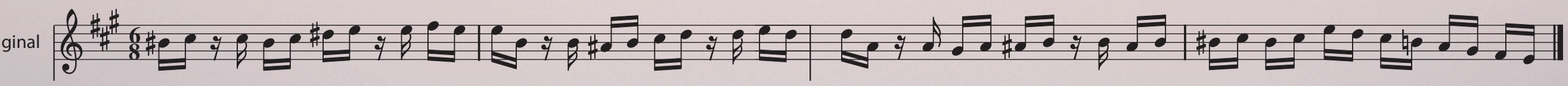


Propagate and prune

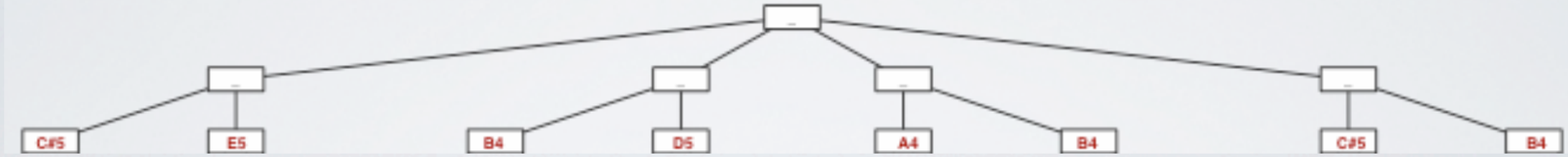




Variation I

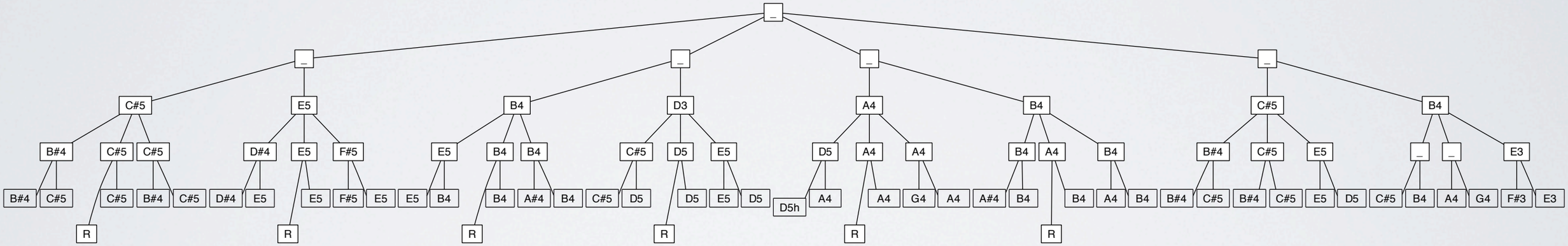


Reduction #2

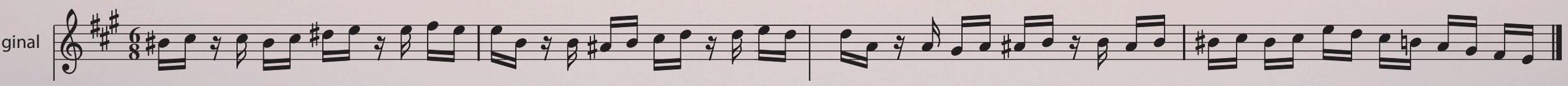


Theme tree = Variation tree

Propagate and prune



Variation I



PROPAGATION

PROPAGATION

- Previous example is an **ideal** hand made case

PROPAGATION

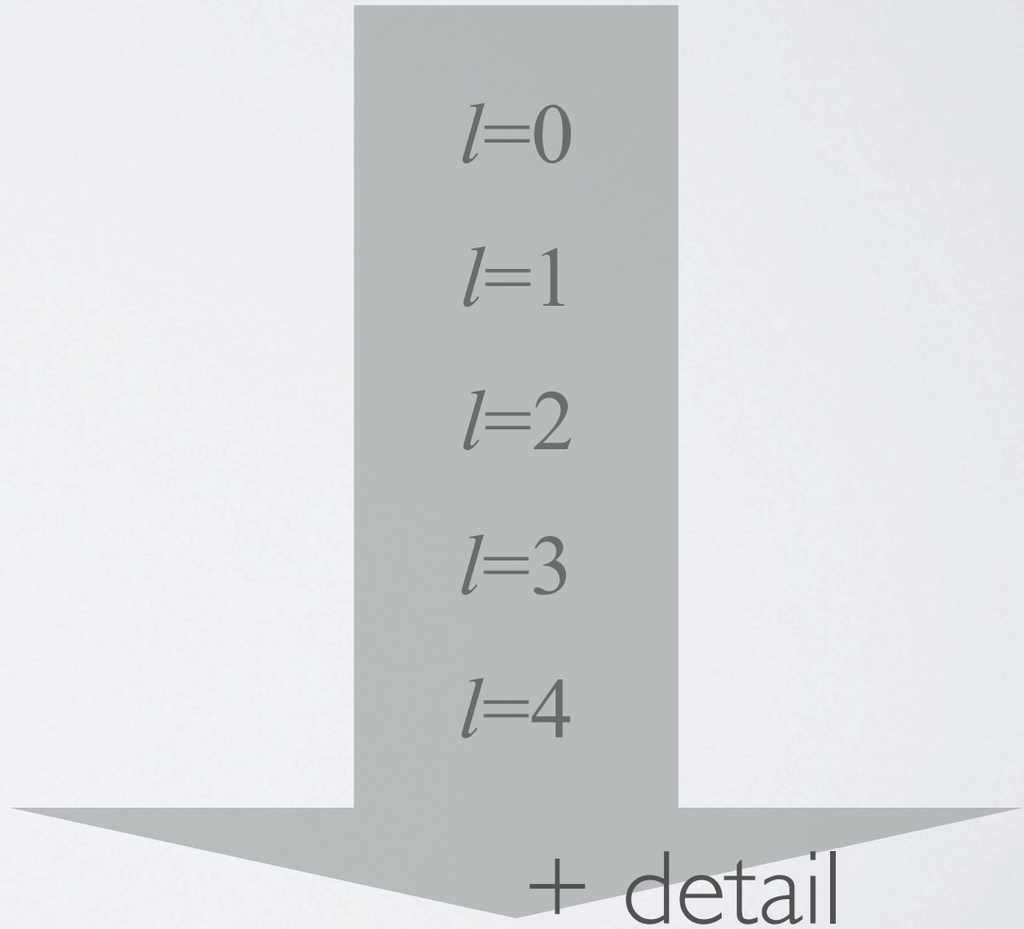
- Previous example is an **ideal** hand made case
- **How** perform propagation?

PROPAGATION

- Previous example is an **ideal** hand made case
- **How** perform propagation?
  - Two decisions must be taken

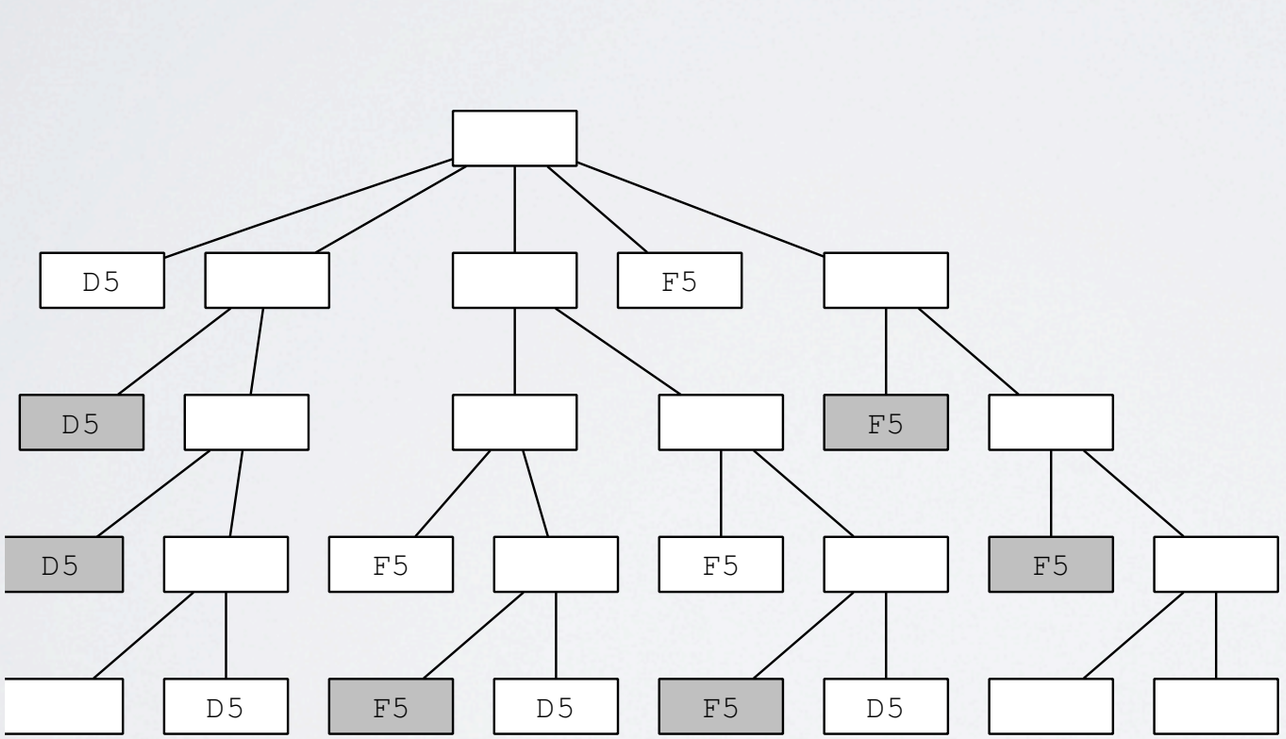
PROPAGATION

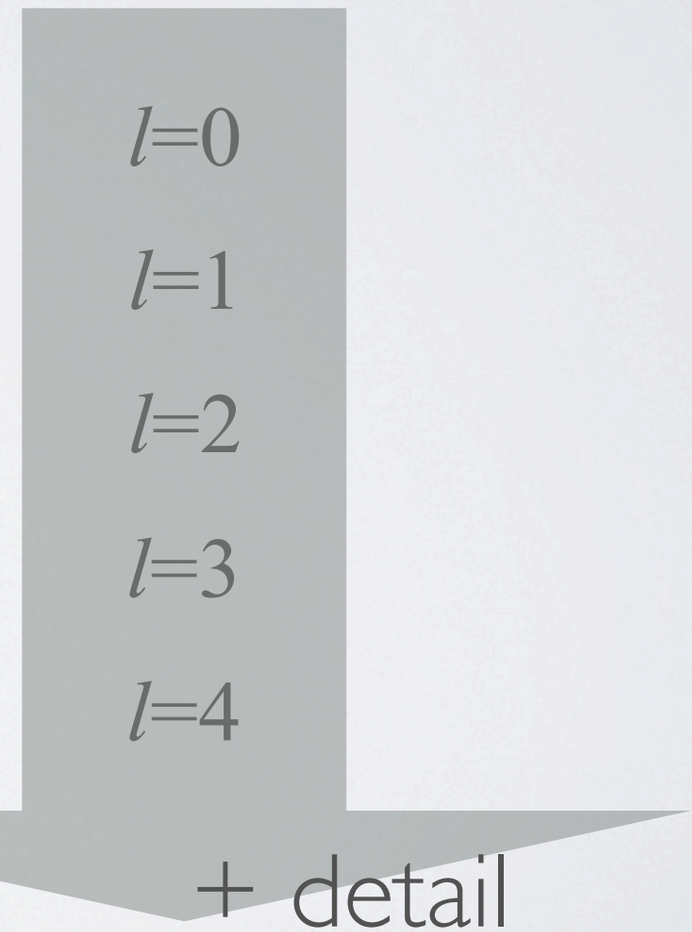
Decision 1:
what level prune?



PROPAGATION

Decision 1:
what level prune?



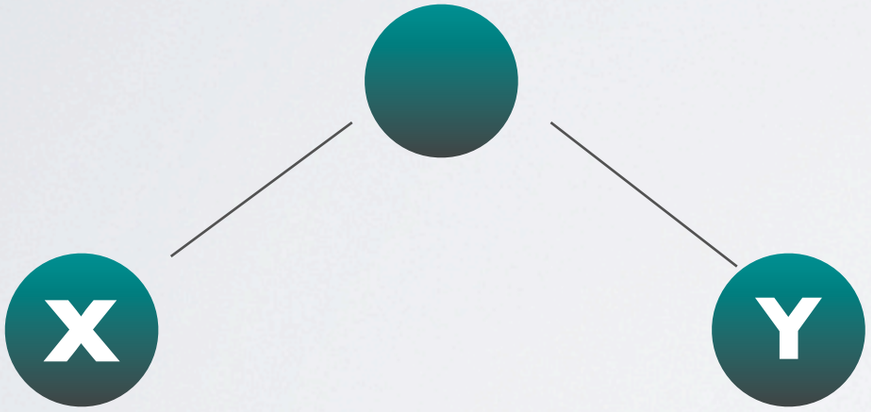


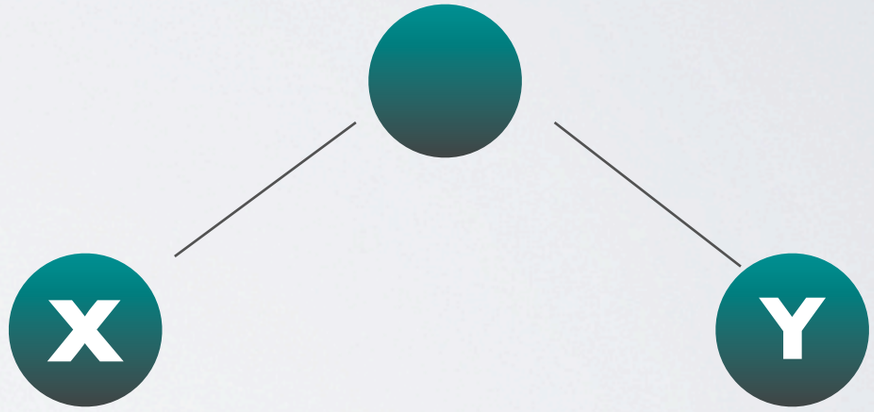
PROPAGATION

Decision 2:
Which node propagate?

PROPAGATION

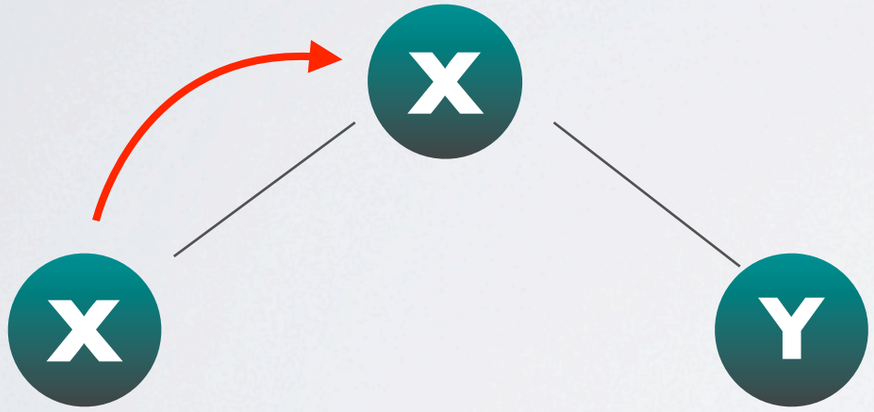
Decision 2:
Which node propagate?



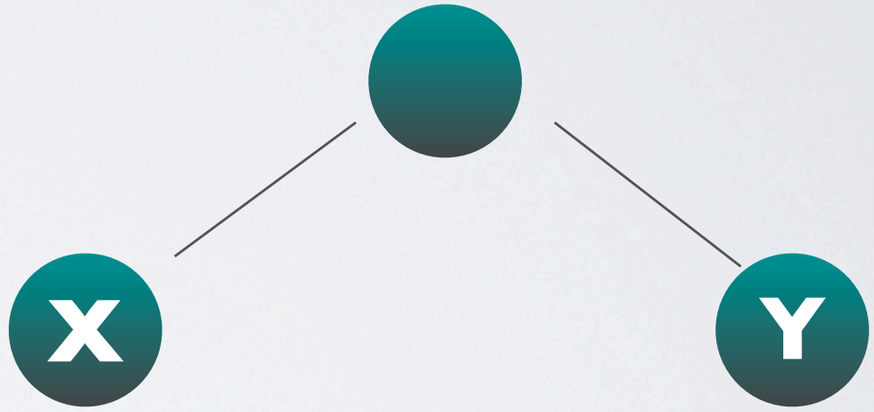


PROPAGATION

Decision 2:
Which node propagate?

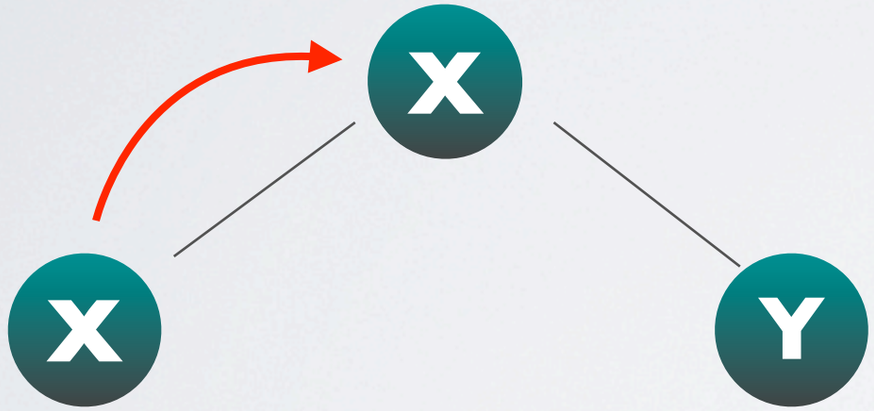


X is more important than **Y**



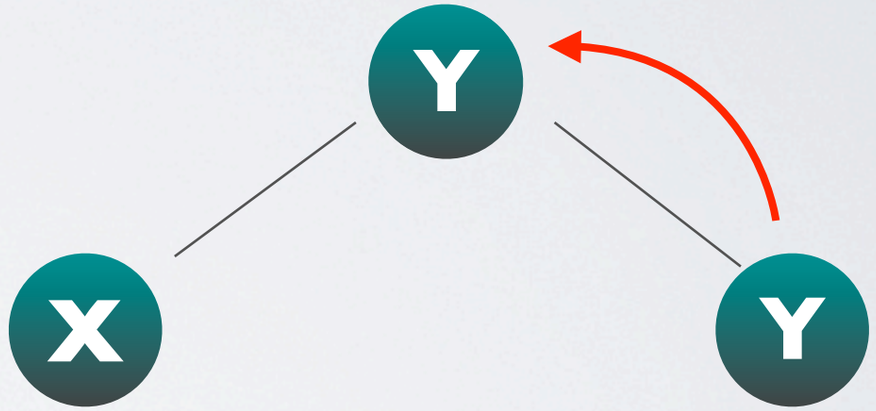
PROPAGATION

Decision 2:
Which node propagate?



X is more important than **Y**

or

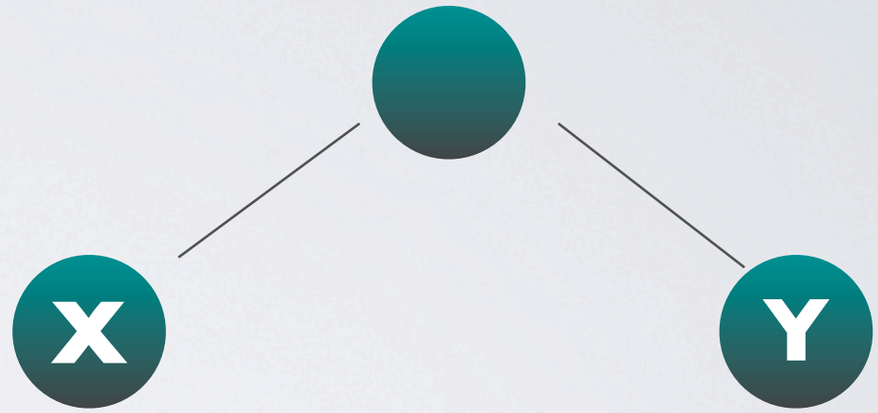


Y is more important than **X**

PROPAGATION

Several propagation schemes are proposed:

* Heuristic
* Melodic analysis
  * *Partial propagation*
* Always propagate left
* Always propagate right
* Schenker analysis



Decision:
Which node propagate?

PROPAGATION

Heuristic

Melodic analysis

Partial propagation

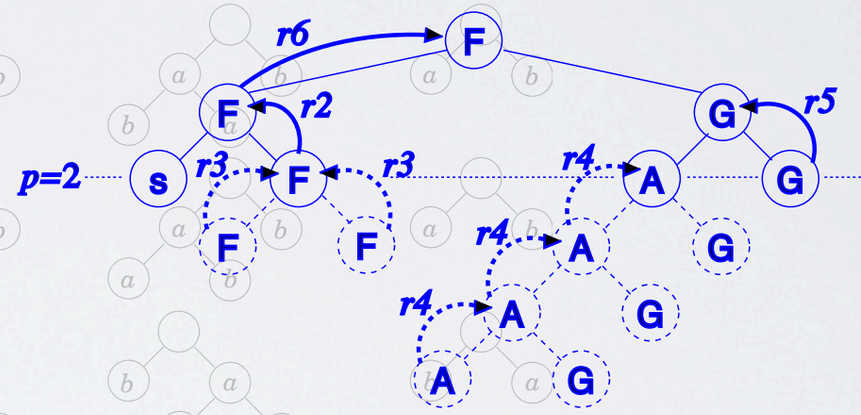
Always propagate left

Always propagate right

Schenker analysis

Decision:

Rules based on **empirical** experience



Consolidate repeated,

remove **rests**,

long notes over very short

left (strong) over right (weak)

PROPAGATION

Heuristic

Melodic analysis

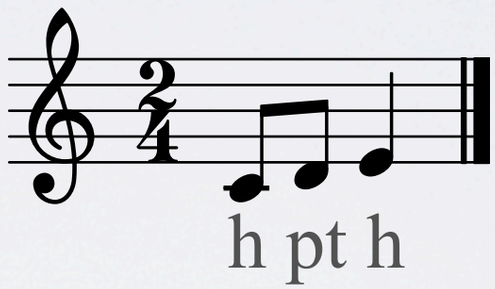
Partial propagation

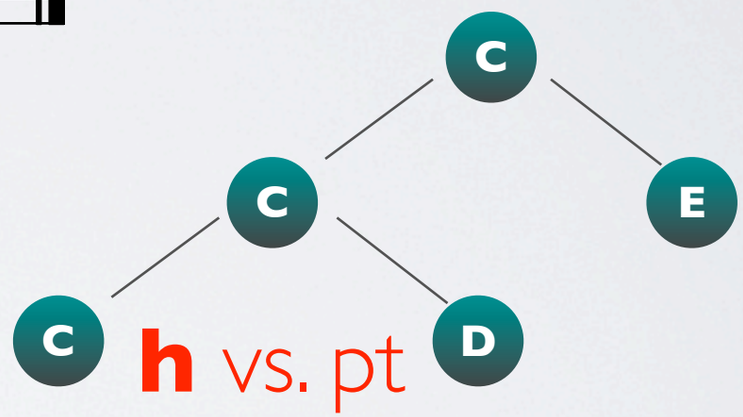
Always propagate left

Always propagate right

Schenker analysis

Decision:
Propagate harmonic notes over non-harmonic notes





PROPAGATION

Heuristic

Melodic analysis

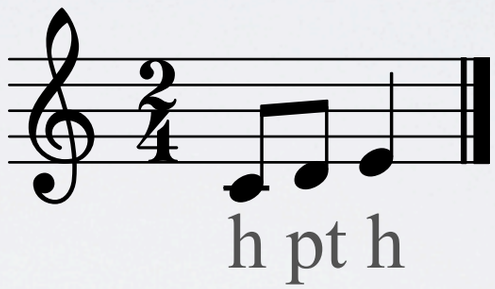
Partial propagation

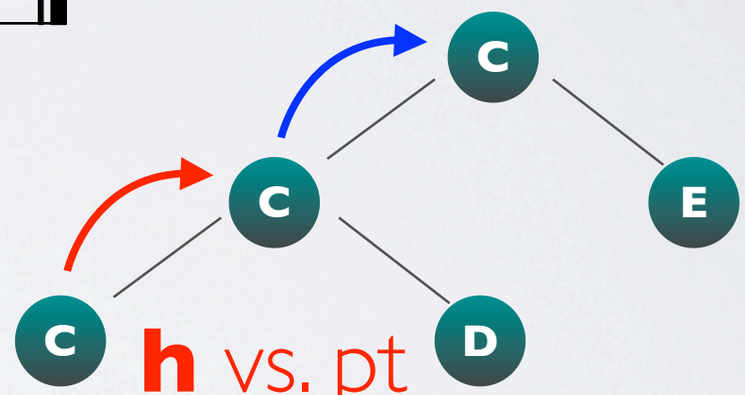
Always propagate left

Always propagate right

Schenker analysis

Decision:
Propagate harmonic notes over non-harmonic notes





PROPAGATION

Heuristic

Melodic analysis

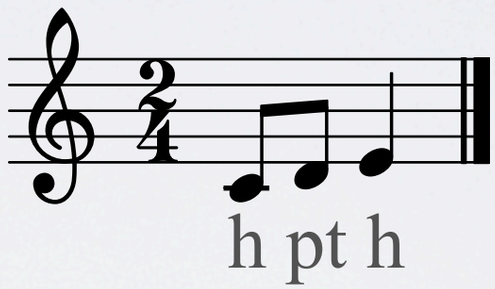
Partial propagation

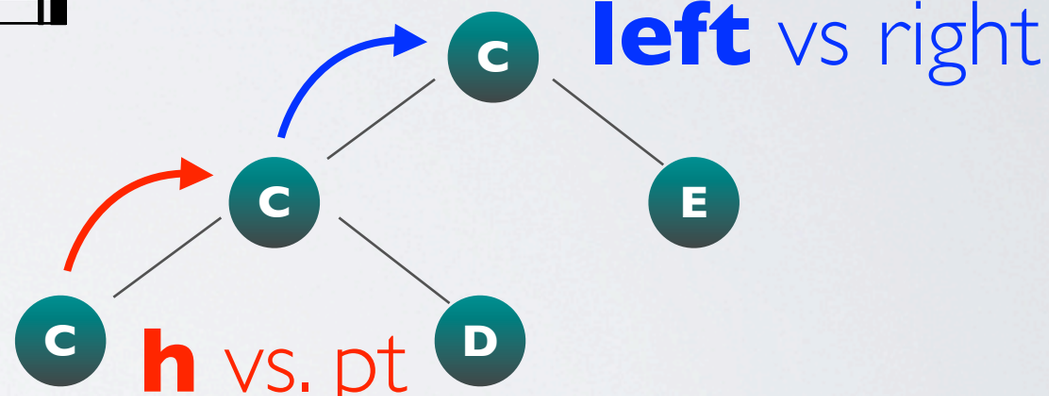
Always propagate left

Always propagate right

Schenker analysis

Decision:
Propagate harmonic notes over non-harmonic notes





When both are harmonic, propagate left

PROPAGATION

Heuristic

Melodic analysis

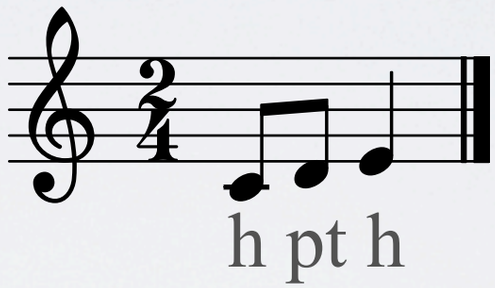
Partial propagation

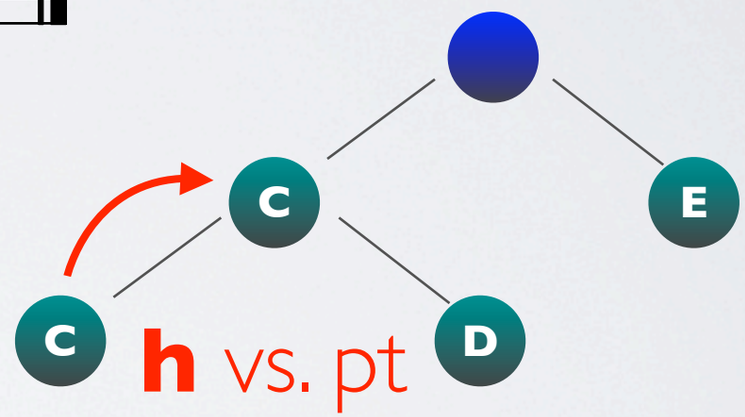
Always propagate left

Always propagate right

Schenker analysis

Decision:
Propagate harmonic notes over non-harmonic notes





When both are harmonic,
don't propagate

PROPAGATION

Heuristic

Melodic analysis

Partial propagation

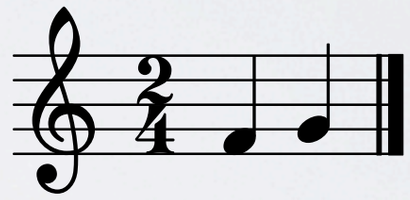
Always propagate left

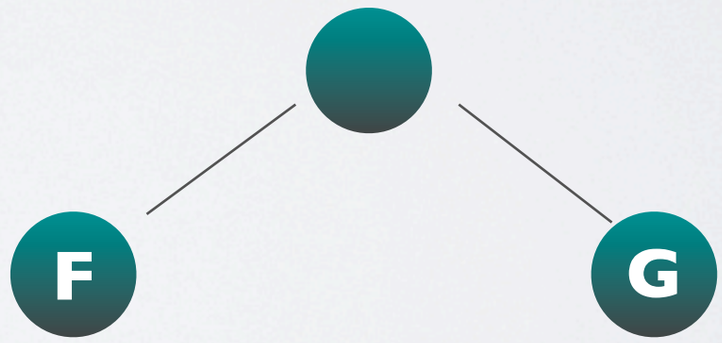
Always propagate right

Schenker analysis

Left over right

Importance to stronger notes





PROPAGATION

Heuristic

Melodic analysis

Partial propagation

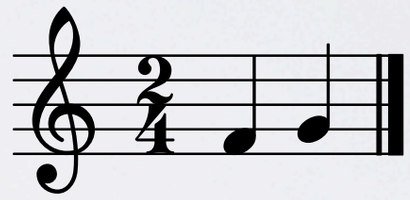
Always propagate left

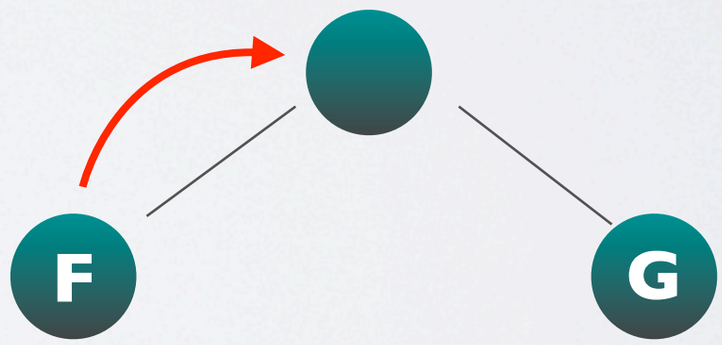
Always propagate right

Schenker analysis

Left over right

Importance to stronger notes





PROPAGATION

Heuristic

Melodic analysis

Partial propagation

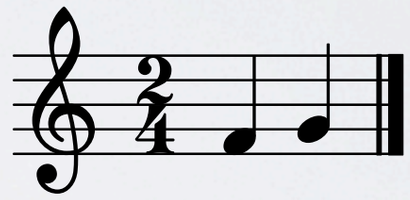
Always propagate left

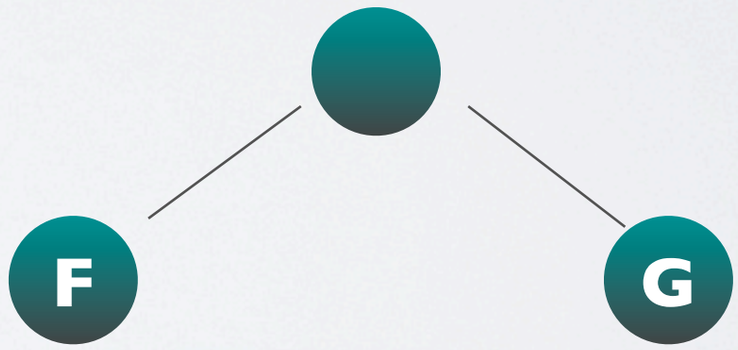
Always propagate right

Schenker analysis

Right over left

Importance to weaker notes





PROPAGATION

Heuristic

Melodic analysis

Partial propagation

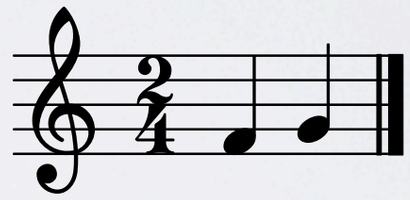
Always propagate left

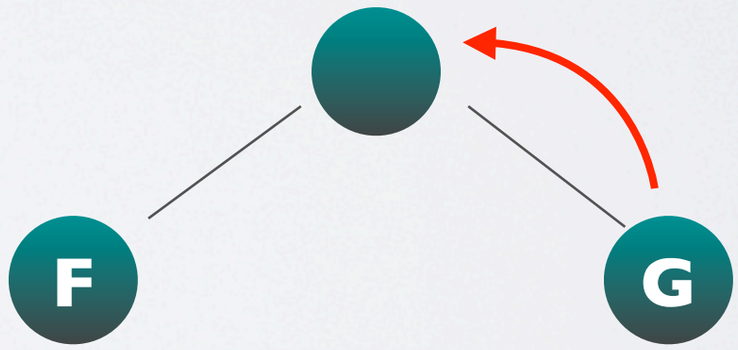
Always propagate right

Schenker analysis

Right over left

Importance to weaker notes





PROPAGATION

Heuristic

Melodic analysis

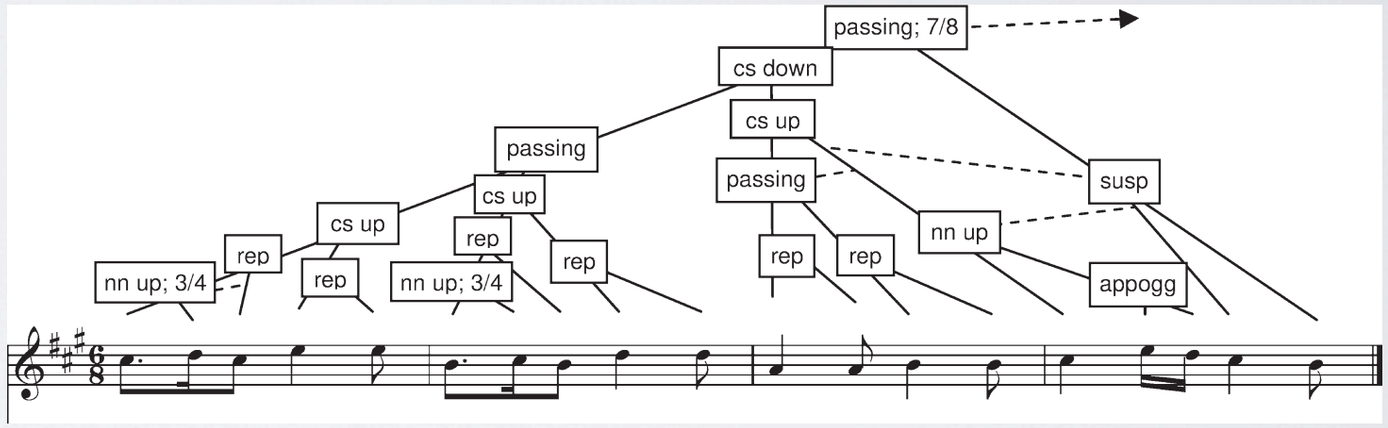
Partial propagation

Always propagate left

Always propagate right

Schenker analysis

≈



Computer approach by Alan Marsden

PROPAGATION

Heuristic

Melodic analysis

Partial propagation

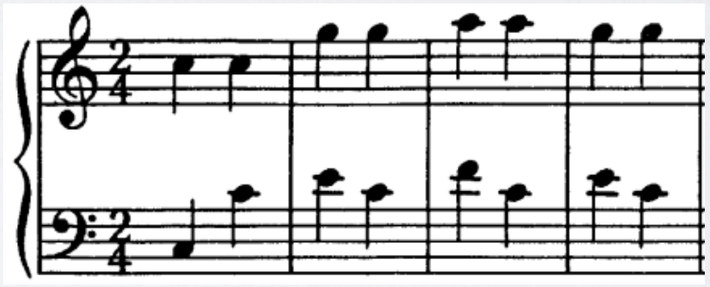
Always propagate left

Always propagate right

Schenker analysis

Computer approach by Alan Marsden

G5 C5 C3 C4							
G5 C5 C3 C4				G5 C4			
C5 C3		G5 C4		A5 C4		G5 C4	
C5 C3	C5 C4	G5 E4	G5 C4	A5 F4	A5 C4	G5 E4	G5 C4



- surface = actual notes
- reduction ‘chart’ with all possible reductions

PROPAGATION

Computer approach by Alan Marsden

Heuristic

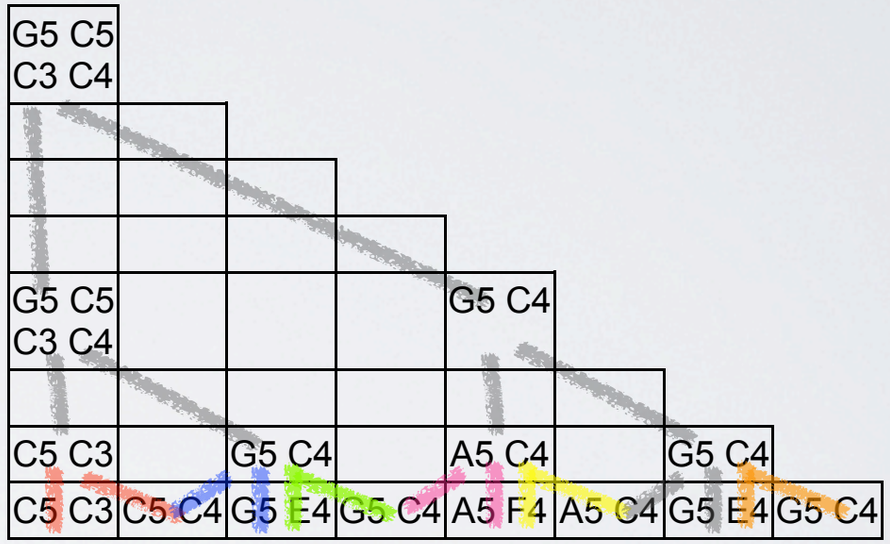
Melodic analysis

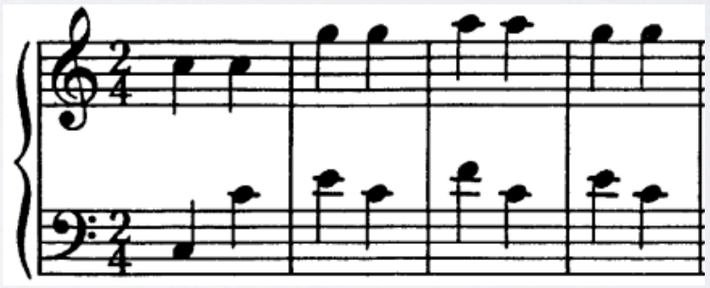
Partial propagation

Always propagate left

Always propagate right

Schenker analysis





- surface = actual notes
- reduction ‘chart’ with all possible reductions

PROPAGATION

Heuristic

Melodic analysis

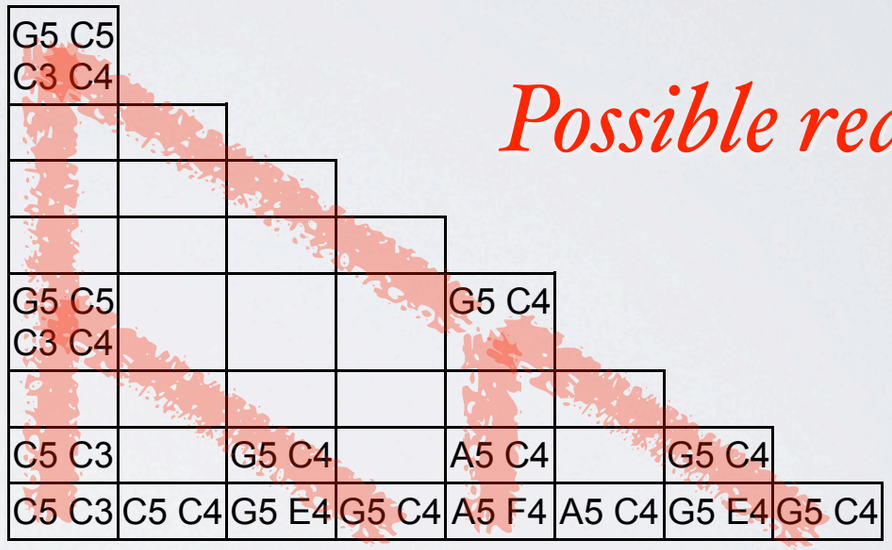
Partial propagation

Always propagate left

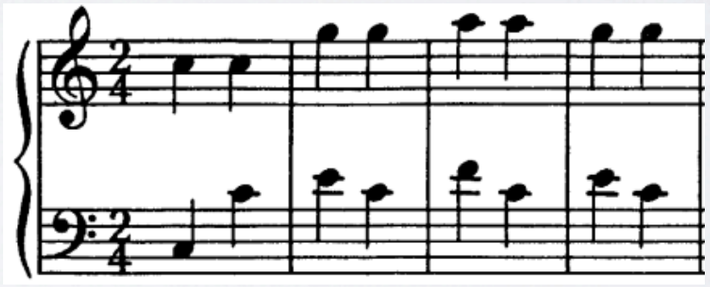
Always propagate right

Schenker analysis

Computer approach by Alan Marsden



Possible reduction



- surface = actual notes
- reduction 'chart' with all possible reductions

PROPAGATION

Heuristic

Melodic analysis

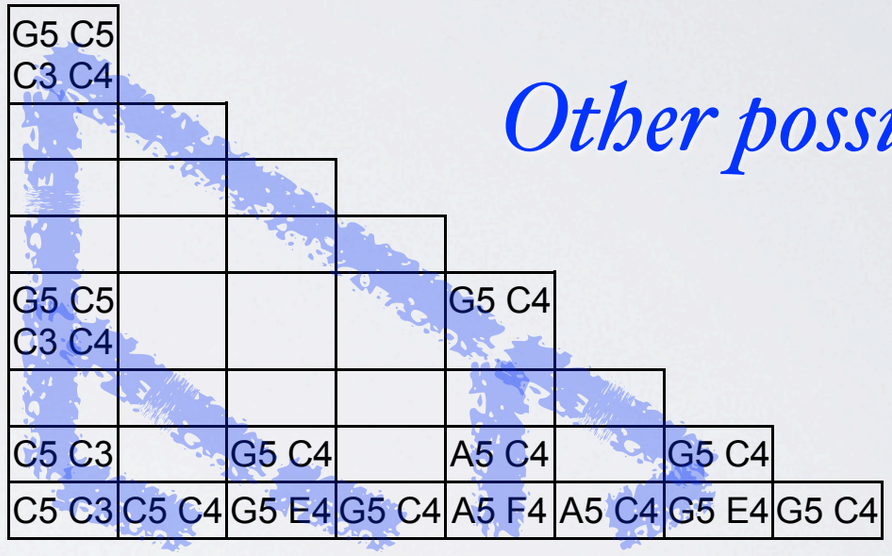
Partial propagation

Always propagate left

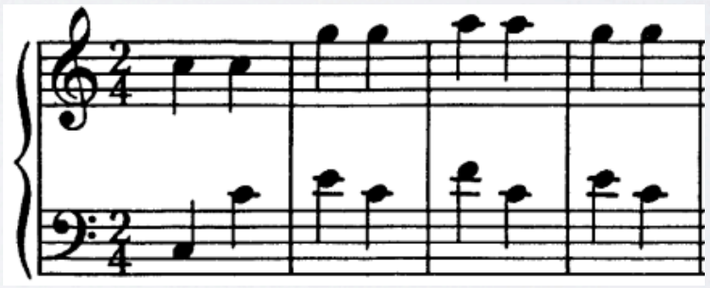
Always propagate right

Schenker analysis

Computer approach by Alan Marsden



Other possibility



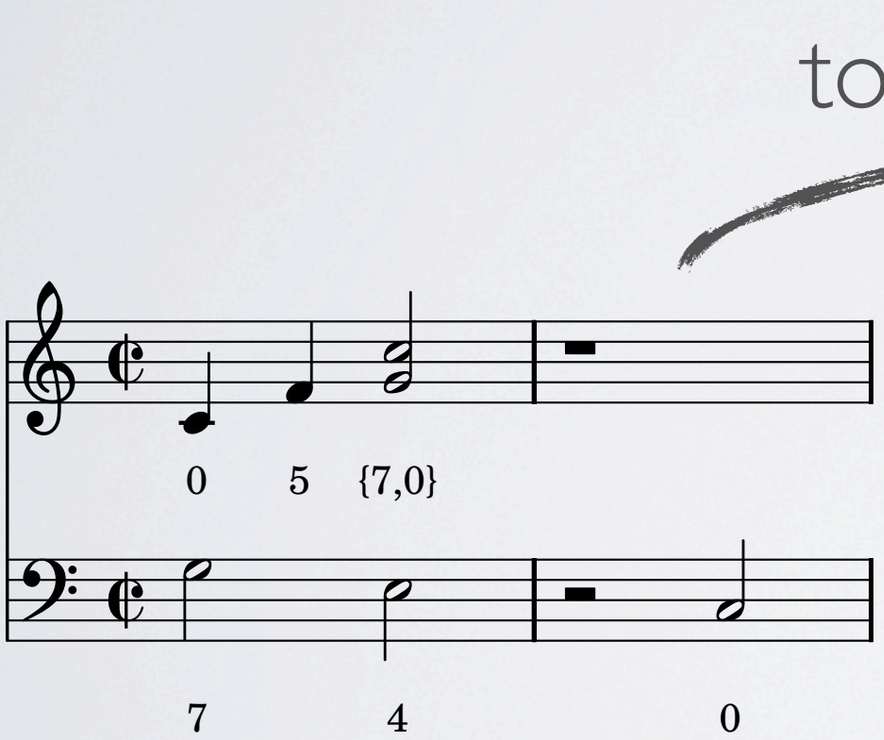
- surface = actual notes
- reduction 'chart' with all possible reductions

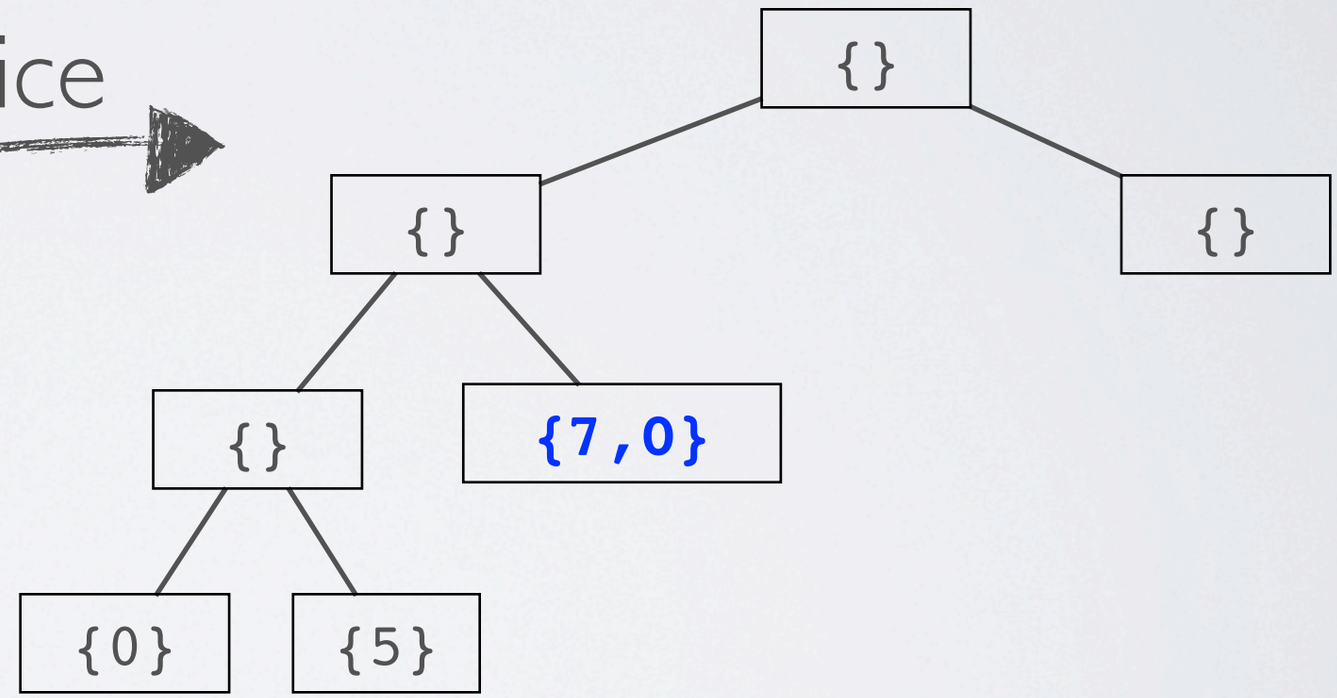
POLYPHONIC TREES

POLYPHONIC TREES

- Merge all voices in the same tree

top voice

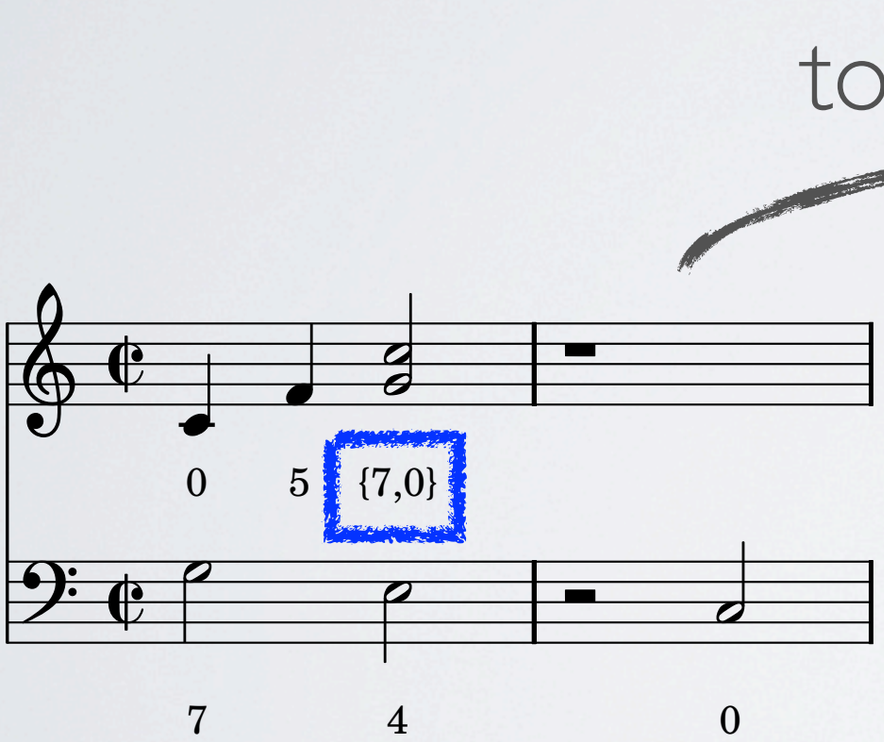


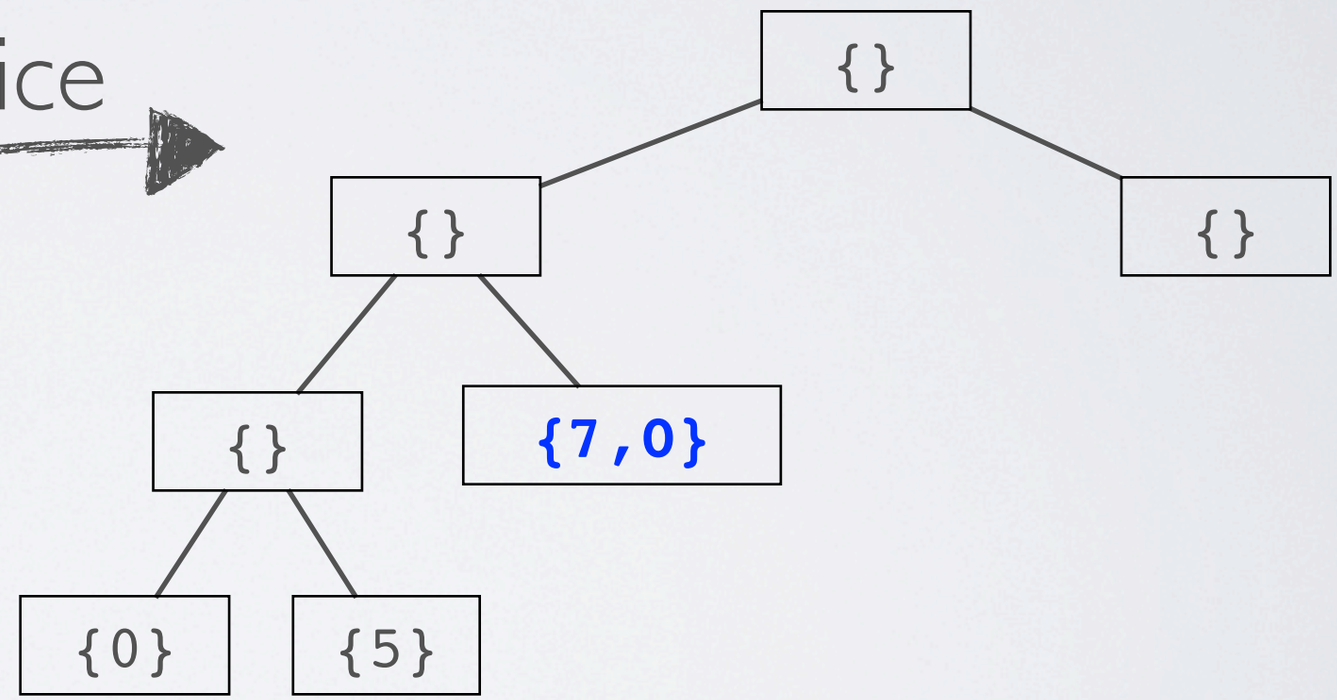


POLYPHONIC TREES

- Merge all voices in the same tree
- Chord notes share node
  - node labels are now sets of pitch classes

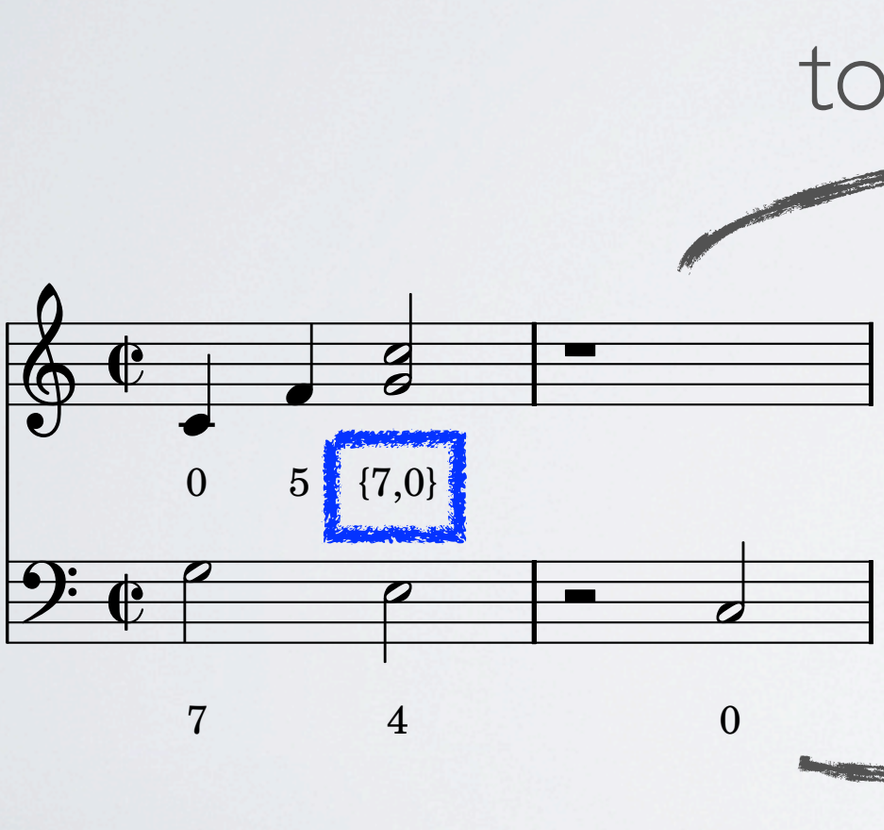
top voice

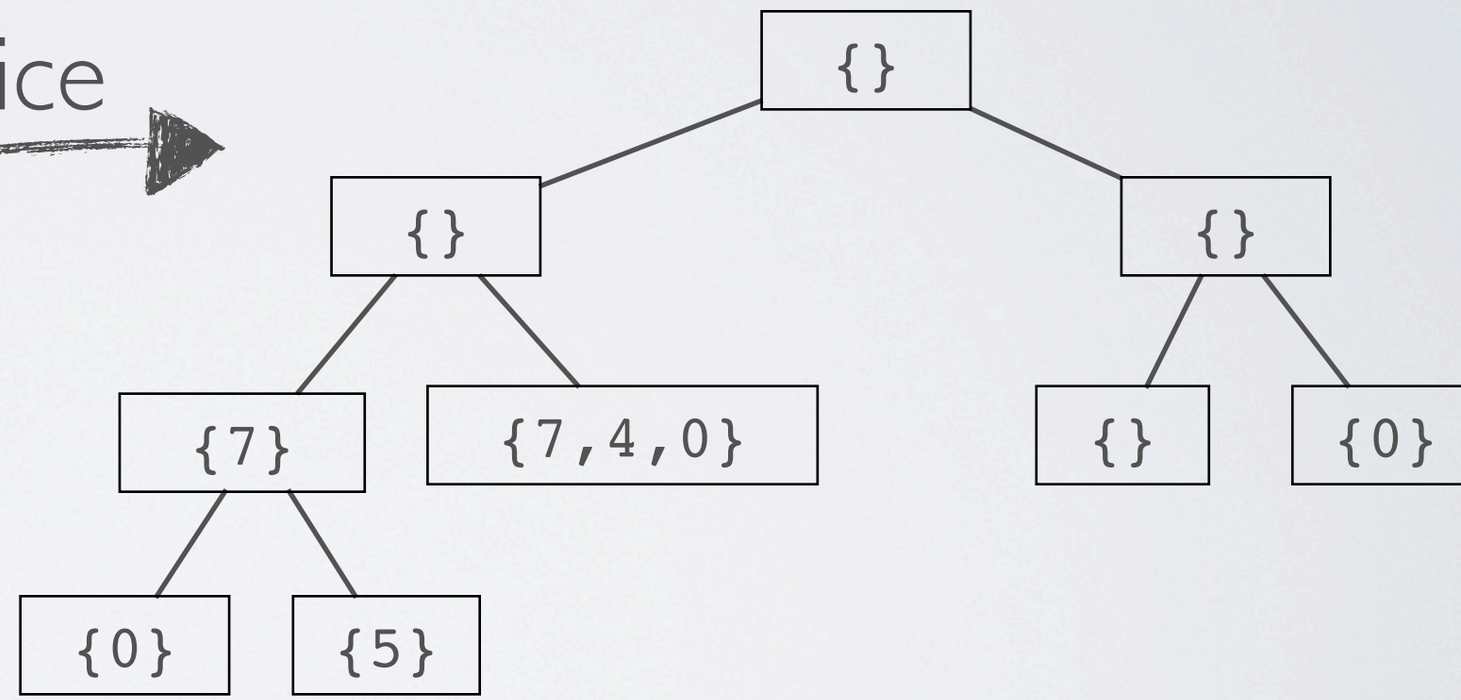




POLYPHONIC TREES

- Merge all voices in the same tree
- Chord notes share node
  - node labels are now sets of pitch classes

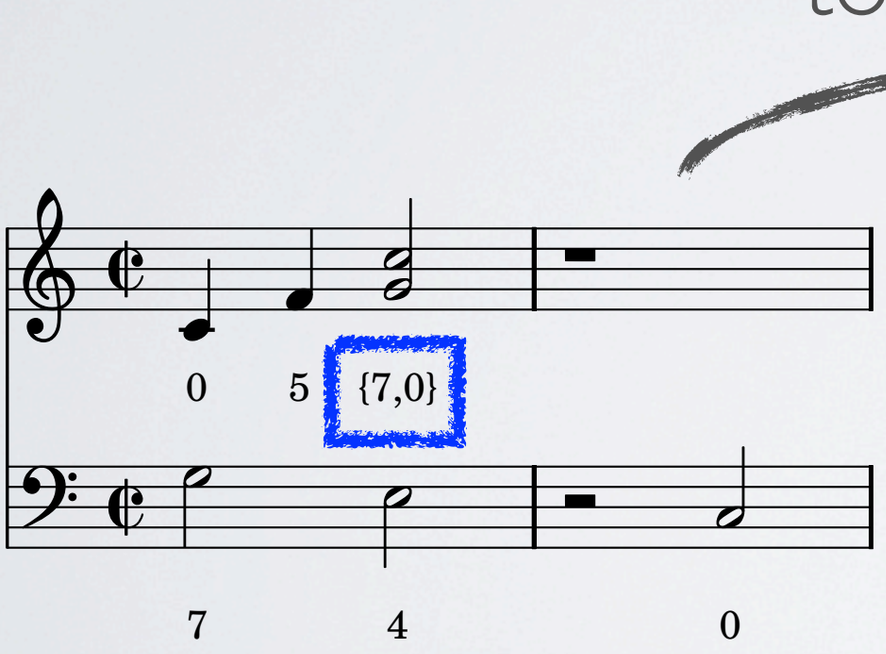




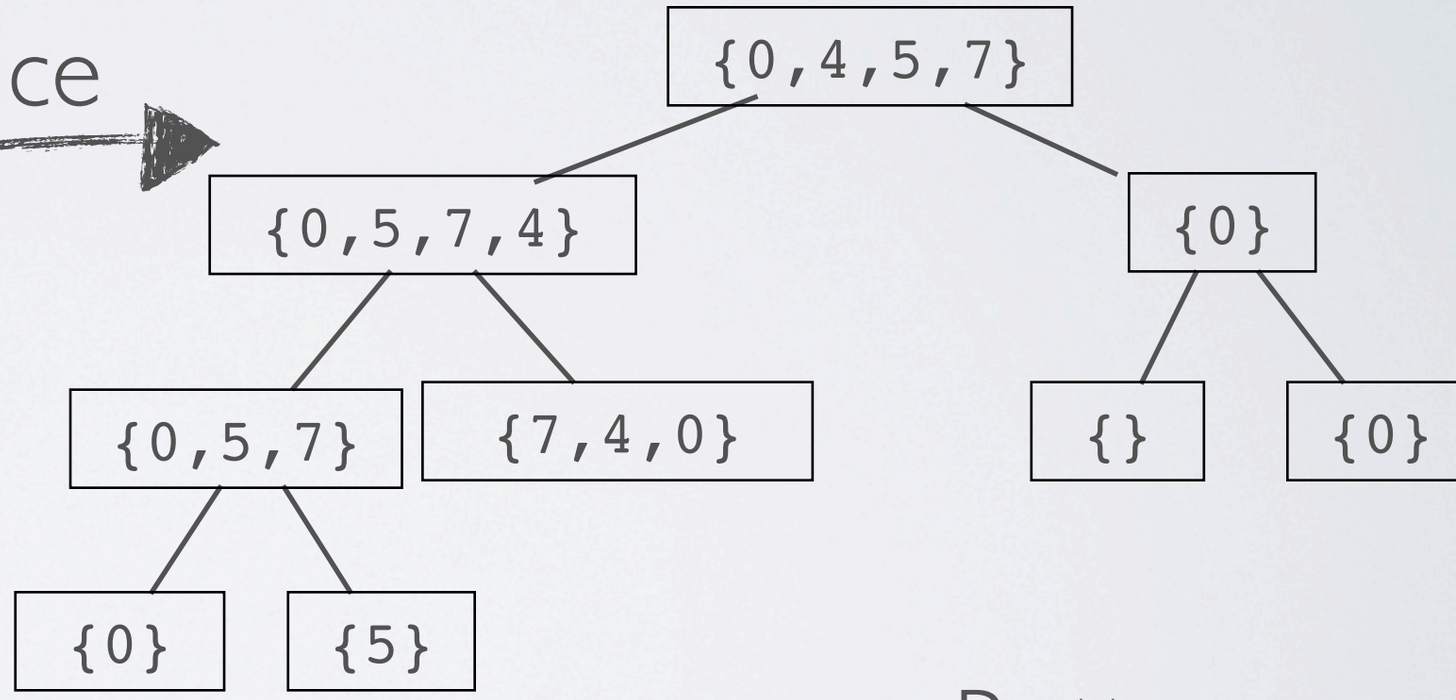
POLYPHONIC TREES

- Merge all voices in the same tree
- Chord notes share node
  - node labels are now sets of pitch classes

top voice



bottom voice



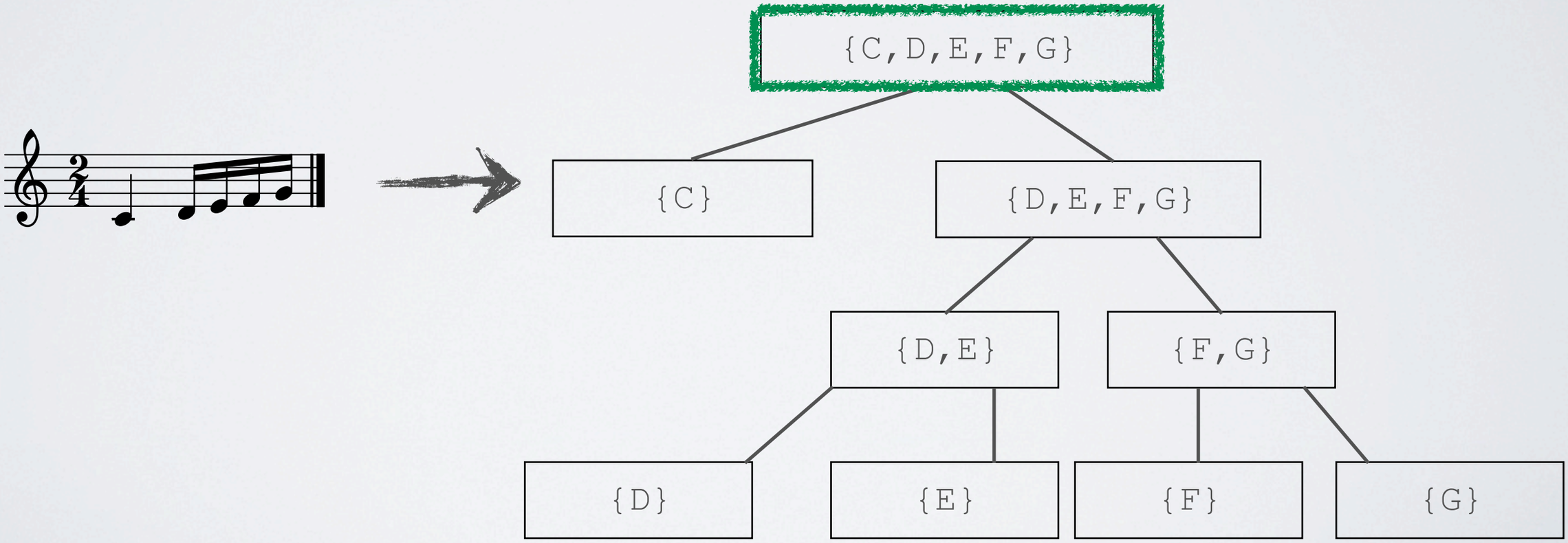
Bottom up propagation using set union

POLYPHONIC TREES

Problem:
set union does not respect durations

POLYPHONIC TREES

Problem:
set union does not respect durations



POLYPHONIC TREES

Solution:
use multisets to encode cardinalities

set algebra: {X} ∪ {X,Y} = {X,Y}

POLYPHONIC TREES

Solution:
use multisets to encode cardinalities

set algebra: $\{X\} \cup \{X,Y\} = \{X,Y\}$

multiset algebra: set $\{X\}$ as $\{\{X\},\{1\}\}$

set $\{X,Y\}$ as $\{\{X,Y\},\{1,1\}\}$

$\{\{X\},\{1\}\} \cup \{\{X,Y\},\{1,1\}\} = \{\{X,Y\},\{2,1\}\}$

POLYPHONIC TREES

Solution:
use multisets to encode cardinalities

set algebra: $\{X\} \cup \{X,Y\} = \{X,Y\}$

multiset algebra: set $\{X\}$ as $\{\{X\},\{1\}\}$

set $\{X,Y\}$ as $\{\{X,Y\},\{1,1\}\}$

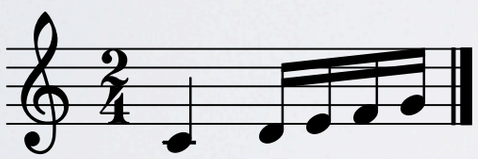
$\{\{X\},\{1\}\} \cup \{\{X,Y\},\{1,1\}\} = \{\{X,Y\},\{2,1\}\}$

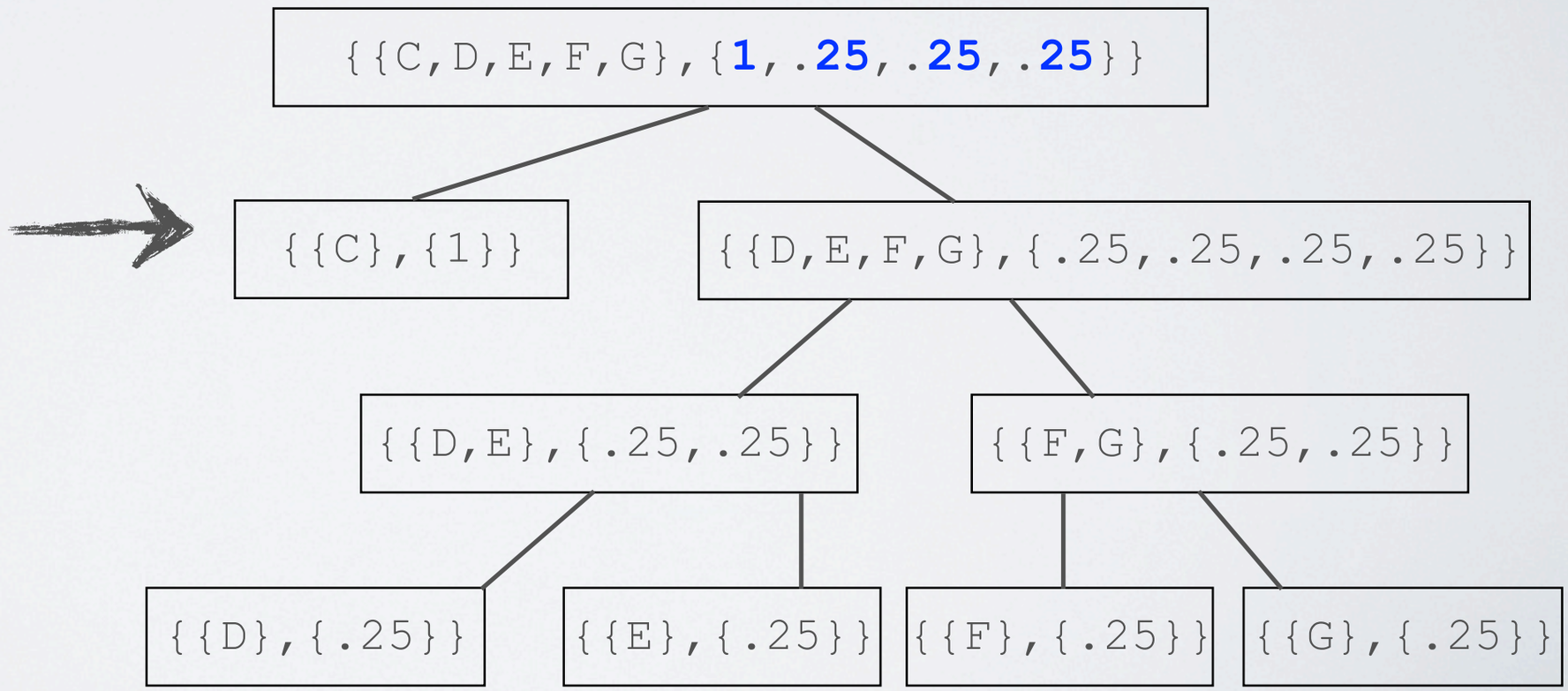
Cardinalities in multisets = note durations

POLYPHONIC TREES

Solution:
use multisets to encode cardinalities

Cardinalities in multisets = normalized note durations





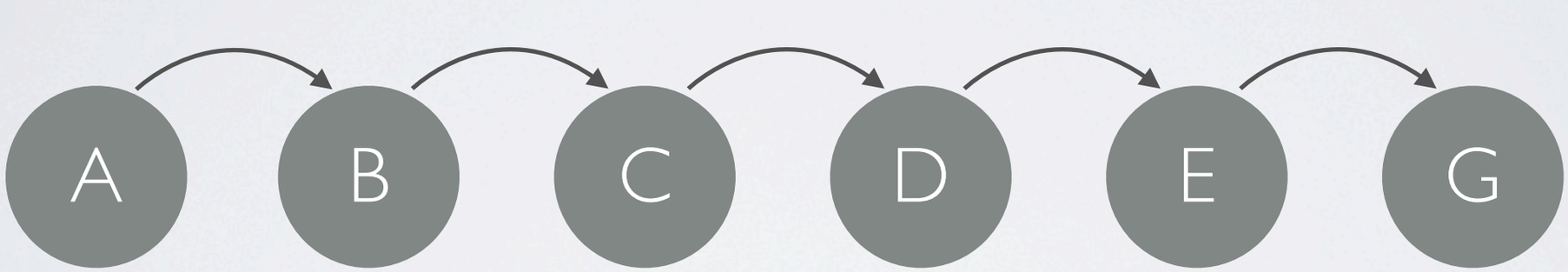
PROBABILISTIC K-TESTABLE TREE LANGUAGES

TREE AUTOMATA

- String languages generated and recognized by automata
- Tree languages generated and recognized by tree automata

K-TESTABILITY

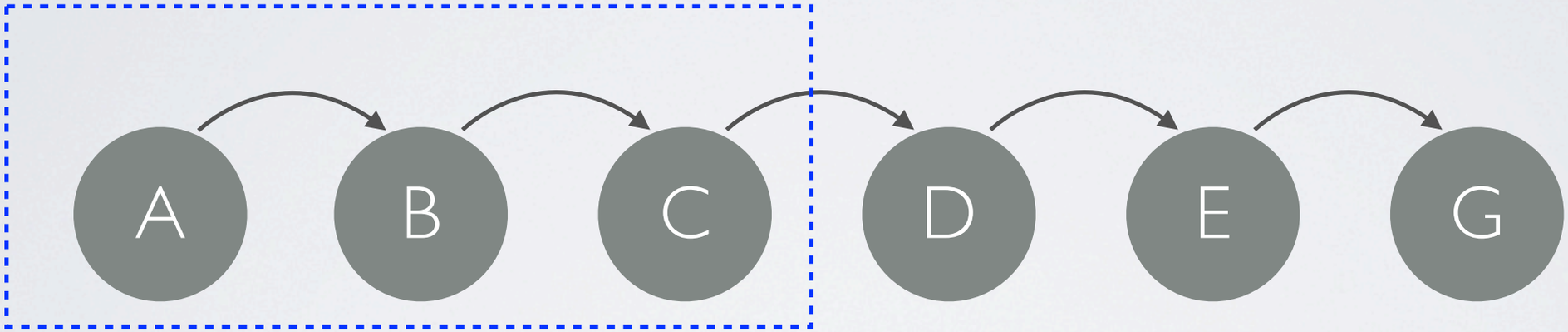
- String: use k symbols to say a string ∈ language
- If probabilities are added ⇒ *k*-grams



- Trees: k-testables

K-TESTABILITY

- String: use k symbols to say a string ∈ language
- If probabilities are added ⇒ *k*-grams

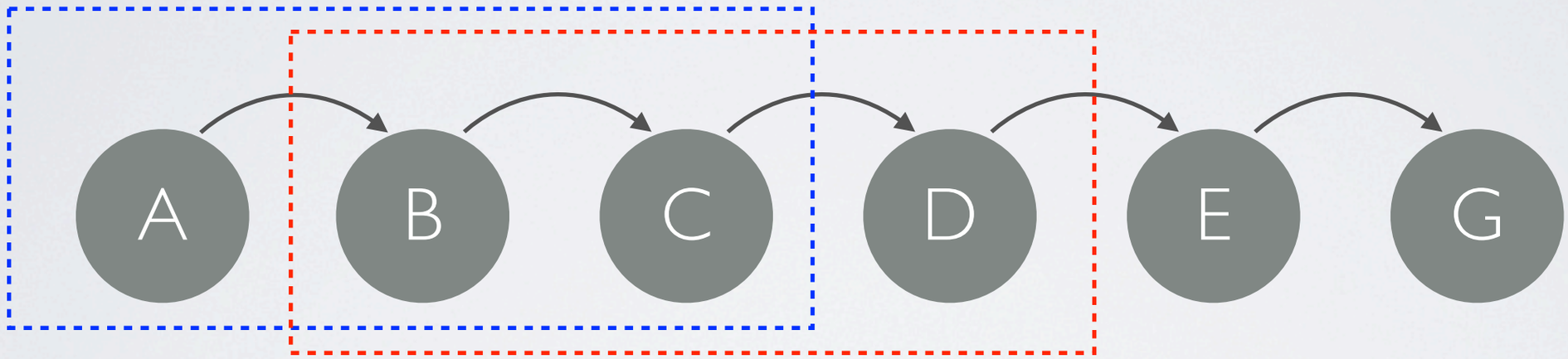


ABC

- Trees: k-testables

K-TESTABILITY

- String: use k symbols to say a string ∈ language
- If probabilities are added ⇒ *k*-grams

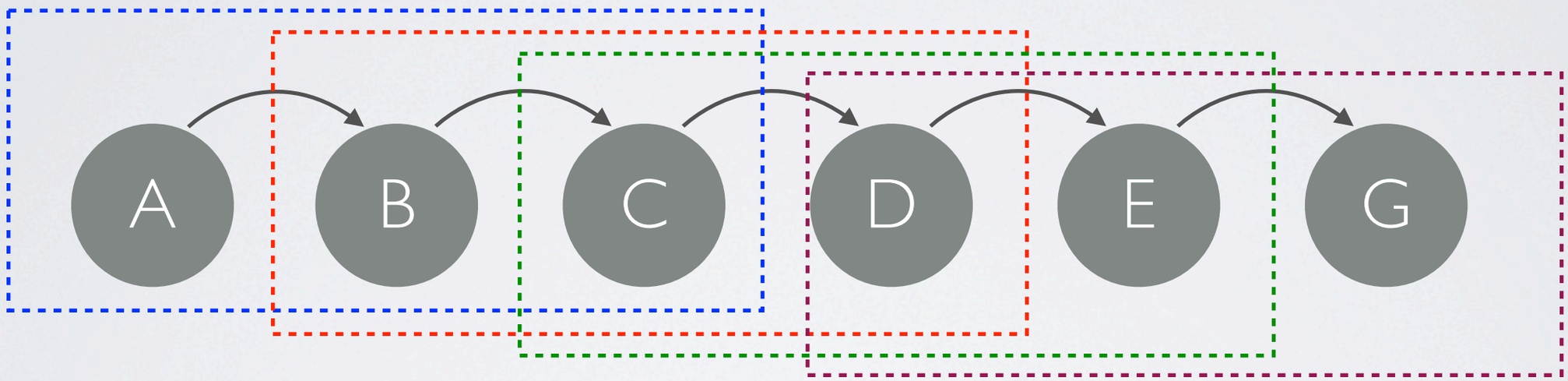


ABC BCD

- Trees: k-testables

K-TESTABILITY

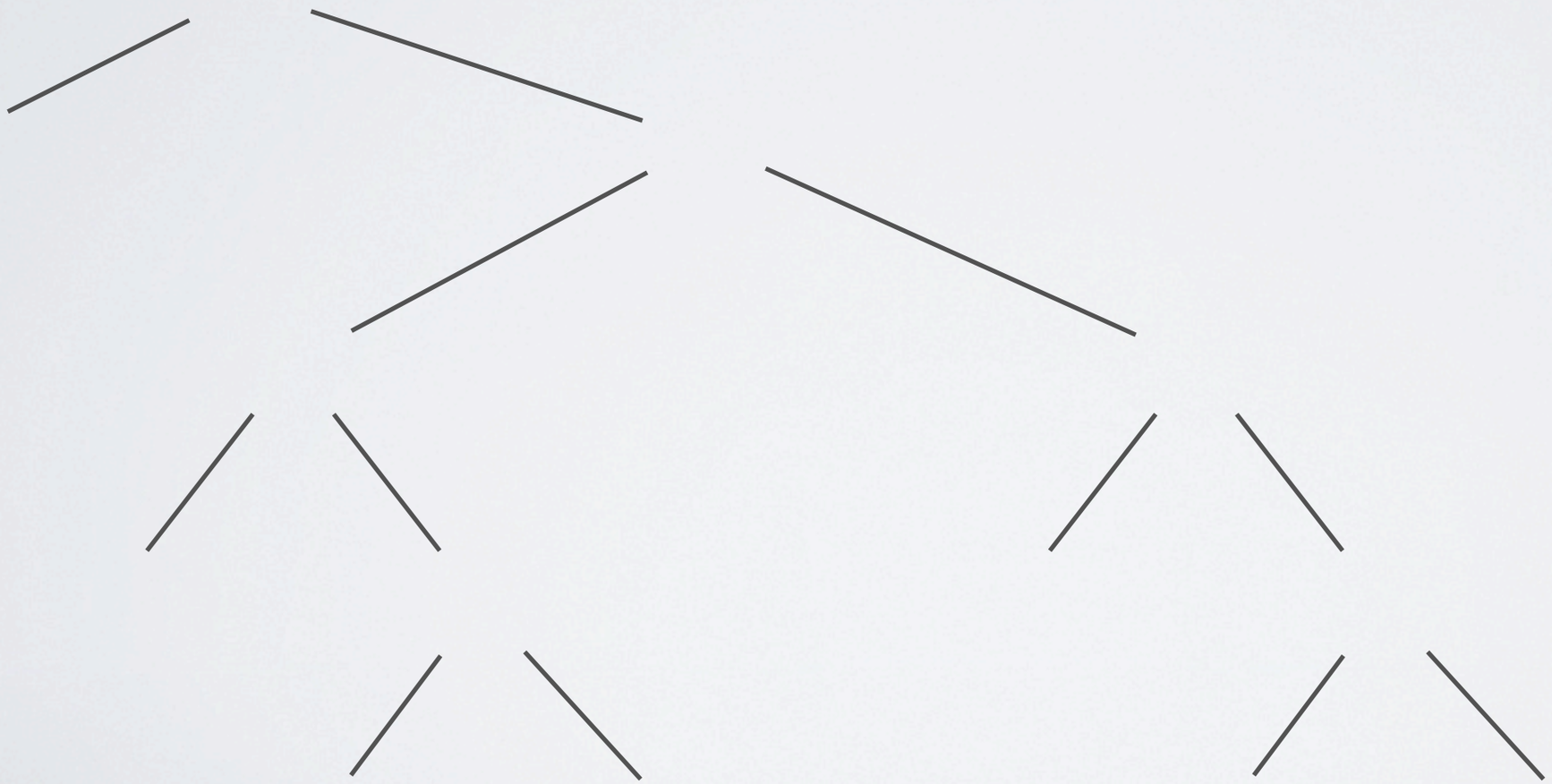
- String: use k symbols to say a string ∈ language
- If probabilities are added ⇒ *k*-grams



- Trees: k-testables

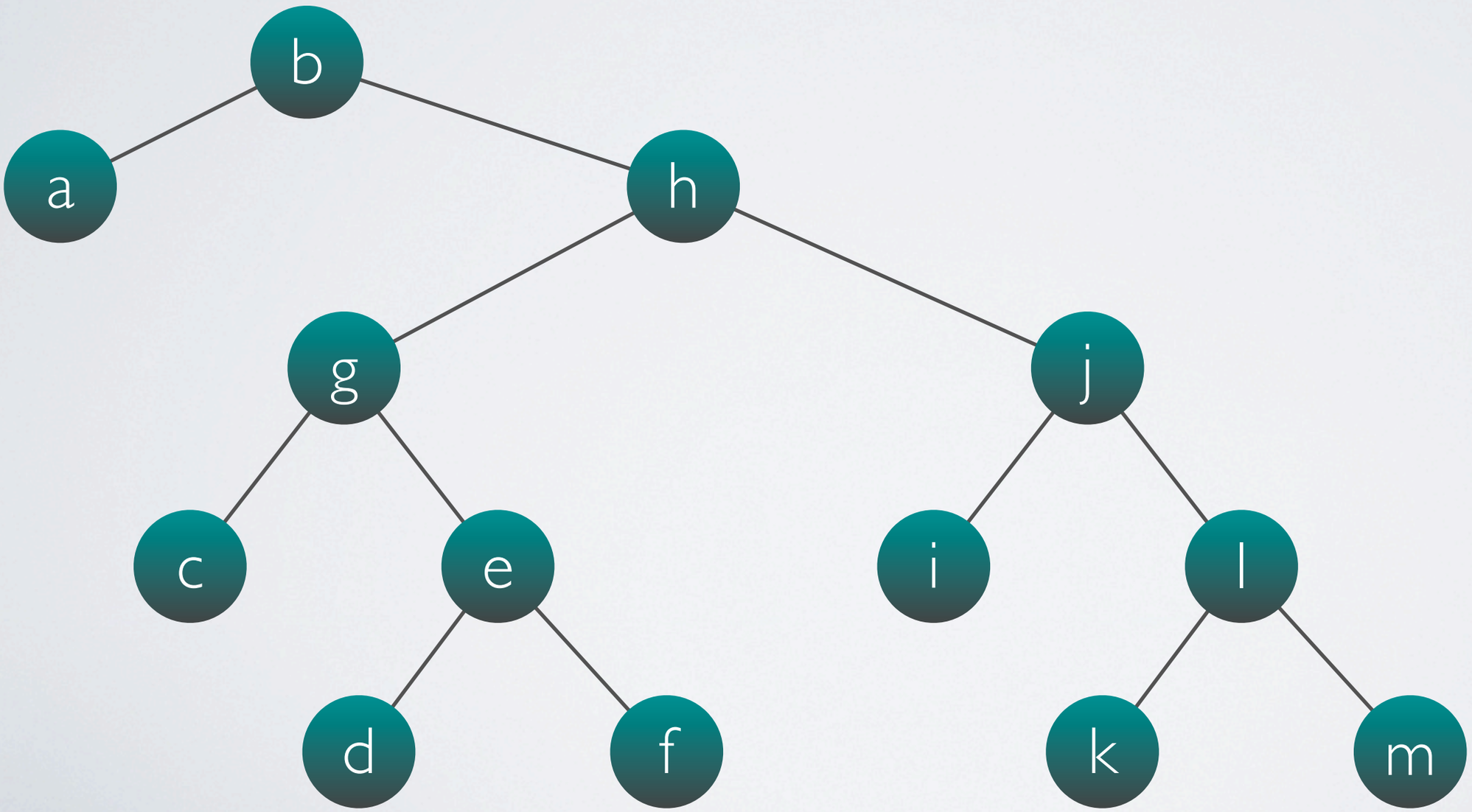
K-TESTABILITY

- Trees: k-testables



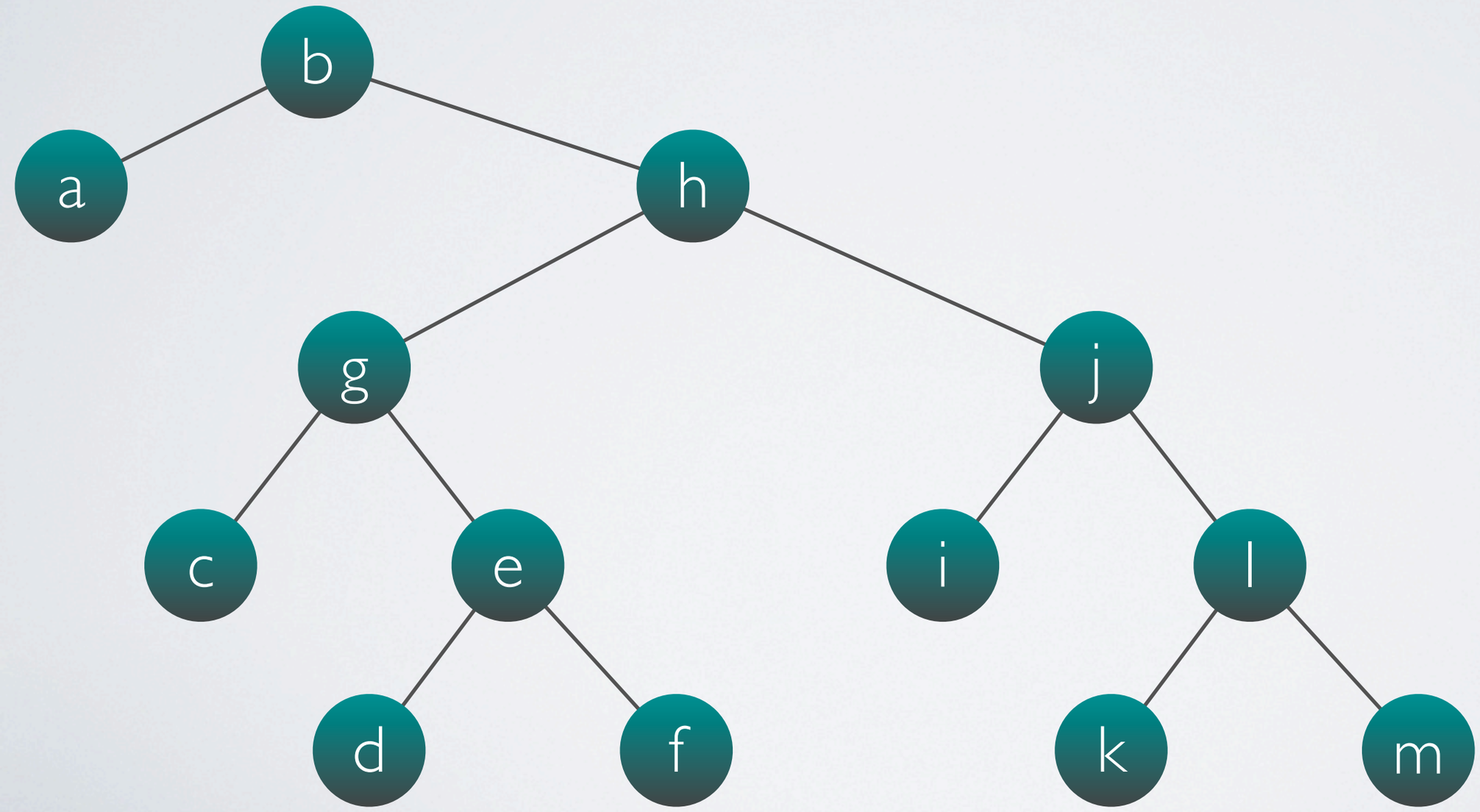
K-TESTABILITY

- Trees: k-testables



K-TESTABILITY

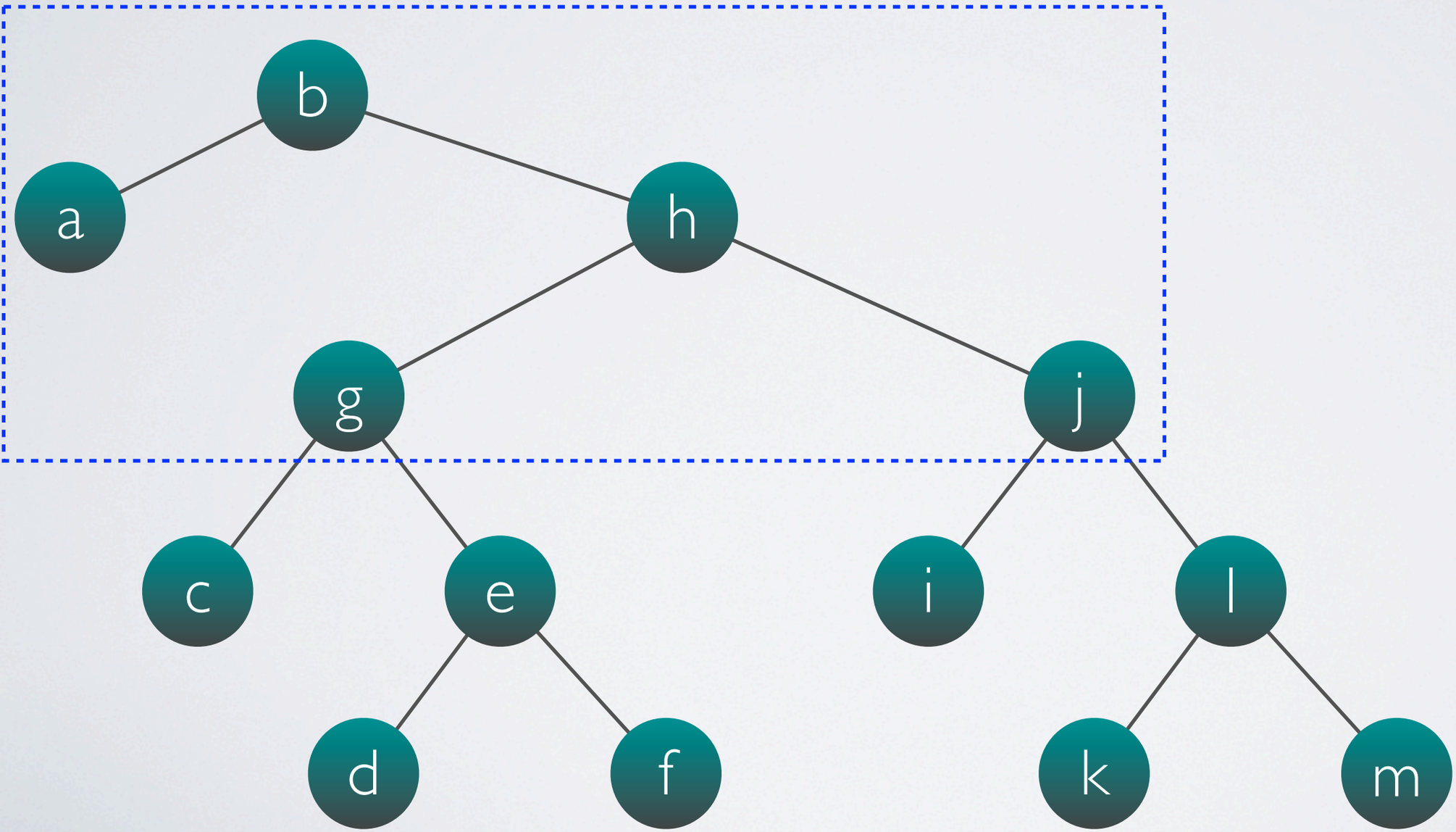
- Trees: k-testables e.g k=3, *max depth=3*



K-TESTABILITY

- Trees: k-testables e.g k=3, *max depth=3*

b(a,h(g,j))

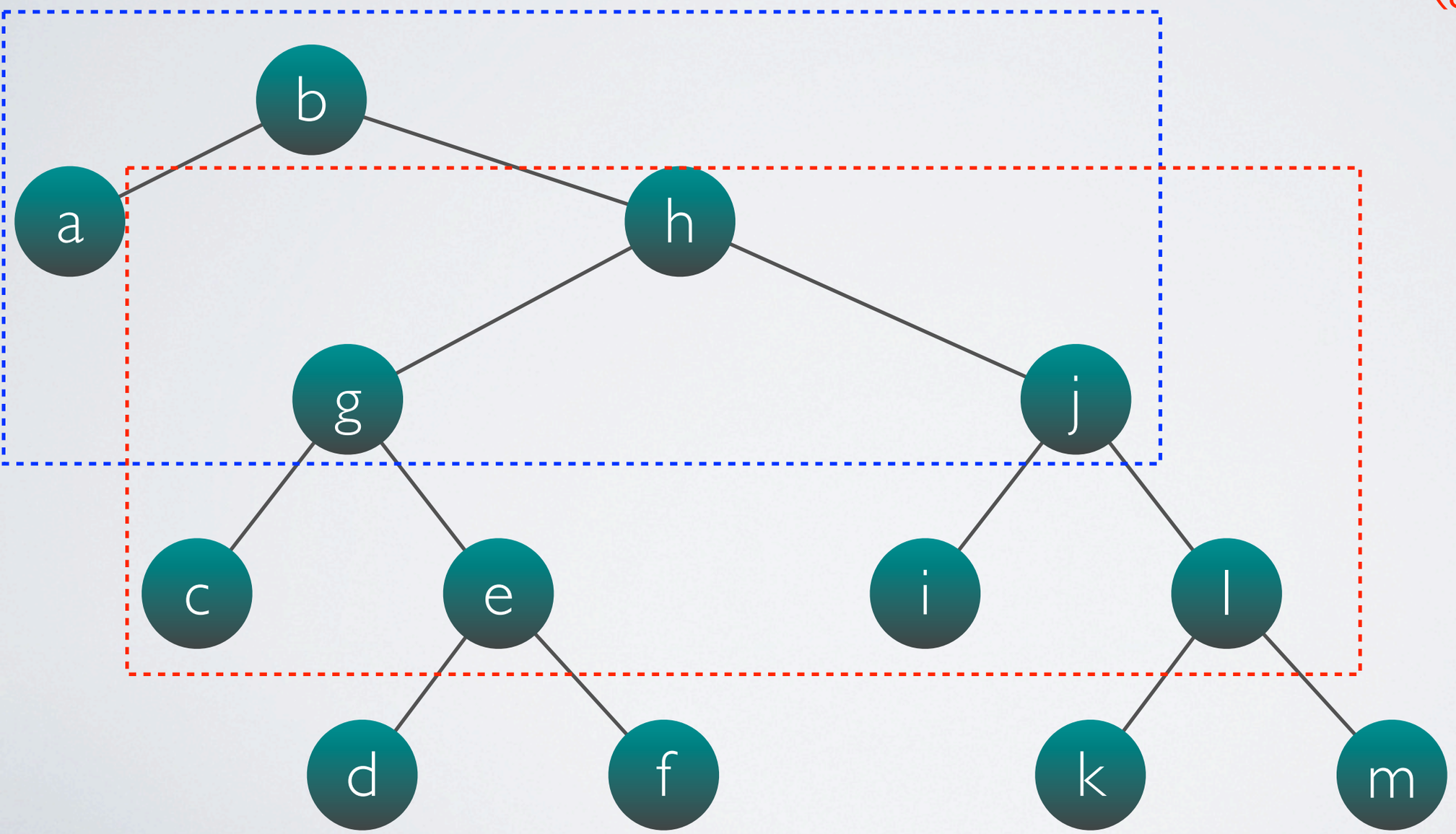


K-TESTABILITY

- Trees: k-testables e.g k=3, *max depth=3*

b(a,h(g,j))

h(g(c,e),j(i,l))



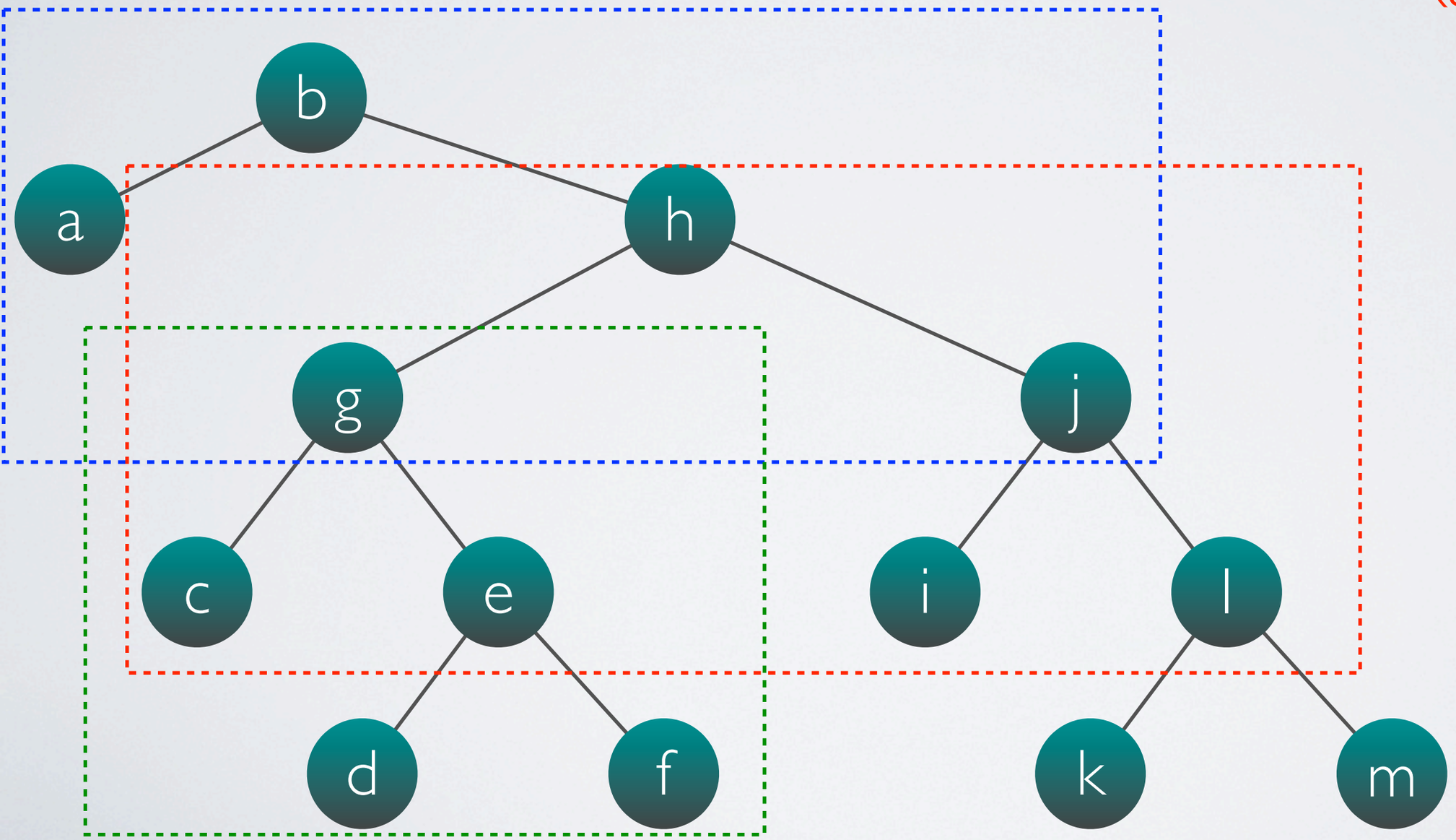
K-TESTABILITY

- Trees: k-testables e.g k=3, *max depth=3*

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))



K-TESTABILITY

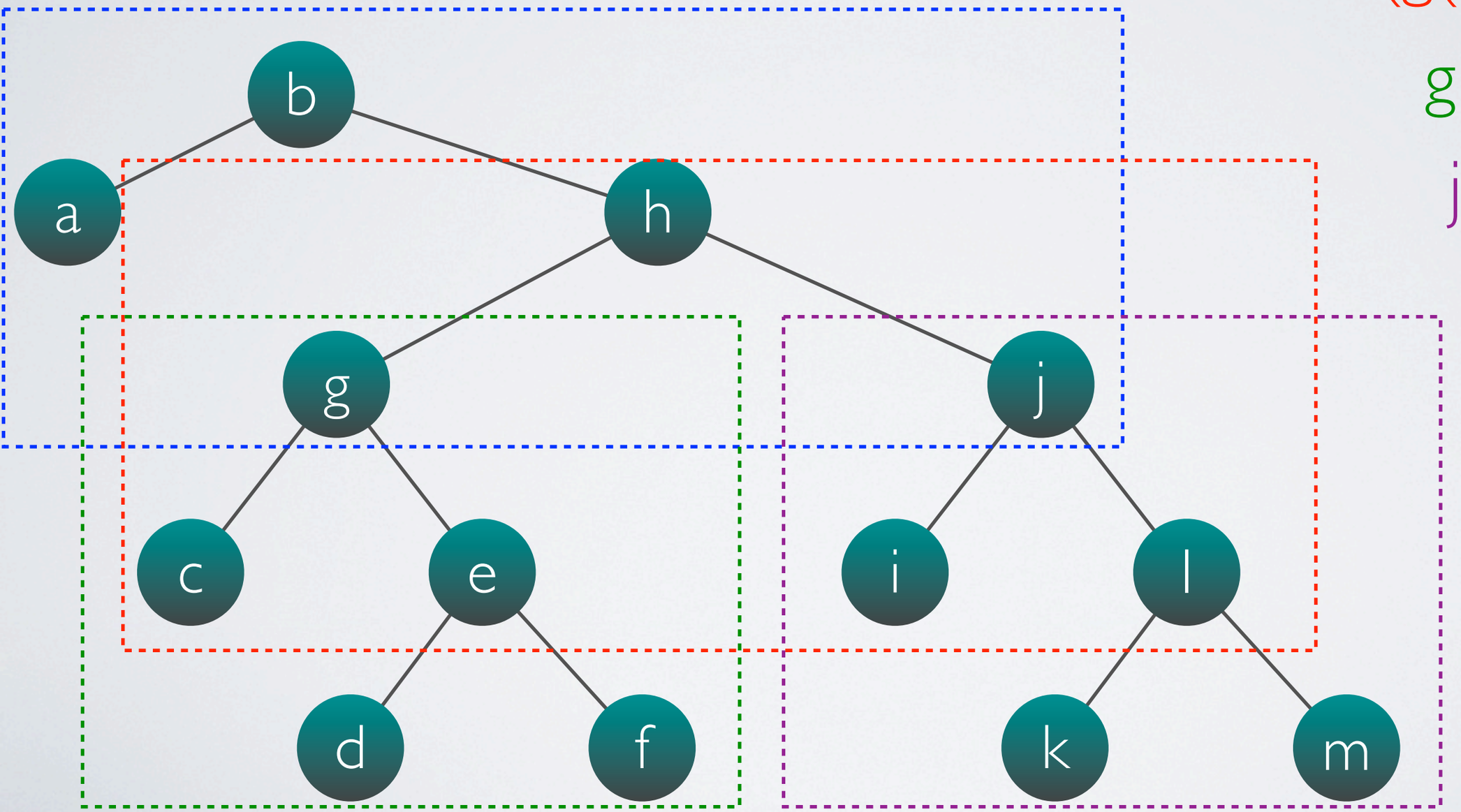
- Trees: k-testables e.g k=3, *max depth=3*

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))

j(i,l(k,m))



K-TESTABILITY

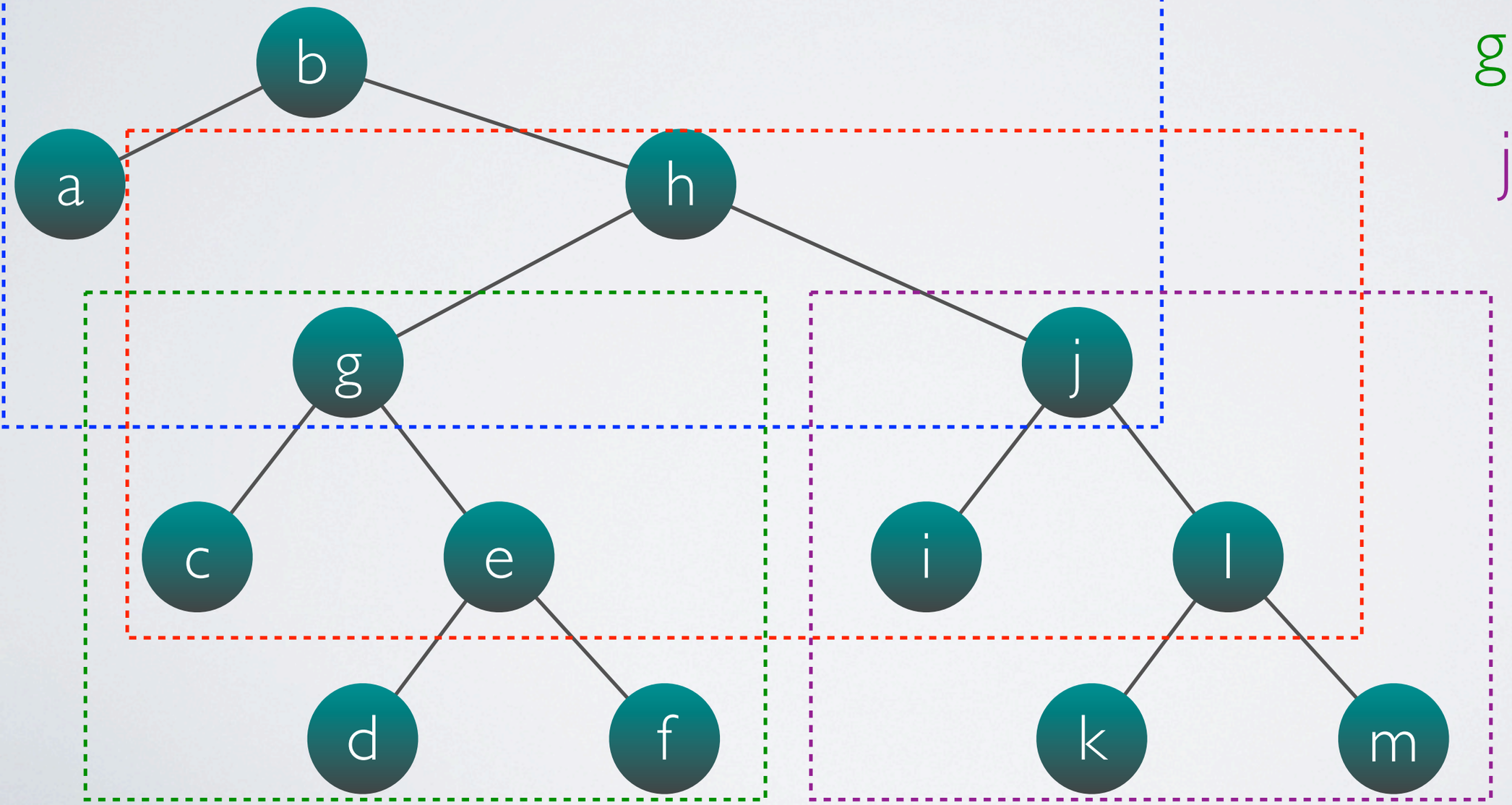
These subtrees are named 3-forks

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))

j(i,l(k,m))



K-TESTABILITY

These subtrees are named 3-forks

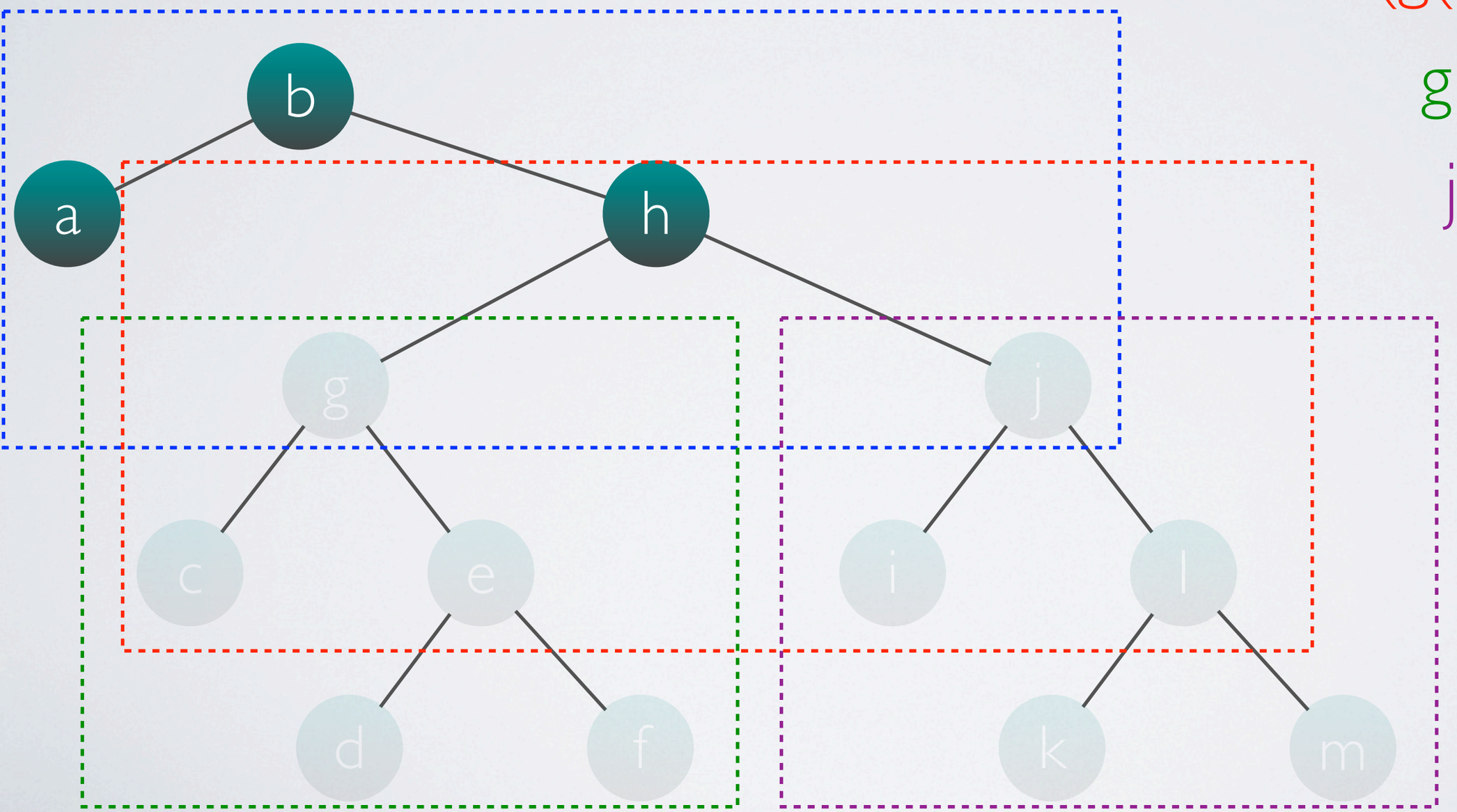
By removing one level they are 2-roots

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))

j(i,l(k,m))



K-TESTABILITY

These subtrees are named 3-forks

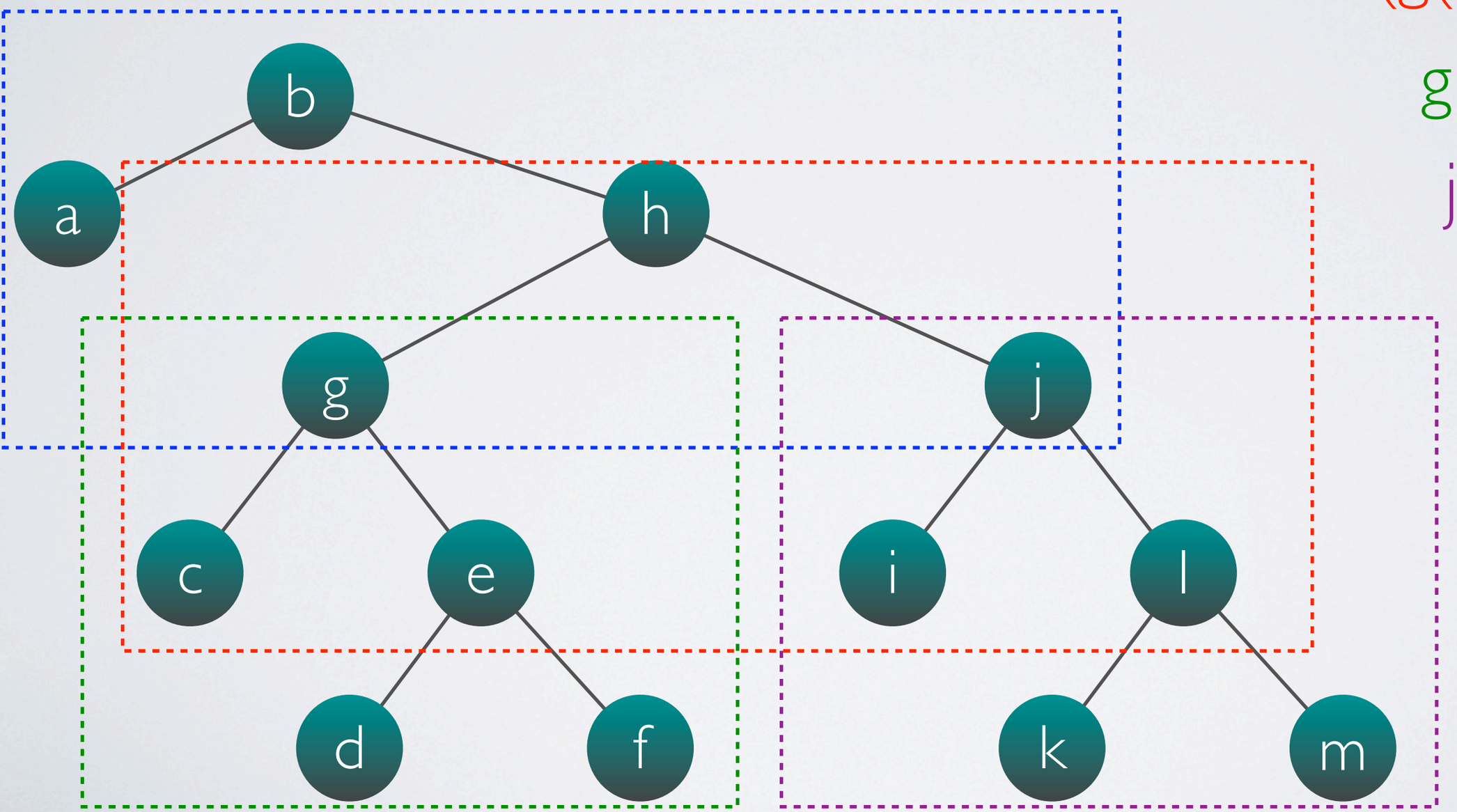
By removing one level they are 2-roots

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))

j(i,l(k,m))



K-TESTABILITY

These subtrees are named 3-forks

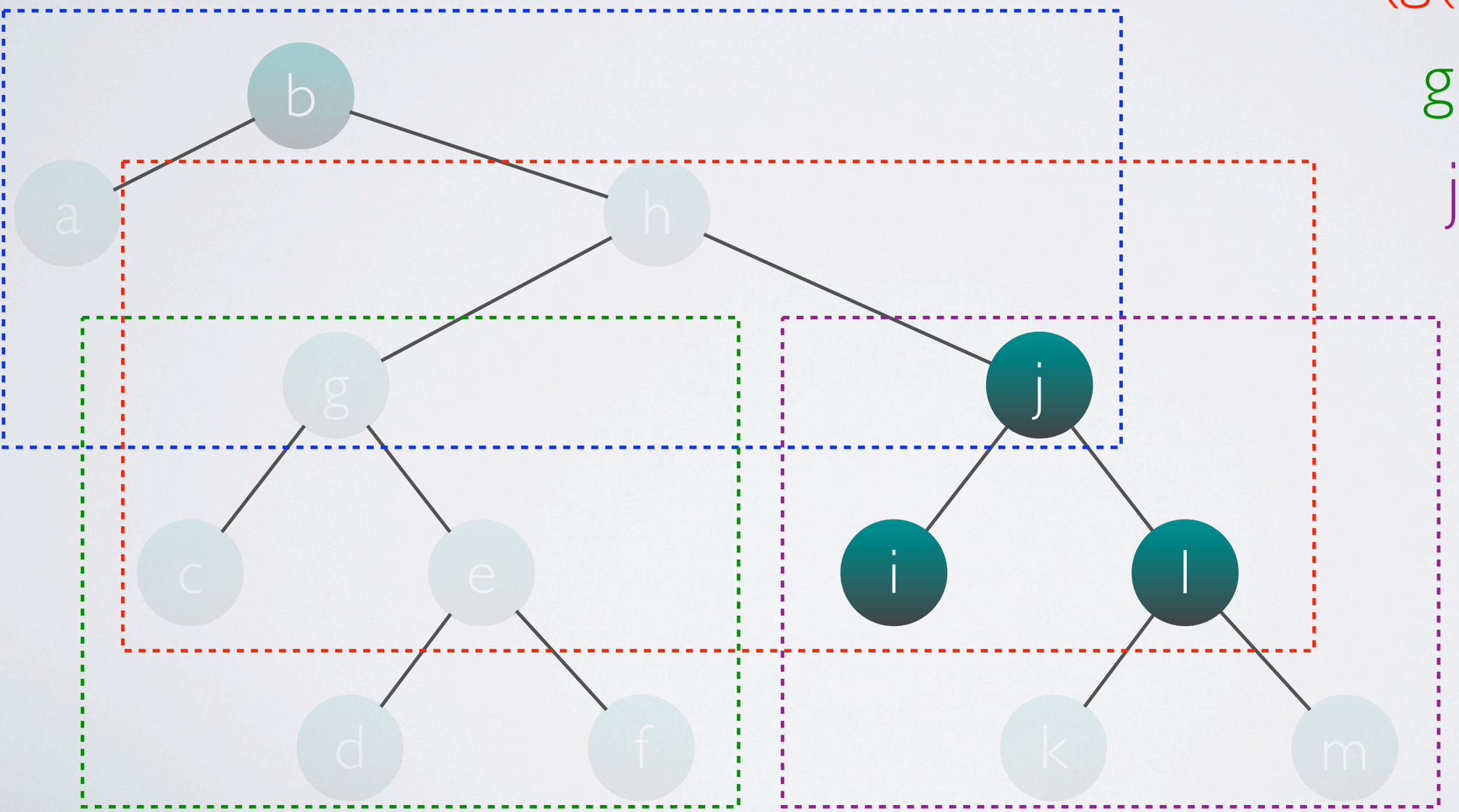
By removing one level they are 2-roots

b(a,h(g,j))

h(g(c,e),j(i,l))

g(c,e(d,f))

j(i,l(k,m))



K-TESTABILITY

- A deterministic tree automata (DTA) can be built using a positive sample of positive examples

- Extension to probabilistic languages (stochastic DTA) equivalent to string *n*-grams

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

$\quad \delta_0(b) = q_2,$

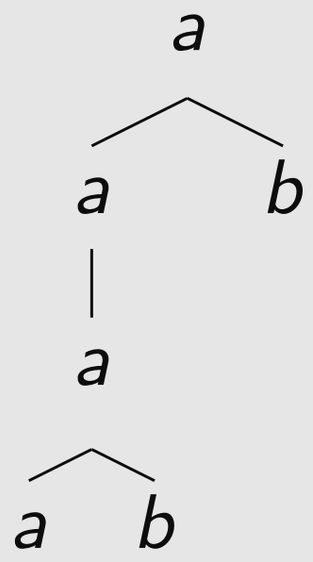
$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

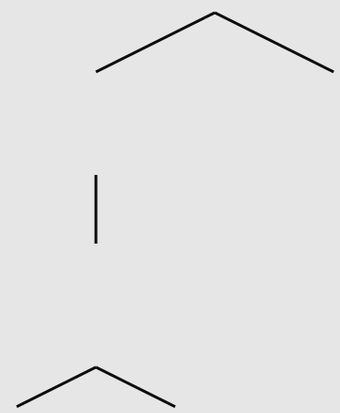
$\}$

$F = \{q_2\}$

Input tree



Recognized part



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\delta_0(a) = q_1,$

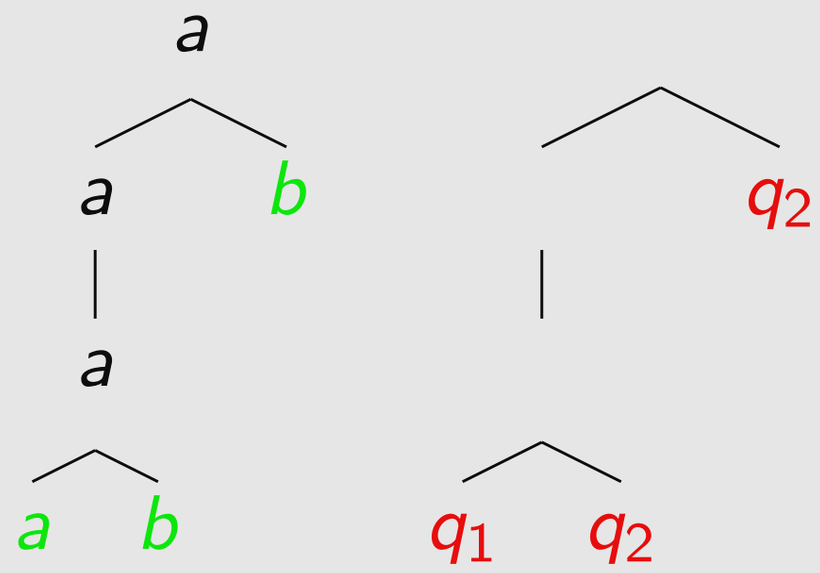
$\delta_0(b) = q_2,$

$\delta_1(a, q_2) = q_1,$

$\delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

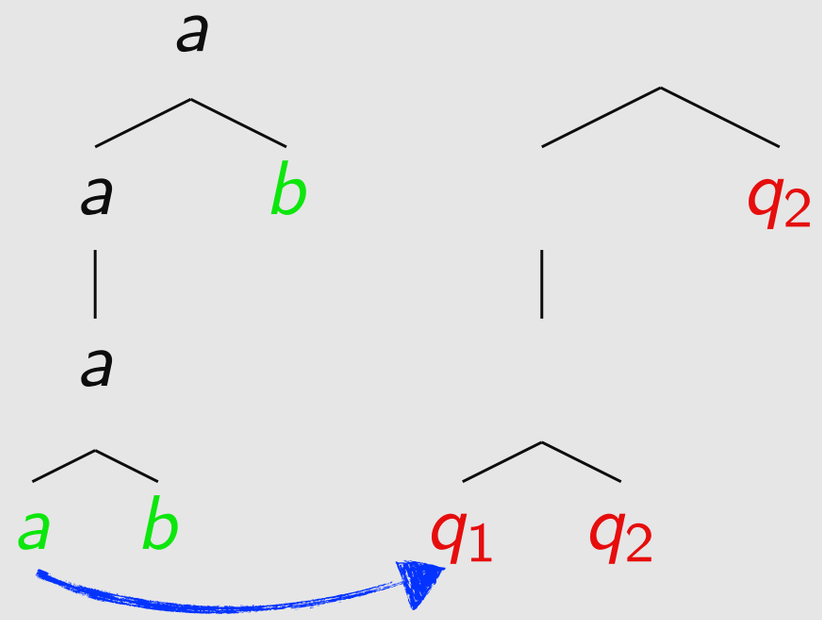
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



For input symbol **a**

state **q1** is reached using transition $\delta_0(a)$

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

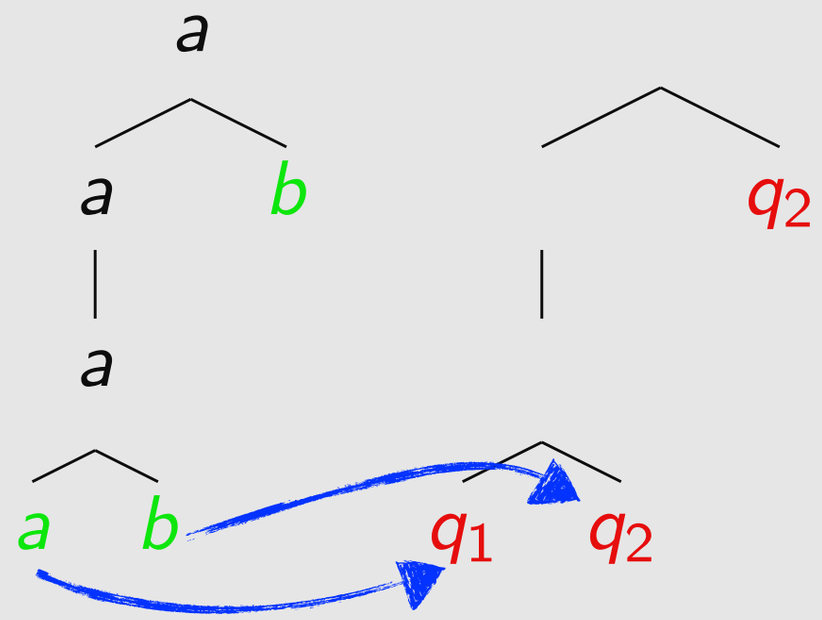
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



For input symbol **a**

state **q1** is reached using transition $\delta_0(a)$

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\delta_0(a) = q_1,$

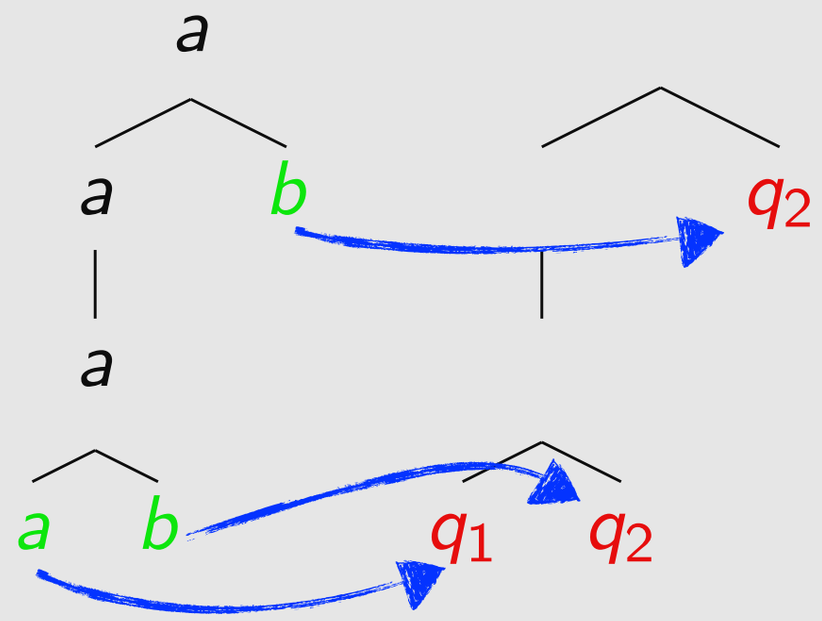
$\delta_0(b) = q_2,$

$\delta_1(a, q_2) = q_1,$

$\delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



For input symbol **a**

state **q1** is reached using transition $\delta_0(a)$

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

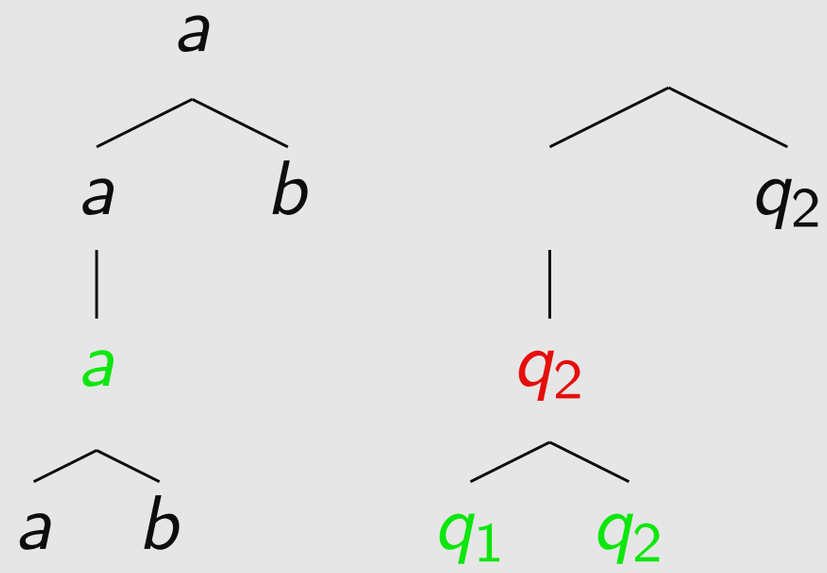
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

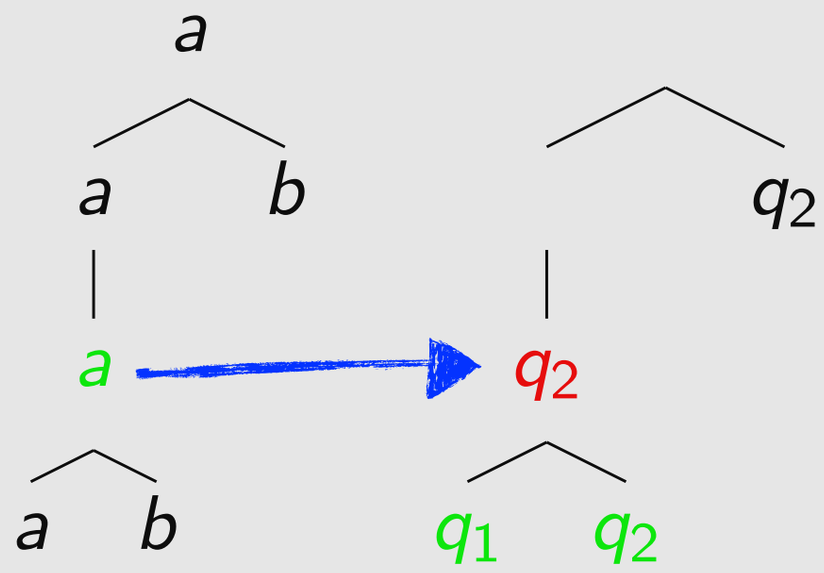
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

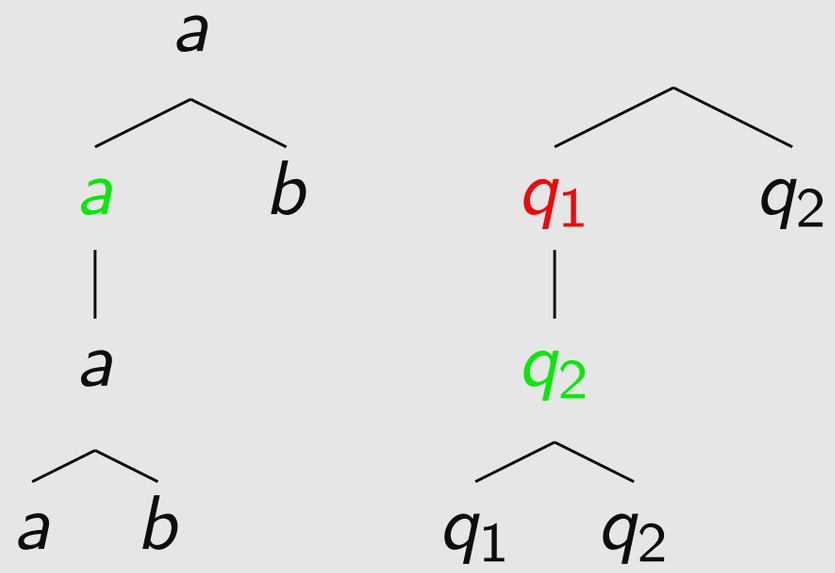
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

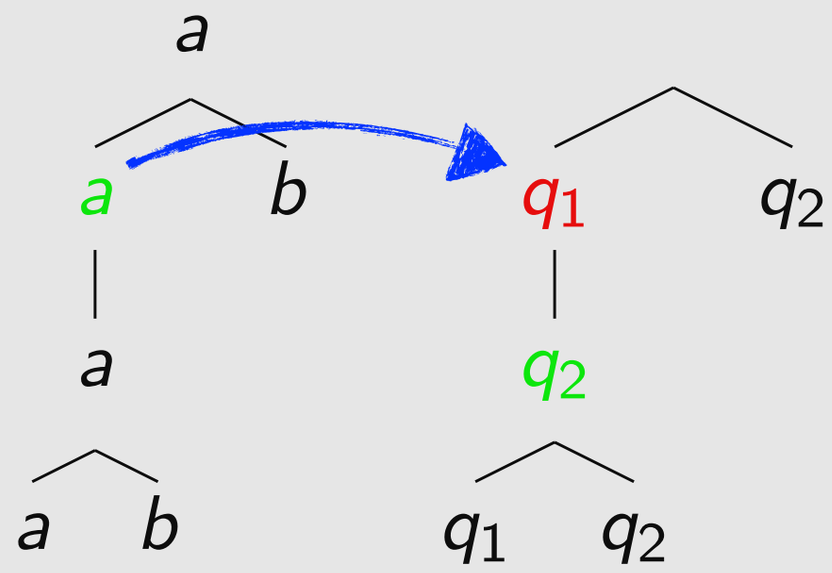
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

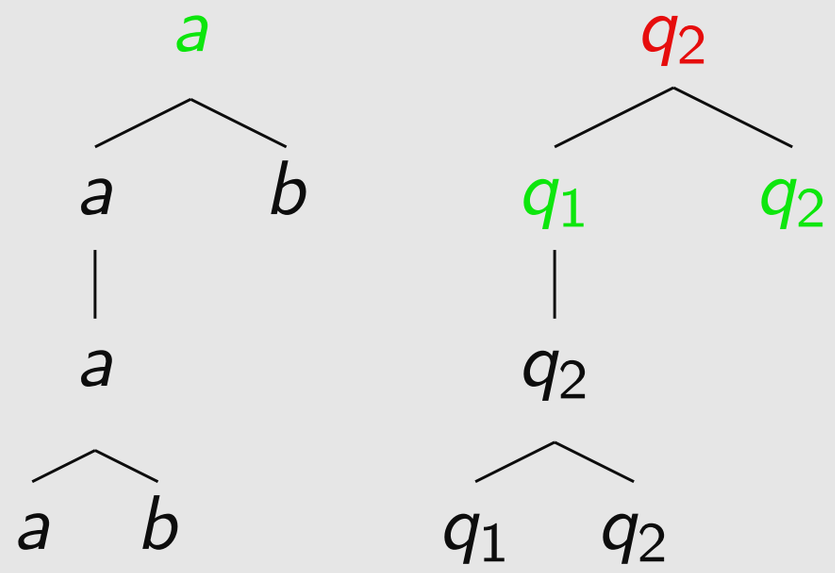
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, F)$$

where Q : states, Σ : alphabet, Δ : transitions, F : accepting states.

Example:

$Q = \{q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Delta = \{$

$\quad \delta_0(a) = q_1,$

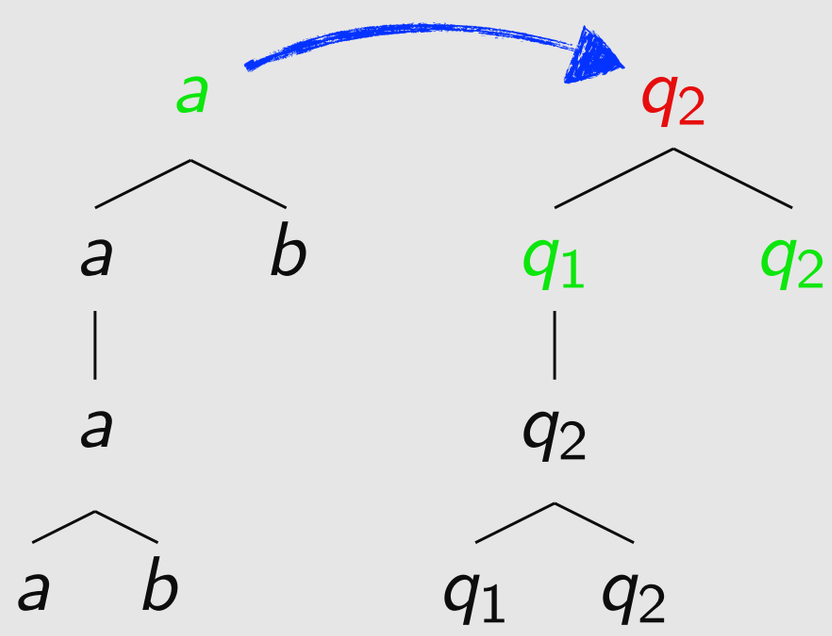
$\quad \delta_0(b) = q_2,$

$\quad \delta_1(a, q_2) = q_1,$

$\quad \delta_2(a, q_1, q_2) = q_2$

$\}$

$F = \{q_2\}$



STOCHASTIC DTA

$$A = (Q, \Sigma, \Delta, P, \rho)$$

where Q : states, Σ : alphabet, Δ : transitions, P : probabilities, ρ p. of the root.

Example:

Q, Σ, and Δ like before,

but now

$P = \{$

$p_0(a) = 0.5,$

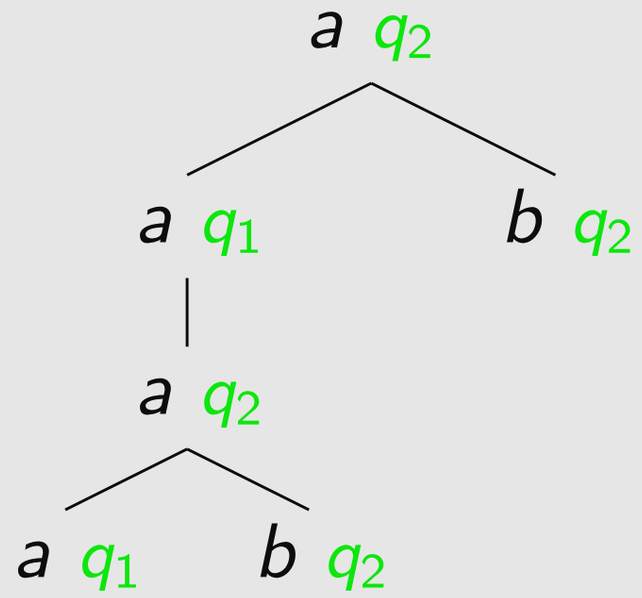
$p_0(b) = 0.3,$

$p_1(a, q_2) = 0.5,$

$p_2(a, q_1, q_2) = 0.7$

$\}$

$\rho(q_2) = 1$



Add a probability to each transition and state being root

DETERMINISTIC TREE AUTOMATA

$$A = (Q, \Sigma, \Delta, P, \rho)$$

where Q : states, Σ : alphabet, Δ : transitions, P : probabilities, ρ p. of the root.

Example:

Q, Σ, and Δ like before,

but now

$P = \{$

$p_0(a) = 0.5,$

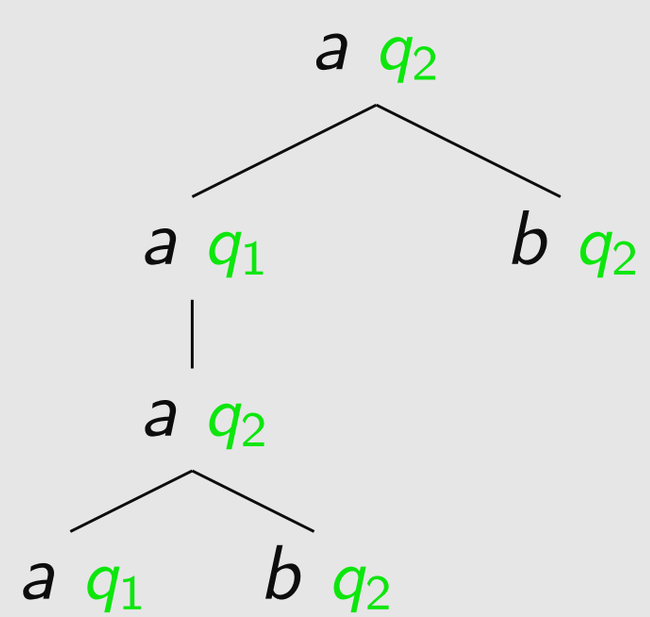
$p_0(b) = 0.3,$

$p_1(a, q_2) = 0.5,$

$p_2(a, q_1, q_2) = 0.7$

$\}$

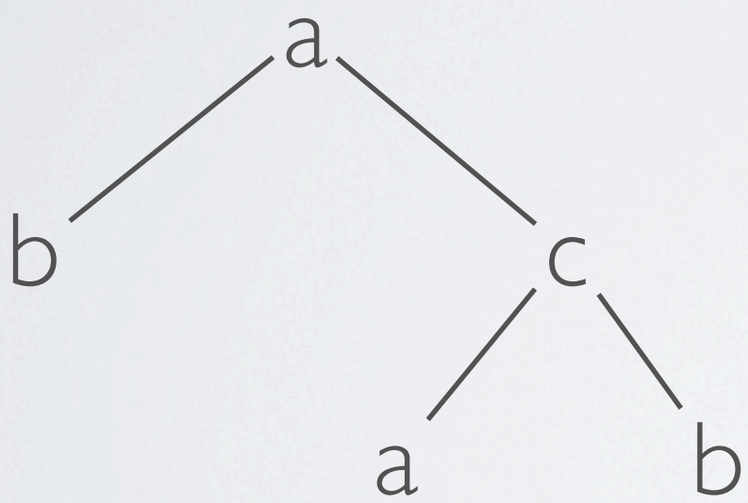
$\rho(q_2) = 1$



$P = 0.3 \times 0.3 \times 0.5 \times 0.7 \times 0.5 \times 0.7 \times 1$

STATE NOMENCLATURE

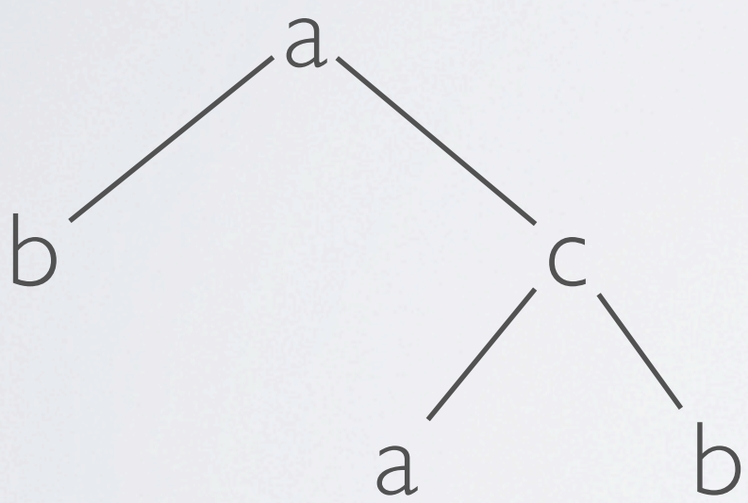
From now on, instead of using q_0, q_1, etc... we will use tree functional notation



STATE NOMENCLATURE

From now on, instead of using q_0, q_1, etc... we will use tree functional notation

If k=2, k-1 states are 1-roots

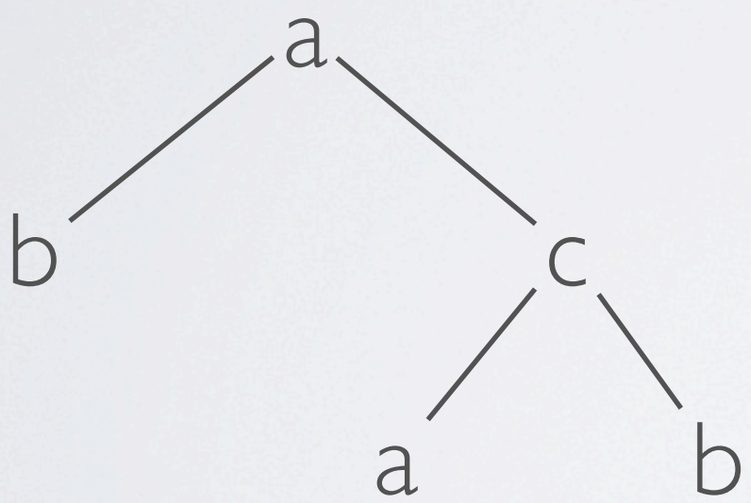


STATE NOMENCLATURE

From now on, instead of using q_0, q_1, etc... we will use tree functional notation

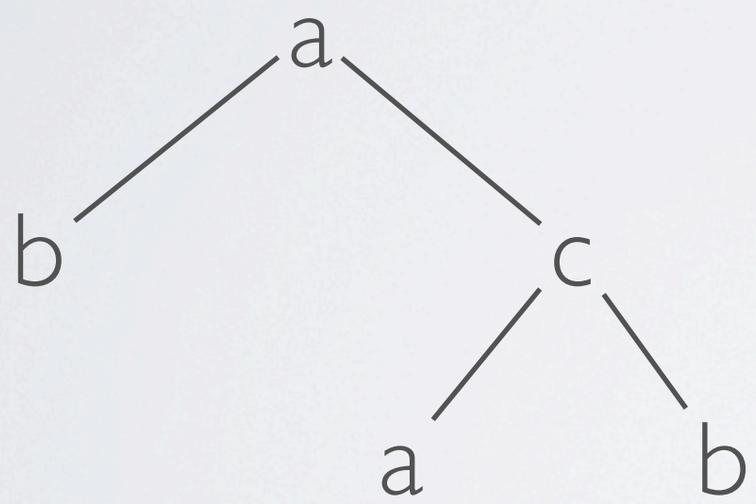
If k=2, k-1 states are 1-roots

States = {a,b,c}



STATE NOMENCLATURE

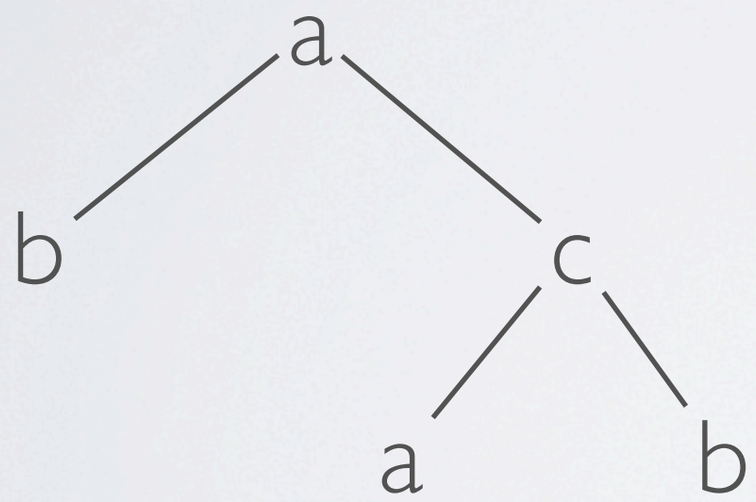
From now on, instead of using q_0, q_1, etc... we will use tree functional notation



If k=3, k-1 states are 2-roots

STATE NOMENCLATURE

From now on, instead of using q_0, q_1, etc... we will use tree functional notation



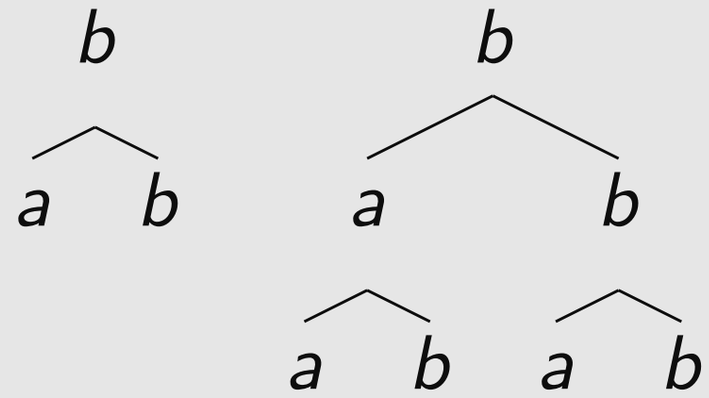
If k=3, k-1 states are 2-roots

States = { a(b,c), c(a,b) }

K=2 TREE PROBABILITIES LEARNING

EXAMPLE : k=2

Training samples



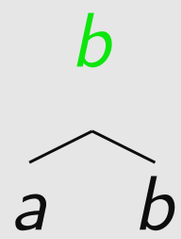
q	$\rho(q)$	δ	$p(\delta)$

Root state probability
= State / TotalStates

Transition probability

EXAMPLE : k=2

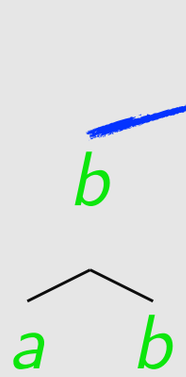
First training tree



q	$\rho(q)$	δ	$p(\delta)$
b	$1/1$		

Recall that states are named
as the root label of the tree
they represent (k-1 root = 1-root)

EXAMPLE : k=2

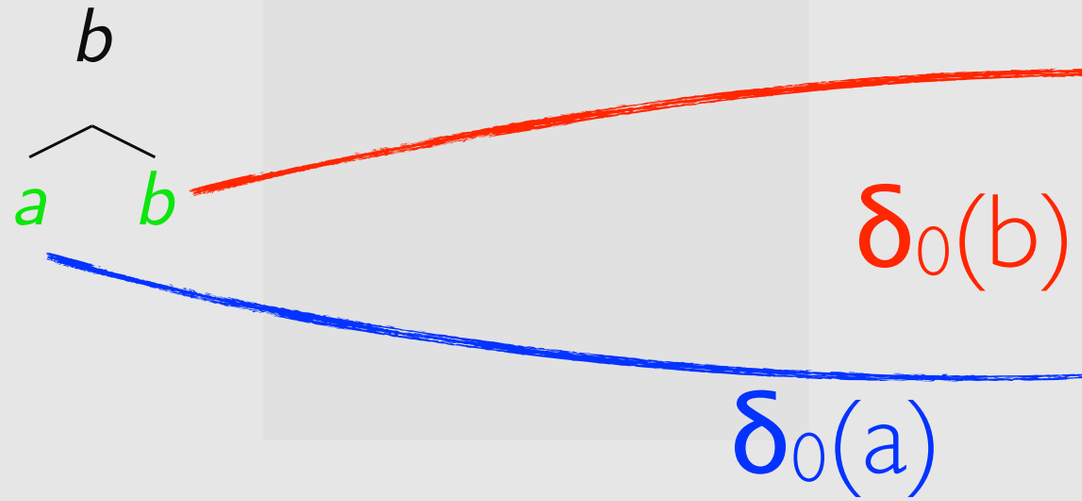


a is not root

q	$\rho(q)$	δ	$p(\delta)$
b	1/1	b, a, b	1/1
a	0		

Both states a and b
input b

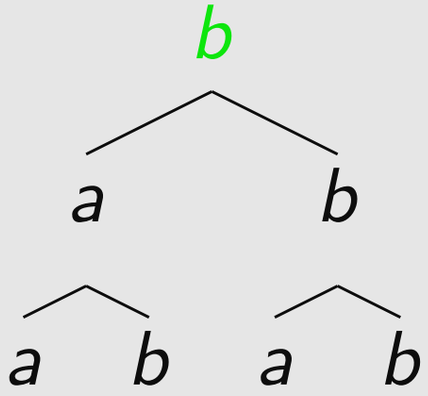
EXAMPLE : k=2



q	$\rho(q)$	δ	$p(\delta)$
b	1/1	b, a, b	1/2
		b	1/2
a	0	a	1/1

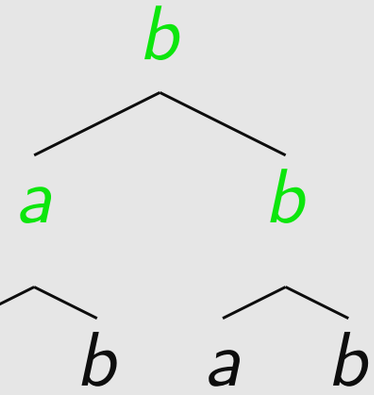
EXAMPLE : k=2

Second training tree



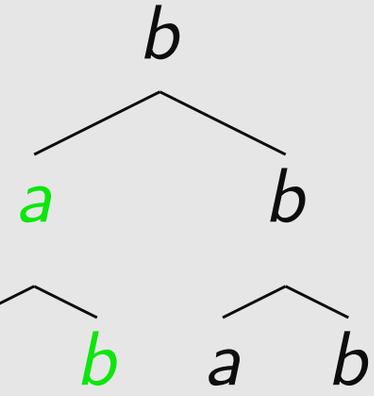
q	$\rho(q)$	δ	$p(\delta)$
b	$2/2$	b, a, b	$1/2$
		b	$1/2$
a	0	a	$1/1$

EXAMPLE : k=2



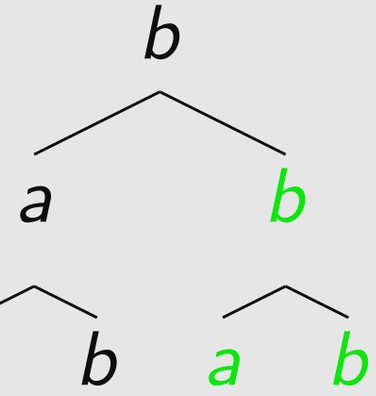
q	$\rho(q)$	δ	$p(\delta)$
b	$2/2$	b, a, b	$2/3$
		b	$1/3$
a	0	a	$1/1$

EXAMPLE : k=2



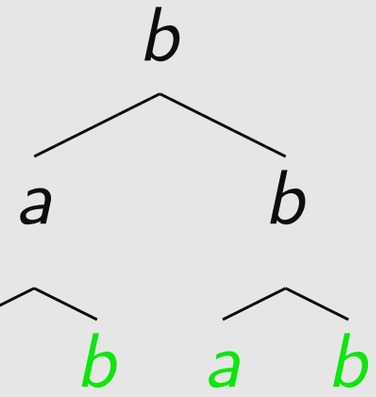
q	$\rho(q)$	δ	$p(\delta)$
b	$2/2$	b, a, b	$2/3$
		b	$1/3$
a	0	a	$1/2$
		a, a, b	$1/2$

EXAMPLE : k=2



q	$\rho(q)$	δ	$p(\delta)$
b	$2/2$	b, a, b	$3/4$
		b	$1/4$
a	0	a	$1/2$
		a, a, b	$1/2$

EXAMPLE : k=2



q	$\rho(q)$	δ	$p(\delta)$
b	2/2	b, a, b	3/6
		b	3/6
a	0	a, a, b	1/4
		a	3/4

Probabilities are computed online
as a counter quotient

SMOOTHING

- The problem occurs when parsing a tree, a transition is found never seen in the training set
  - therefore with a zero probability
- Solution: follow a back-off scheme similar to strings approach
  - i.e. if working with *K*, parse tree using *(K-1)*
- Then: all $1 \leq k \leq K$ models are built

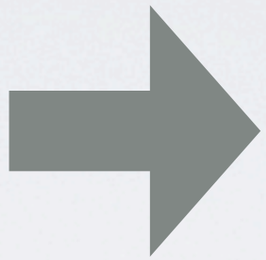
CLASSIFICATION

- With probabilistic k-testable tree languages

CLASSIFICATION

- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

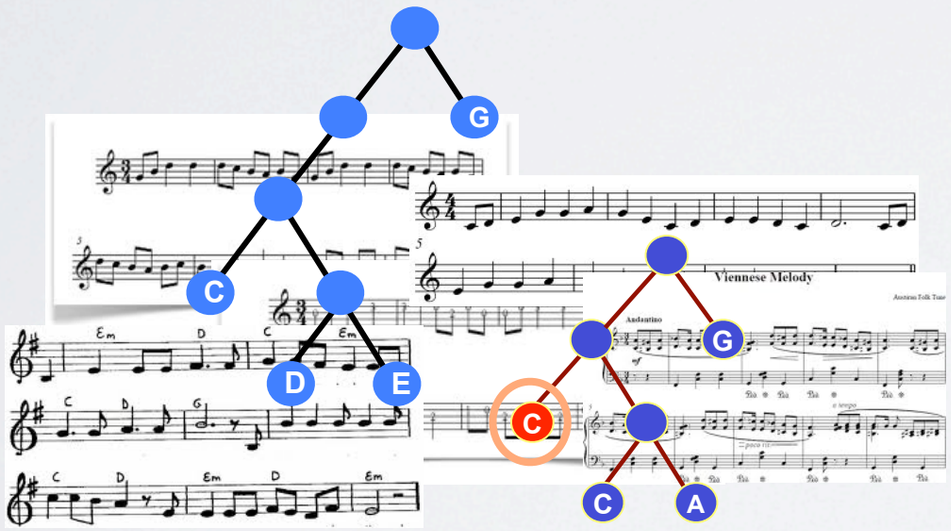
CLASSIFICATION

- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)

Model for each class
(= probabilistic tree automaton including backoff)

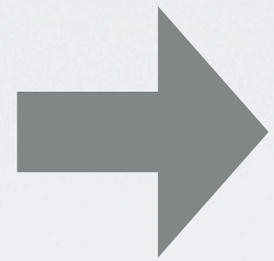
Jazz



CLASSIFICATION

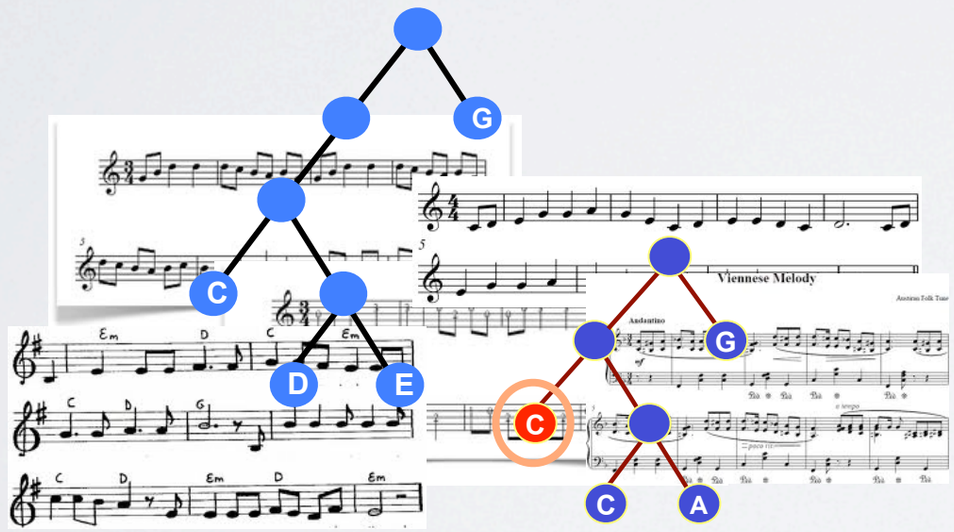
- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

Jazz



$$A_{jazz} = (Q, \Sigma, \Delta, P_{jazz}, \rho_{jazz})$$

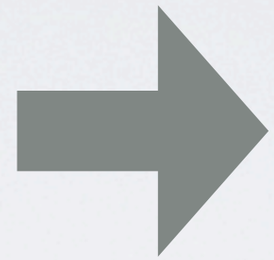
$$P_{jazz} = \{$$
$$p0(a) = 0.5,\ p0(b) = 0.3,$$
$$p1(a, q2) = 0.5,$$
$$p2(a, q1, q2) = 0.7$$
$$\}$$

$$\rho_{jazz}(q2) = 1$$

CLASSIFICATION

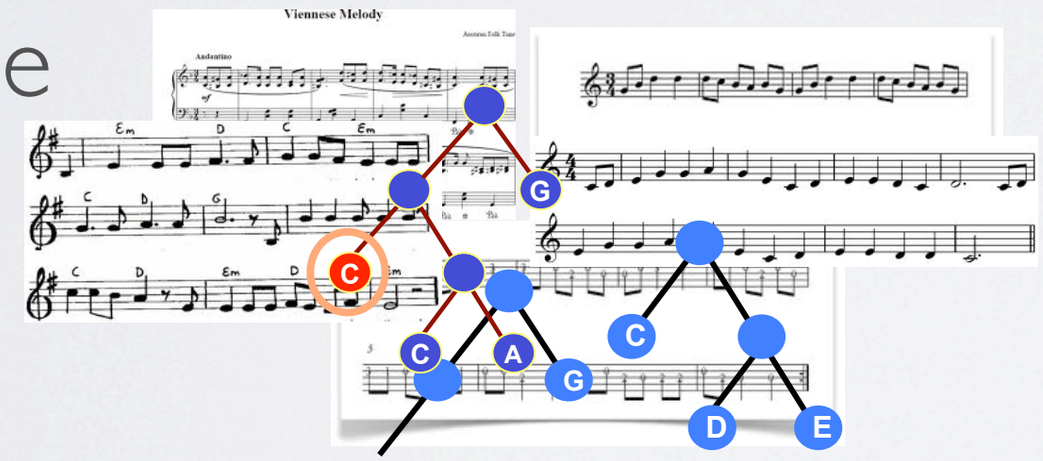
- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

Baroque



$$A_{baro} = (Q, \Sigma, \Delta, P_{bar}, \rho_{baro})$$

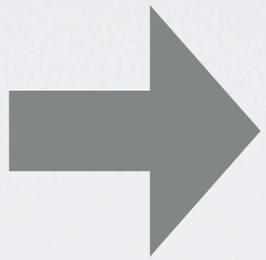
$$P_{baro} = \{$$
$$p0(a) = 0.2,\ p1(b) = 0.2,$$
$$p1(a, q3) = 0.1,$$
$$p2(b, q1, q1) = 0.9$$
$$\}$$

$$\rho_{jazz}(q2) = 1$$

CLASSIFICATION

- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)

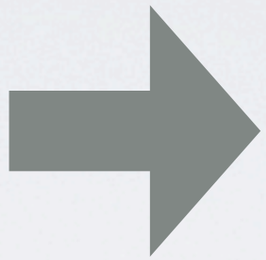


Model for each class
(= probabilistic tree automaton including backoff)

CLASSIFICATION

- With probabilistic k-testable tree languages

Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

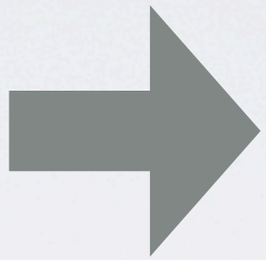
Classification

Melody tree

CLASSIFICATION

- With probabilistic k-testable tree languages

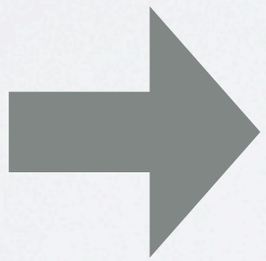
Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

Classification

Melody tree

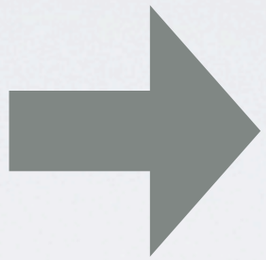


Run the probabilistic tree automata for each model

CLASSIFICATION

- With probabilistic k-testable tree languages

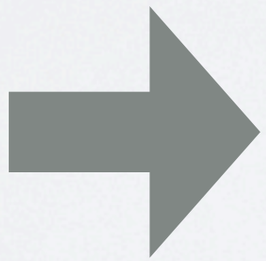
Training samples
Set of trees representing
different classes (e.g genres)



Model for each class
(= probabilistic tree automaton including backoff)

Classification

Melody tree



Run the probabilistic tree automata for each model

Classify as **highest probability class**

GENERATION

- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$ q2

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$

GENERATION

- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$

look q2 as target

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$

q2

GENERATION

- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$ ←

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$ ←

q2

Random choose

GENERATION

- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$ ←

q2

Random choose
e.g

GENERATION

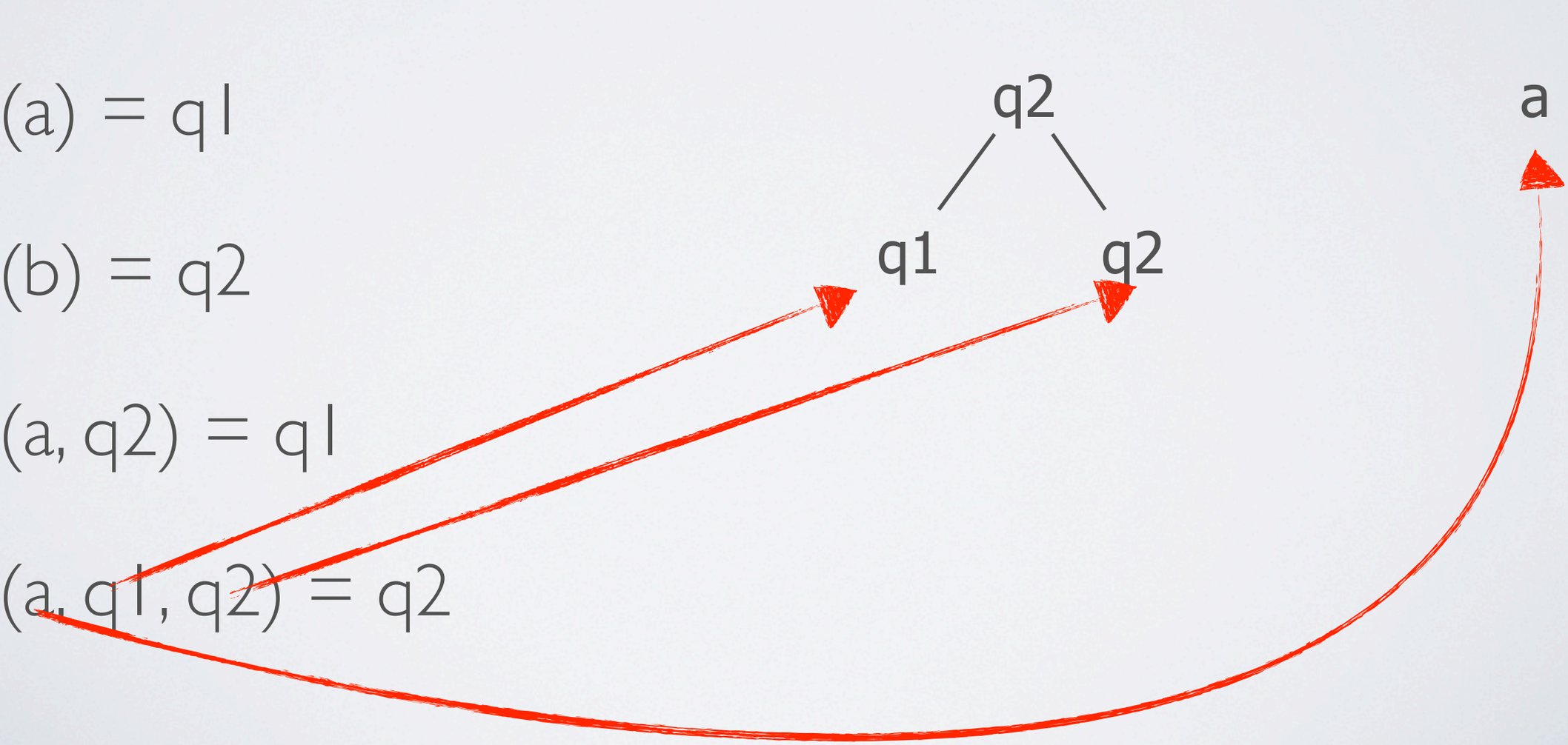
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$



GENERATION

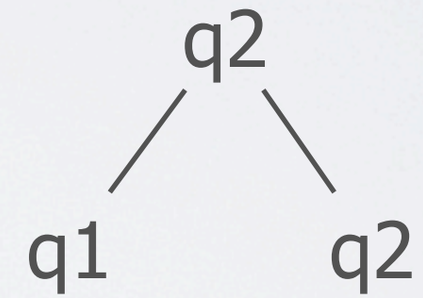
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$



Now repeat operation
for each state: q1 and q2

GENERATION

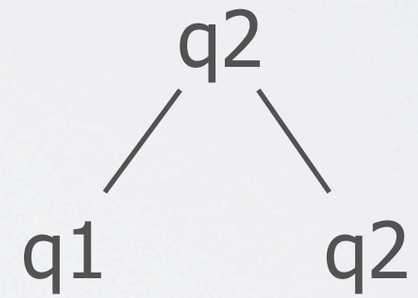
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$



a

GENERATION

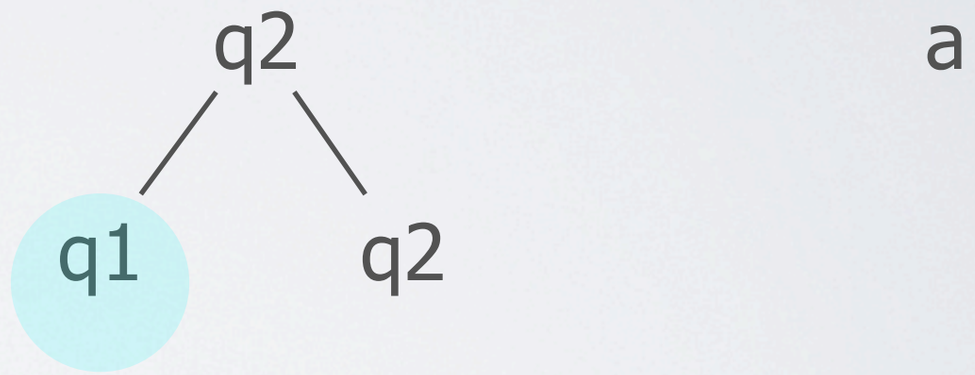
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$ ←

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$ ←

$\delta_2(a, q1, q2) = q2$



look q1 as target and random choose one

GENERATION

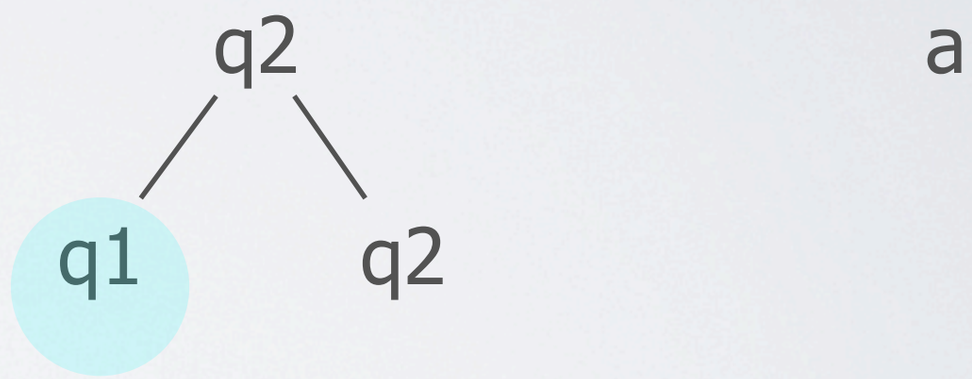
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

$\delta_0(a) = q1$ ← *q1 becomes leaf*

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$



GENERATION

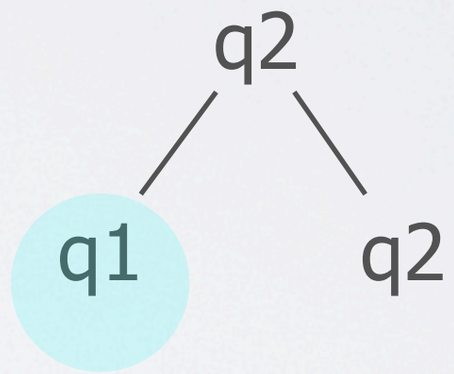
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

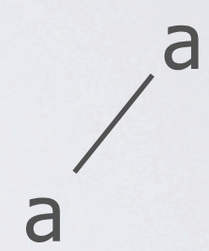
$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$





GENERATION

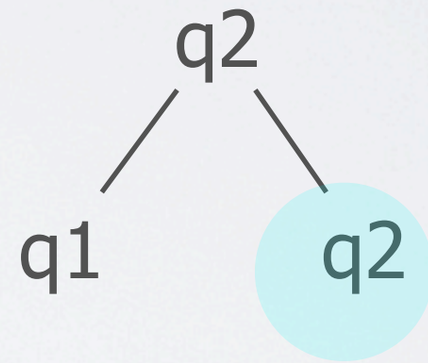
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

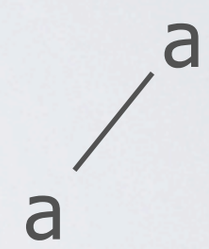
$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$





GENERATION

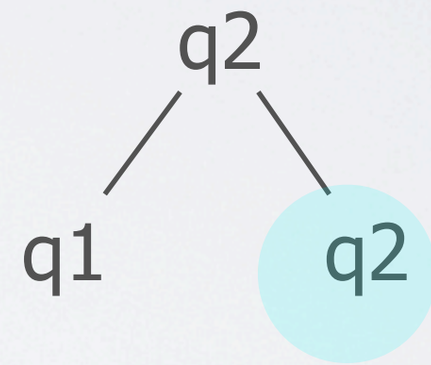
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

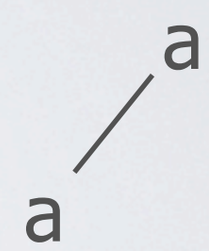
$\delta_0(a) = q1$

$\delta_0(b) = q2$ ←

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$





look q2 as target and random choose one

GENERATION

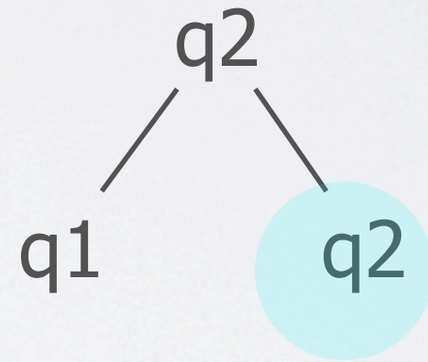
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

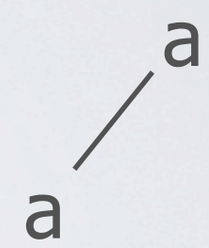
$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$





GENERATION

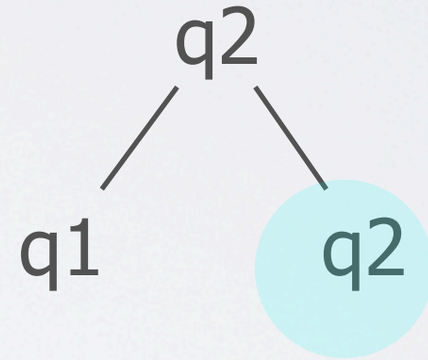
- Tree automata with F = {q2}:
  - given a state q1 random choose a transition where this state is the target and produce a symbol and move:

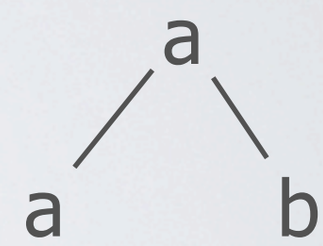
$\delta_0(a) = q1$

$\delta_0(b) = q2$

$\delta_1(a, q2) = q1$

$\delta_2(a, q1, q2) = q2$





GENERATION

GENERATION

- Probabilistic tree automata:

GENERATION

- Probabilistic tree automata:
- just follow probability distribution

GENERATION

- Probabilistic tree automata:
  - just follow probability distribution
  - non probabilistic tree automata are just probabilistic tree automata with equal probability for all transitions

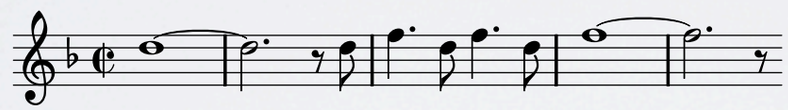
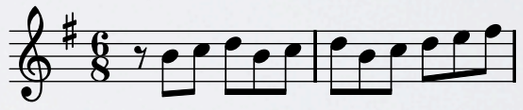
LEARNING ⇒GENERATION

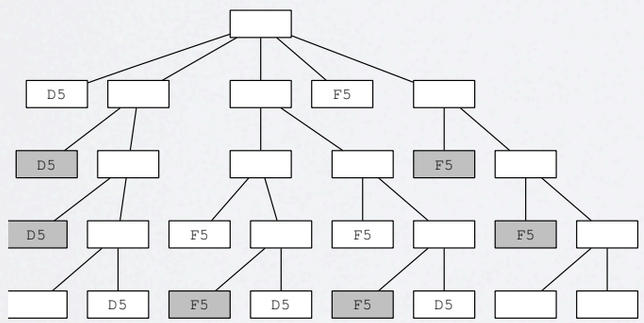
- Style = just set of trees
  - learnt using probabilistic k-testable tree languages
- Generate new trees that belong to that style

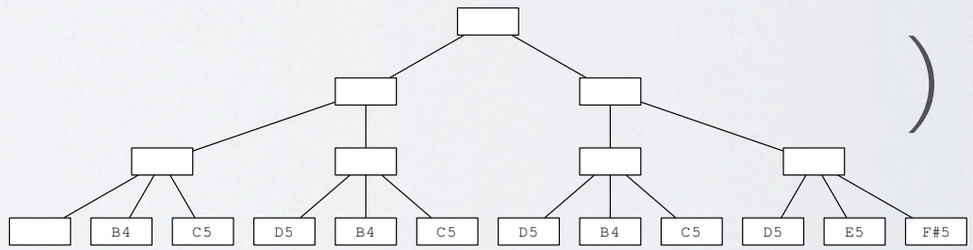
METRIC TREE COMPARISON

COMPARISON

`similarity` function or
Inverse of the distance

`similarity(`  `,`  `)`

`similarity(` 

`,` 

`)`

COMPARISON

distances or similarity measures

`similarity` function or
Inverse of the distance

Literature

- Shasha & Zhang
- Selkow
- G.Valiente
- Jiang's alignment

Our proposal

- Similarity measure between partially labeled trees

COMPARISON

distances or similarity measures

`similarity` function or
Inverse of the distance

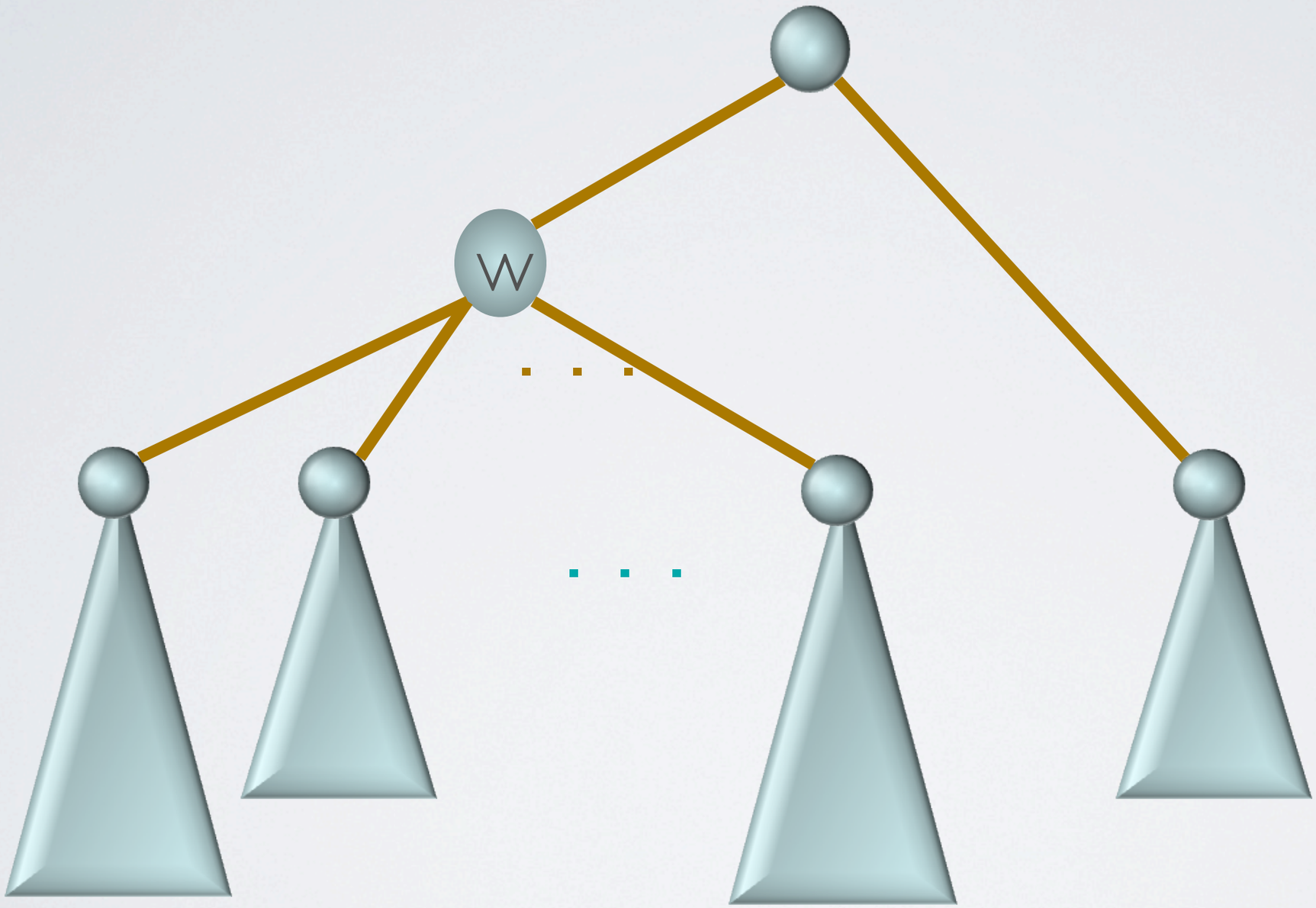
Literature

- Shasha & Zhang
- Selkow
- G.Valiente

Our proposal

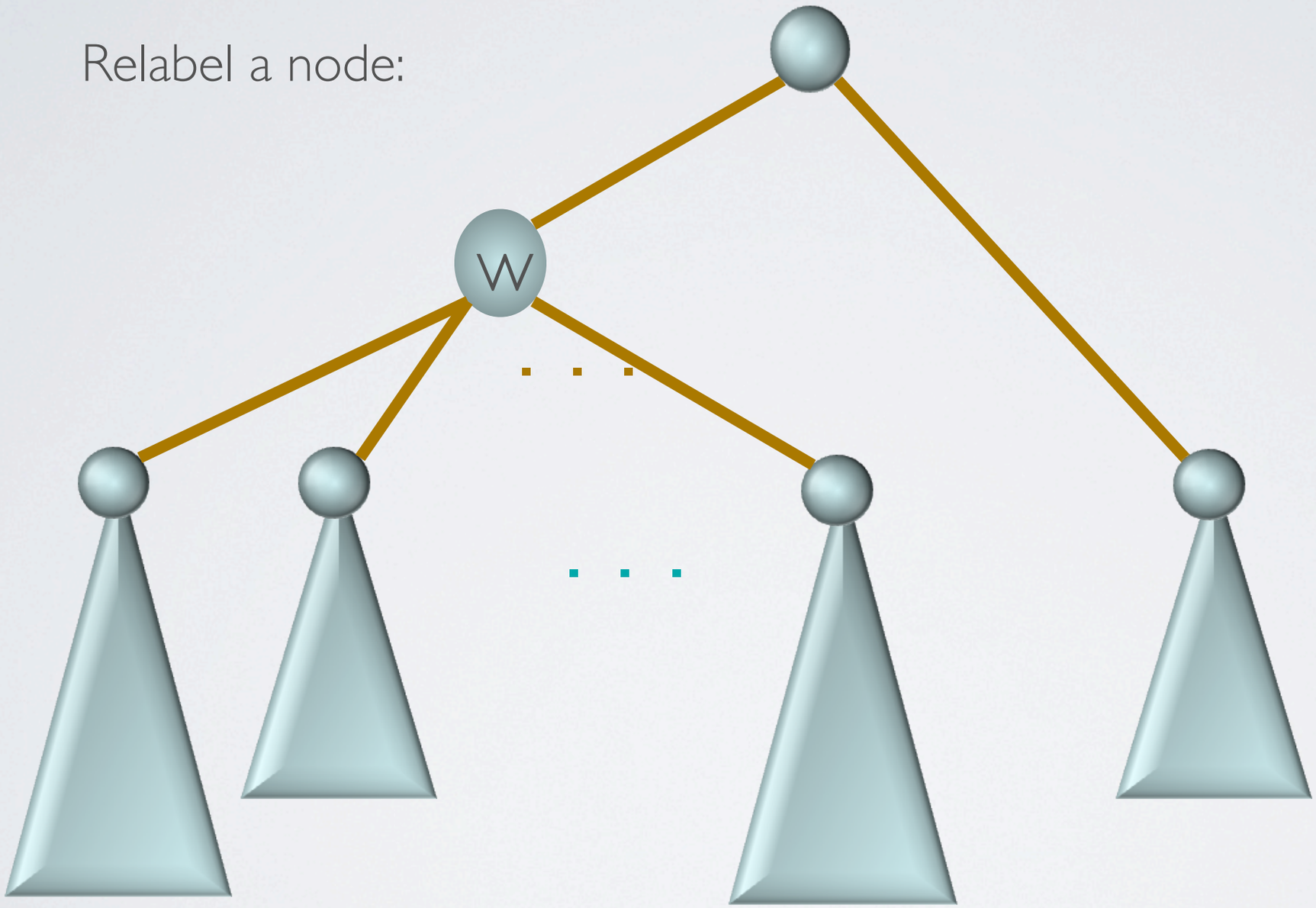
- Similarity measure between partially labeled trees
- Jiang's alignment
  (time improved alternative)

TREE EDIT OPERATIONS



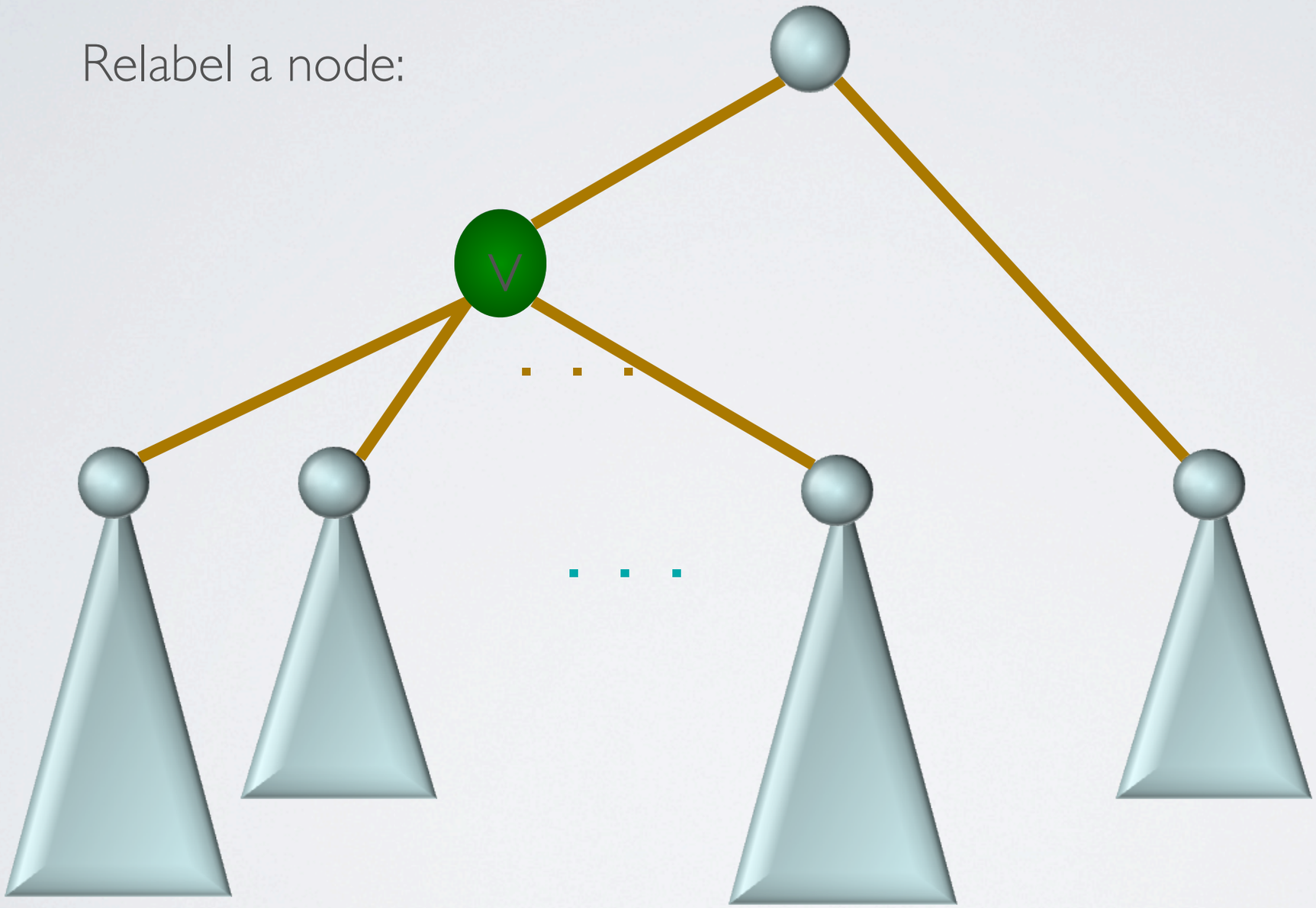
TREE EDIT OPERATIONS

Relabel a node:



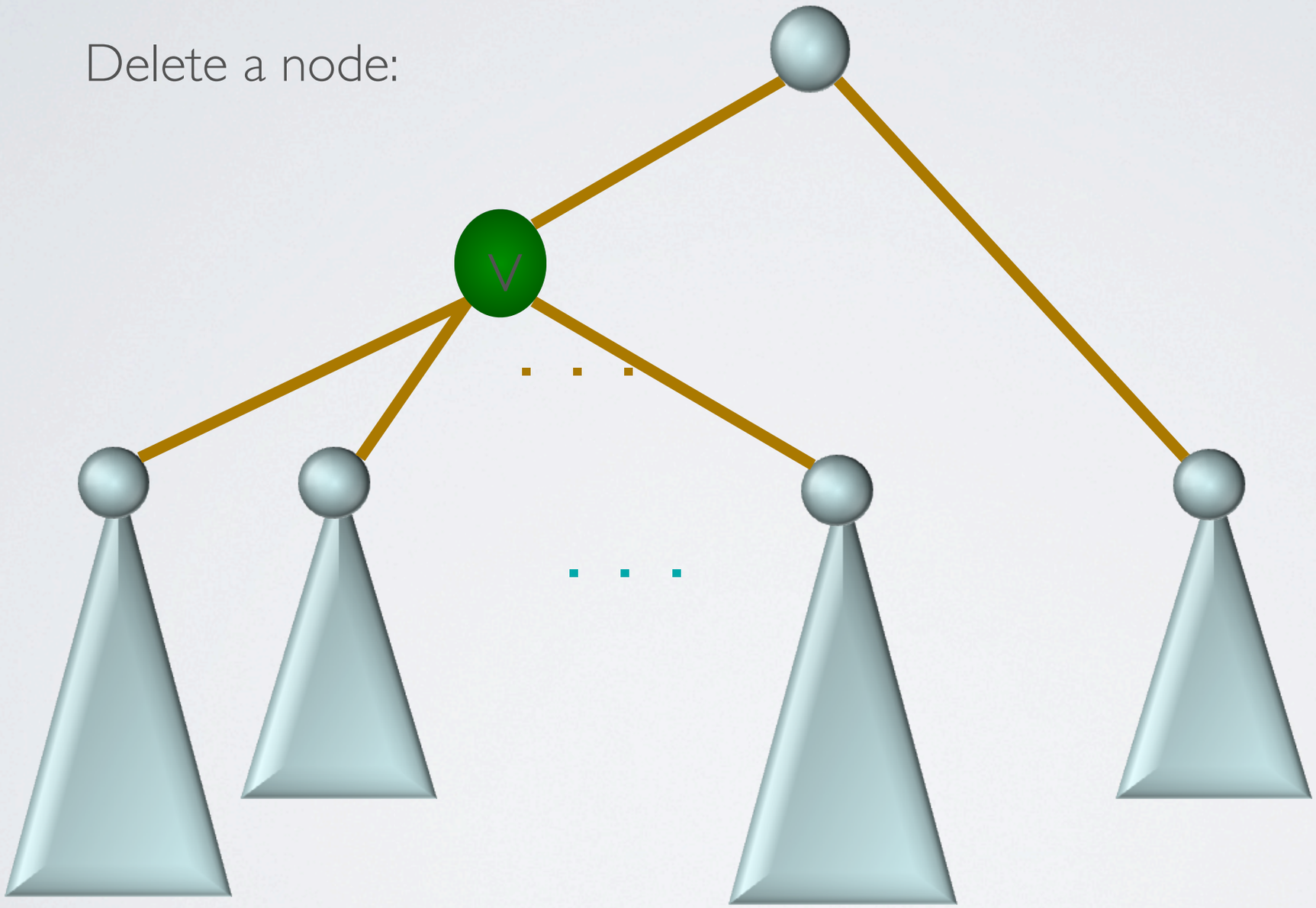
TREE EDIT OPERATIONS

Relabel a node:



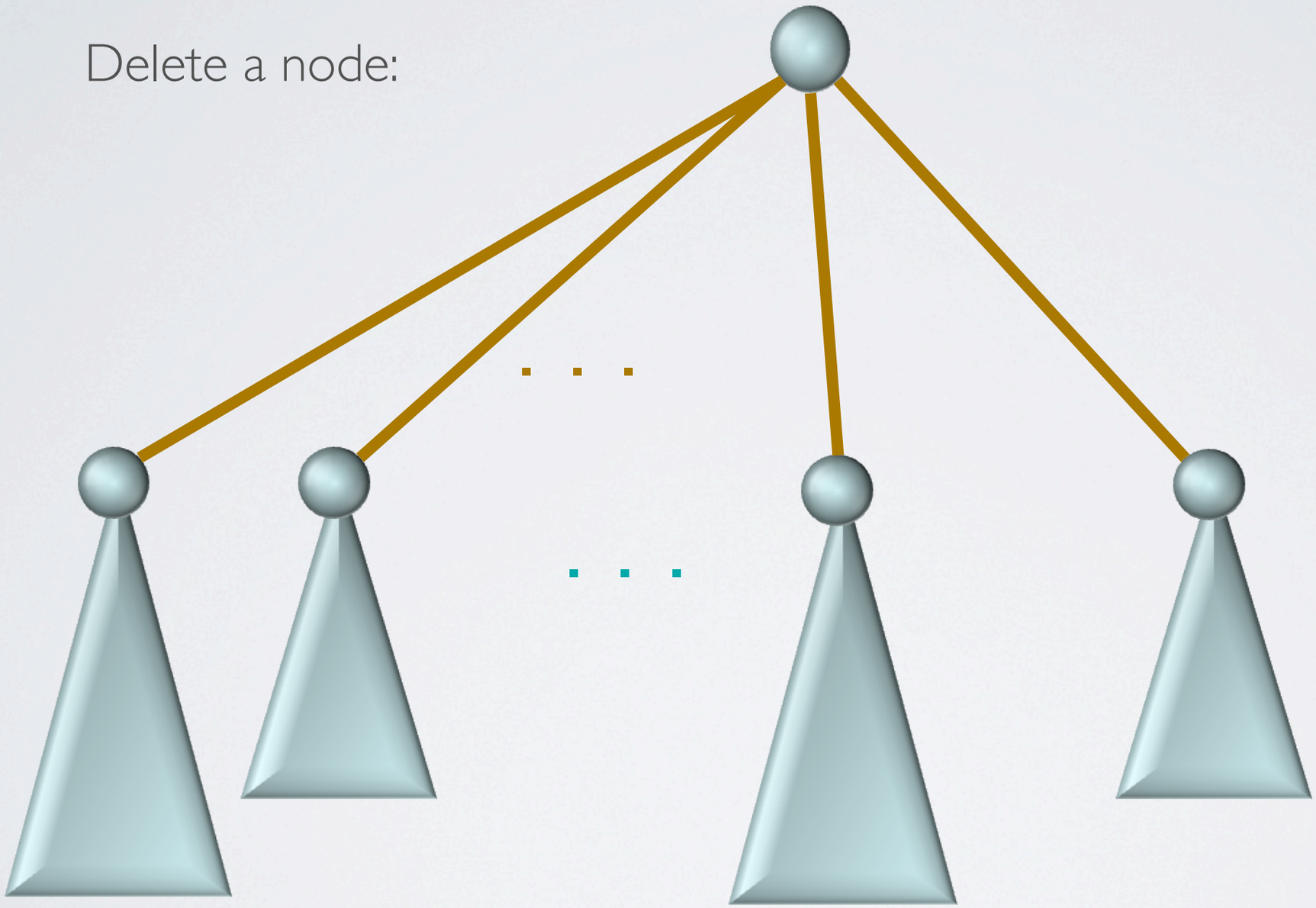
TREE EDIT OPERATIONS

Delete a node:



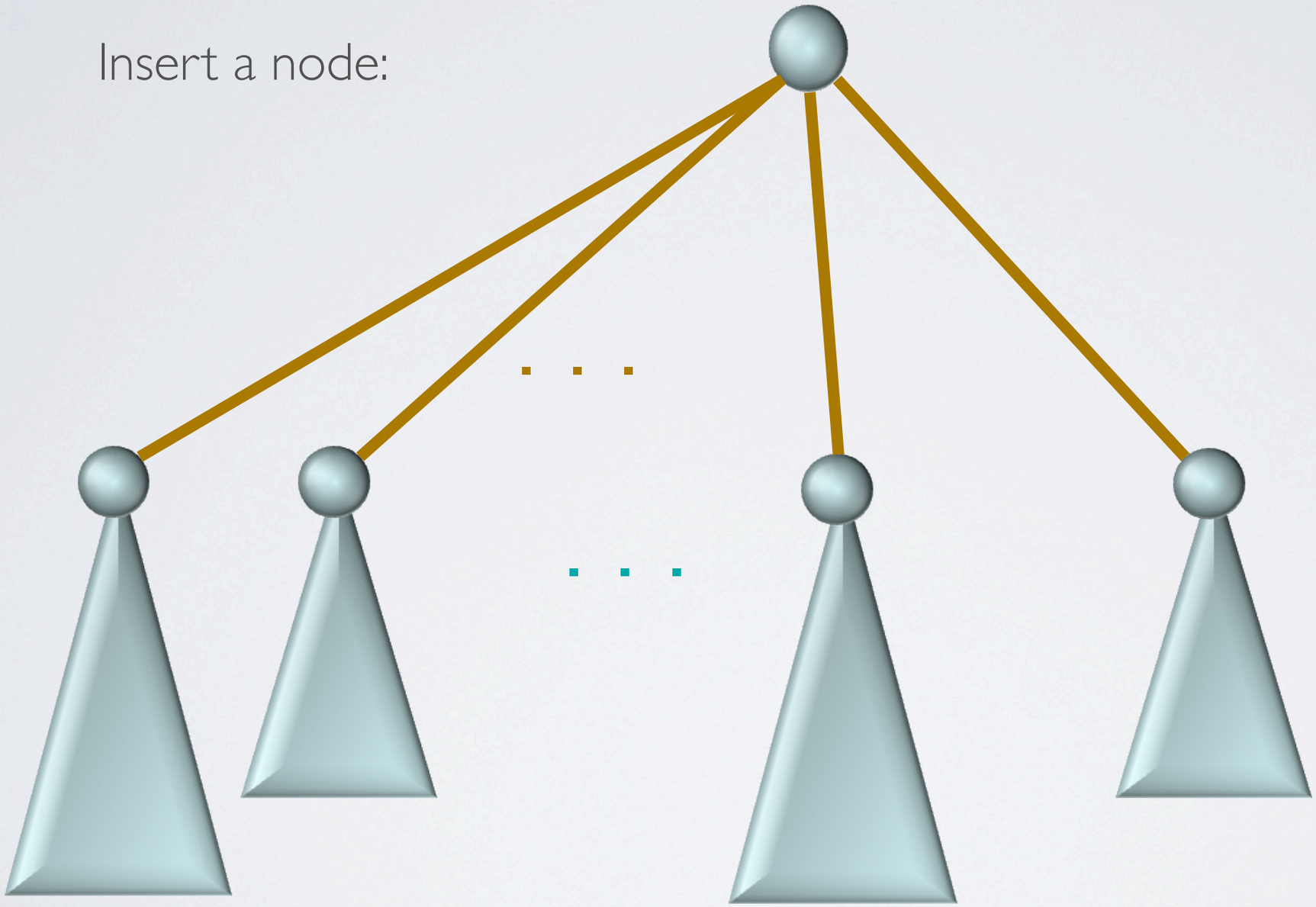
TREE EDIT OPERATIONS

Delete a node:



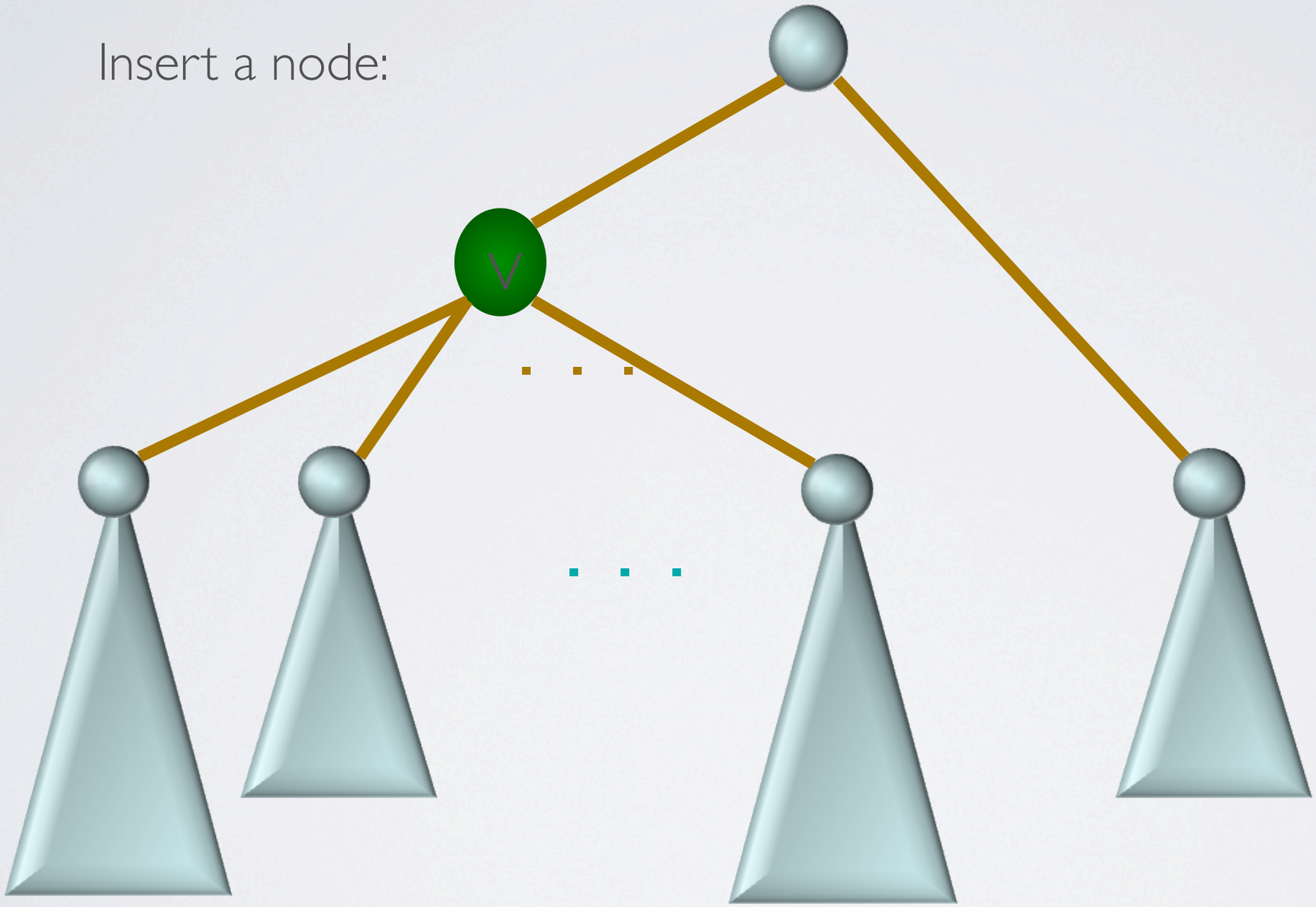
TREE EDIT OPERATIONS

Insert a node:



TREE EDIT OPERATIONS

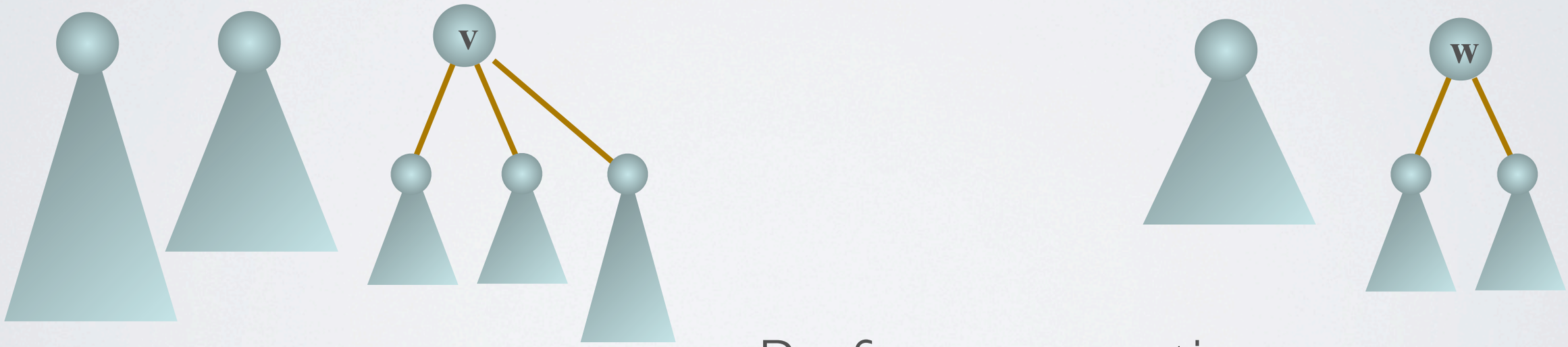
Insert a node:



TREE EDIT OPERATIONS

Recurs on the rightmost root:

$$d(F,G) = \min \begin{cases} \text{Delete v} \\ \text{Delete w} \\ \text{Match v and w} \end{cases}$$



Perform operations **recursively** and keep **lowest cost** edit sequence

POLYPHONIC TREES

Comparison

Use Selkow's tree edit distance

Symbols to compare now are not pitches but multisets

POLYPHONIC TREES

Comparison

Use Selkow's tree edit distance

Symbols to compare now are not pitches but multisets

Convert multiset
into a vector

POLYPHONIC TREES

Comparison

Use Selkow's tree edit distance

Symbols to compare now are not pitches but multisets

Convert multiset into a vector

```
{{C,G,A},{2,1.5,0.25}}
```

=

```
{{C,C#,D,D#,E,F,F#,G,G#,A,A#,B},{2,0,0,0,0,0,0,1.5,0.25,0,0}}
```

POLYPHONIC TREES

Comparison

Use Selkow's tree edit distance

Symbols to compare now are not pitches but multisets

Convert multiset into a vector

```
{{C,G,A},{2,1.5,0.25}}
```

=

```
{{C,C#,D,D#,E,F,F#,G,G#,A,A#,B},{2,0,0,0,0,0,0,1.5,0.25,0,0}}
```

Vector, position determines pitch

```
[2,0,0,0,0,0,0,1.5,0.25,0,0]
```

POLYPHONIC TREES

Comparison

Use Selkow tree edit distance

Symbols to compare now are not pitches but multisets

Vector, position
determines pitch

Use classical vector similarity measures

Manhattan (L1), Cosine similarity, Euclidean distance (L2), Jaccard coefficient^{-1},
Log distance, Matching coefficient, Multisets distance, Overlap coefficient, Probabilities,
Variational distance, Hellinger distance, Harmonic Mean

POLYPHONIC TREES

Comparison

Use Selkow tree edit distance

Symbols to compare now are not pitches but multisets

Vector, position determines pitch

```
[2,0,0,0,0,0,0,1.5,0.25,0,0]
```

Use classical vector similarity measures

Manhattan (L1), Cosine similarity, Euclidean distance (L2), Jaccard coefficient^{-1},
Log distance, Matching coefficient, Multisets distance, Overlap coefficient, Probabilities,
Variational distance, Hellinger distance, Harmonic Mean

ALGORITHMIC COMPOSITION

ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**

Sequence generation system

ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**

Sequence generation system

↓

Tree encoding

ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**

Sequence generation system

↓

Tree encoding → **Genetic operators using trees**

ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

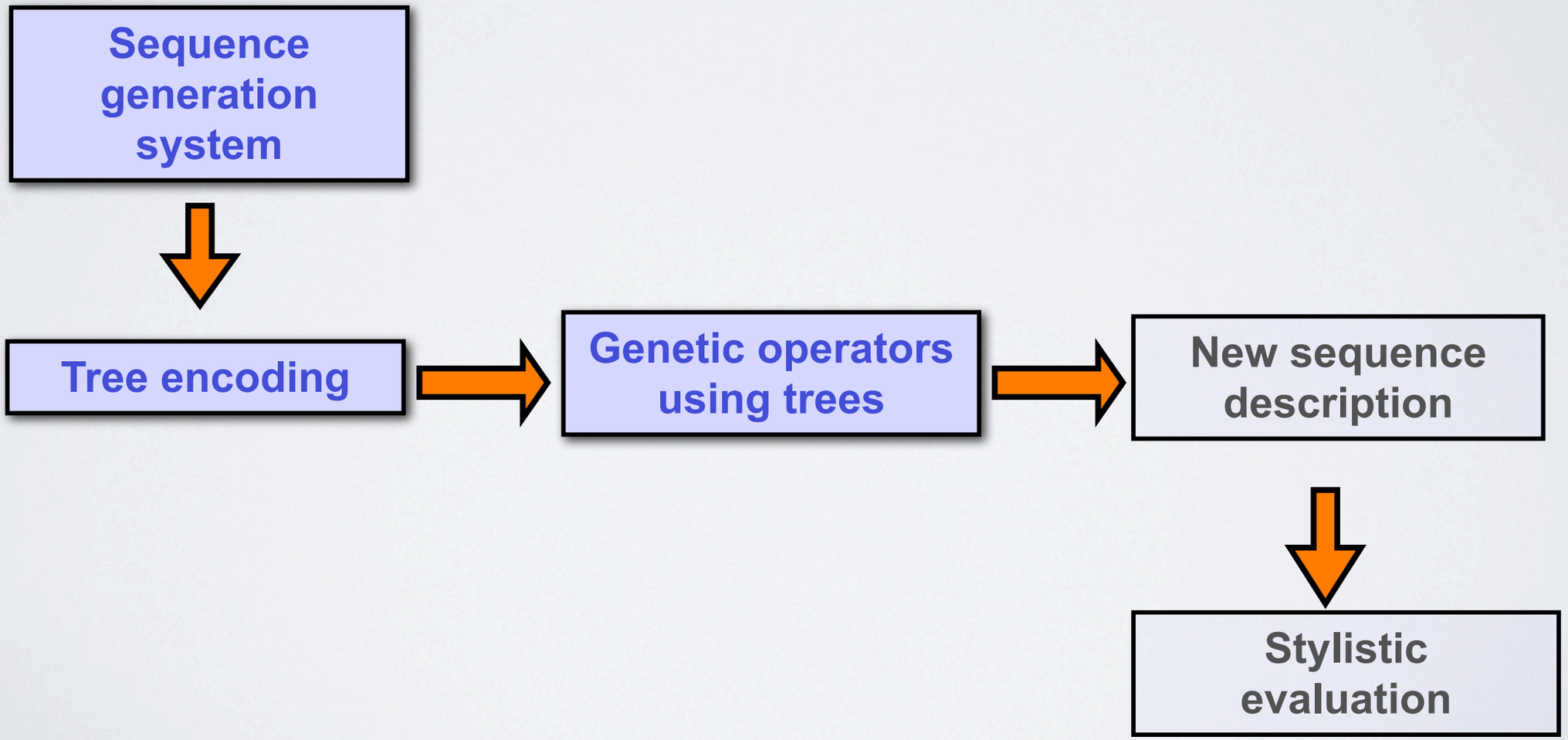
Composer ⟷ **interaction** ⟷ **Specialized critic**

ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**

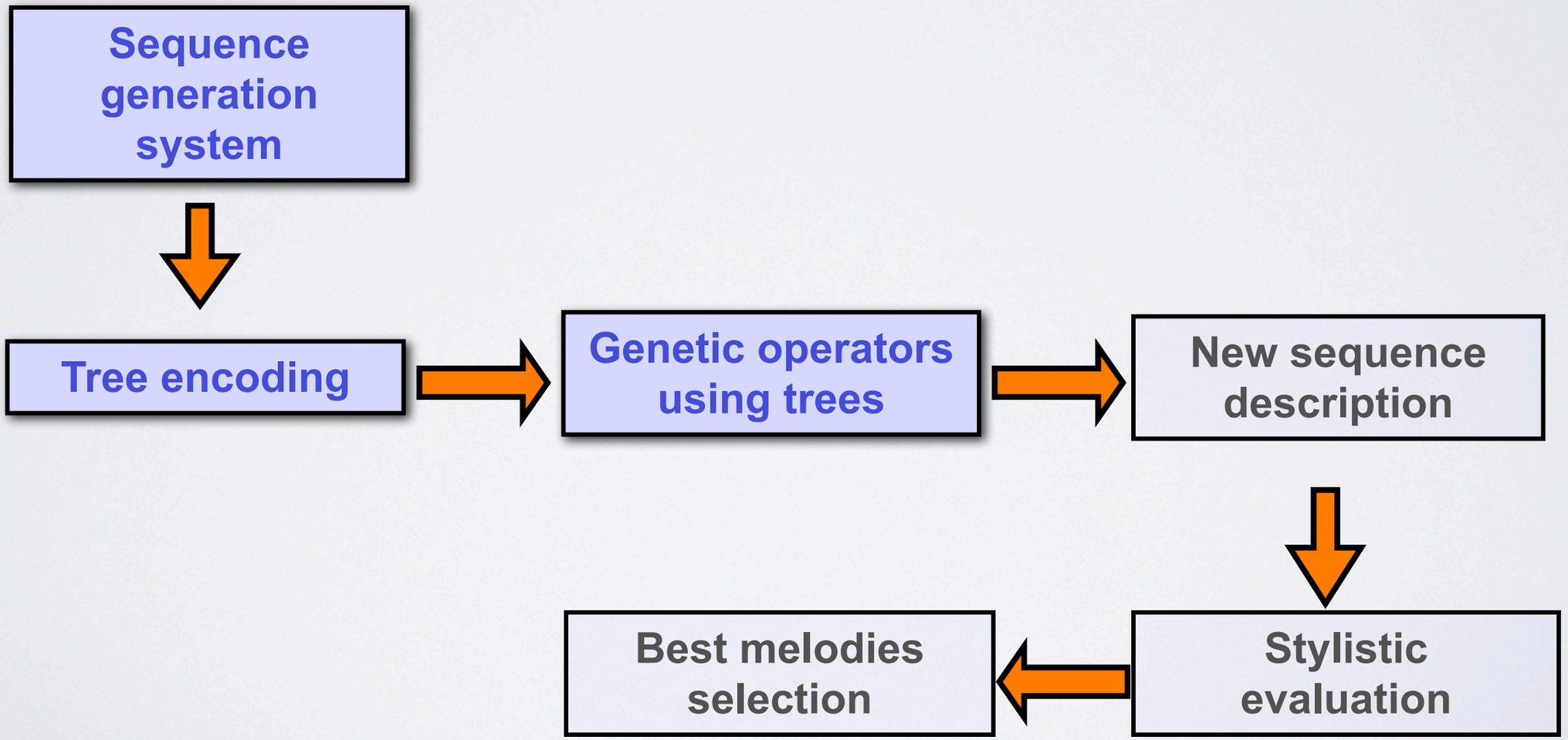


ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**

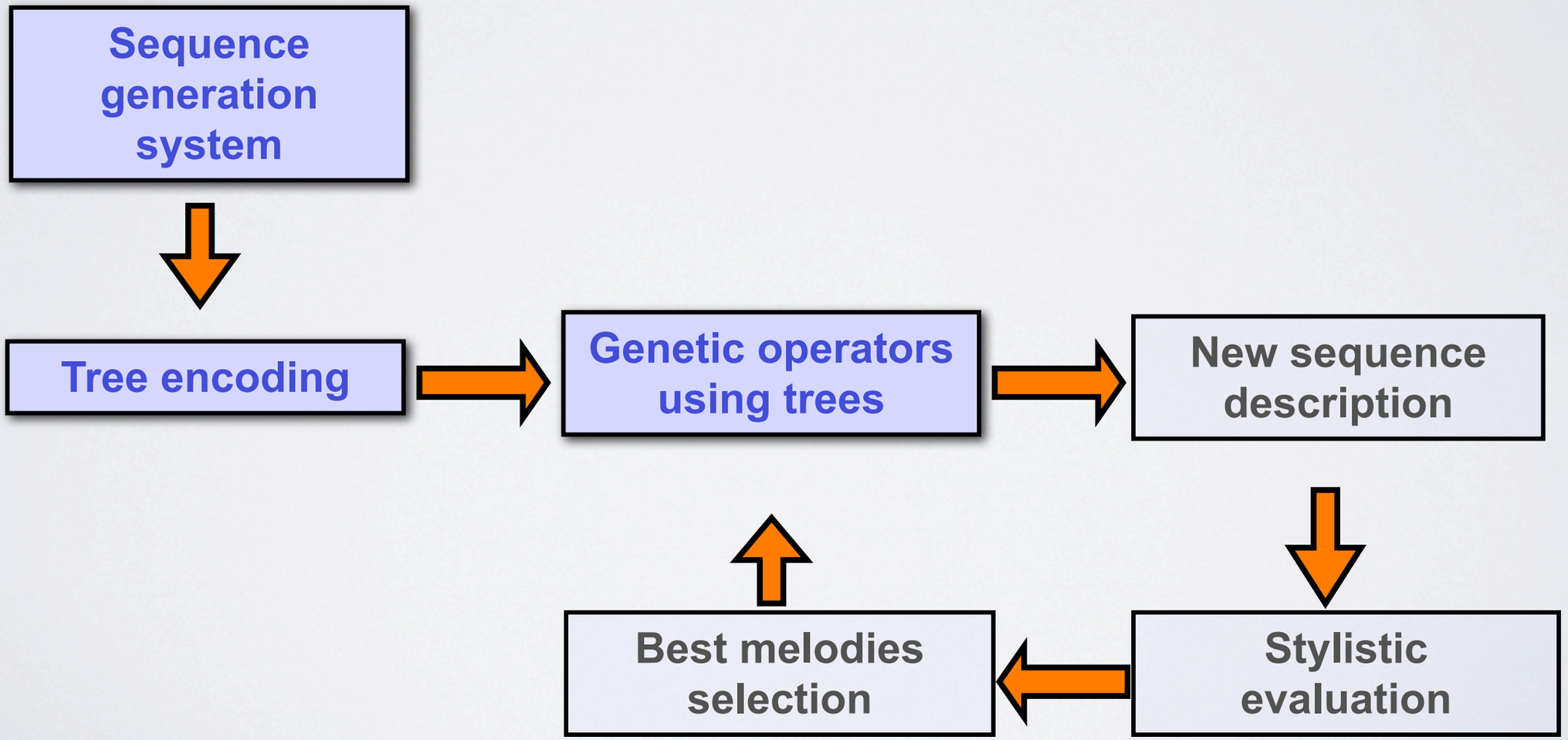


ALGORITHMIC COMPOSITION

Genetic algorithms and Pattern Recognition

Genetic system based on style

Composer ⟷ **interaction** ⟷ **Specialized critic**



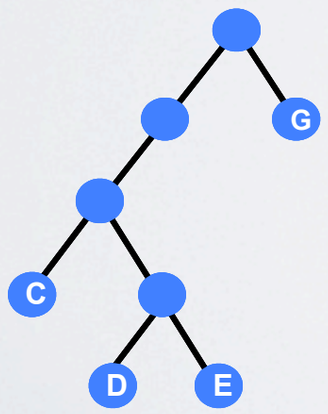
ALGORITHMIC COMPOSITION

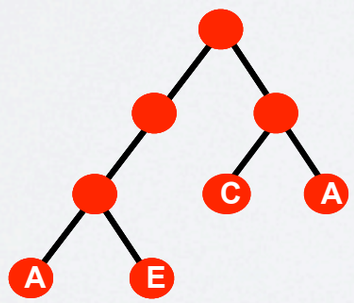
The composer

Sequence generation system

Tree encoding

Genetic operators using trees





ALGORITHMIC COMPOSITION

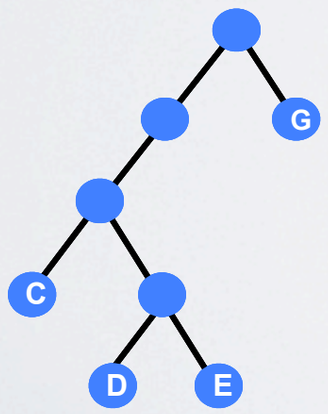
The composer

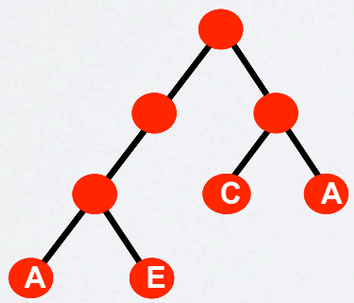
Sequence generation system

Random generators
Markov chains,
…

Tree encoding

Genetic operators using trees





ALGORITHMIC COMPOSITION

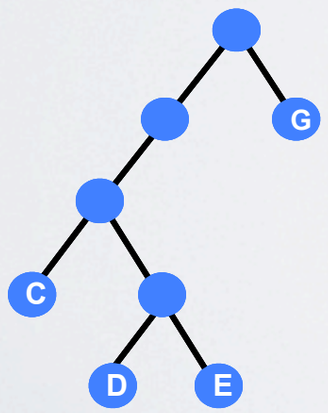
The composer

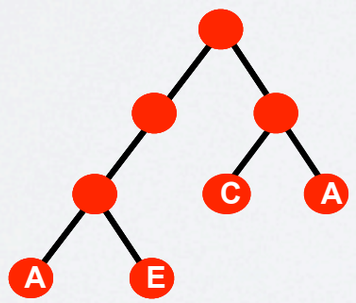
Sequence generation system

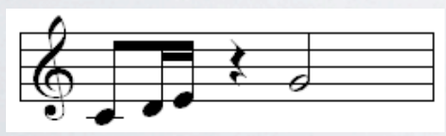
Random generators
Markov chains,
…

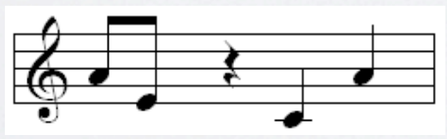
Tree encoding

Genetic operators using trees









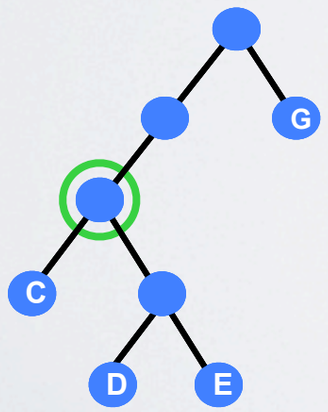
ALGORITHMIC COMPOSITION

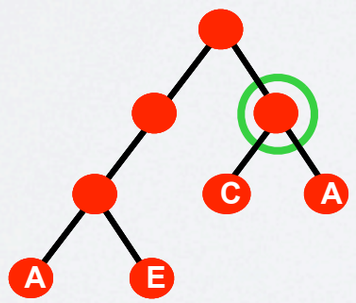
The composer

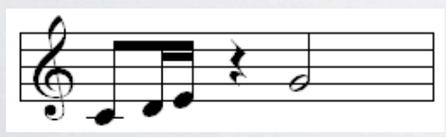
Sequence generation system

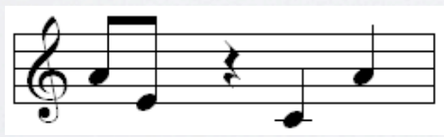
Random generators
Markov chains,
…

Tree encoding → **Genetic operators using trees**









ALGORITHMIC COMPOSITION

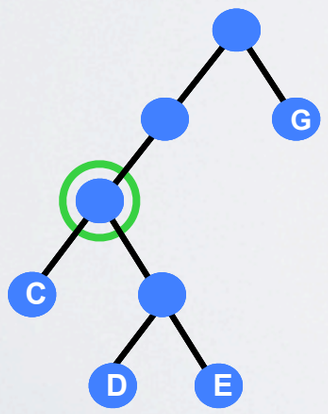
The composer

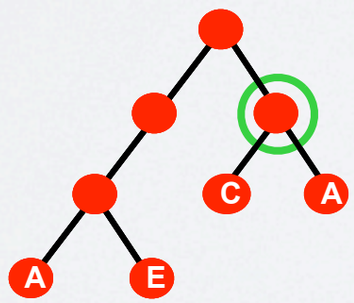
Sequence generation system

Random generators
Markov chains,
…

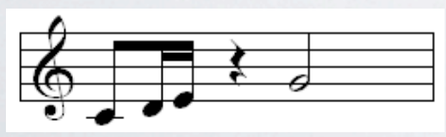
Tree encoding

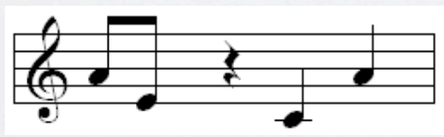
Genetic operators using trees





crossover





ALGORITHMIC COMPOSITION

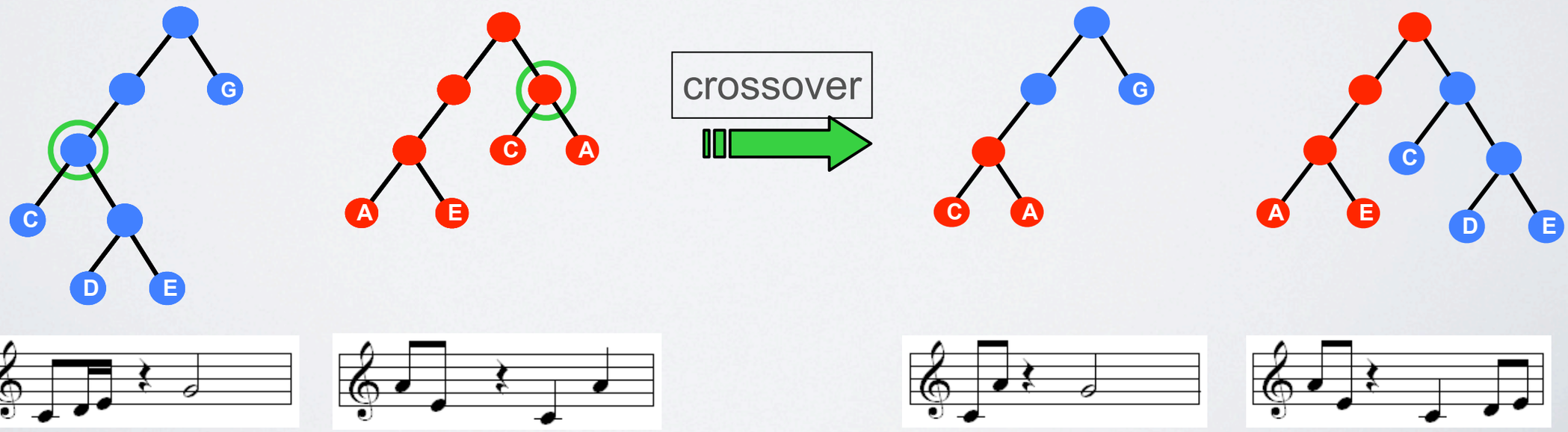
The composer

Sequence generation system

Random generators
Markov chains,
…

Tree encoding

Genetic operators using trees



CONCLUSIONS

CONCLUSIONS

- Trees encode **structure** in a natural way

CONCLUSIONS

- Trees encode **structure** in a natural way
  - Over strings: **remove** a **parameter** to be tuned (**rhythm**) as it gets encoded implicitly encoded

CONCLUSIONS

- Trees encode **structure** in a natural way
  - Over strings: **remove** a **parameter** to be tuned (**rhythm**) as it gets encoded implicitly encoded
- Easy to **manipulate** with meaningful results

CONCLUSIONS

- Trees encode **structure** in a natural way
  - Over strings: **remove** a **parameter** to be tuned (**rhythm**) as it gets encoded implicitly encoded
    - Easy to **manipulate** with meaningful results
- Ready to incorporate in the structure **more** things **than just pitch**

CONCLUSIONS

- Trees encode **structure** in a natural way
  - Over strings: **remove** a **parameter** to be tuned (**rhythm**) as it gets encoded implicitly encoded
    - Easy to **manipulate** with meaningful results
- Ready to incorporate in the structure **more** things **than just pitch**
- We are able to **learn** structures from examples

CONCLUSIONS

- Trees encode **structure** in a natural way
  - Over strings: **remove** a **parameter** to be tuned (**rhythm**) as it gets encoded implicitly encoded
    - Easy to **manipulate** with meaningful results
- Ready to incorporate in the structure **more** things **than just pitch**
- We are able to **learn** structures from examples
  - this structures can be **re-generated** in a composition scenario

SELECTED PUBLICATIONS

- "**Symbolic music comparison with tree data structures**"
  Rizo, David
  PhD Thesis. Universidad de Alicante (2010)
  http://www.dlsi.ua.es/gent/drizo/thesis/thesisdrizo.pdf

- "**Melodic Identification Using Probabilistic Tree Automata**"
  Bernabeu, José F. and Calera-Rubio, Jorge and Iñesta, José M. and Rizo, David
  Journal of New Music Research, vol. 40, pp. 93-103 (2011)

- "**Polyphonic Music Retrieval with Classifier Ensembles**"
  Rizo, David and Iñesta José M. and Lemström, Kjell
  Journal of New Music Research, vol. 40, pp. 313-324 (2011)

- "**New partially labelled tree similarity measure: a case study**"
  Rizo, David and Iñesta, J.Manuel
  Structural, Syntactic, and Statistical Pattern Recognition, ISBN: 978-3-642-14979-5, pp. 296--305, Cesme, Turkey (2010)
  http://grfia.dlsi.ua.es/repositori/grfia/pubs/248/ssspr10-cr.pdf

Merci pour votre attention

MaMuX
March, 2012

David Rizo
Univ. Alicante

ircam Centre Pompidou

PRAIg

IUII

Universitat d'Alacant
Universidad de Alicante