Improving and Adapting Finite State Transducer Methods for Musical Accompaniment

Jonathan P. Forsyth, Rachel M. Bittner,
Michael Musick, & Juan P. Bello
Music and Audio Research Lab (MARL)
New York University
New York, NY

NYU | STEINHARDT

MARL

introduction

given an input melody,
generate harmonic accompaniment

data-driven approaches

finite state machines are a natural fit for solving this problem, and have been used extensively in natural language processing

NYU | STEINHARDT

MARL

outline

introduction to finite state machines

general approach to generating musical accompaniment

application of this approach

NYU | STEINHARDT

MARL

finite state machines

used in natural language processing, bioinformatics, computer vision

directed graphs (nodes, edges)

edges labelled with symbols

edges and nodes can have weights (e.g. probabilities)

NYU | STEINHARDT

MARL

finite state machines

make it relatively easy to break down a complex problem into simpler sub-problems

allow use of a purely data-driven approach

can also incorporate higher-level knowledge into system

require relatively little training data

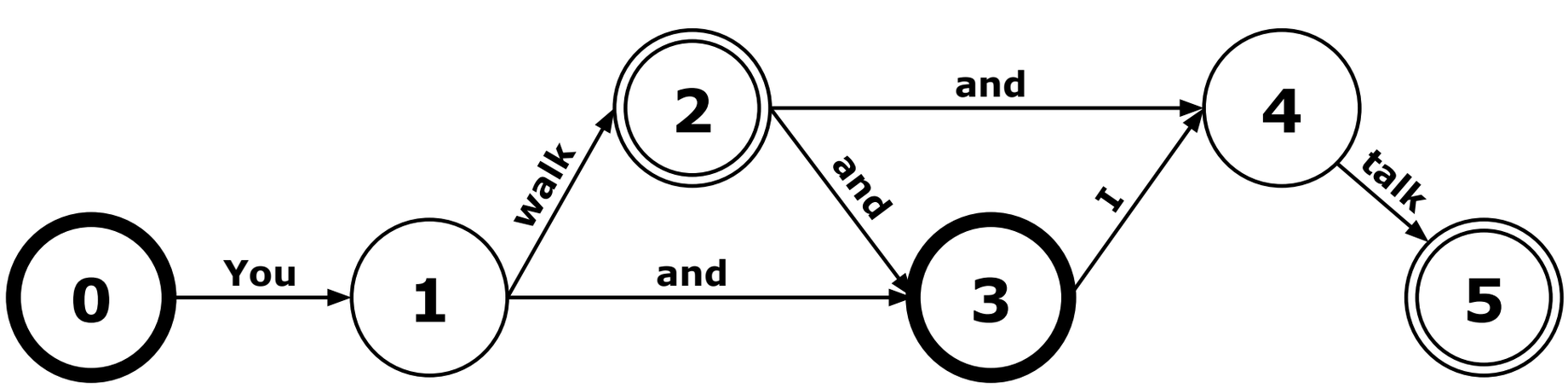
NYU | STEINHARDT

MARL

finite state automaton

FSA defined by the 5-tuple:

$$\langle \Sigma, Q, I, F, \delta \rangle$$

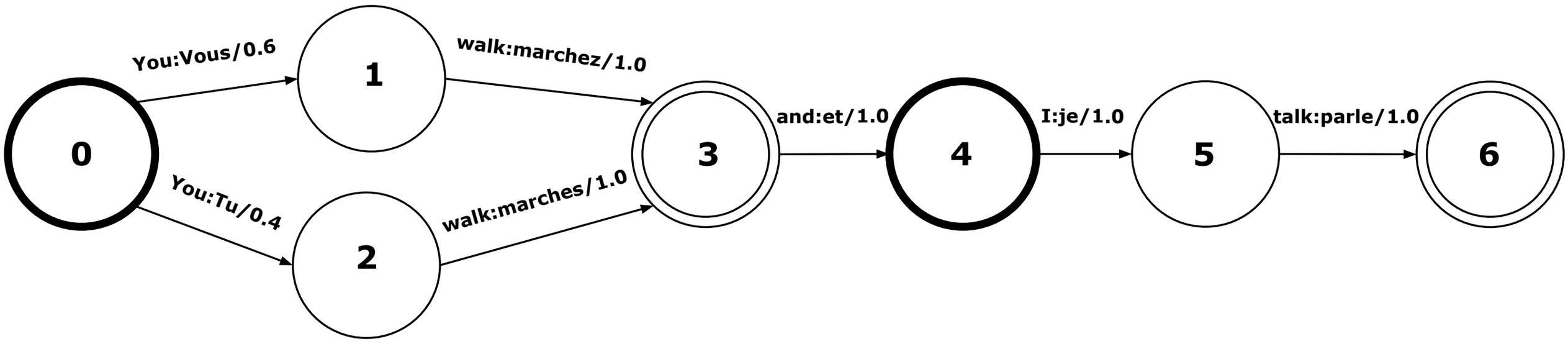


(sometimes referred to as an “acceptor”)

finite state transducer

FST defined by the 6-tuple:

$$\langle \Sigma, \Delta, Q, I, F, \delta \rangle$$



NYU STEINHARDT

MARL

finite state operations

composition

determinization

minimization

shortest path

NYU | STEINHARDT

MARL

generating accompaniment

approach to accompaniment generation
borrowed from speech recognition

speech recognition problem:

find the most likely sequence of words
given a sequence of phonemes

NYU | STEINHARDT

MARL

generating accompaniment

general framework:

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\Sigma^*}{\arg\max} \Pr\left[\mathbf{c} \mid \mathbf{m}\right] = \underset{\mathbf{c}\in\Sigma^*}{\arg\max} \Pr\left[\mathbf{m} \mid \mathbf{c}\right] \cdot \Pr[\mathbf{c}]$$

$\mathbf{m}$: input melodic sequence

$\hat{\mathbf{c}}$: most likely sequence of accompaniment chords

NYU | STEINHARDT

MARL

generating accompaniment

$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \Sigma^*}{\arg\max} \Pr\left[\mathbf{c} \mid \mathbf{m}\right] = \underset{\mathbf{c} \in \Sigma^*}{\arg\max} \Pr\left[\mathbf{m} \mid \mathbf{c}\right] \cdot \Pr[\mathbf{c}]$$

model components separately:

FST (*L*) to model Pr[m | c]

FSA (*G*) to model Pr[c]

NYU | STEINHARDT

MARL

generating accompaniment

input alphabet (Σ_i): pitch classes

output alphabet (Σ_o): chord symbols or quantized chroma vectors[1]

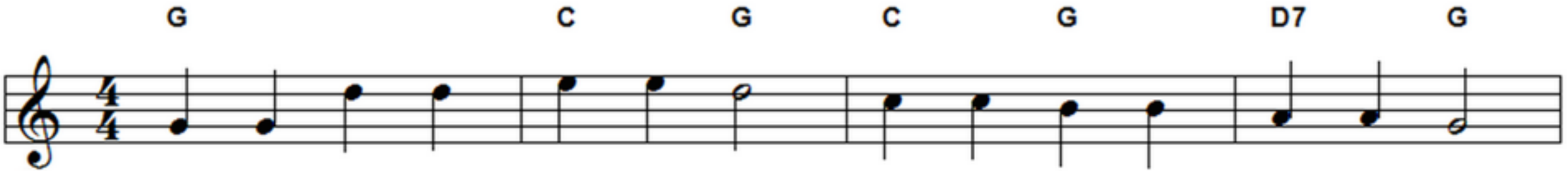
training data: pairs of sequences of melodic symbols and sequences of harmonic symbols

[1] Forsyth, J. P., & Bello, J. P. (2013). Generating Musical Accompaniment Using Finite State Transducers. In *Proceedings of the16th International Conference on Digital Audio Effects (DAFx-13)*.

NYU | STEINHARDT

MARL

training *L* FST



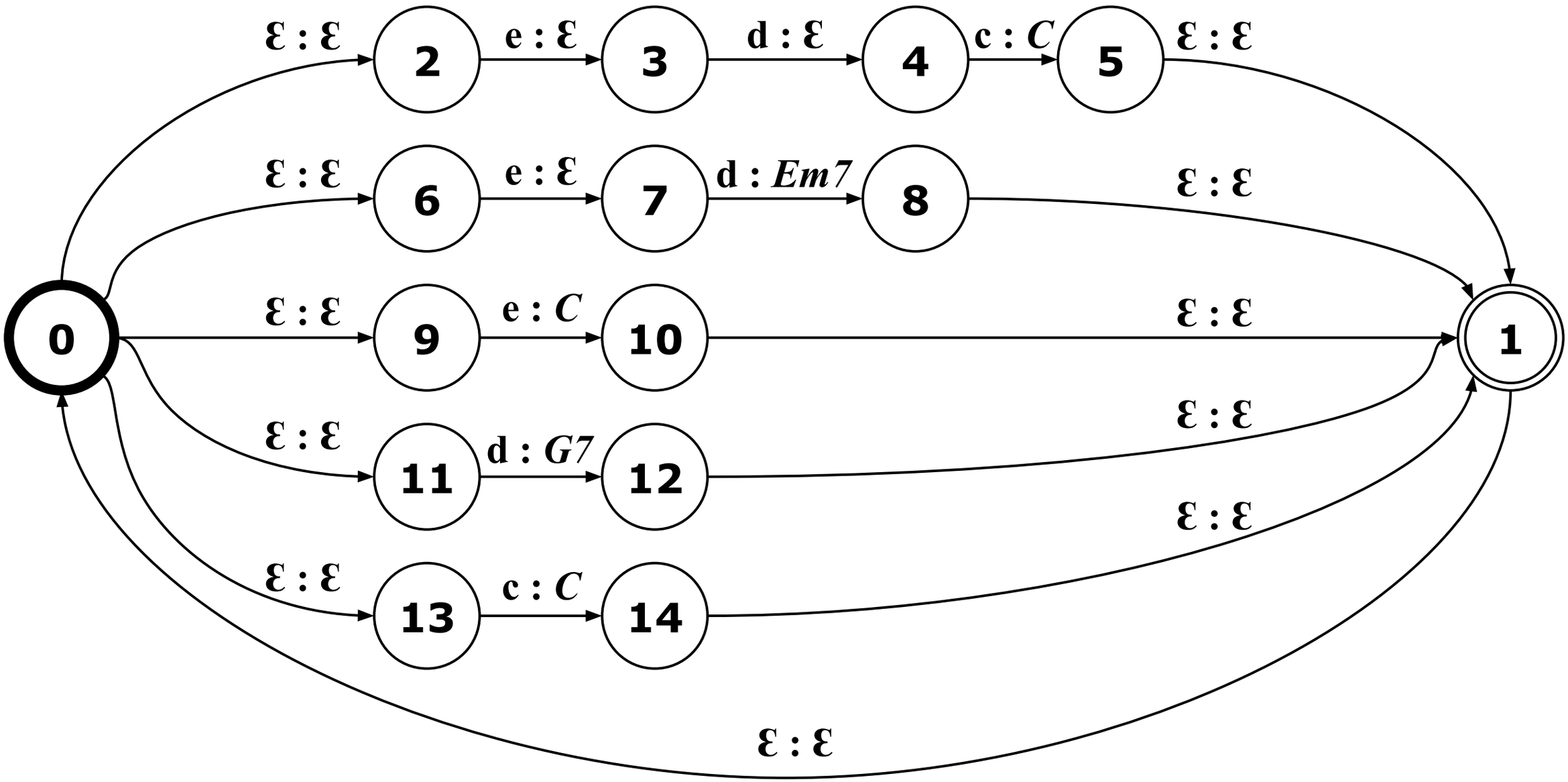
example training sequence

training *L* FST

generates the training sequences:

g,g,d,d ⟶ G
e,e ⟶ C
d ⟶ G
c,c ⟶ C
b,b ⟶ G
a,a ⟶ D7
g ⟶ G

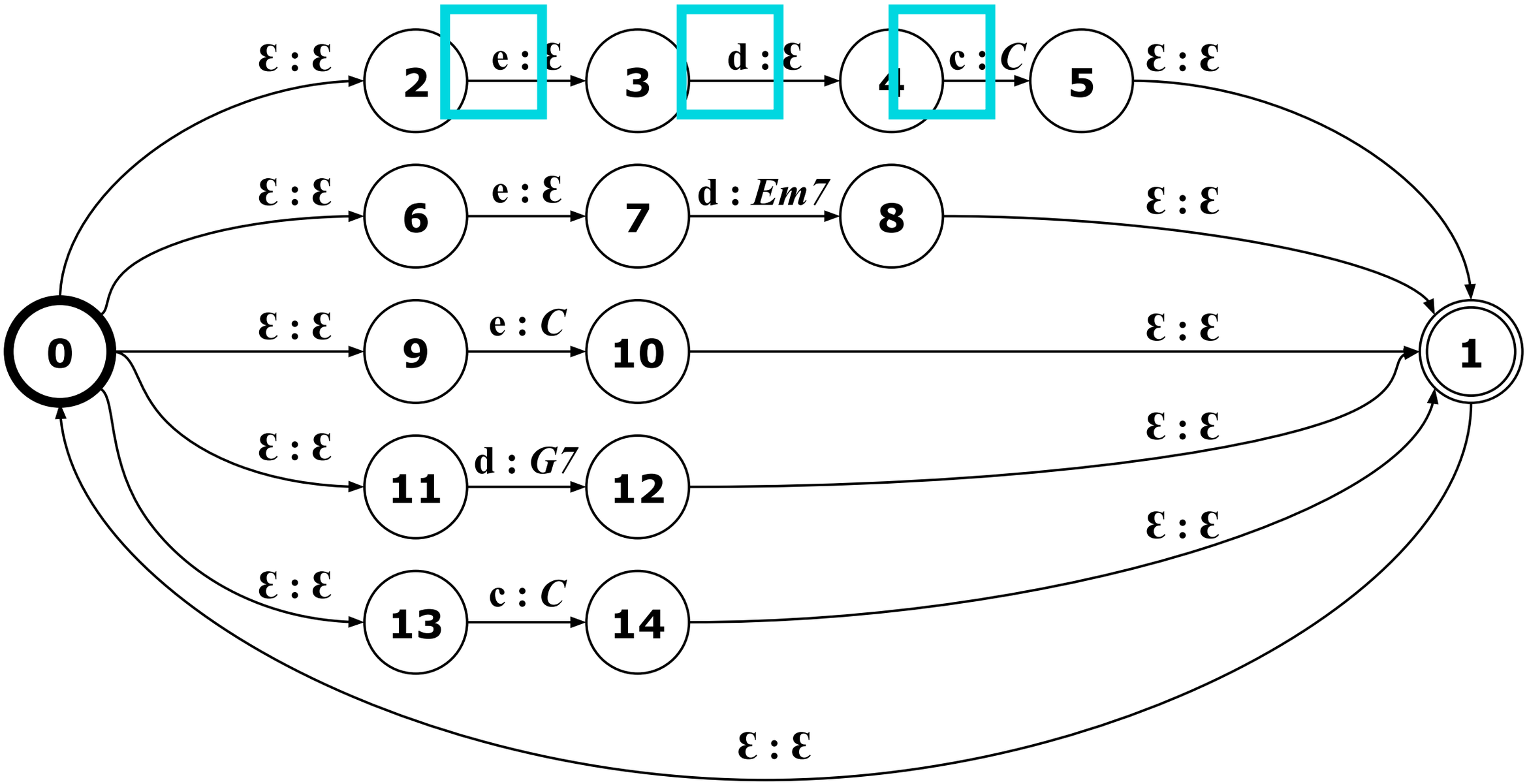
training *L* FST



NYU | STEINHARDT

MARL

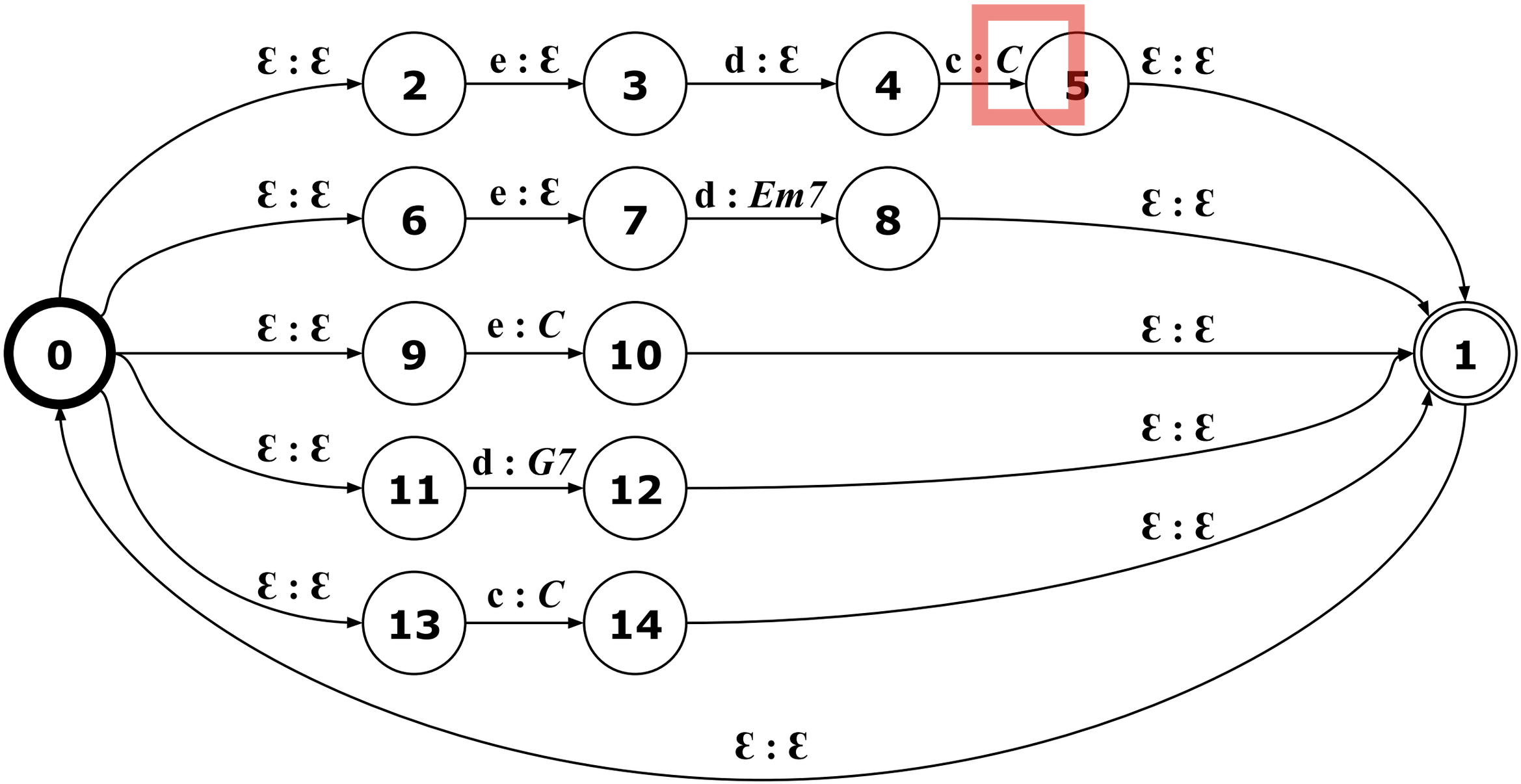
training *L* FST



NYU STEINHARDT

MARL

training *L* FST



NYU | STEINHARDT

MARL

training *L* FST

same melodic sequence may map to different chords

(e.g. [*c,e,g*] may map to *C* or *Amin7*)

training *L* FST

same melodic sequence may map to different chords

(e.g. [*c,e,g*] may map to *C* or *Amin7*)

analogous problem in speech recognition:

same set of phonemes map to “red” and “read”

training *L* FST

an FST with conflicting mappings cannot be determinized, which is required for minimization

NYU | STEINHARDT

MARL

training *L* FST

solution:

append different disambiguation symbol (typically, ‘#0’, ‘#1’, etc.) to each sequence that maps to more than one chord

NYU | STEINHARDT

MARL

training *L* FST

e.g., our training data contains pairs
[*c,e,g* : *C*] and [*c,e,g* : *Amin7*]

add disambiguation symbols to get the pairs

[*c,e,g,#0* : *C*] and [*c,e,g,#1* : *Amin7*]

NYU STEINHARDT

MARL

training *L* FST

after training full model, replace disambiguation symbols with **ε**

NYU | STEINHARDT

MARL

training *G* FSA

use an n-gram trained on all length-n chord sequences in training data

n-grams model sequential data with context of n-1 symbols

n-gram represented as an FSA (*G*)

NYU | STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU | STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU | STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU | STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU STEINHARDT

MARL

full model

combine the individual models:

$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU STEINHARDT

MARL

full model

combine the individual models:

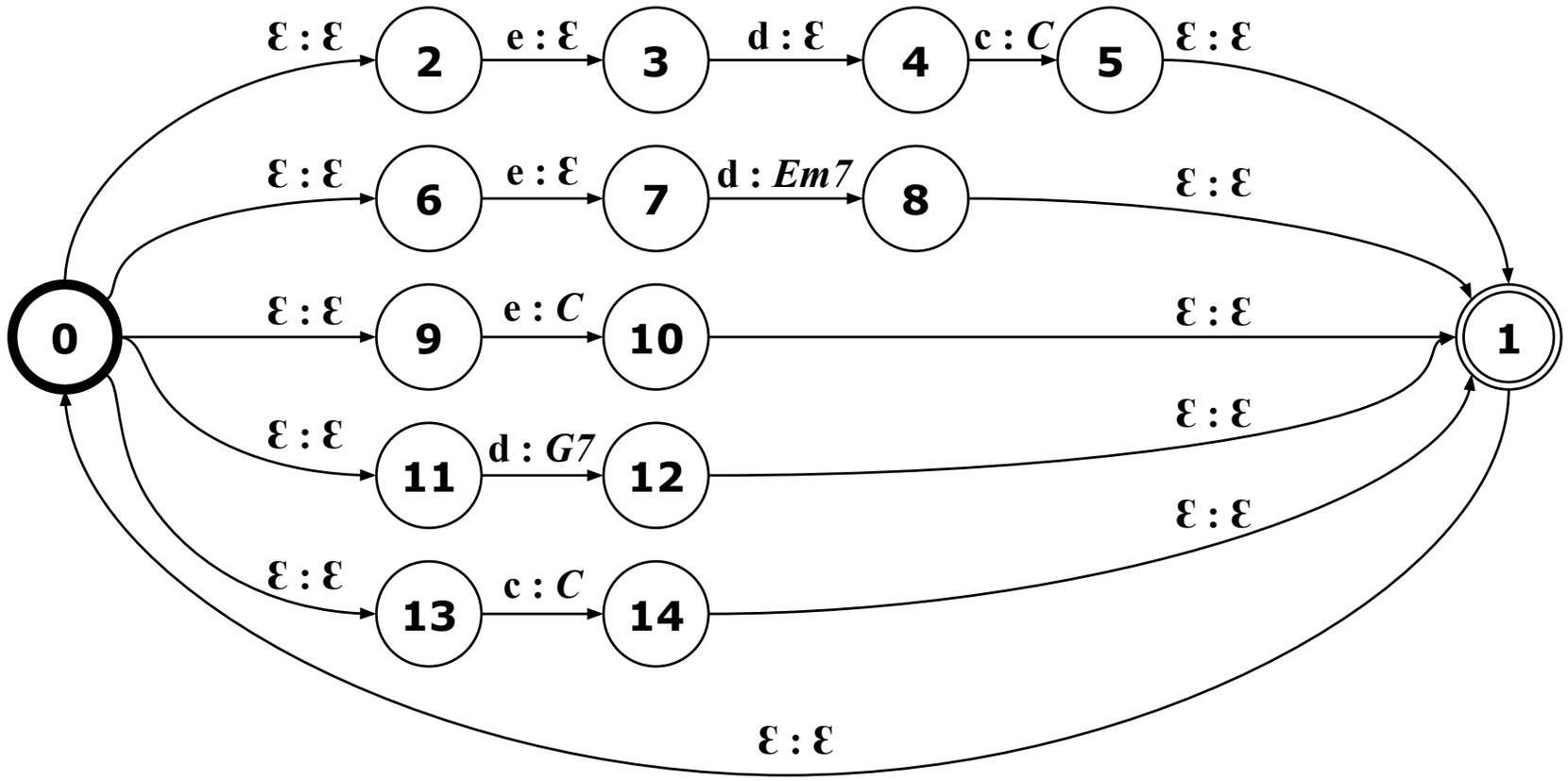
$$\mathcal{T} = \pi_\epsilon(min(det(\tilde{\mathcal{L}} \circ \mathcal{G})))$$

NYU STEINHARDT

MARL

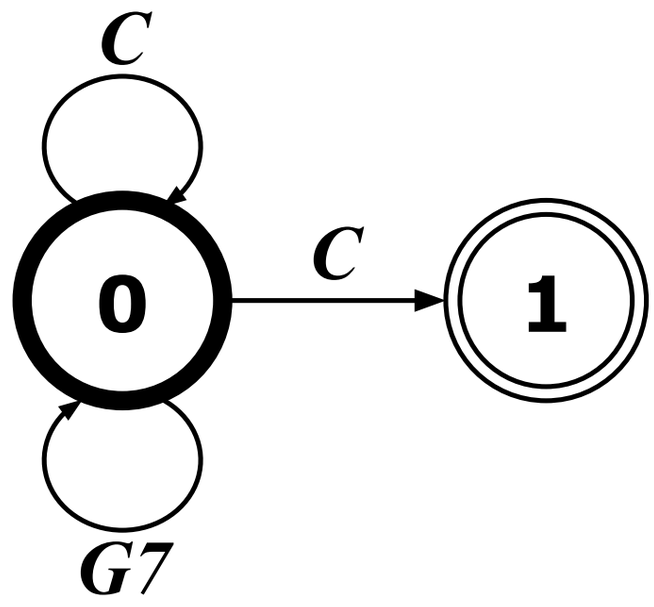
chord generation example

melody->chord FST (*L*):



chord generation example

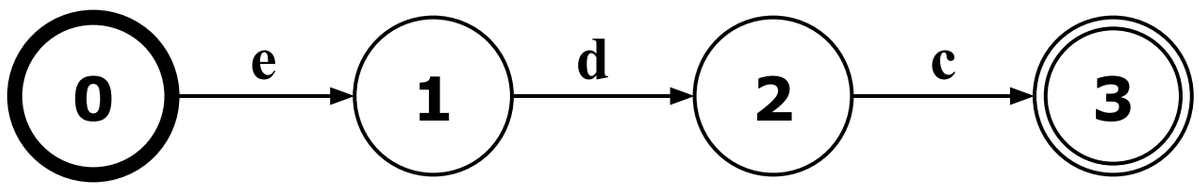
chord model FSA (*G*):



chord generation example

input melody sequence m = [*e*,*d*,*c*]

construct “linear chain” FST, *M*, from melody:

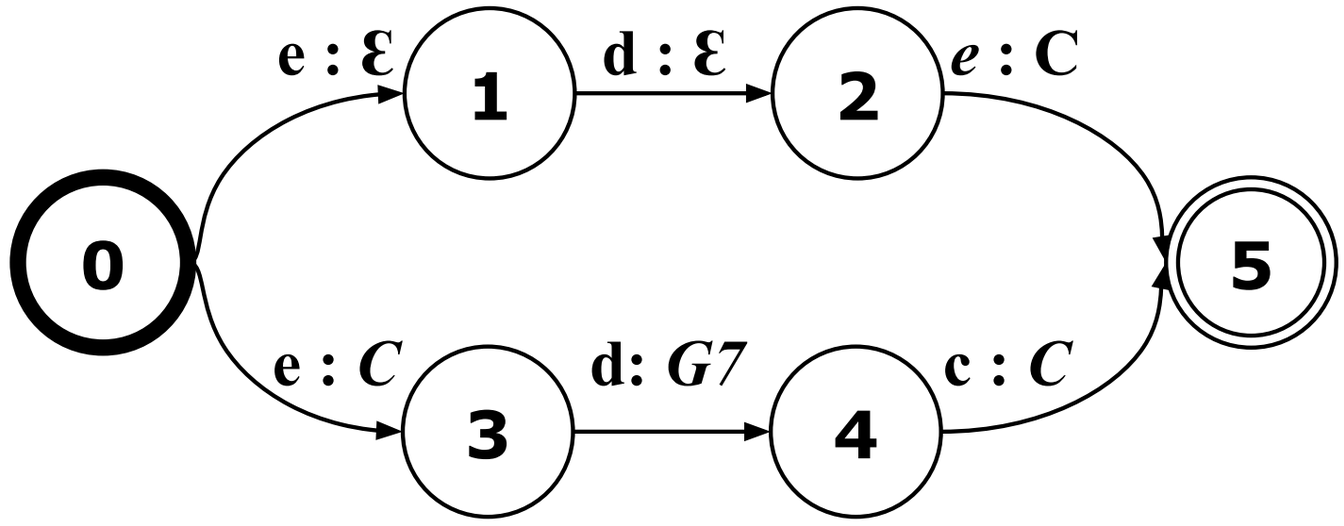


NYU STEINHARDT

MARL

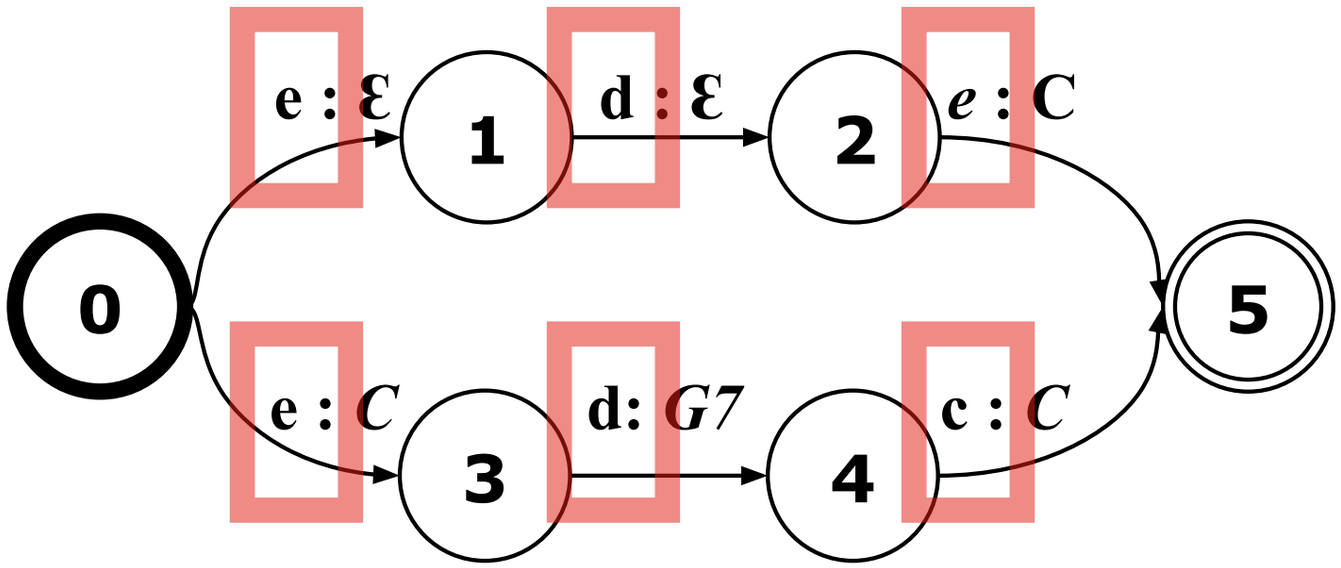
chord generation example

compute $C = M \circ T$



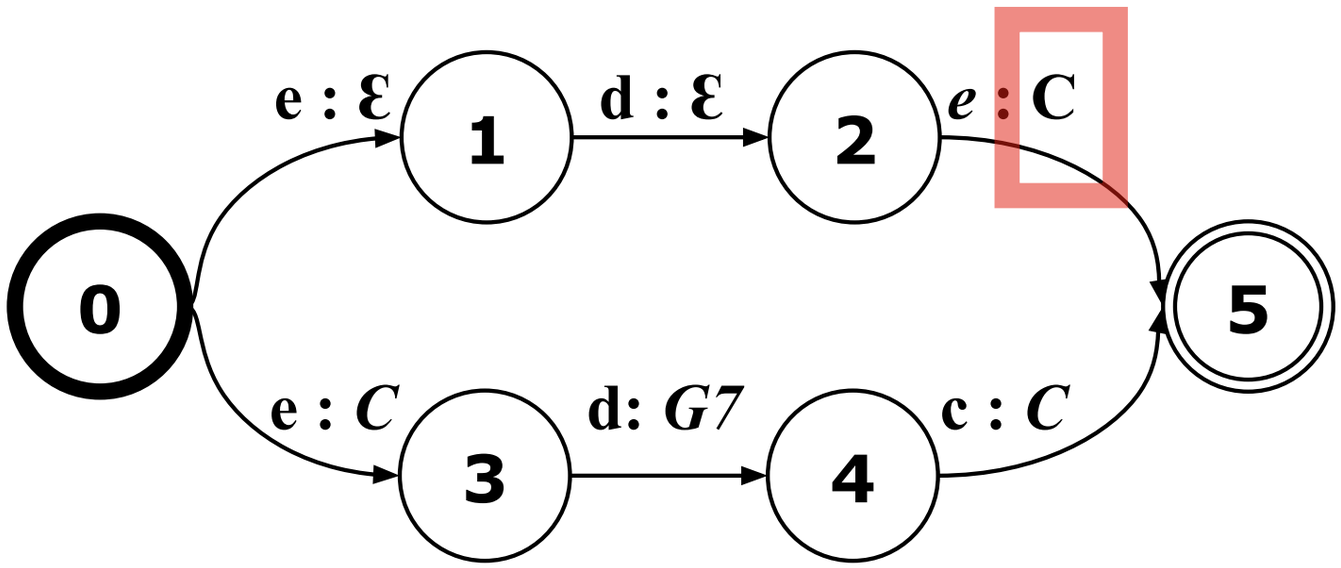
chord generation example

compute $C = M \circ T$



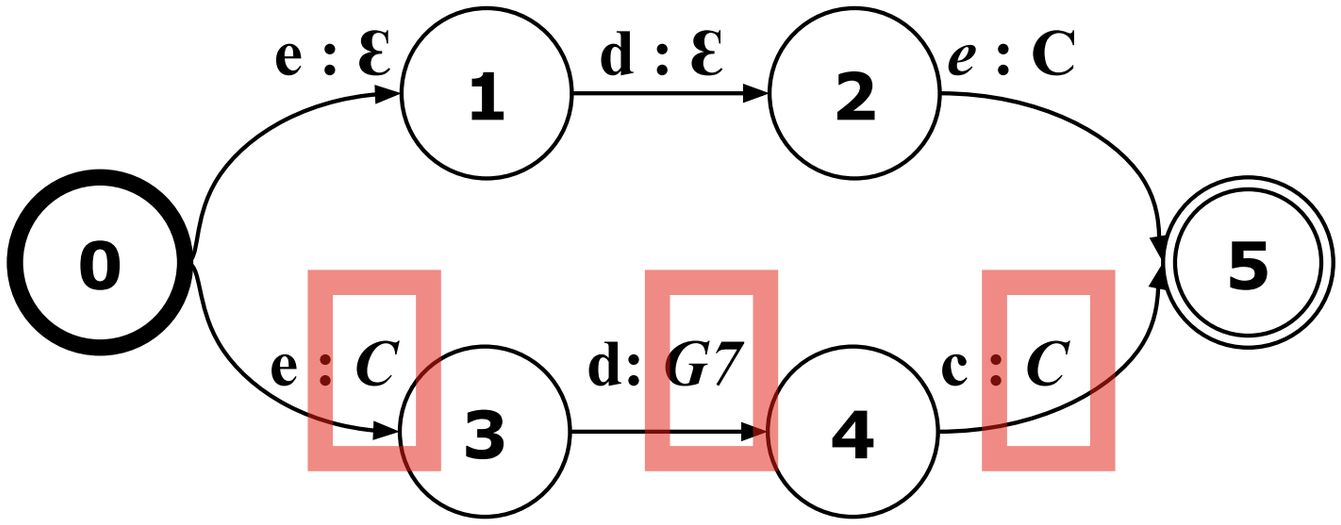
chord generation example

compute $C = M \circ T$



chord generation example

compute $C = M \circ T$



NYU STEINHARDT

MARL

chord generation example

finding the accompaniment sequence from *C*:

shortest path (most likely chords)

randomly generate (uniform sampling or weights as negative log probabilities)

NYU | STEINHARDT

MARL

extension to rhythm

harmonic accompaniment model doesn't incorporate rhythm, so we add an separate rhythmic model

extension to rhythm

use the same basic approach as in the case of generating harmonic accompaniment :

$$\hat{\mathbf{r}} = \arg\max_{\mathbf{r} \in \Sigma^*} \Pr\left[\mathbf{r} \mid \mathbf{s}\right] = \arg\max_{\mathbf{r} \in \Sigma^*} \Pr\left[\mathbf{s} \mid \mathbf{r}\right] \cdot \Pr[\mathbf{r}]$$

alphabet consists of quantized inter-onset times

NYU | STEINHARDT

MARL

The Harmonically Ecosystemic Machine

collaboration with Michael Musick

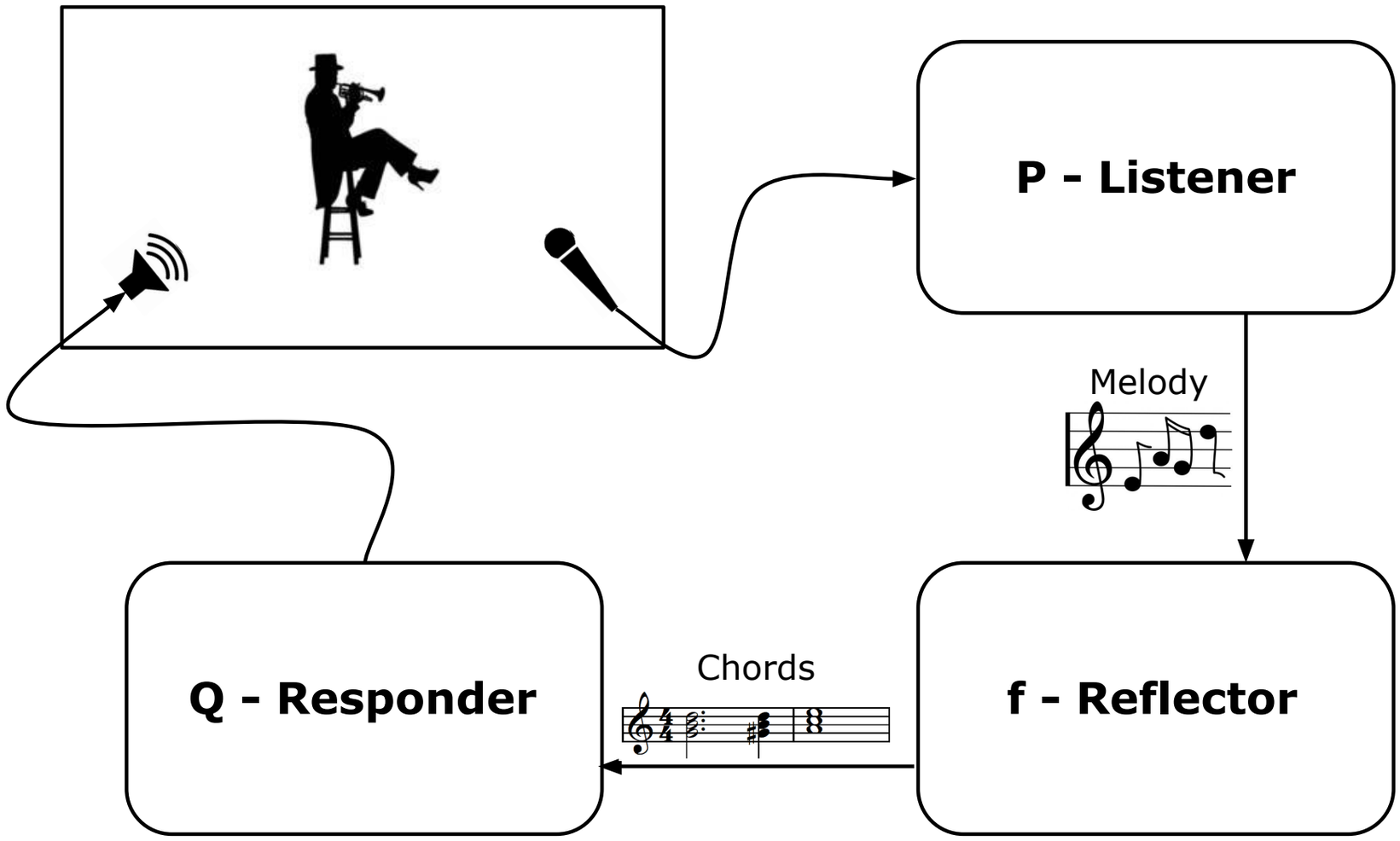
interactive installation that incorporates FST-based harmony and rhythm generation[2]

[2] Musick, M., Forsyth, J.P. , & Bittner, R. (2015) Building a Harmonically Ecosystemic Machine: Combining Sonic Ecosystems with Models of Contemporary Harmonic Language. In *Proc. of the 41th International Computer Music Conference (ICMC), Denton, TX.*

NYU | STEINHARDT

MARL

The Harmonically Ecosystemic Machine



NYU STEINHARDT

MARL

The Harmonically Ecosystemic Machine

Listener and *Responder* modules implemented in SuperCollider

Reflector/Decision Maker module implemented in Python

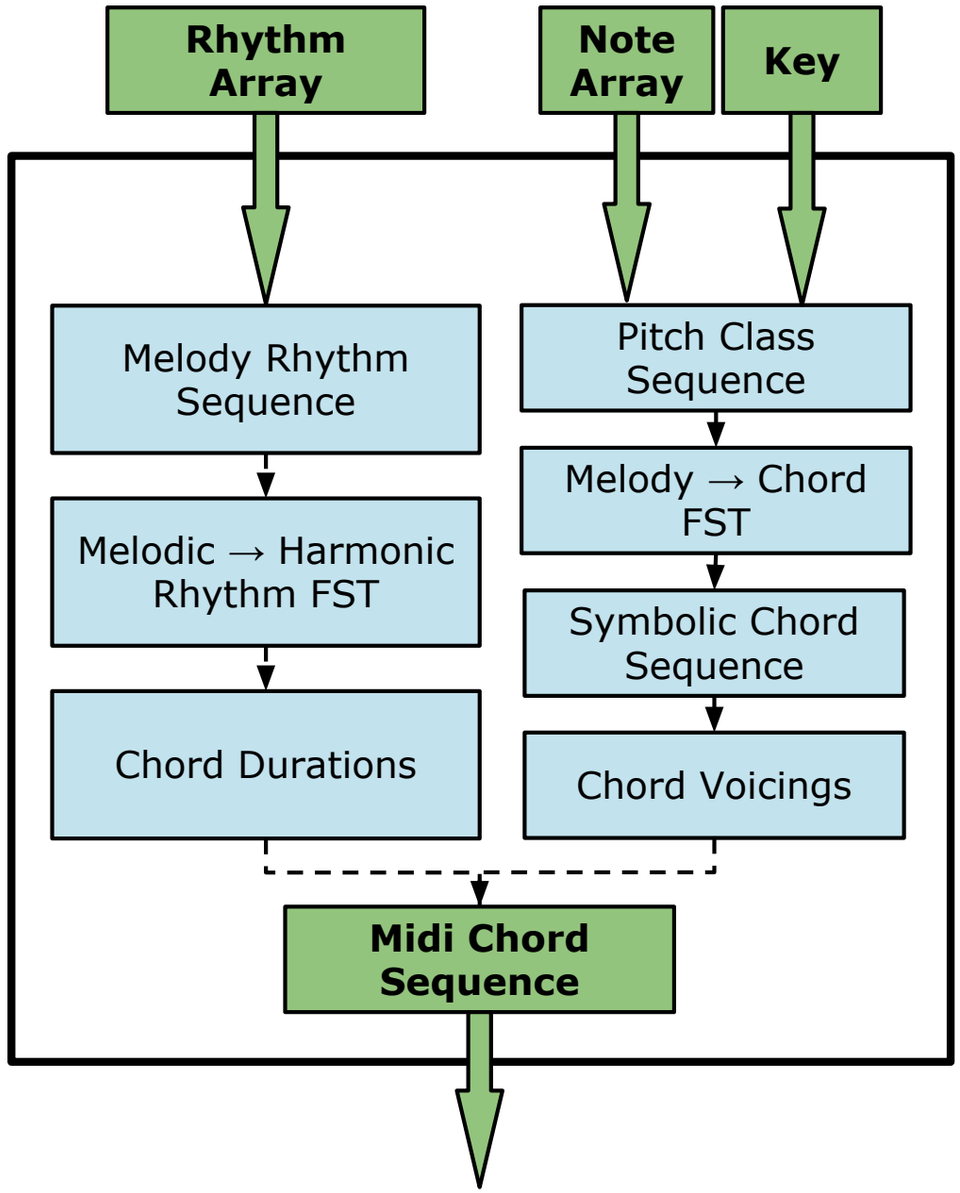
uses OpenFst and OpenGrm libraries

communication via OSC

NYU | STEINHARDT

MARL

The Harmonically Ecosystemic Machine



NYU STEINHARDT

MARL

The Harmonically Ecosystemic Machine

training data:

harmony: Rock Corpus dataset

rhythm: works by Mozart

NYU | STEINHARDT

MARL

The Harmonically Ecosystemic Machine

why?

aesthetics: create familiar harmony and rhythms for participants

practicality: limited number of useful datasets

NYU | STEINHARDT

MARL

The Harmonically Ecosystemic Machine

installation is located in the UNT Music Building room 2009

NYU | STEINHARDT

MARL

adapting to real-time

computational efficiency more important in real-time than offline system

improving efficiency involves tradeoffs

NYU | STEINHARDT

MARL

adapting to real-time

reduce size of *L* FST to improve efficiency:

key normalize melody/harmony

... but this requires key estimation, which is error-prone

NYU | STEINHARDT

MARL

adapting to real-time

reduce size of *G* FST to improve efficiency:

limit n-gram order

... but longer context may be preferable

NYU | STEINHARDT

MARL

adapting to real-time

length of melodic input also impacts efficiency

- longer input melody is more computationally expensive
- ... but longer context generally provides better harmonizations
- also impacts system latency

adapting to real-time

for our implementation:

key-normalized *L* FST

key independent chord representation

n-gram order = 3

input melody length of 8 beats

NYU | STEINHARDT

MARL

conclusions

harmonic generation system was only offline

project was an opportunity to explore system in a real-time setting

practical and aesthetic benefits

NYU | STEINHARDT

MARL

Thanks!

Music Building room 2009

http://steinhardt.nyu.edu/marl/research/sonic_spaces

jpf211@nyu.edu

NYU | STEINHARDT

MARL