

Chapter 10

WFST Decoder

김지환

서강대학교 컴퓨터공학과

Table of contents

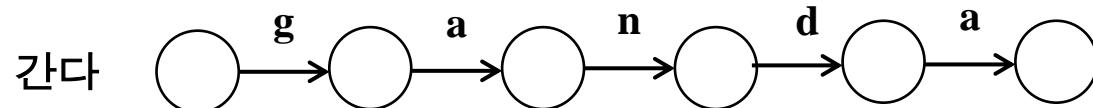
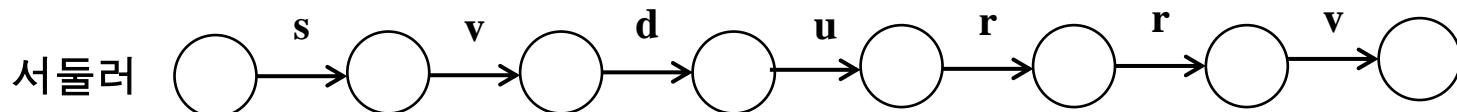
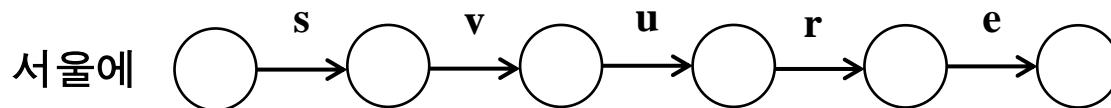
10.1 Lexical Tree 기반 Search

10.2 WFST

10.1 Lexical Tree 기반 Search

■ 예제

- 현재 어휘 사전은 아래 3개의 단어로 구성되어 있음
 - 서울에
 - 서둘러
 - 간다



10.1 Lexical Tree 기반 Search

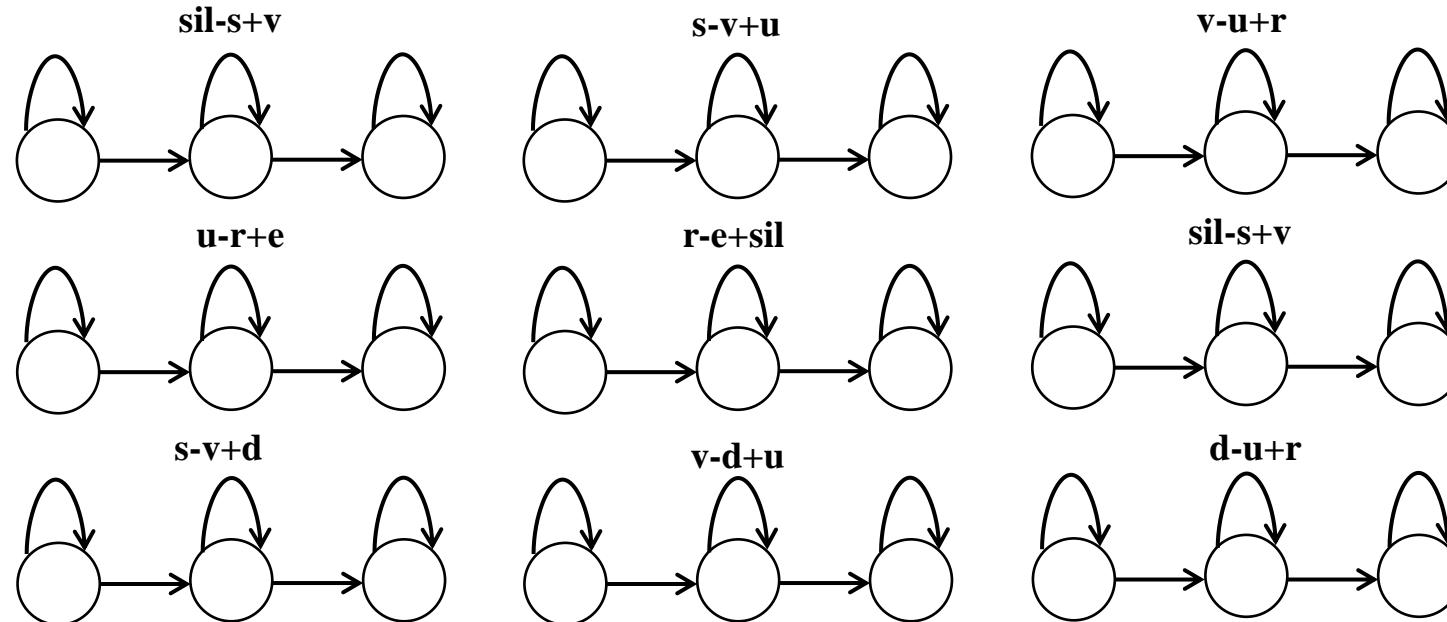
■ 예제

- Tri-gram 언어모델에서 만들 수 있는 문장은 아래와 같음
(3개의 단어로 길이가 최대 3단어인 문장은 총 39가지를 생성할 수 있음)
 - 서울에 서둘러 간다
 - 서둘러 서울에 간다
 - 간다 서울에 서둘러
 - 서울에 간다 서울에
 - 서울에 간다
 - 간다 서울에
 - 서둘러 간다
 - 서울에 서둘러
 - 서둘러 서울에
 - 서울에
 - 서둘러
 - 간다
 - ...

10.1 Lexical Tree 기반 Search

■ 예제

- 3개의 단어에 대한 HMM 구조는 아래와 같음
 - Tri-state에 Tri-phone을 가정함
 - * Silence phone은 sil로 표현함

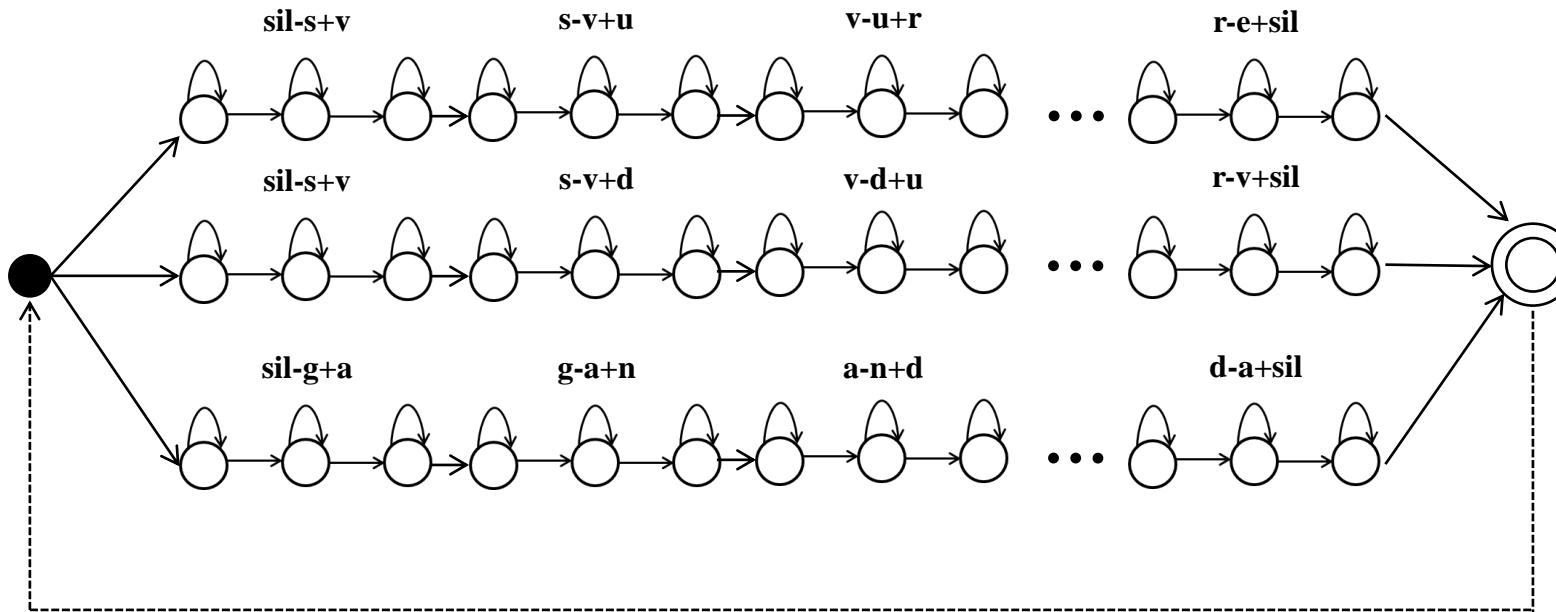


...

10.1 Lexical Tree 기반 Search

■ 예제

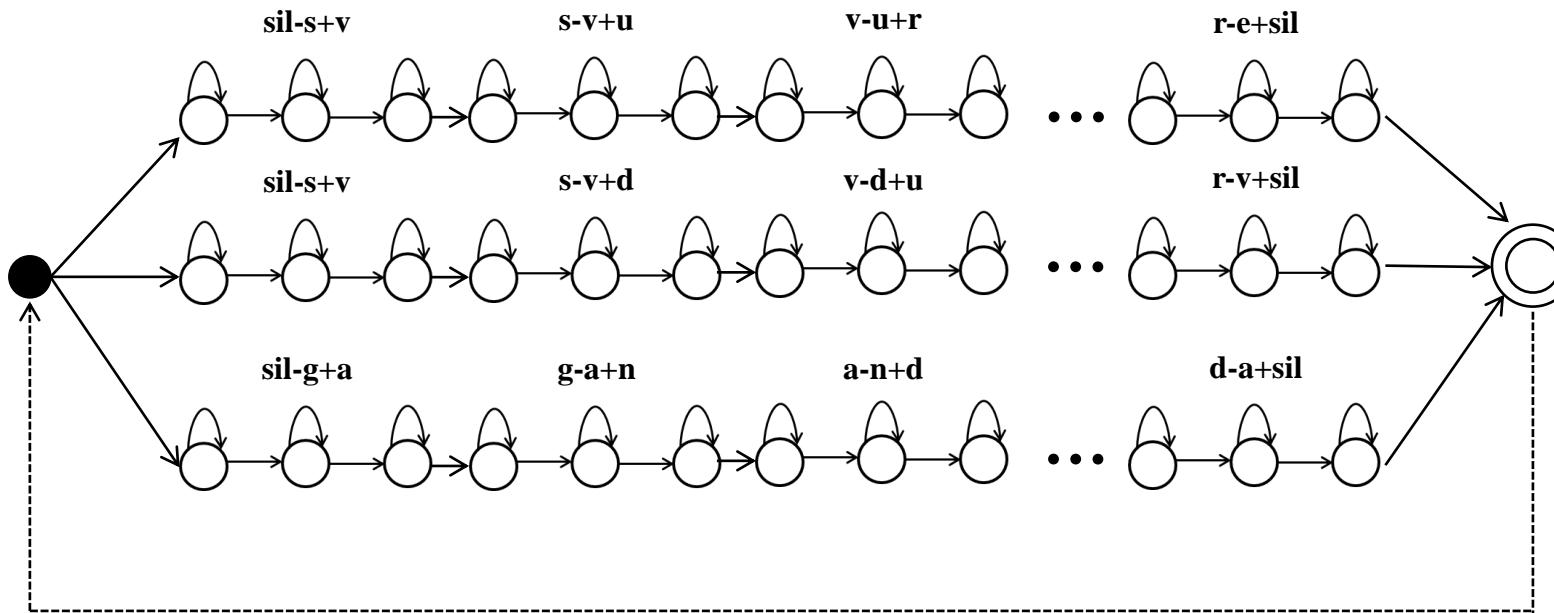
- 생성될 수 있는 search network는 아래와 같음



10.1 Lexical Tree 기반 Search

■ 예제

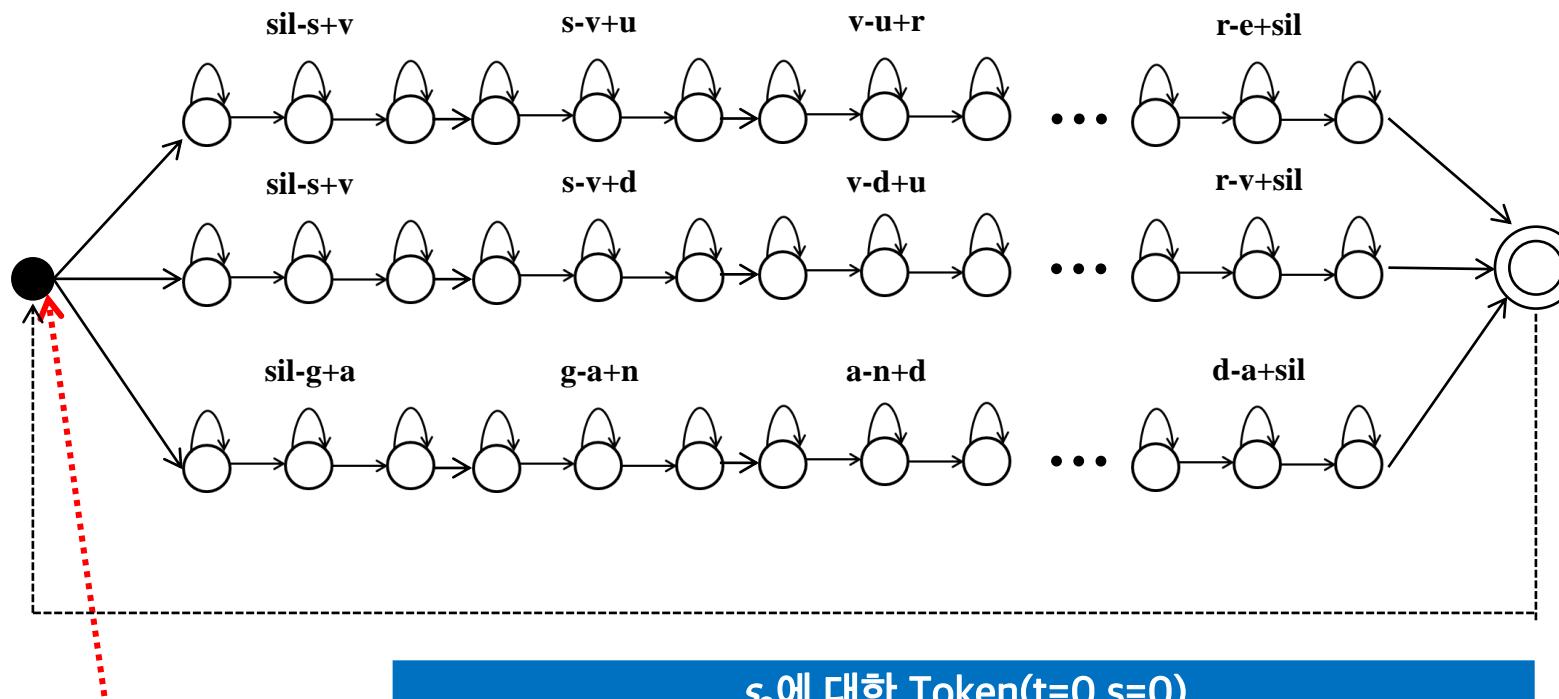
- ‘서울에 서둘러 간다’라고 발성했다고 가정



10.1 Lexical Tree 기반 Search

■ 예제

- Time $t = 0$



Start state s_0 에서 시작

s_0 에 대한 Token($t=0, s=0$)

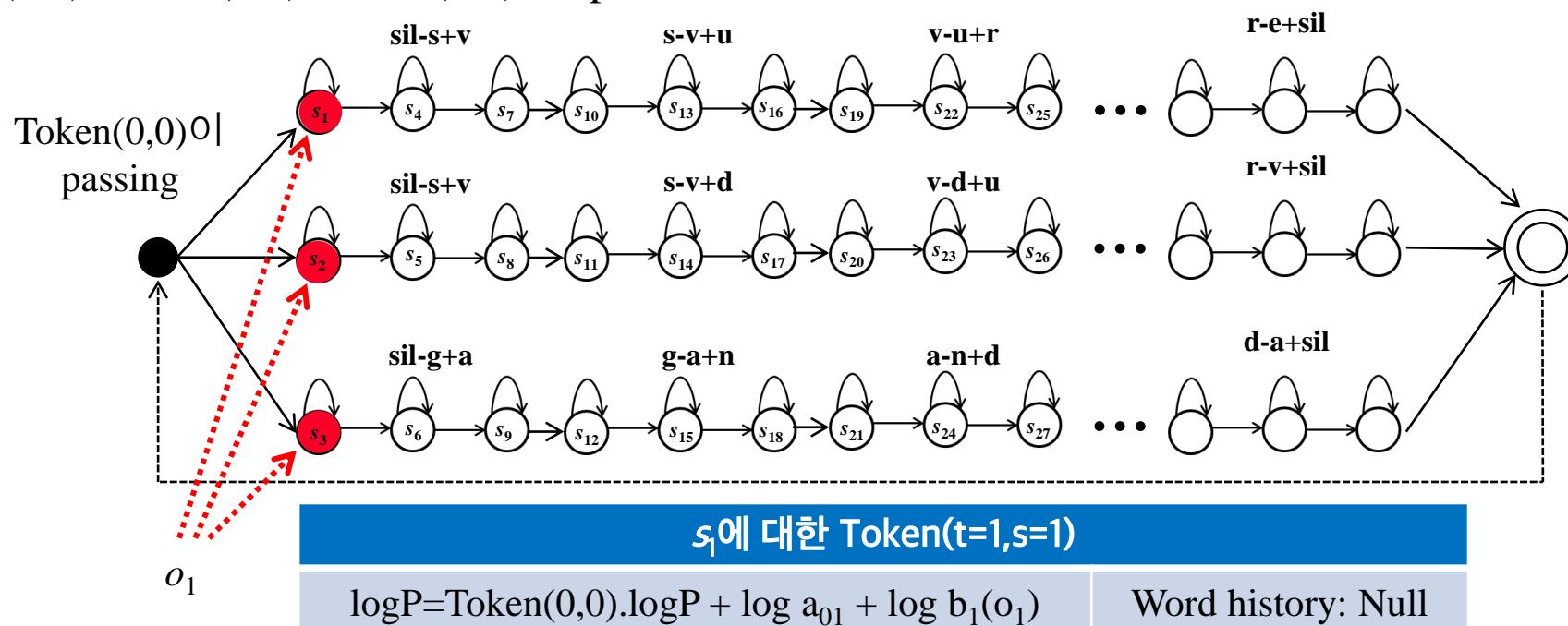
$\log P = 0$

Word history: Null

10.1 Lexical Tree 기반 Search

■ 예제

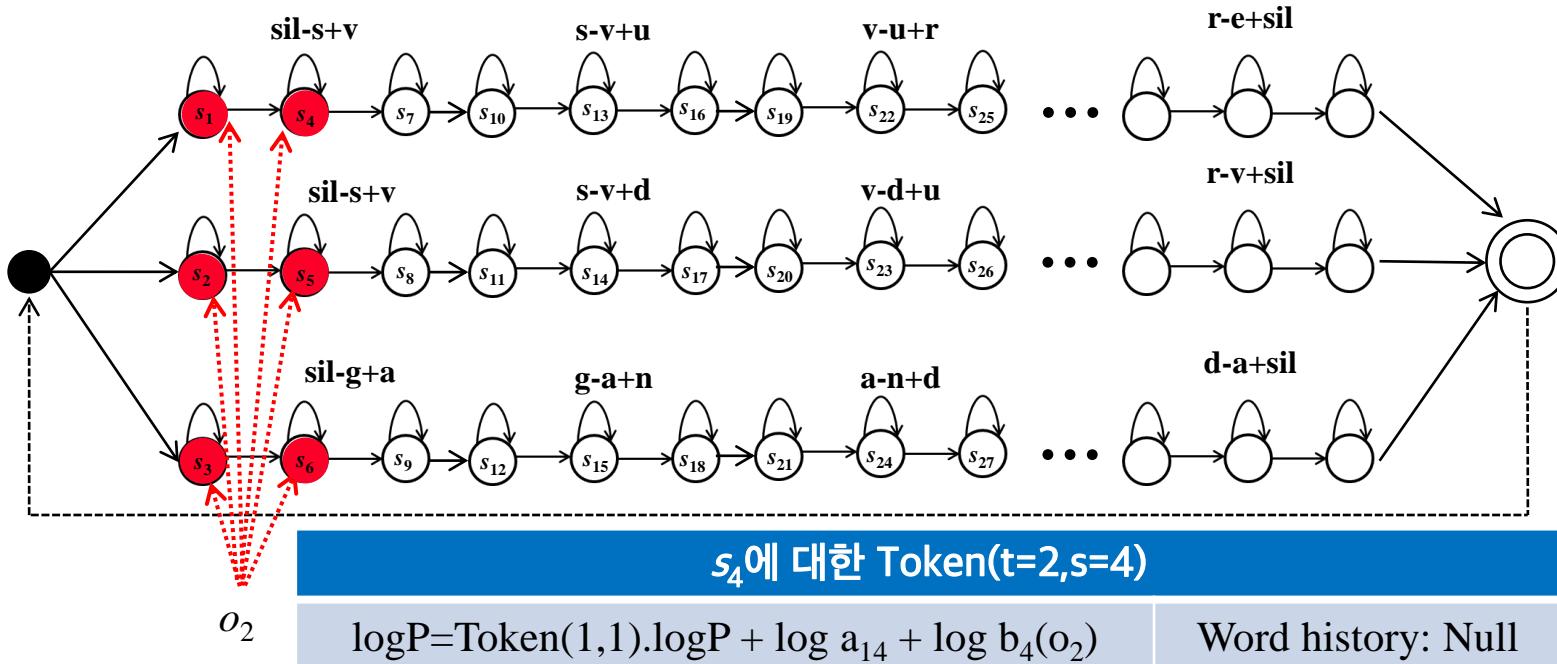
- Time t = 1 (특징벡터 o1)
 - s1, s2, s3가 activation 됨
 - s1, s2, s3에 Token(0,0)가 copy 됨
 - Token(1,1), Token(1,2), Token(1,3) 0| update 됨



10.1 Lexical Tree 기반 Search

■ 예제

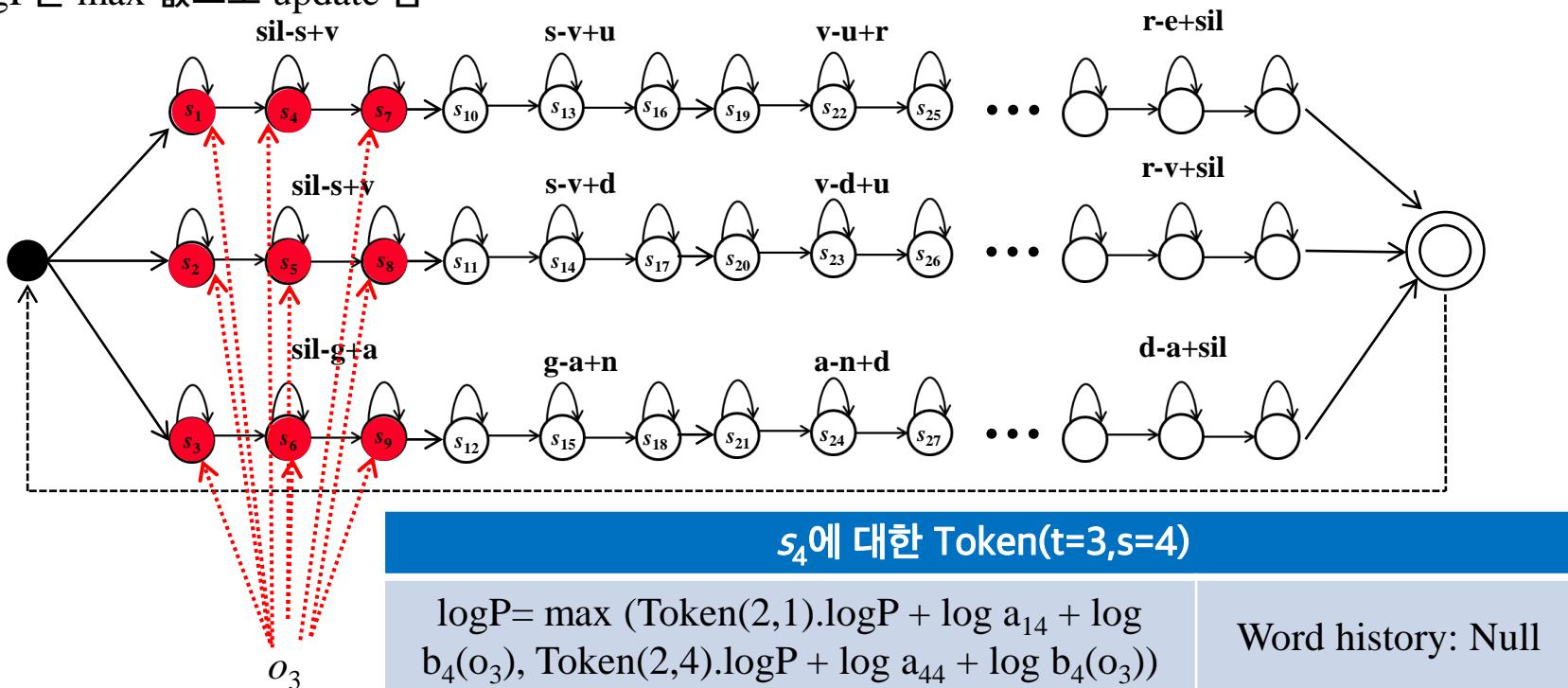
- Time t = 2 (특징벡터 o2)
 - s1, s2, s3, s4, s5, s6 가 activation 됨
 - s1, s4에는 Token(1,1)0| copy, s2, s5에는 Token(1,2)0| copy, s3, s6에는 Token(1,3)0| copy 됨
 - Token(2,1), Token(2,2), Token(2,3), Token(2,4), Token(2,5), Token(2,6)0| update 됨



10.1 Lexical Tree 기반 Search

■ 예제

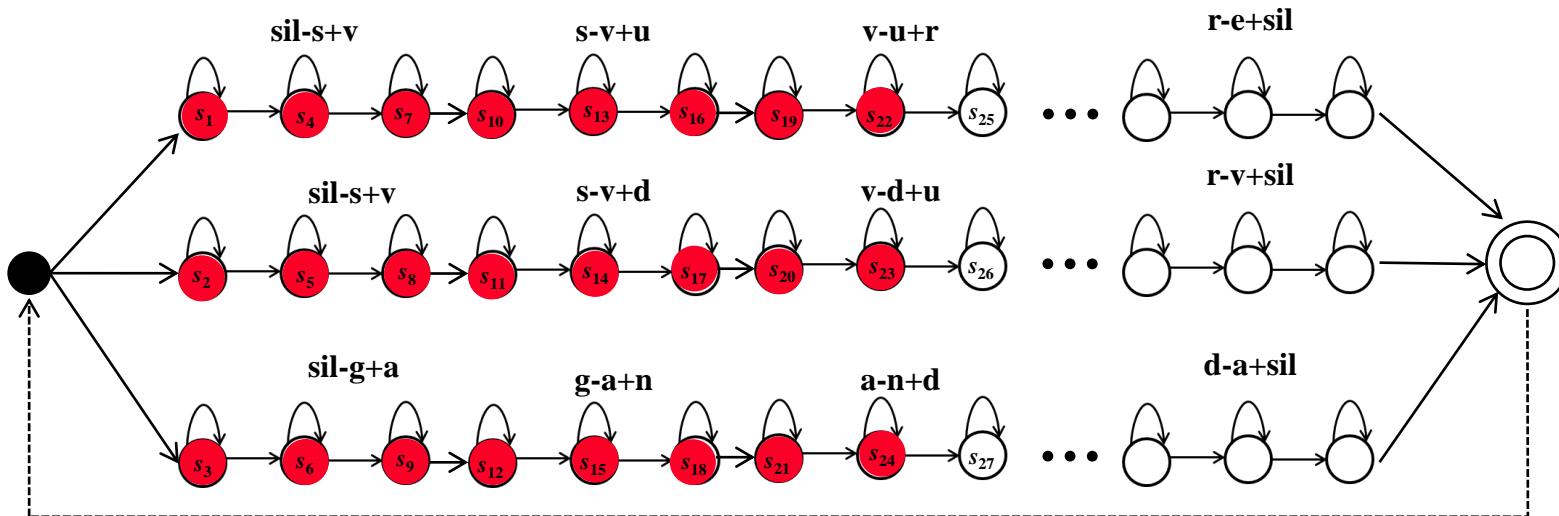
- Time t = 3 (특징벡터 o3)
 - s1~s9 이 activation 됨
 - S4의 경우, 경로가 2가지가 발생 (s1, s1, s4 / s1, s4, s4)
 - * 이 때, logP는 max 값으로 update 함



10.1 Lexical Tree 기반 Search

■ 예제

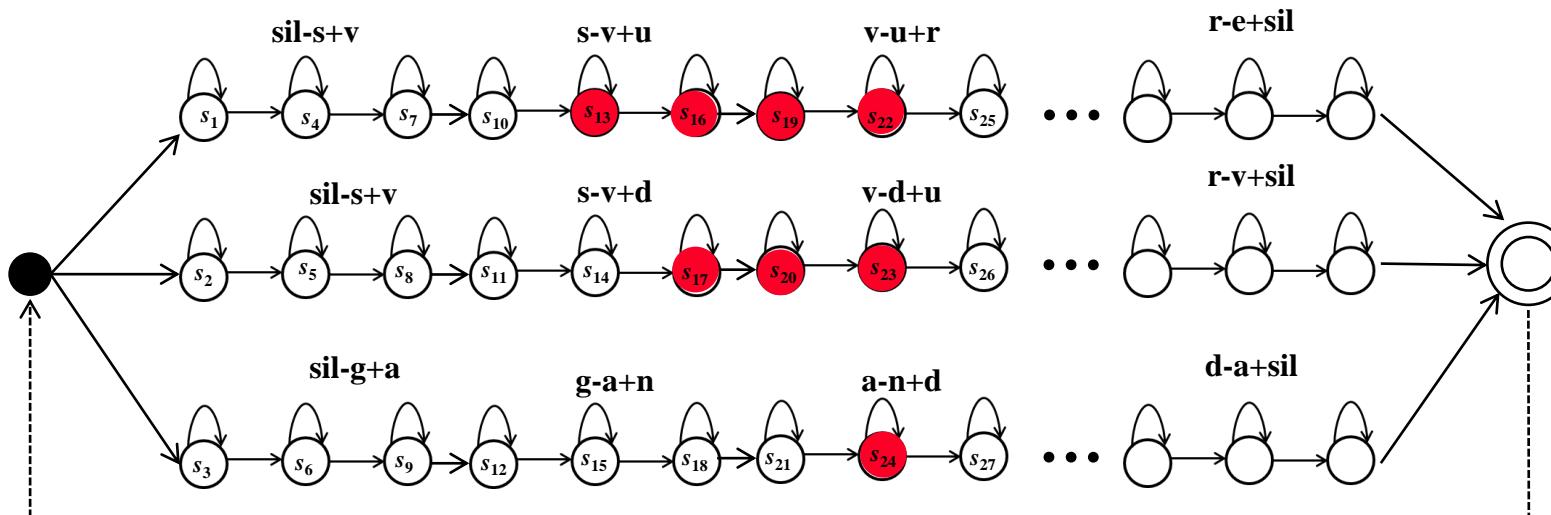
- 특징벡터가 입력될 때마다, activation state가 지속적으로 증가
 - Activation state가 많아질수록, 저장되는 token의 개수 증가
 - Search space가 증가하여, search 속도가 저하됨



10.1 Lexical Tree 기반 Search

■ 예제

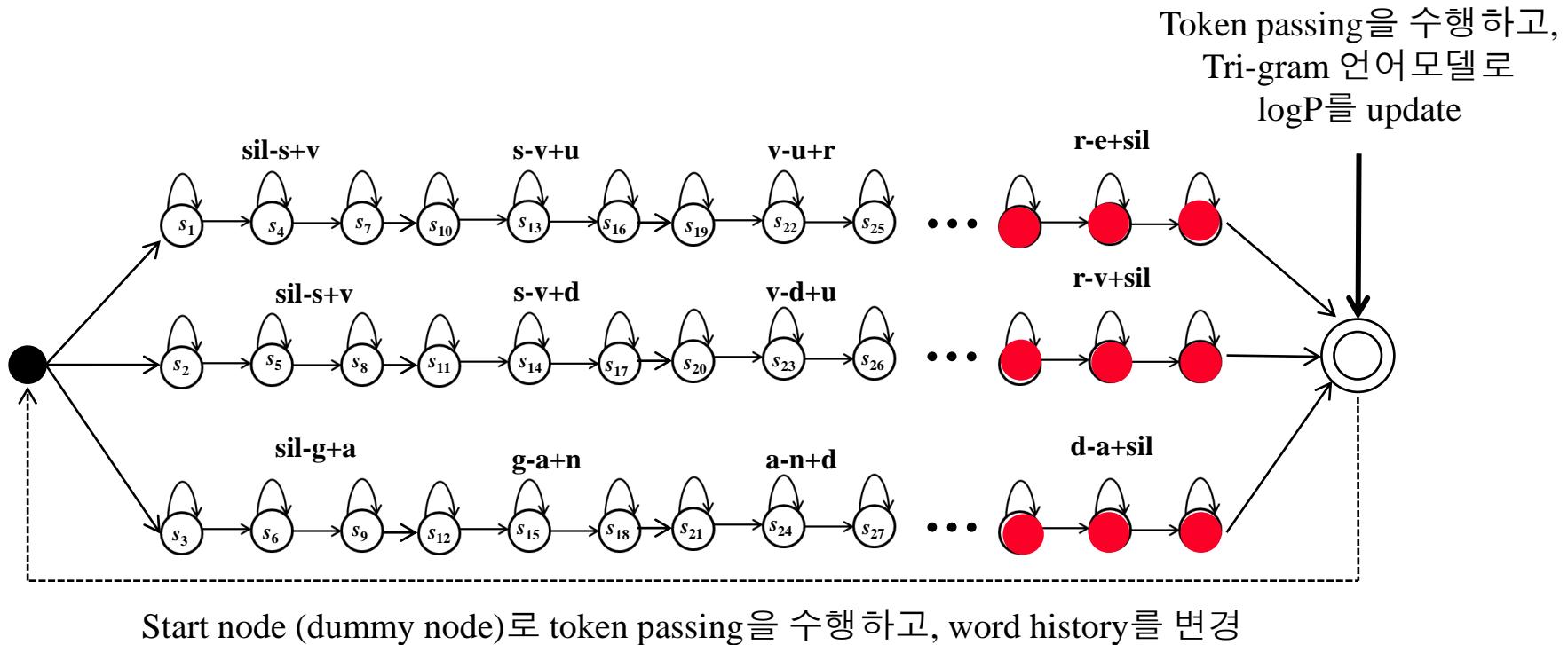
- Beam pruning을 수행하여, activation state의 개수를 줄임
 - Search space가 감소하여, search 속도를 향상시킴
 - 값이 가장 큰 best logP를 기준으로, 일정 범위 내의 logP를 갖는 state들만 남김
 - * 또는 상위 몇 개의 state만 남김



10.1 Lexical Tree 기반 Search

■ 예제

- 각 단어의 final node가 activation 된 경우에, dummy final node로 token passing을 수행하고, tri-gram 언어모델로 logP를 update
- Start node(dummy node)로 token passing을 수행하고, word history를 변경



10.2.1 Weighted Finite State Transducer(WFST) 개요

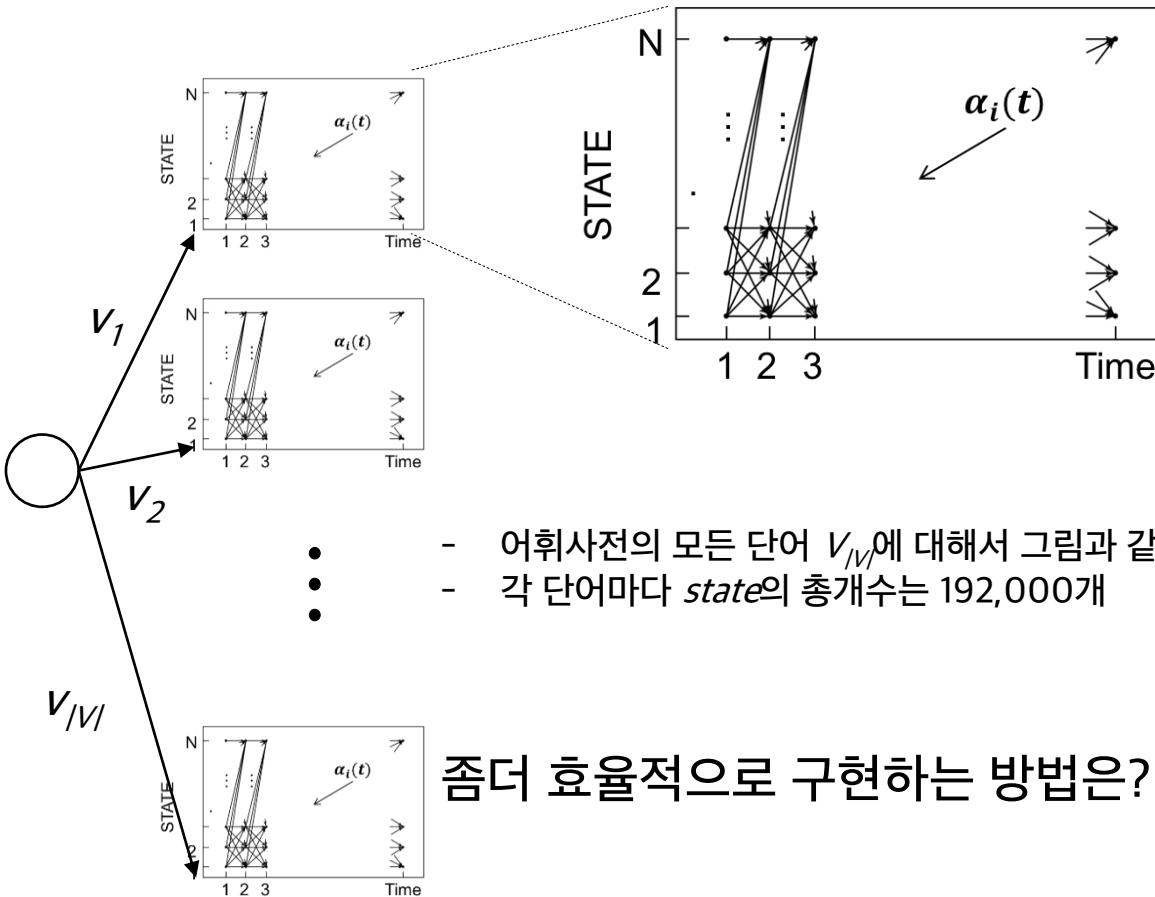
■ 고립단어 음성인식

- 가정

- 발음사전에 정의된 phone의 개수: 40
- State clustering 및 unseen tri-phone이 없다고 가정
 - * Context-dependent phone: Tri-phone ($40^3 = 64,000$)
 - * HMM의 총 state 개수: $40^3 \times 3 = 192,000$ 개
 - 하나의 tri-phone을 tri-state HMM 모델로 표현
- 입력의 길이: T
- 어휘 : $V = \{v_1, v_2, \dots, v_{|V|}\}$

10.2.1 WFST 개요

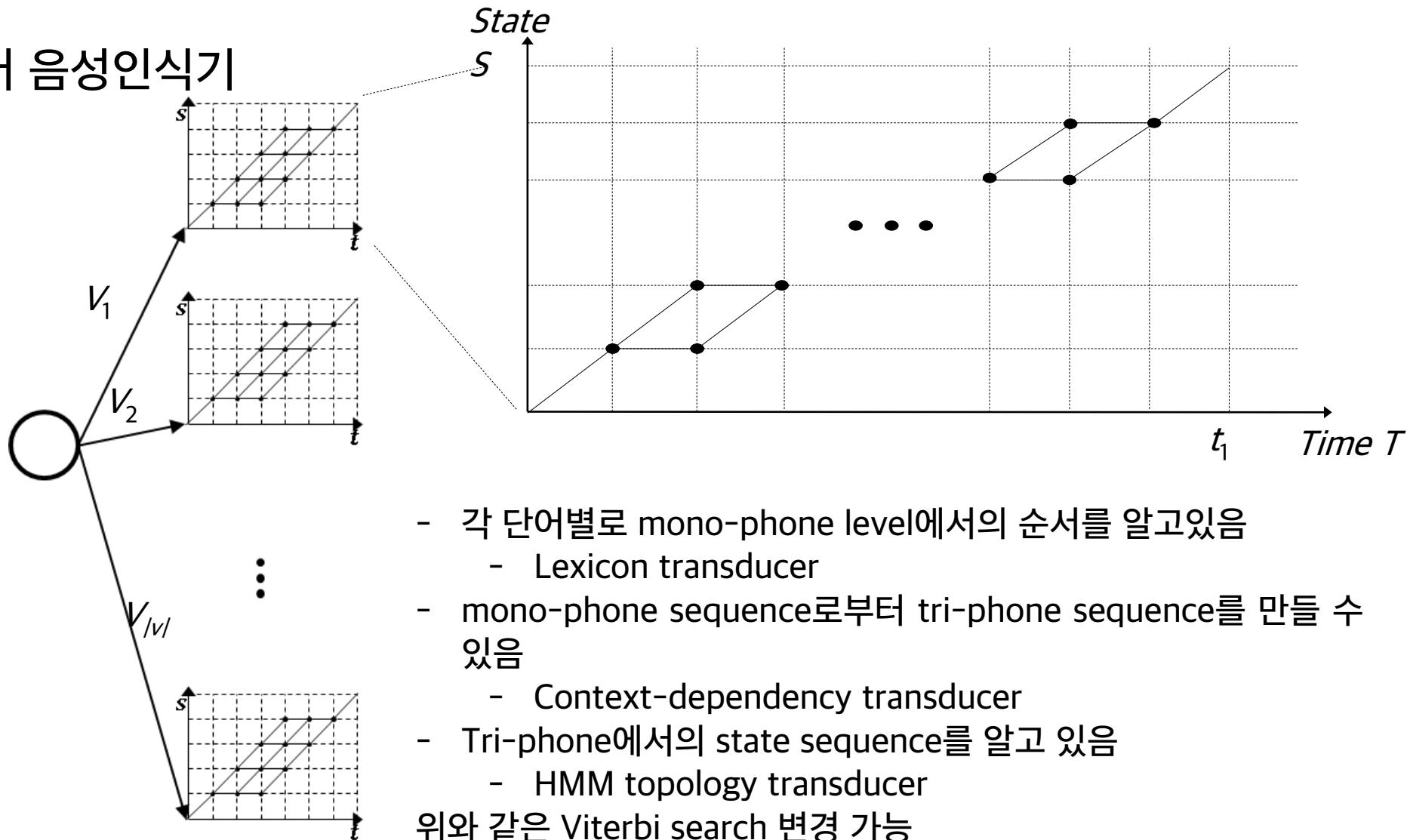
■ 고립단어 음성인식



10.2.1 WFST 개요

■ 고립단어 음성인식기

- 예제



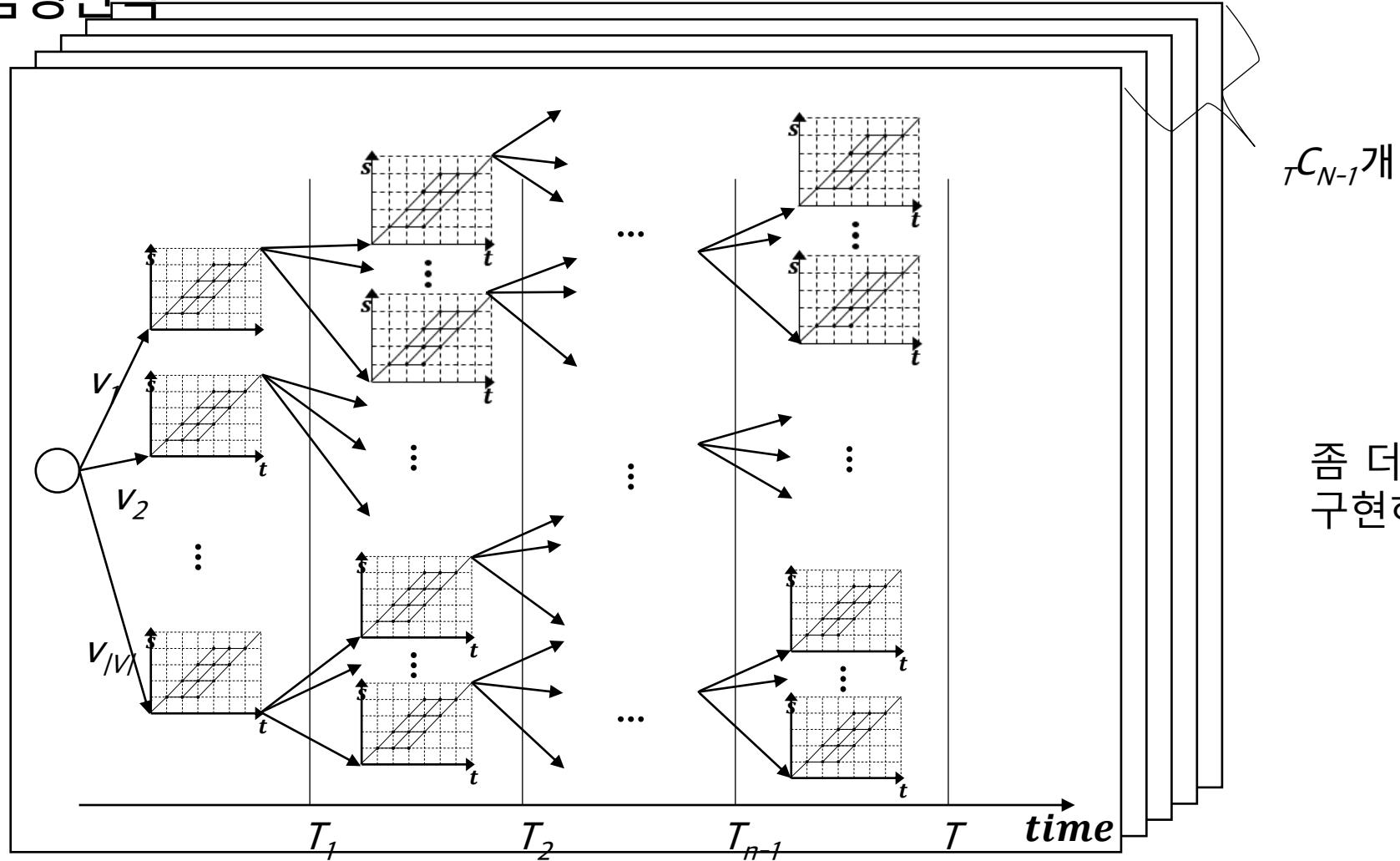
10.2.1 WFST 개요

■ 연속 음성인식의 어려운 점

- 문장을 이루는 단어의 개수를 모름
- 각 단어의 경계를 모름
- 가정
 - 한 문장에서 단어는 N개로 가정
 - i 번째 단어가 끝나는 지점은 T_i
 - * 각 단어의 경계를 찾는 문제
 - T 개의 frame에서 $N - 1$ 개의 경계를 찾는 문제임
 - 총 가지 수는 $_T C_{N-1}$ 개임

10.2.1 WFST 개요

■ 연속 음성인식

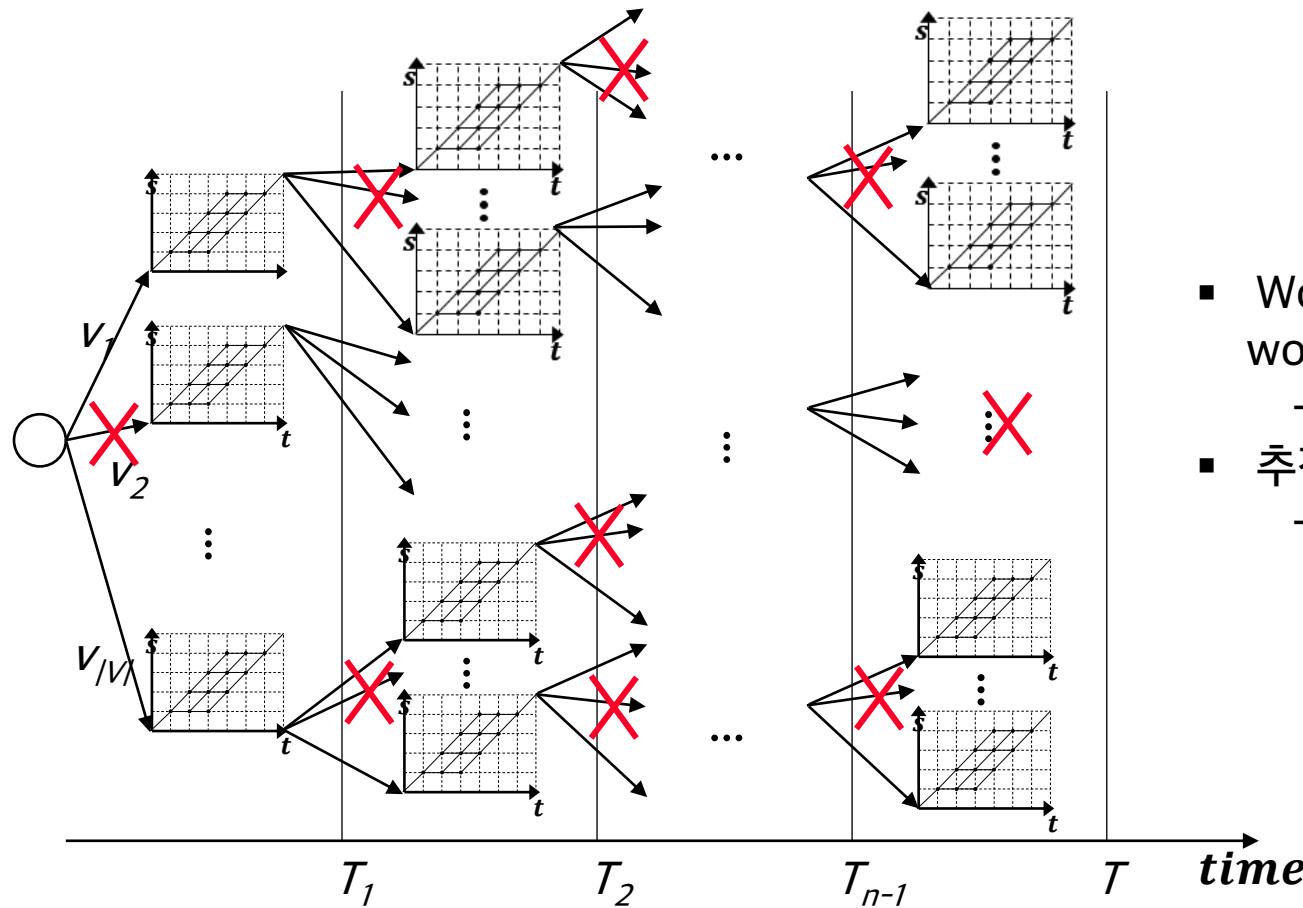


음성인식

10.2.1 WFST 개요

■ 연속 음성인식기

• 예제



- Word history에 따라 다음 word를 추정가능
 - Grammar transducer
- 추정되는 평균 단어의 수 -perplexity

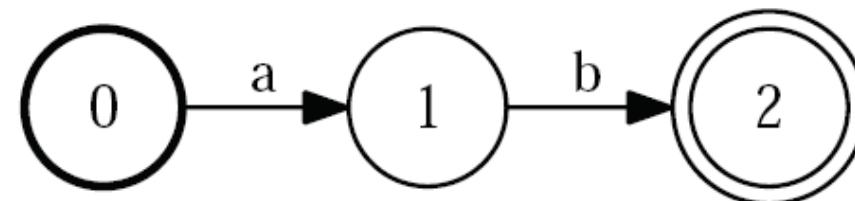
10.2.1 WFST 개요

- Lexical tree기반의 연속 음성인식
 - HMM, Context dependency, Lexicon, Grammar가 별도로 구현되어 있음.
 - Runtime에서 T가 증가하며 Network을 차례로 만들어 나가는 방식
 - Search space가 너무 커지는 것을 막기 위하여 beam pruning을 적용
 - 단점 : 속도가 느리다
- WFST(Weighted Finite State Transducer)가 추구 하는 방향
 - HMM, Context dependency, Lexicon, Grammar를 결합한 효율적인 Network를 미리 만들어 놓는다
 - Composition, Determinization, Minimization이 필요
 - 단점
 - 메모리를 많이 요구한다
 - 어휘 또는 학습 텍스트의 약간의 변화만 있어도 Network를 다시 구성해야함
 - 대용량 언어모델을 사용하는 경우 Network가 커짐

10.2.1 WFST 개요

■ Finite State Acceptor (FSA)

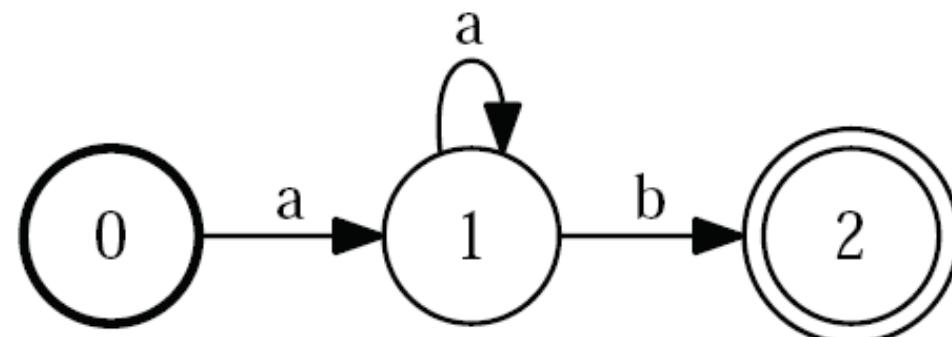
- FSA *accepts* a set of strings
 - a string is a sequence of symbols
- View FSA as a representation of a possibly infinite set of strings
- Numbers in circles are state labels
- Labels on arcs are the symbols
- Start state(s) bold; Final/Accepting states have extra circle
- 예제
 - This FSA accepts just the string ab, i.e. the set {ab}



10.2.1 WFST 개요

■ Less trivial FSA

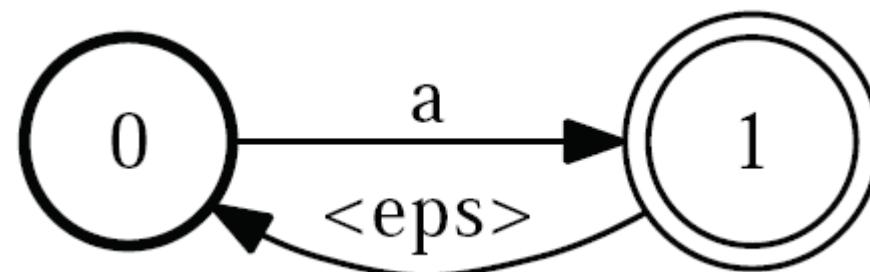
- String is *accepted* (included in the set) if
 - There is a path with that sequence of symbols on it.
 - That path is successful
 - * Starts at an initial state
 - * Ends at a final state
- 예제
 - This example represents the infinite set $\{ab, aab, aaab, \dots\}$



10.2.1 WFST 개요

■ Epsilon symbol (ϵ)

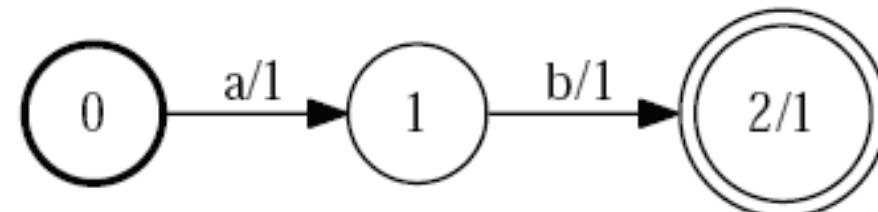
- The symbol ϵ means *no symbol is there*
- 예제
 - This example represents the set of strings $\{a, aa, aaa, \dots\}$
 - If ϵ were treated as a normal symbol, this would be $\{a, a \epsilon a, a \epsilon a \epsilon a, \dots\}$
 - ϵ is sometimes written as `<eps>`



10.2.1 WFST 개요

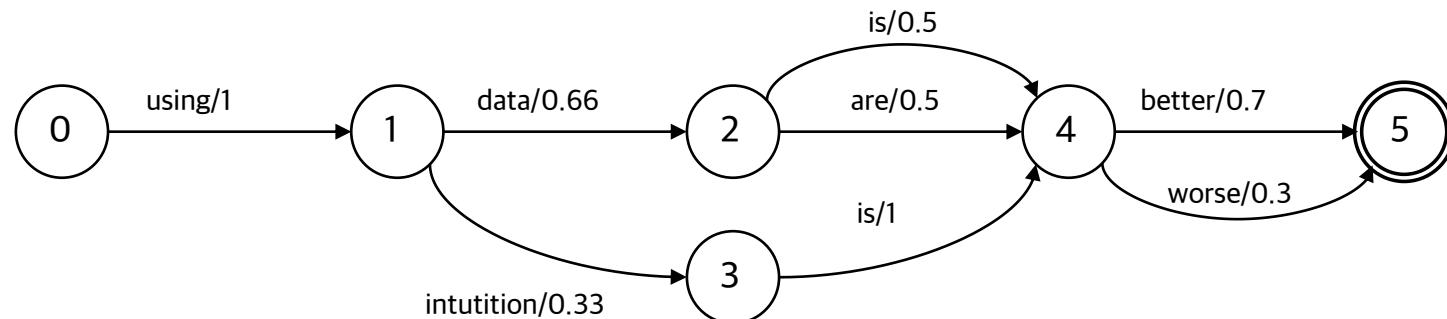
■ Weighted finite state acceptor

- Like a normal FSA but with weights on the arcs and final-states
 - Weight comes after ‘ / ’
 - For final-state, ‘ 2/1 ’ means final-cost 1 on state 2
- View WFSA as a function from a string to a cost
- In this view, unweighted FSA is:
 - $f : \text{string} \rightarrow \{0, \infty\}$
- If multiple paths have the same string, take the one with the lowest weight
- 예제
 - Maps ‘ab’ to $(3 = 1 + 1 + 1)$, all else to 1.

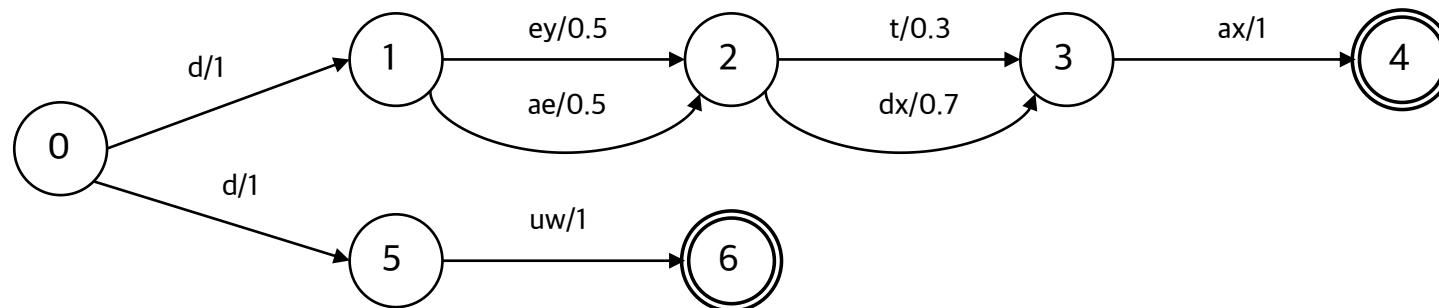


10.2.1 WFST 개요

■ 예제



(a) 언어모델

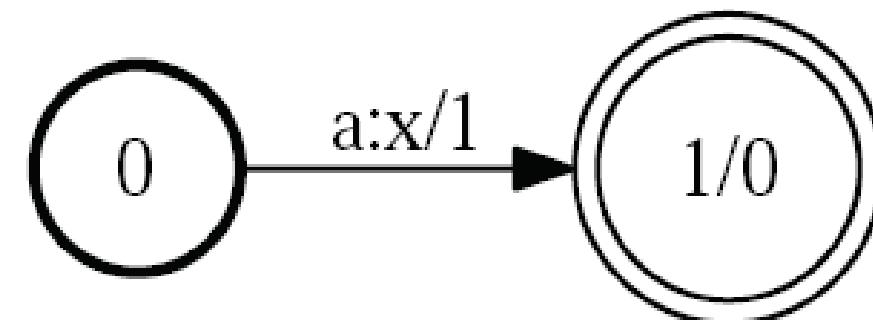


(b) 단어 'data'와 'dew'에 대한 발음사전

10.2.1 WFST 개요

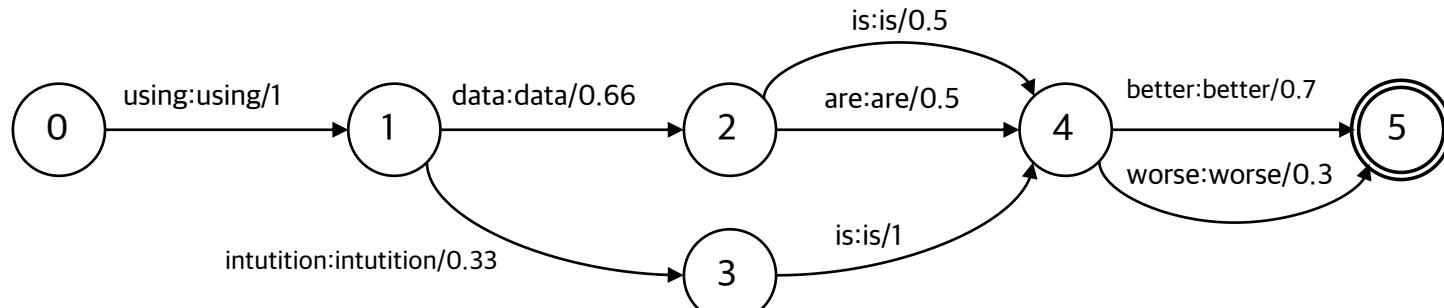
■ Weighted finite state transducer

- Like a WFSA except with two labels on each arc
- View it as a function from a pair of strings to a weight
- Symbols on the left and right are termed input and output symbols
- 예제
 - Input symbol: a
 - Output symbol: x
 - Weight ($a \rightarrow x$): 1

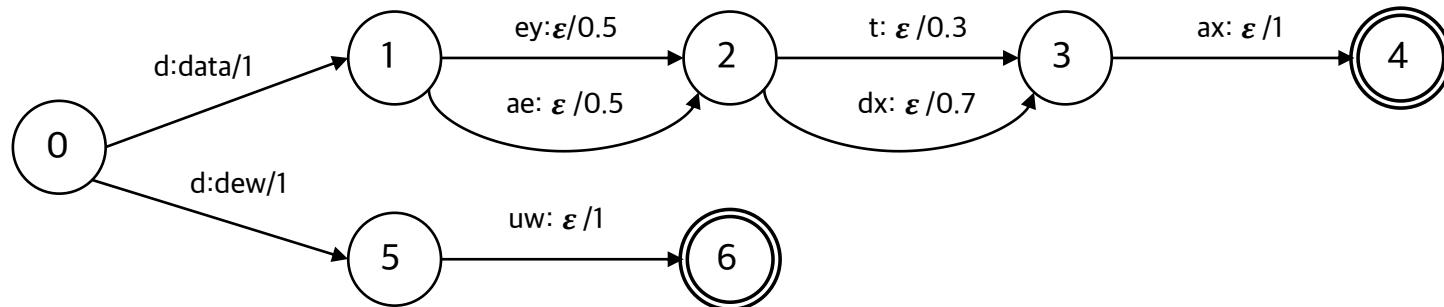


10.2.1 WFST 개요

■ 예제



(a) 언어모델의 WFST



(b) 단어 ‘data’와 ‘dew’에 대한 발음사전의 WFST

10.2.2 WFST 기반 디코딩 구성 요소

- Exploit several knowledge sources (lexicon, grammar, phonetics) to find most likely spoken word sequence [Mohri, 2008]

$$HCLG = H \circ C \circ L \circ G$$

- Create H, C, L, G separately and compose them together
 - Grammar Transducer : G
 - Probabilistic grammar or language model acceptor (word)
 - Lexicon Transducer : L
 - Lexicon (phones to words)
 - Context dependency Transducer : C
 - Context-dependent relabeling (context-dependent phone to context-independent phone)
 - HMM Transducer : H
 - HMM structure (PDF labels to context-dependent phones)

10.2.2 WFST 기반 디코딩 구성 요소

- 본 강의에서 예제는 Kaldi toolkit에서 구축되는 WFST 기반 음성인식 디코딩 방법에 따라 진행
 - [Mohri, 2008]에서 제안하는 WFST 기반 음성인식 디코딩

$$HCLG = rds(\min(\det(H \circ \det(C \circ \det(L \circ G)))))$$

- rds
 - * Remove disambiguation symbols
- min
 - * Minimization
- det
 - * Determinization

10.2.2 WFST 기반 디코딩 구성 요소

- 본 강의에서 예제는 Kaldi toolkit에서 구축되는 WFST 기반 음성인식 디코딩 방법에 따라 진행

- Kaldi toolkit에서 구축되는 WFST 기반 음성인식 디코딩

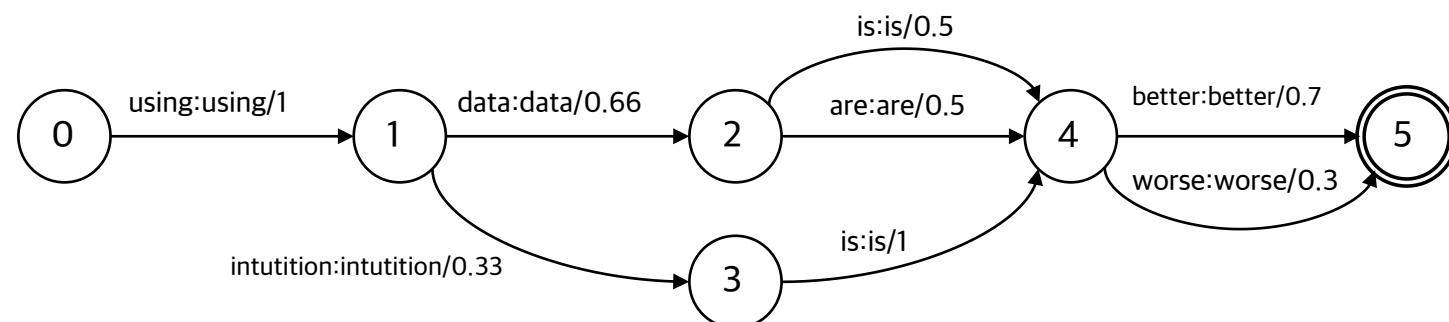
$$HCLG = \text{asl}(\min(rds(\det(H_a \circ \min(\det(C \circ \min(\det(L \circ G))))))))$$

- asl
 - Add self loops
 - H_a
 - Self loop가 없는 WFST H

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- Grammar transducer 로써, 언어모델 arpa 파일로부터 생성됨
- G에 대한 model topology
 - Input과 Output
 - * Word sequence
 - Weight
 - * History dependent word probability

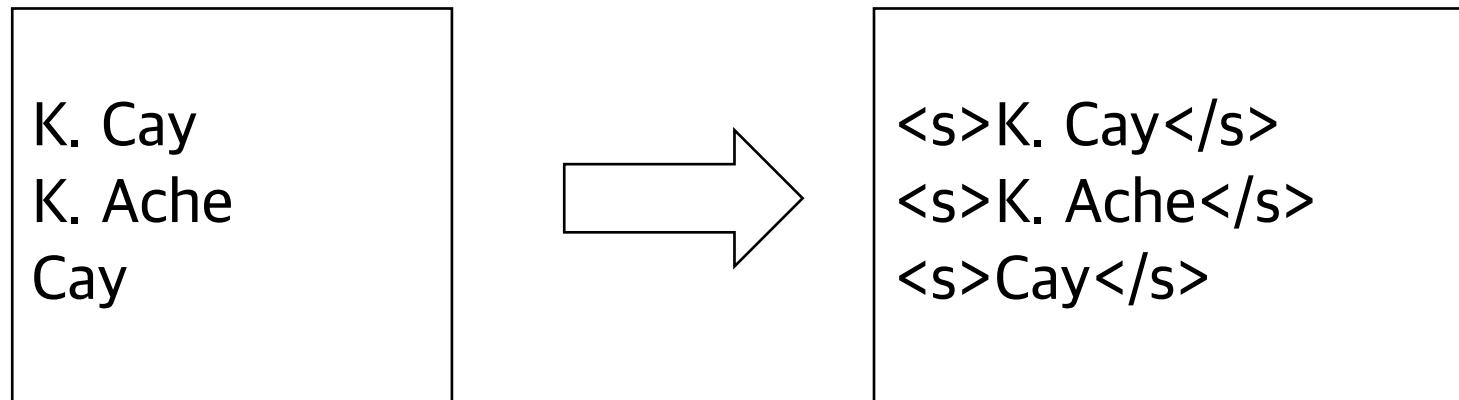


10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

- 아래와 같은 텍스트 코퍼스를 갖고 있다고 가정
 - * Start symbol (<s>) 및 End symbol (</s>) 도 함께 고려함



10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

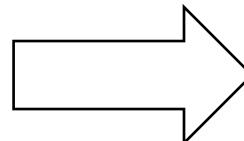
- 예제
 - 주어진 텍스트 코퍼스로부터 ARPA 파일 생성
 - * 예제에서는 uni-gram 및 bi-gram을 모두 고려함

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

```
<s>K. Cay</s>
<s>K. Ache</s>
<s>Cay</s>
```



```
₩data₩
ngram 1=5
ngram 2=6
₩1-grams:
-0.4259687 </s>
-99 <s> -0.30103
-0.90309 Ache -0.09691
-0.60206 Cay -0.2730013
-0.60206 K. -0.2730013
₩2-grams:
-0.60206 <s> Cay
-0.30103 <s> K.
-0.30103 Ache </s>
-0.1760913 Cay </s>
-0.4771213 K. Ache
-0.4771213 K. Cay
₩end₩
```

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

₩data₩
ngram 1=5
ngram 2=6

₩1-grams:
-0.4259687 </s>
-99 <s> -0.30103
-0.90309 Ache -0.09691
-0.60206 Cay -0.2730013
-0.60206 K. -0.2730013

₩2-grams:
-0.60206 <s> Cay
-0.30103 <s> K.
-0.30103 Ache </s>
-0.1760913 Cay </s>
-0.4771213 K. Ache
-0.4771213 K. Cay

₩end₩

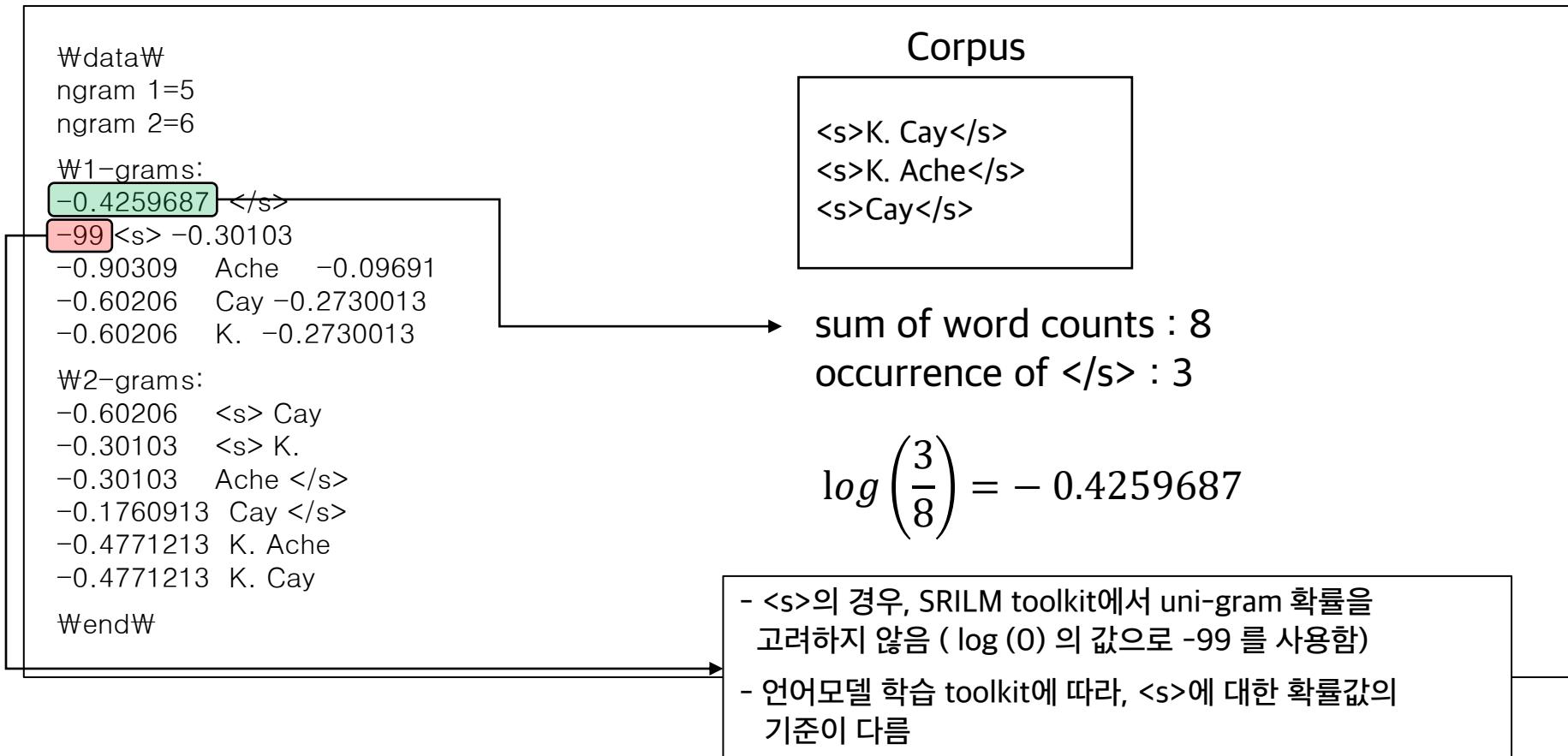
코퍼스로부터 생성된 각 N -gram의 tuple 개수

Column	Description
1	주어진 N -gram에 대한 conditional probability의 \log_{10} 에 대한 값
2	N -gram sequence
3	Backoff weight (Highest-order N-gram의 경우, backoff weight는 계산하지 않음)

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제



10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

₩data₩

ngram 1=5

ngram 2=6

₩1-grams:

-0.4259687 </s>

-99 <s> -0.30103

-0.90309 Ache -0.09691

-0.60206 Cay -0.2730013

-0.60206 K. -0.2730013

₩2-grams:

-0.60206 <=> Cay

-0.30103 <s> K.

-0.30103 Ache </s>

-0.1760913 Cay </s>

-0.4771213 K. Ache

-0.4771213 K. Cay

₩end₩

- SRILM toolkit에서는 bi-gram을 위한 log probability 계산을 위하여, Katz smoothing을 사용함

$$P_{bo}(w_i | w_{i-n+1} \dots w_{i-1}) = \begin{cases} d_{w_{i-n+1} \dots w_i} \frac{C(w_{i-n+1} \dots w_{i-1} w_i)}{C(w_{i-n+1} \dots w_{i-1})} & \text{if } C(w_{i-n+1} \dots w_i) > k \\ \alpha_{w_{i-n+1} \dots w_{i-1}} P_{bo}(w_i | w_{i-n+2} \dots w_{i-1}) & \text{otherwise} \end{cases}$$

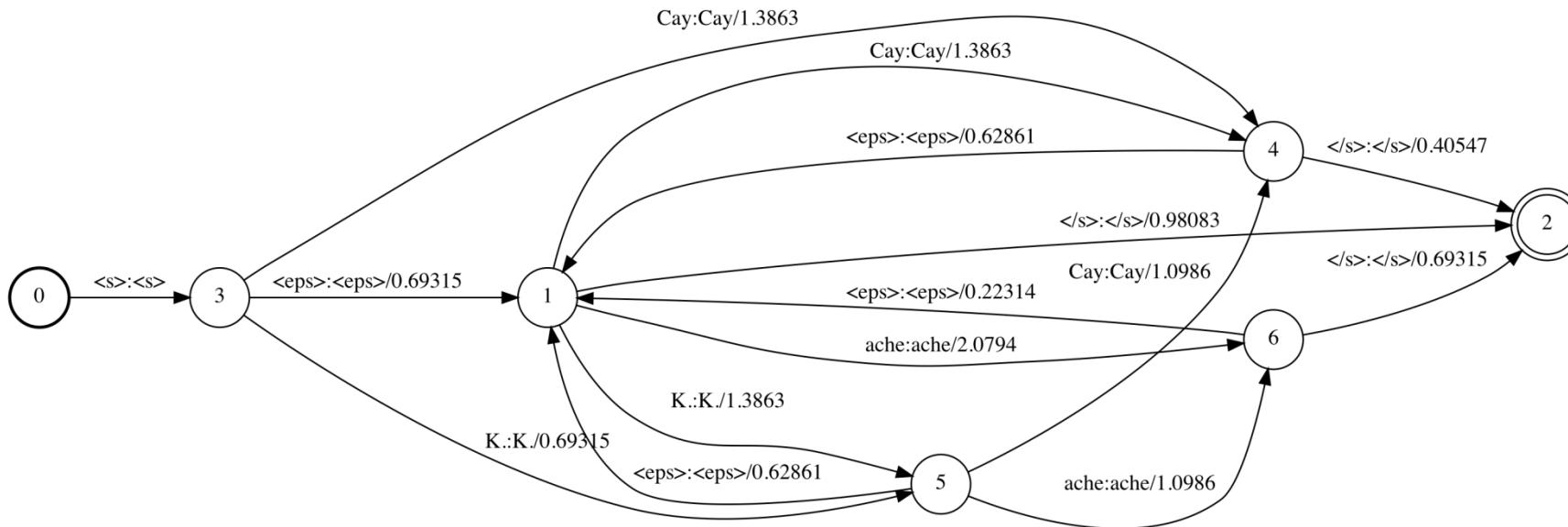
- Unseen bi-gram의 log probability 계산 방법
→ Unseen bi-gram $w_0 w_1$ 에 대해, 아래의 수식으로 계산
→ (Uni-gram w_0 의 backoff weight) + (Uni-gram w_1 의 log probability)

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

- ARPA 파일로부터 language model에 대한 WFST인 G를 생성



<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.1 Grammar transducer(G)

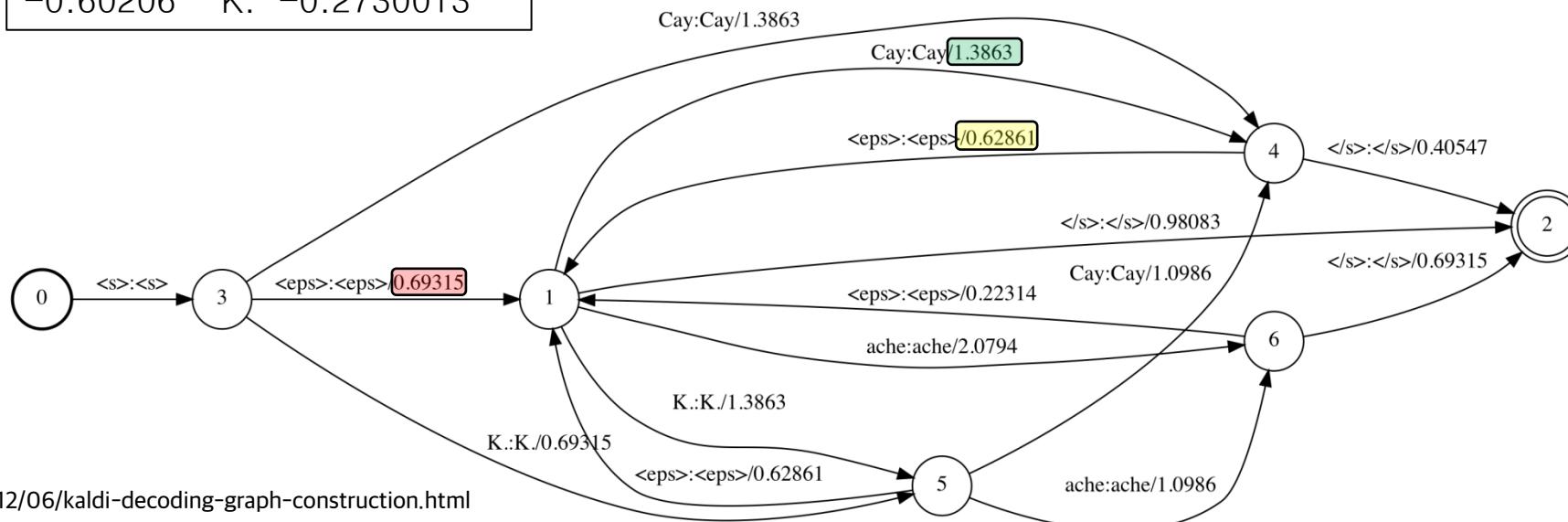
■ Grammar transducer G

- 예제

- ARPA 파일의 log probability 및 backoff weight에 대한 값이 G의 weight로 표현됨

W1-grams:
-0.4259687 </s>
-99 <s> -0.30103
-0.90309 Ache -0.09691
-0.60206 Cay -0.2730013
-0.60206 K. -0.2730013

$$0.69315 = -\ln(10^{-0.30103})$$
$$1.3863 = -\ln(10^{-0.60206})$$
$$0.62861 = -\ln(10^{-0.2730013})$$

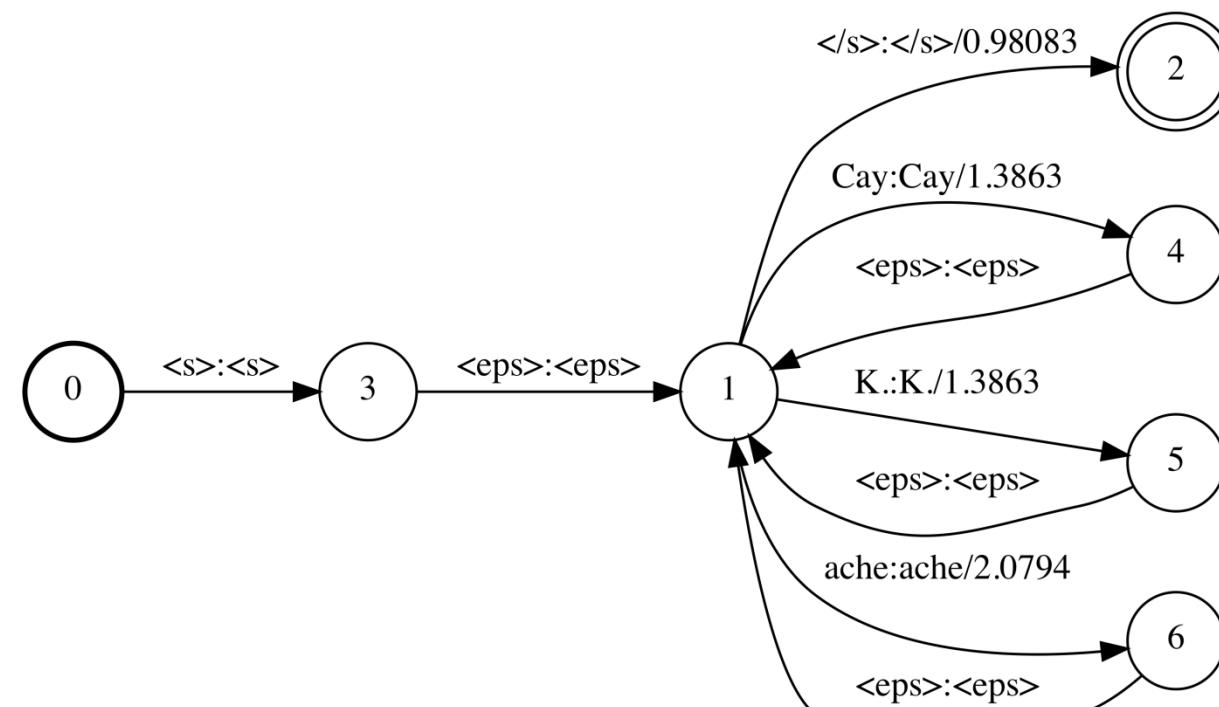


10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

- Uni-gram 언어모델을 이용한 WFST G 생성 방법
 - * Step 1: ARPA 파일로부터 raw WFST 생성



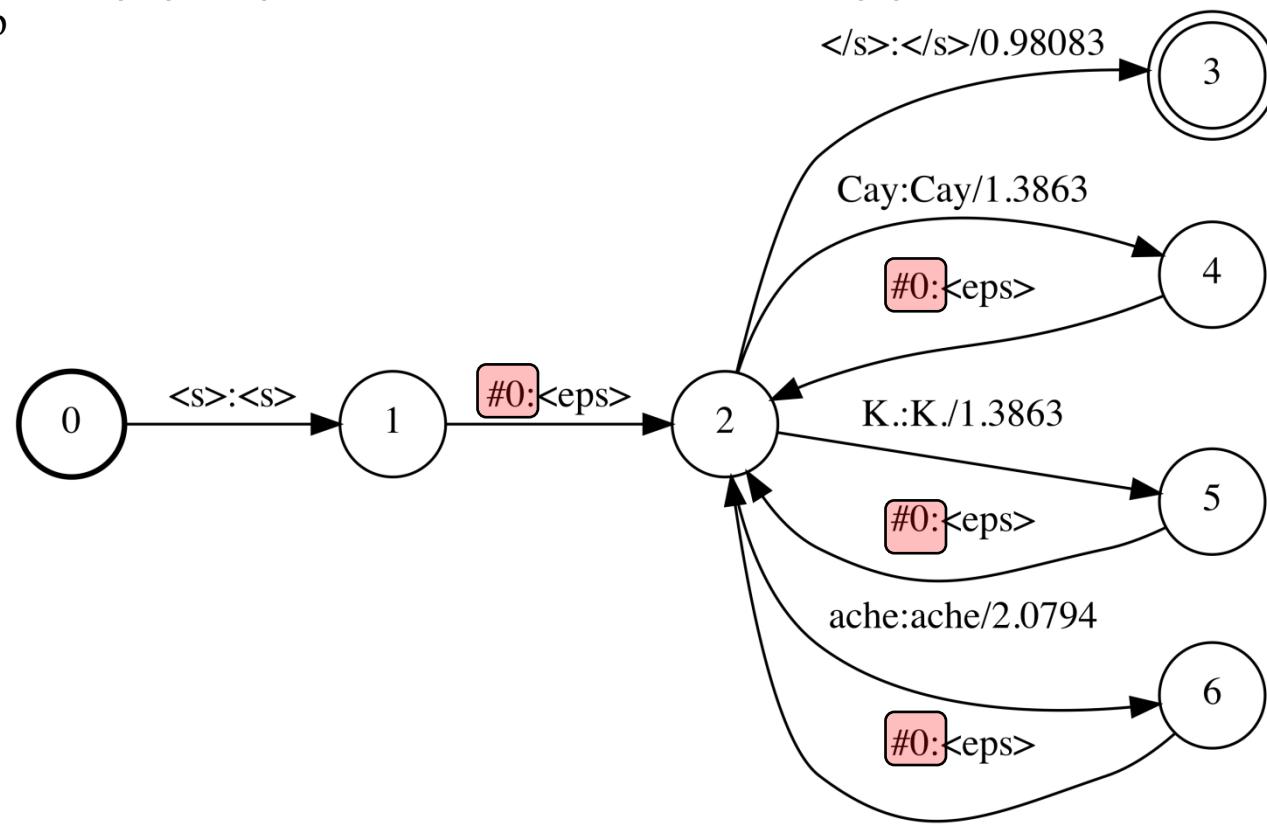
<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

- Uni-gram 언어모델을 이용한 WFST G 생성 방법
 - * Step 2: Input $0|\epsilon$

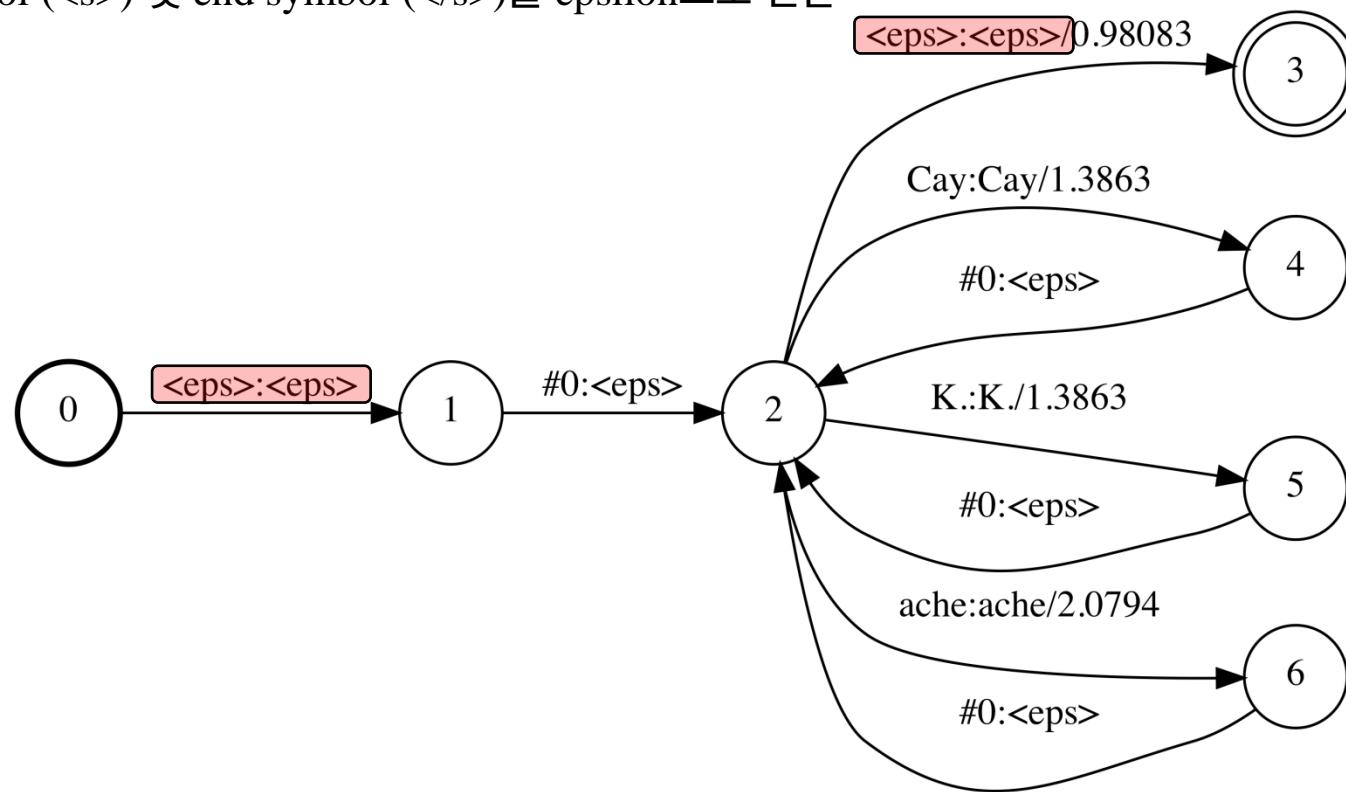


10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

• 예제

- Uni-gram 언어모델을 이용한 WFST G 생성 방법
 - * Step 3: Start symbol (<s>) 및 end symbol (</s>)을 epsilon으로 변환

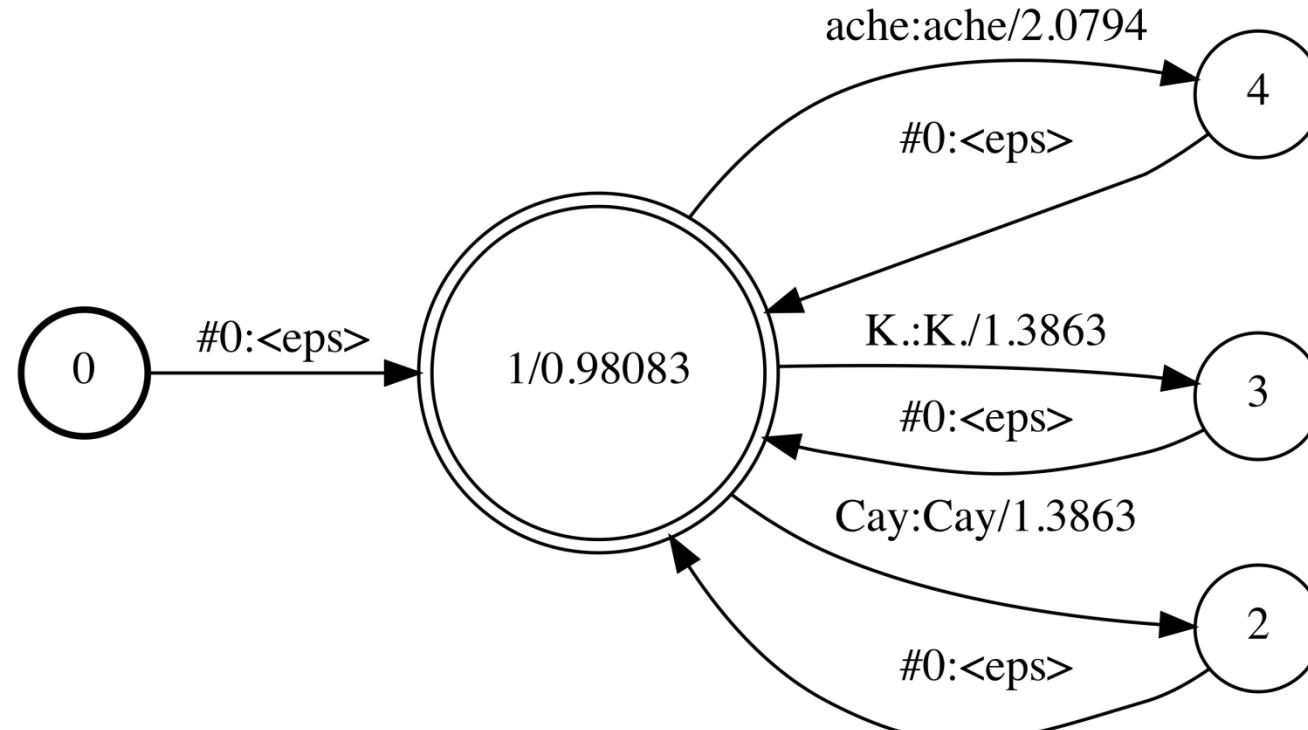


10.2.2.1 Grammar transducer(G)

■ Grammar transducer G

- 예제

- Uni-gram 언어모델을 이용한 WFST G 생성 방법
 - * Step 4: Input과 output의 symbol이 모두 epsilon인 edge를 제거



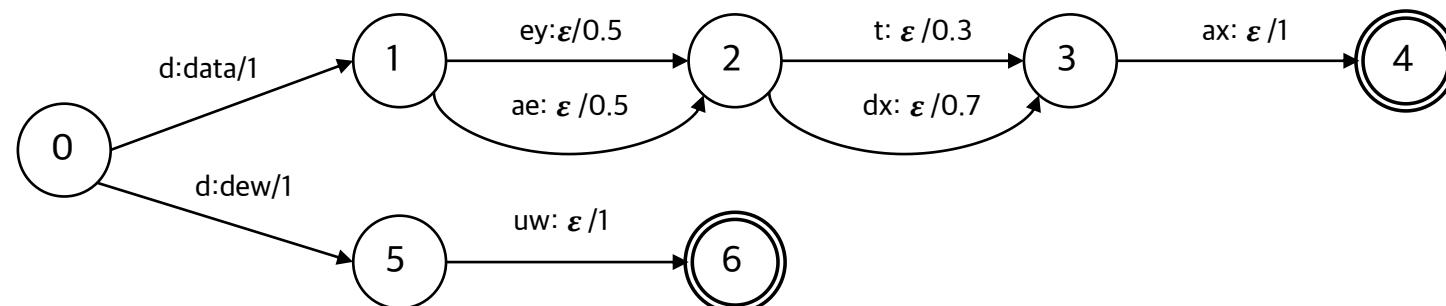
<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.2 Lexicon Transducer(L)

■ Pronunciation lexicon L

- 발음사전 (Pronunciation lexicon)으로 부터 WFST L을 생성함
- L에 대한 model topology

- Input
 - * Context-independent phone (phoneme) sequence
- Output
 - * Word sequence
- Weight
 - * Pronunciation probability



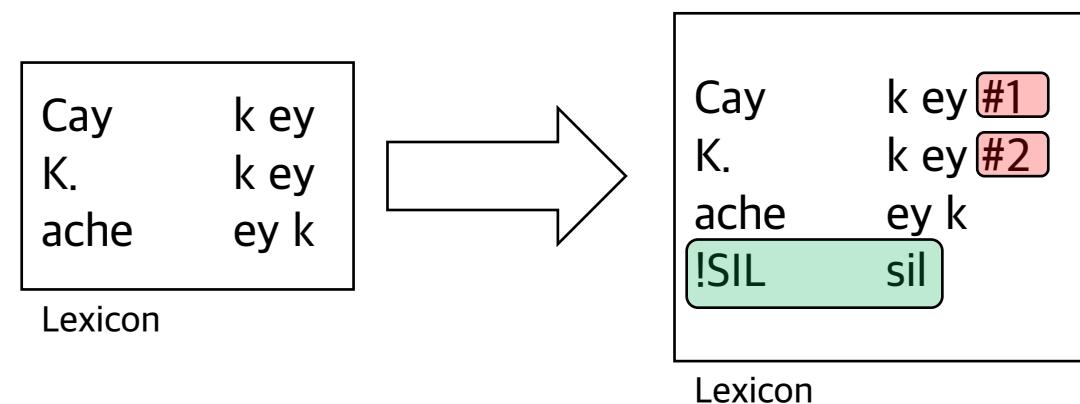
단어 ‘data’와 ‘dew’에 대한 발음사전의 WFST

10.2.2.2 Lexicon Transducer(L)

■ Pronunciation lexicon L

- 예제

- 입력으로 주어진 raw lexicon에 대해, silence phone을 고려
 - * !SIL sil (Kaldi에서 생성되는 lexicon 기준)
- 다른 단어이지만, 동일한 phone sequence에 대해서는 disambiguation symbol을 추가
 - * #1, #2 : Disambiguation symbol



10.2.2.2 Lexicon Transducer(L)

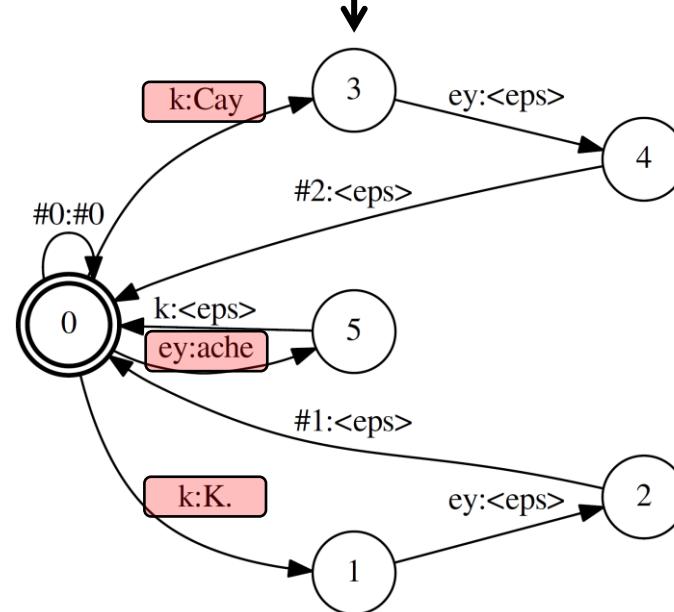
■ Pronunciation lexicon L

- 예제

Cay	k ey #1
K.	k ey #2
ache	ey k

Lexicon

Lexicon의 단어 및 각 단어에 대한 phone sequence를 이용하여, WFST L을 생성



10.2.2.2 Lexicon Transducer(L)

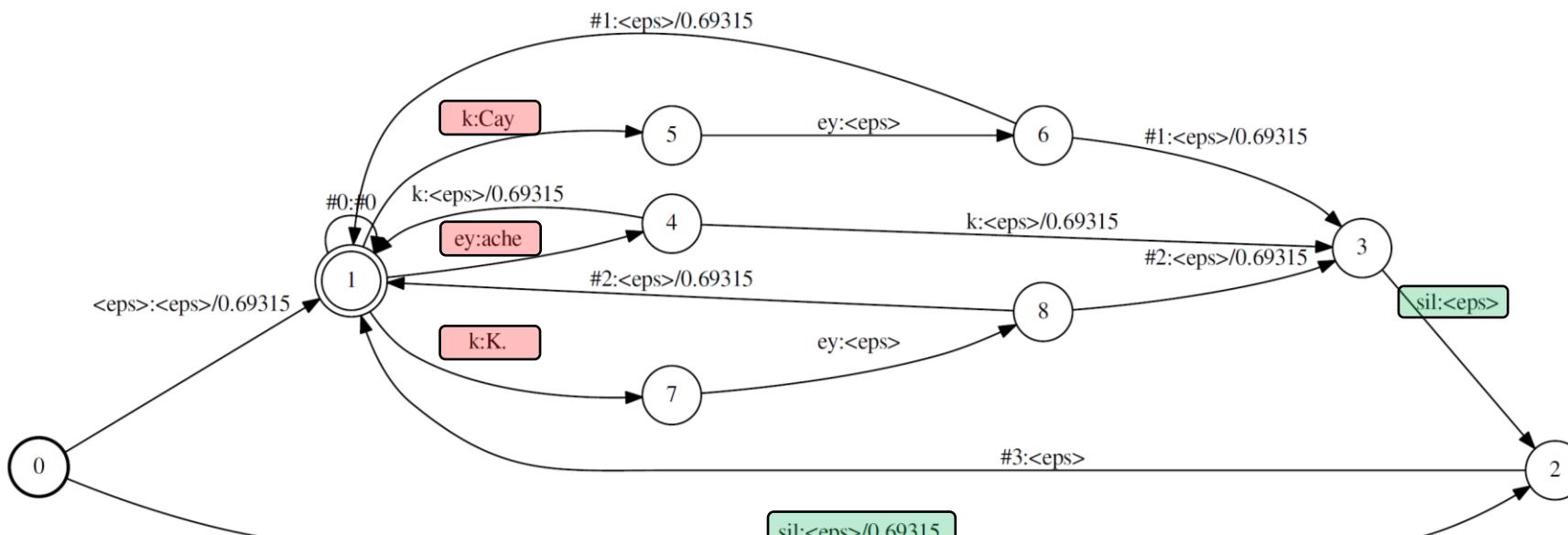
■ Pronunciation lexicon L

- 예제

Cay	k ey #1
K.	k ey #2
ache	ey k
!SIL	sil #3

추가적으로 sil에 대한 경로를 추가
단어의 시작과 끝에
sil가 추가 될 수 있도록 WFST를 구성
sil에 대한 확률은 조정 가능하며 예제는 0.5로 할당

Lexicon

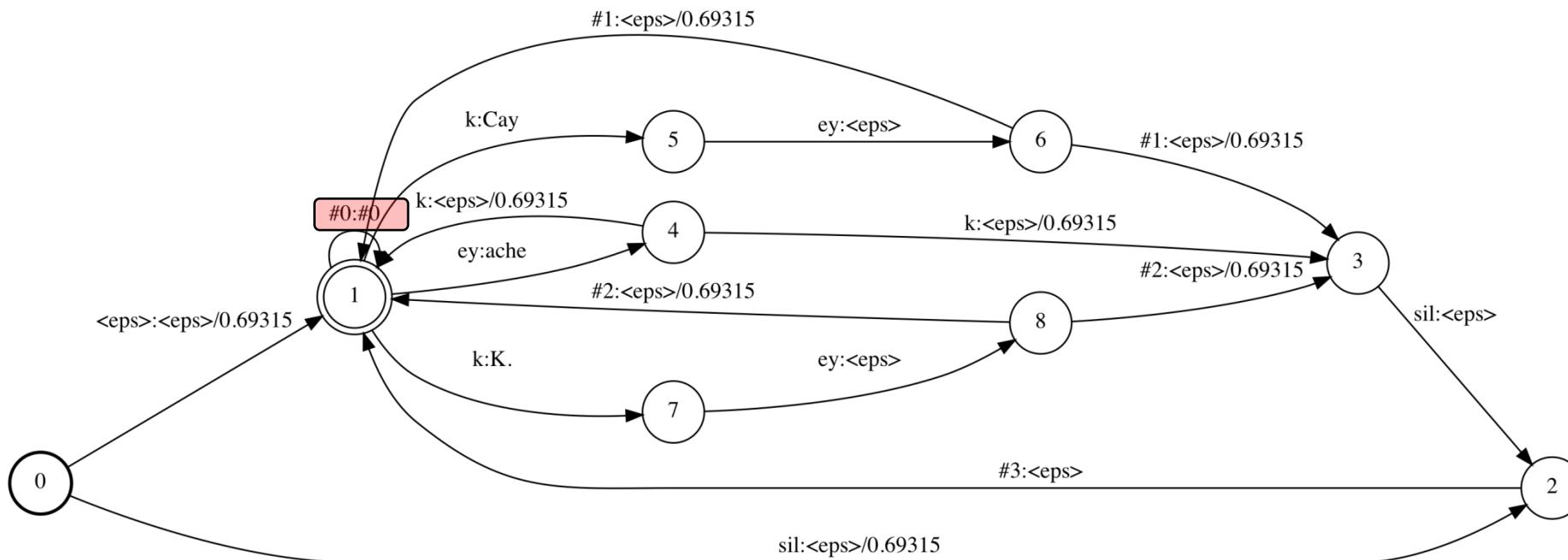


10.2.2.2 Lexicon Transducer(L)

■ Pronunciation lexicon L

- 예제

- L과 G에 대한 WFST를 composition 하기 위하여, disambiguation symbol인 #0를 추가함
 - * #0 : G FST에서 한 word의 phone sequence에 대해 종료까지 대기하는 역할

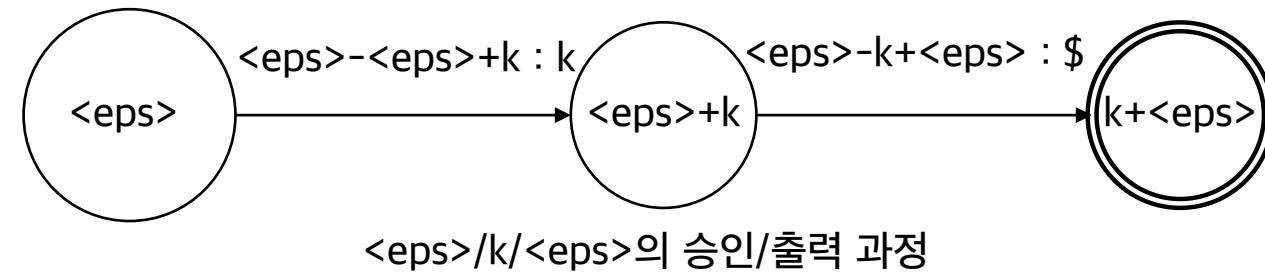


<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.3 Context-dependency transducer(C)

■ Context-dependency transducer C

- Context-dependent phone을 context-independent phone으로 변환하는 WFST
 - Context-dependent phone
 - * Tri-phone
 - Context-independent phone
 - * Phone
- C에 대한 model topology
 - Input
 - * Context-dependent phone (Tri-phone) sequence
 - Output
 - * Context-independent phone (Phone) sequence



10.2.2.3 Context-dependency transducer(C)

■ Context-dependency transducer C

- 예제

- state : left-context phone / central-phone을 나타내는 state에 numbering
- special symbol
 - * #-1 : start state에서 나가는 arc의 input, <eps><eps>를 -1로 표현
 - H의 HMM state sequence의 입력을 대기하기 위해 사용
 - * #0~#3 : C°(L°G)에서 C와 composition 되는 L °G의 disambiguation symbols에 대해서는 모든 state에 self loop를 추가
 - * \$: 최종상태로 향하는 transition의 출력기호

10.2.2.3 Context-dependency transducer(C)

■ Context-dependency transducer C

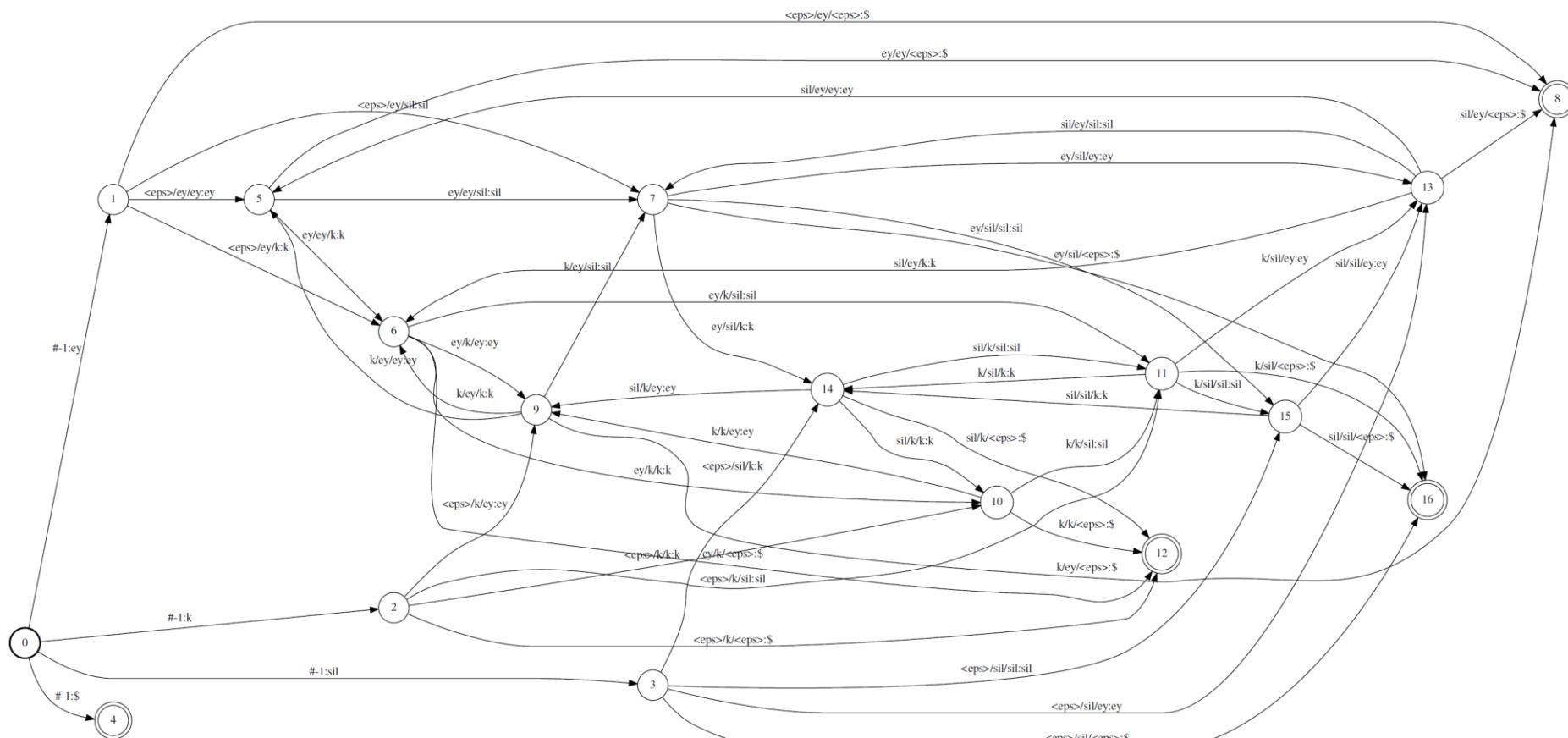
- 예제

- N : mono-phone의 개수
 - * 예제의 N = 4 (<eps>, k, ey, sil)
- State의 개수
 - * start state + left-context × mono-phone : $1+4^2 = 17$ 개
- final state의 개수
 - * mono-phone / <eps> : 4개
- transition의 개수
 - * $O(N^3)$: number of state × mono-phone

10.2.2.3 Context-dependency transducer(C)

■ Context-dependency transducer C

- 예제(self-loop 제거)

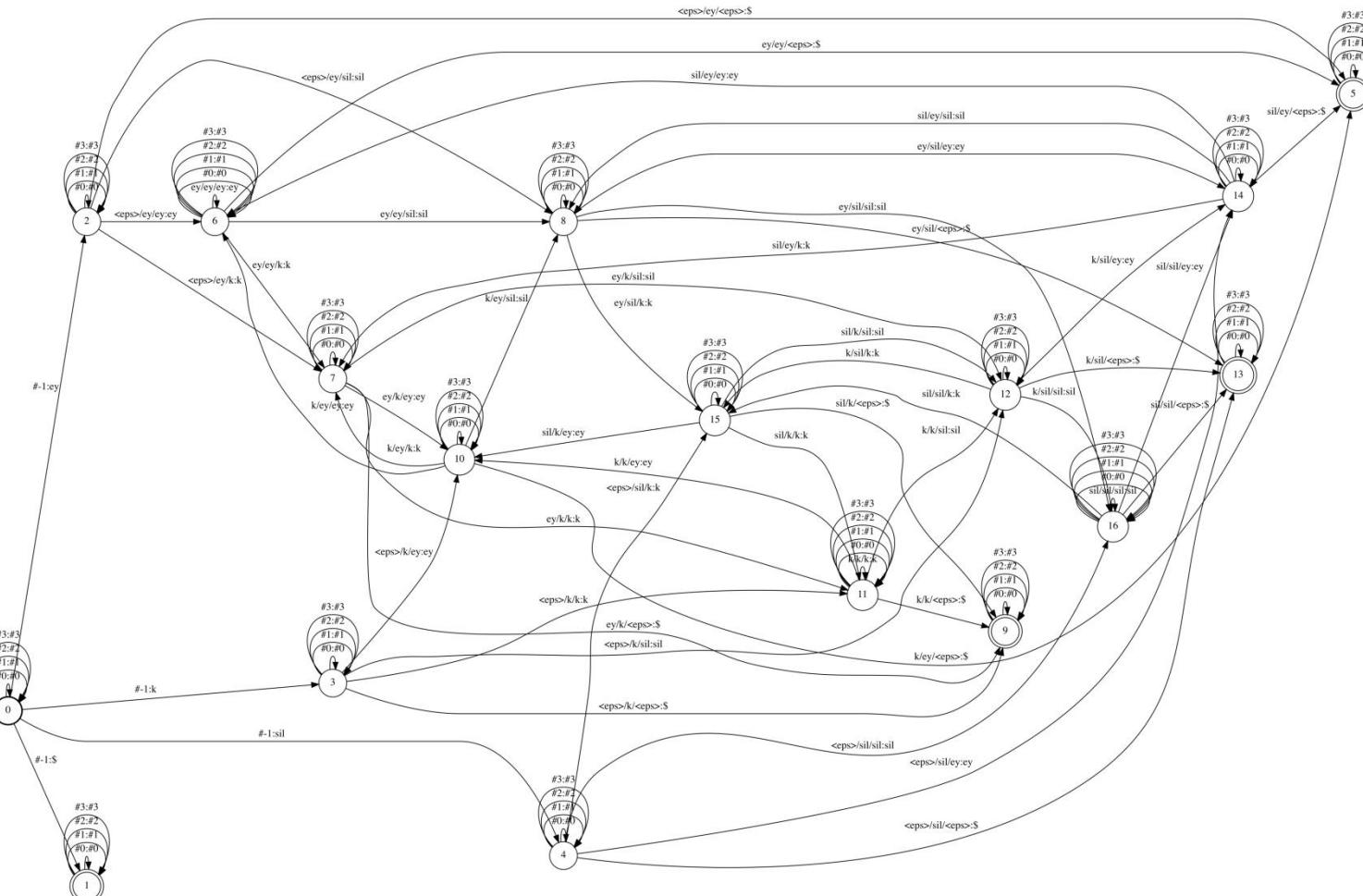


<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.3 Context-dependency transducer(C)

■ Context-dependency transducer C

- 예제



10.2.2.4 HMM topology transducer (H)

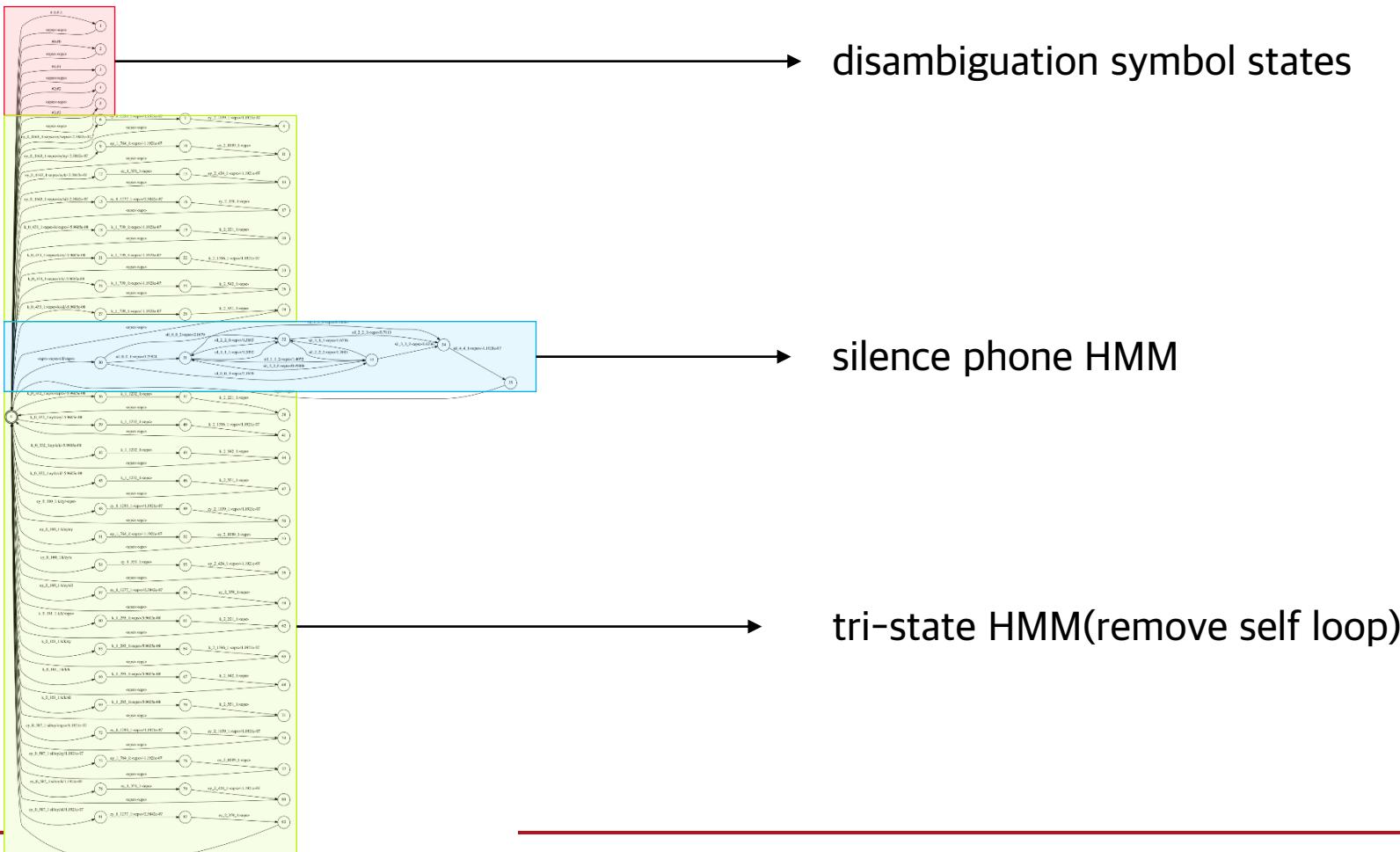
■ HMM topology transducer H

- HMM 구조를 WFST로 표현
 - Maps states to phonemes
- H에 대한 model topology
 - Input
 - * HMM state sequence
 - Output
 - * Context-dependent phone (Tri-phone) sequence
 - Weight
 - * HMM state transition probability

10.2.2.4 HMM topology transducer (H)

- HMM topology transducer H

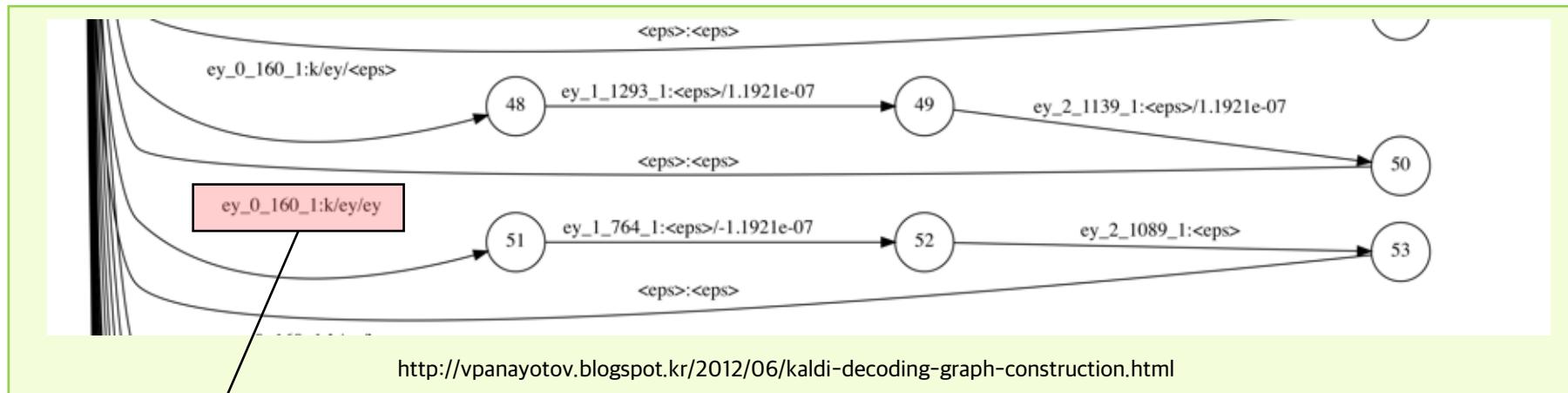
- ## ● 예제



10.2.2.4 HMM topology transducer (H)

■ HMM topology transducer H

- 예제



ey_0_160_1: k/ey/ey

Notation
 $K_S_PDFID_O : \text{tri-phone / W}$

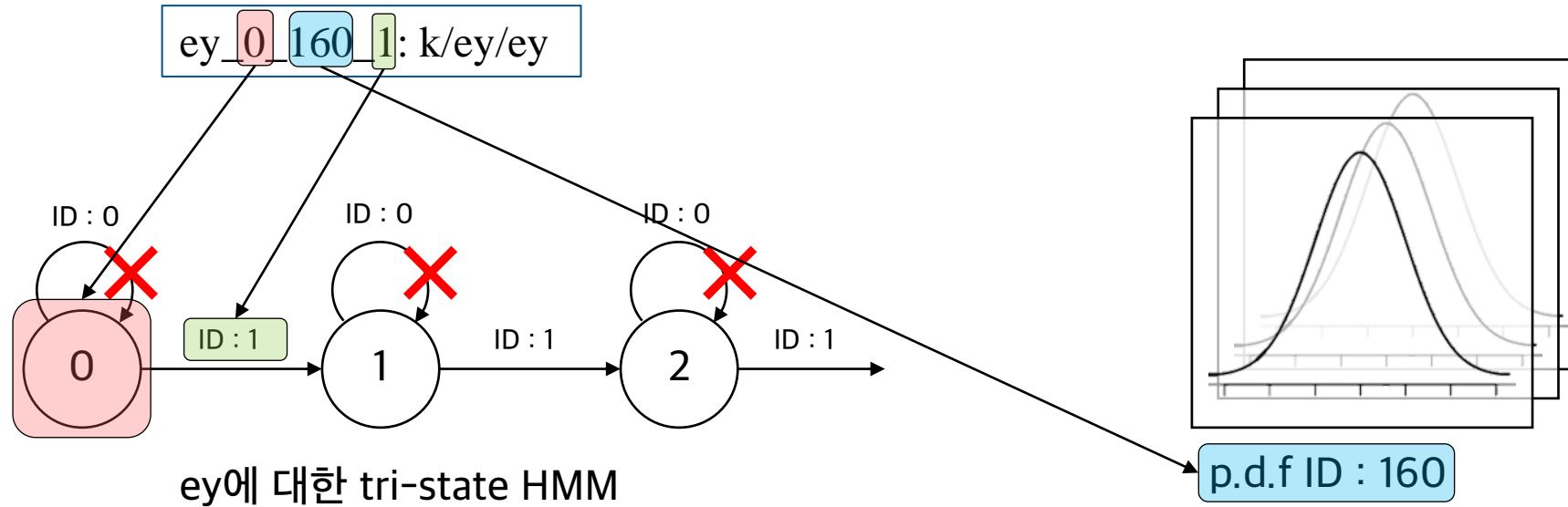
- K : K에 대한 HMM
- S : HMM State의 번호, 일반적으로 tri-state HMM은 0,1,2로 이루어짐
- PDFID : HMM의 transition p.d.f의 ID
- O : S-state에서 나가는 transition의 ID
- W : weight, H FST에서는 일괄적으로 확률값들을 1로 계산

10.2.2.4 HMM topology transducer (H)

■ HMM topology transducer H

- 예제

- H transducer는 composition 과정의 복잡도를 줄이기 위해 self-loop를 제거한 Ha transducer를 이용
- C FST에서 #_1 symbol은 HMM state sequence의 입력을 대기하기 위해 필요
- 제거된 self-loop는 $H \circ C \circ L \circ G$ 를 생성 후 추가

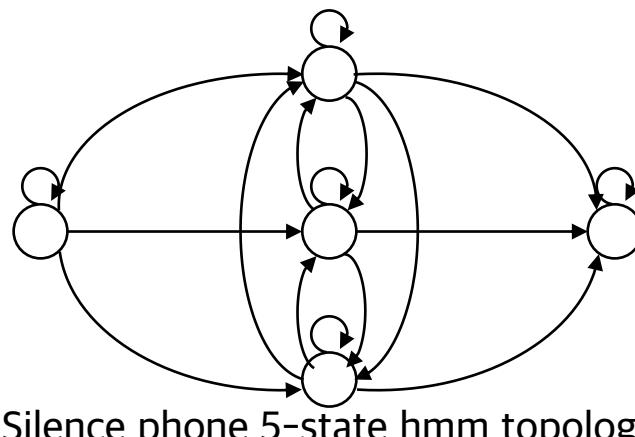
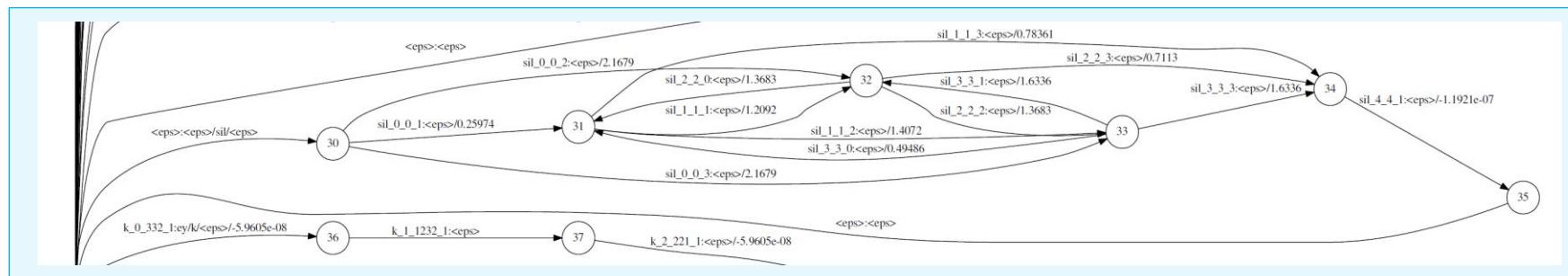


10.2.2.4 HMM topology transducer (H)

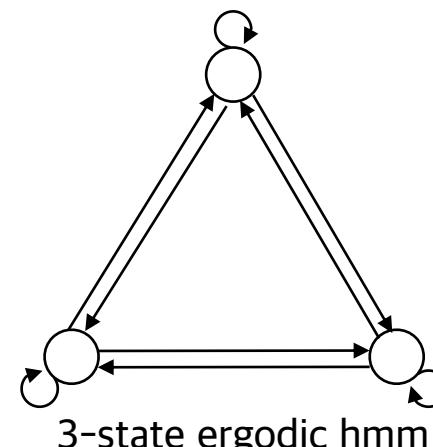
■ HMM topology transducer H

- 예제

- Silence에 대해서는 다음과 같은 5-State HMM topology로 모델링
- Initial state와 final state를 제외한 3개의 state가 모두 연결된 ergodic model



Silence phone 5-state hmm topology



3-state ergodic hmm

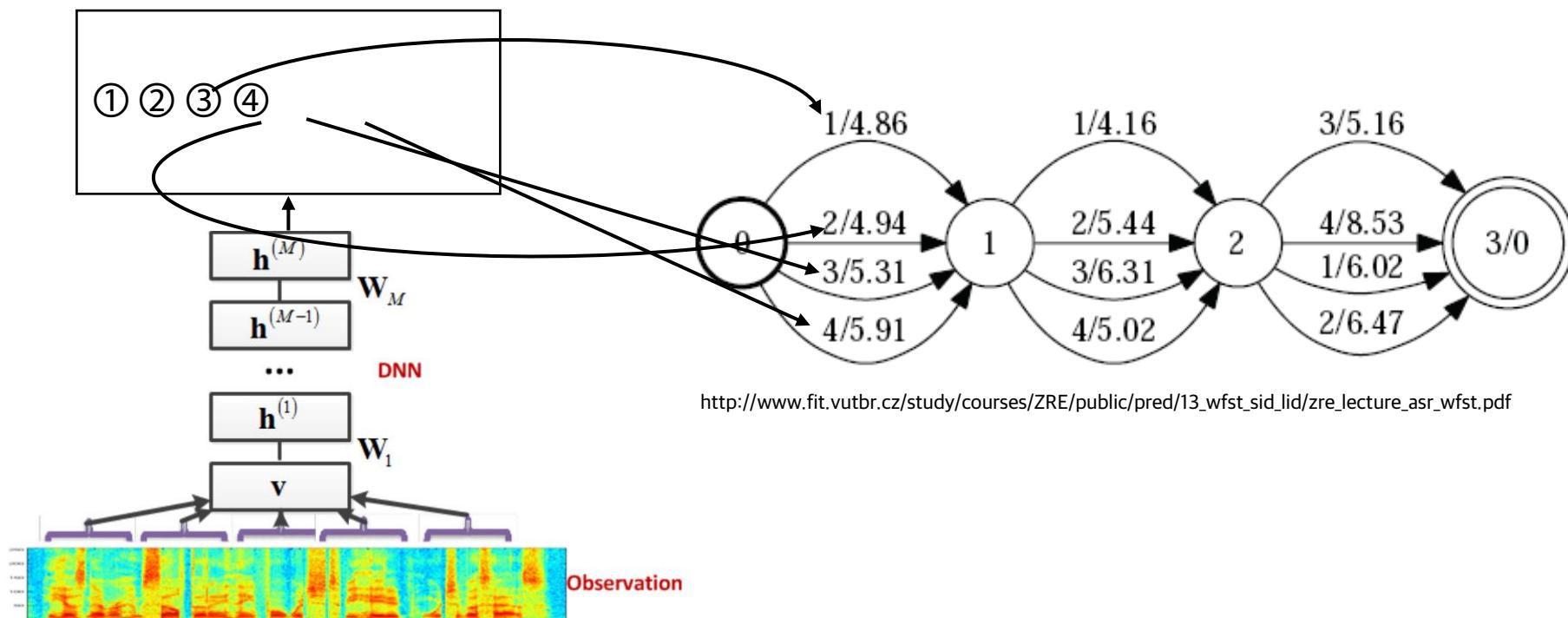
10.2.2.5 Utterance transducer(U)

- Let U be an FST that encodes the acoustic scores of an utterance
 - U 는 실제 입력된 음성 신호에 따라 계속 재구축됨
 - HCLG처럼 미리 모델을 생성할 수 없음

- Let $S = U \circ HCLG$ be called the search graph for an utterance
 - 디코딩 과정에서 U 는 항상 재구축됨
 - S 도 U 에 의해, 디코딩 과정에서 항상 재구축됨

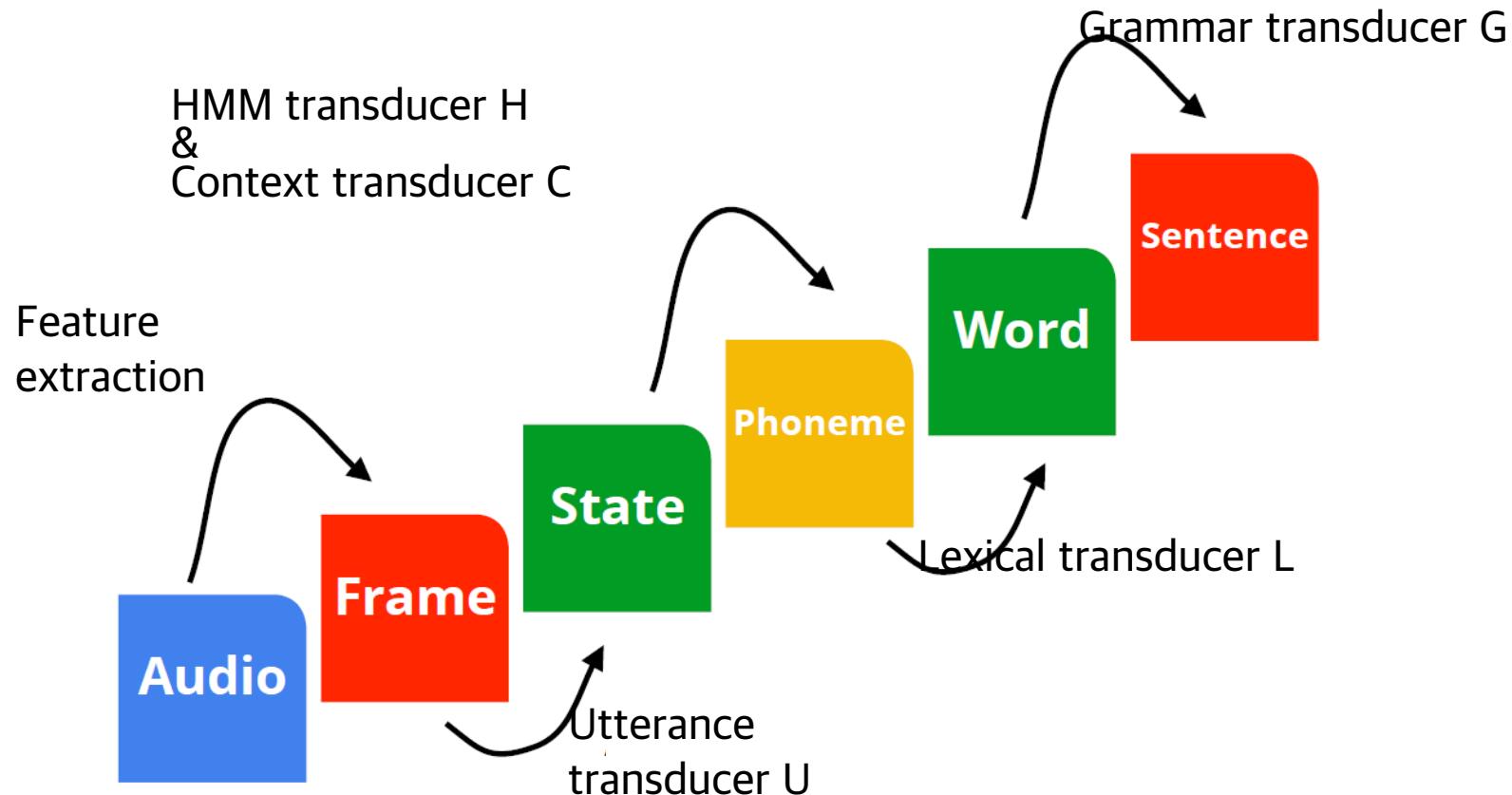
10.2.2.5 Utterance transducer(U)

■ Utterance transducer U



10.2.2.6 WFST 기반 디코딩 구성 요소 – 정리

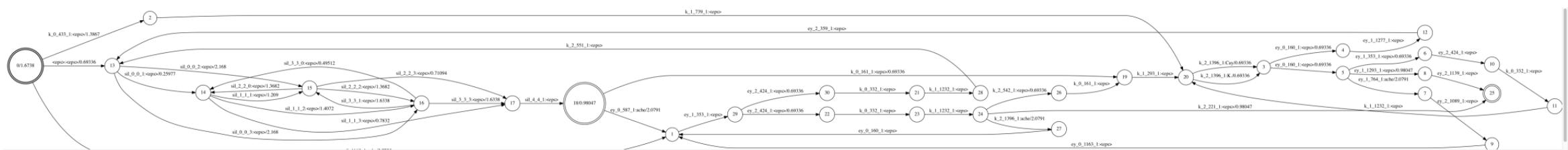
- Speech recognition pipeline on WFST



10.2.2.6 WFST 기반 디코딩 구성 요소 – 정리

■ HCLG에 대한 WFST

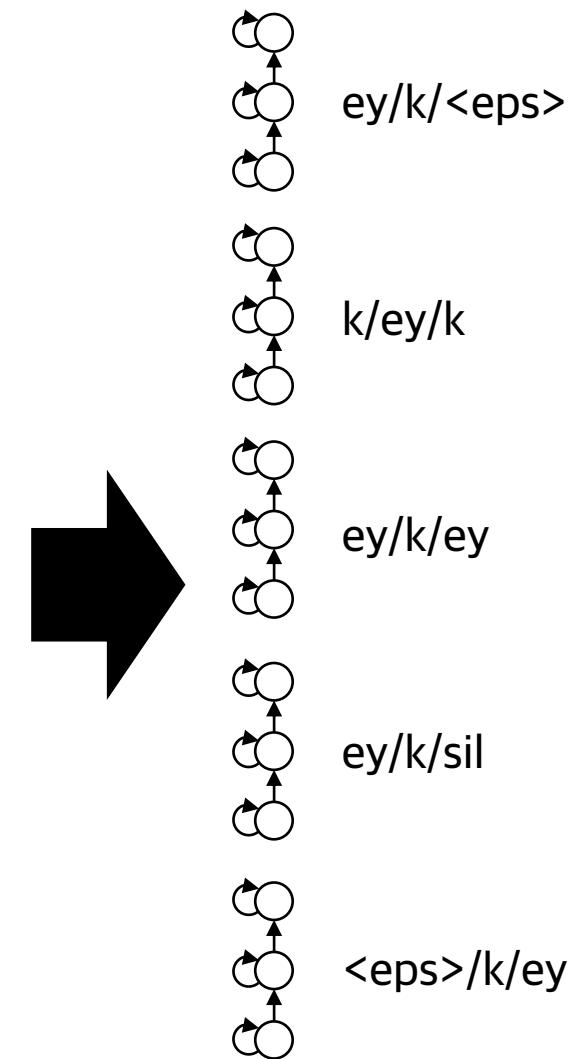
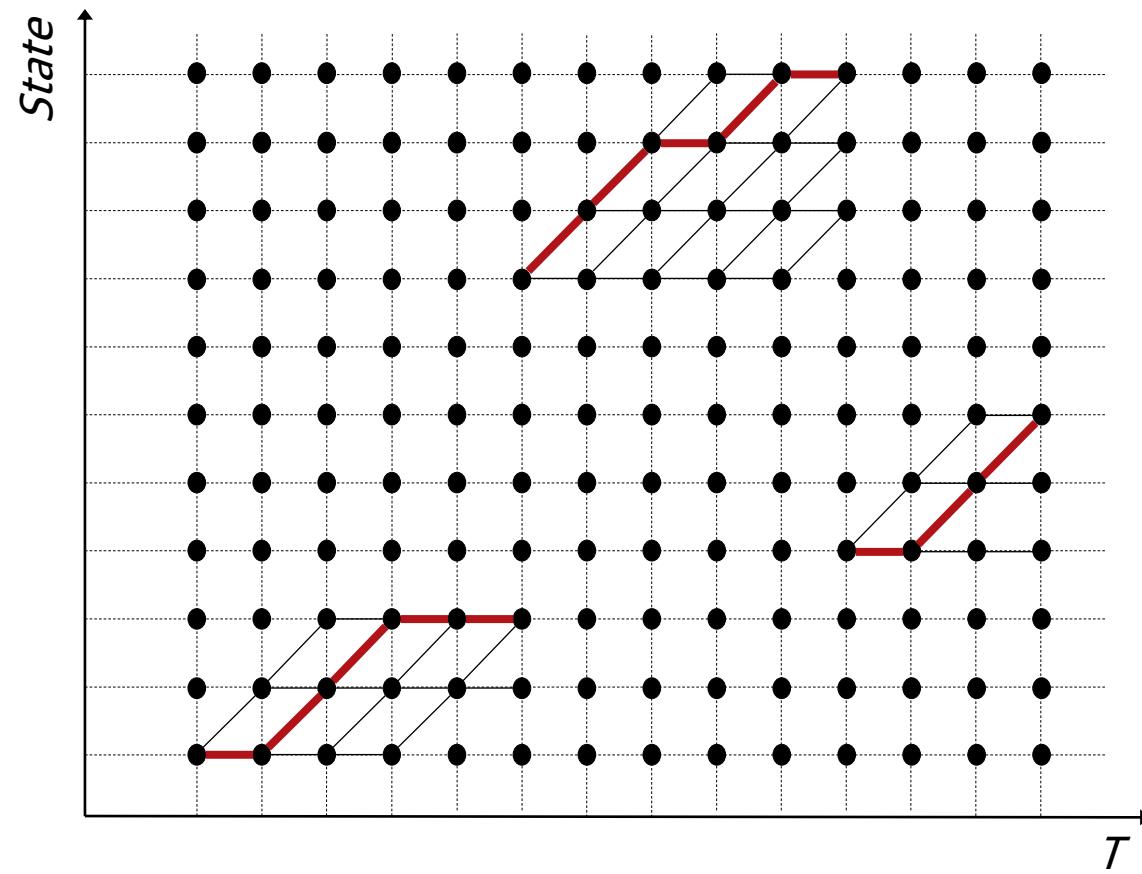
- HCLG에 대한 model topology
 - Input
 - * HMM state
 - Output
 - * Word
 - Weight
 - * H, L, G의 weight 정보를 모두 composition한 value



<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>

10.2.2.6 WFST 기반 디코딩 구성 요소 – 정리

■ WFST based Decoding

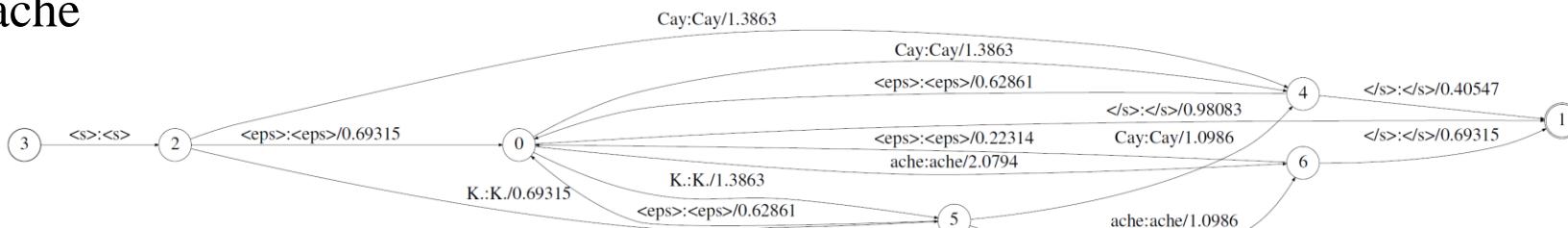


10.2.2.6 WFST 기반 디코딩 구성 요소 – 정리

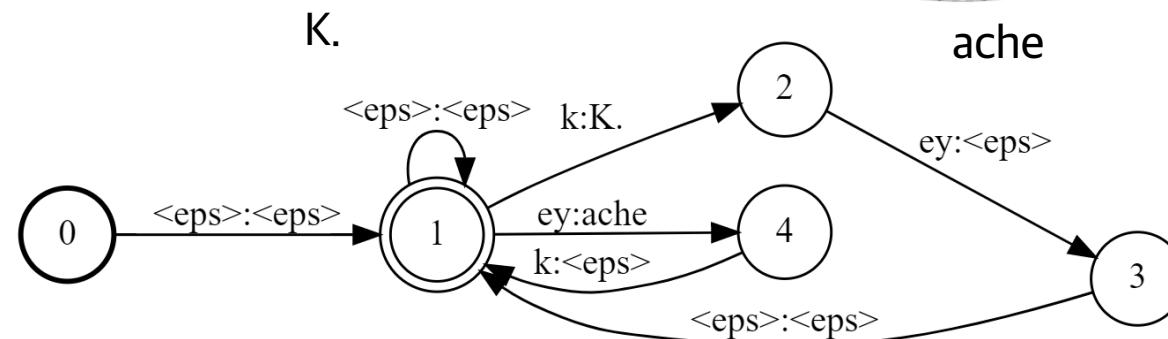
■ WFST based Decoding

Output : K. ache

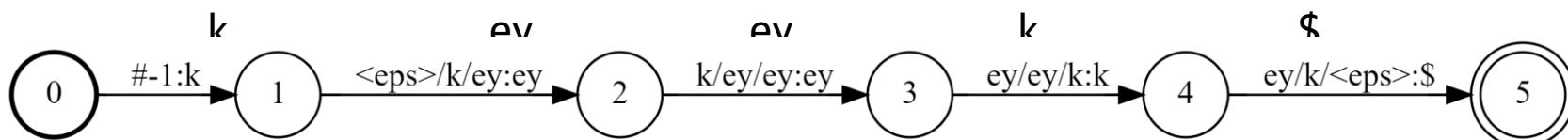
G



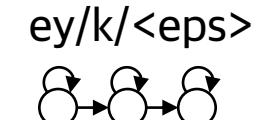
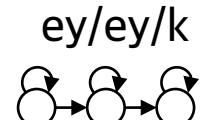
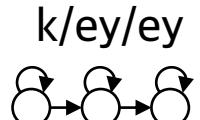
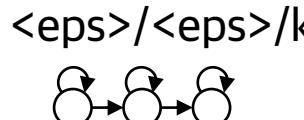
L



C



H



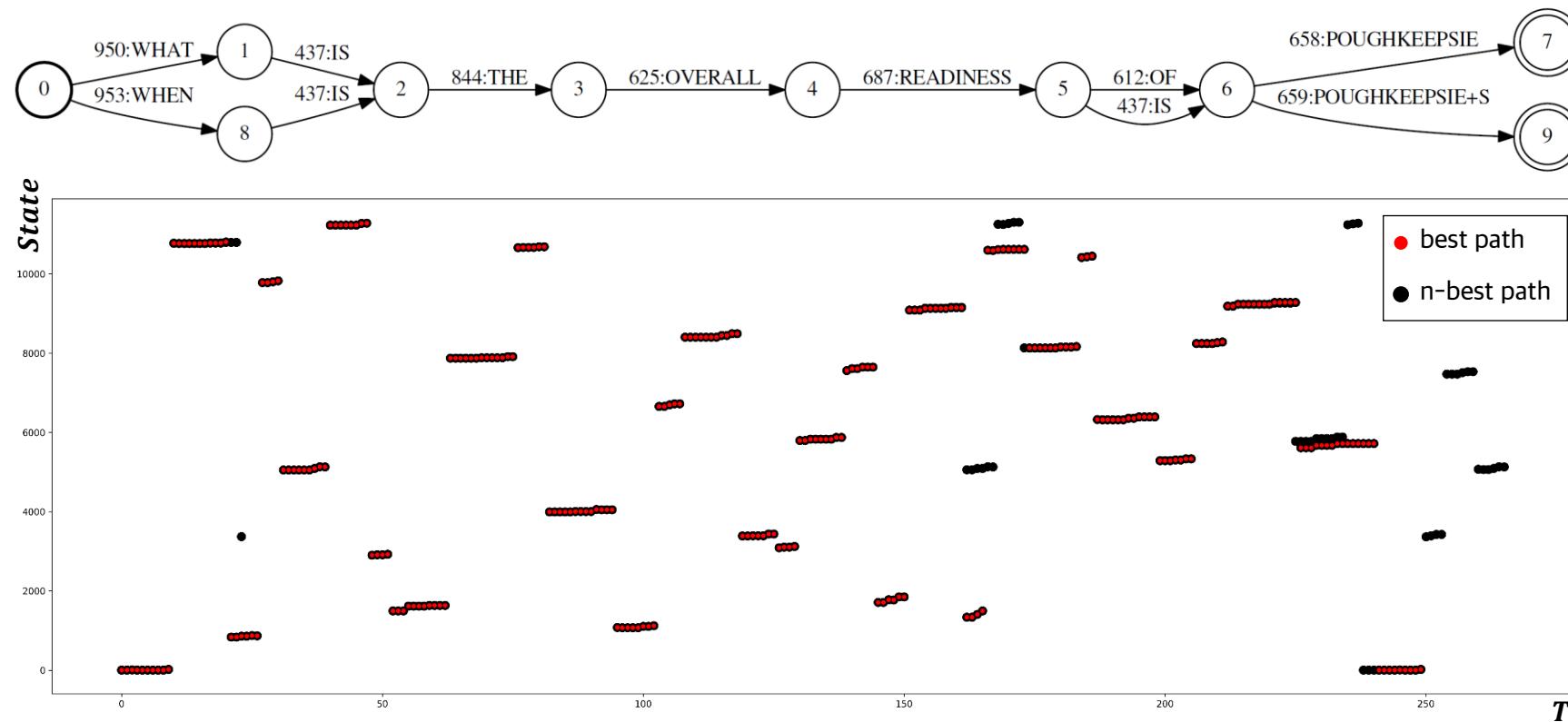
음성인식

10.2.2.6 WFST 기반 디코딩 구성 요소 – 정리

■ WFST based Decoding

- 예제

- 실제 wav파일에 n-best hmm state 출력 결과



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ HCLG를 생성하기 위한 알고리즘

- Composition
- Determinization
- Minimization

■ 본 장에서는 HCLG에서 L과 G를 합치는 과정을 예제로 설명함

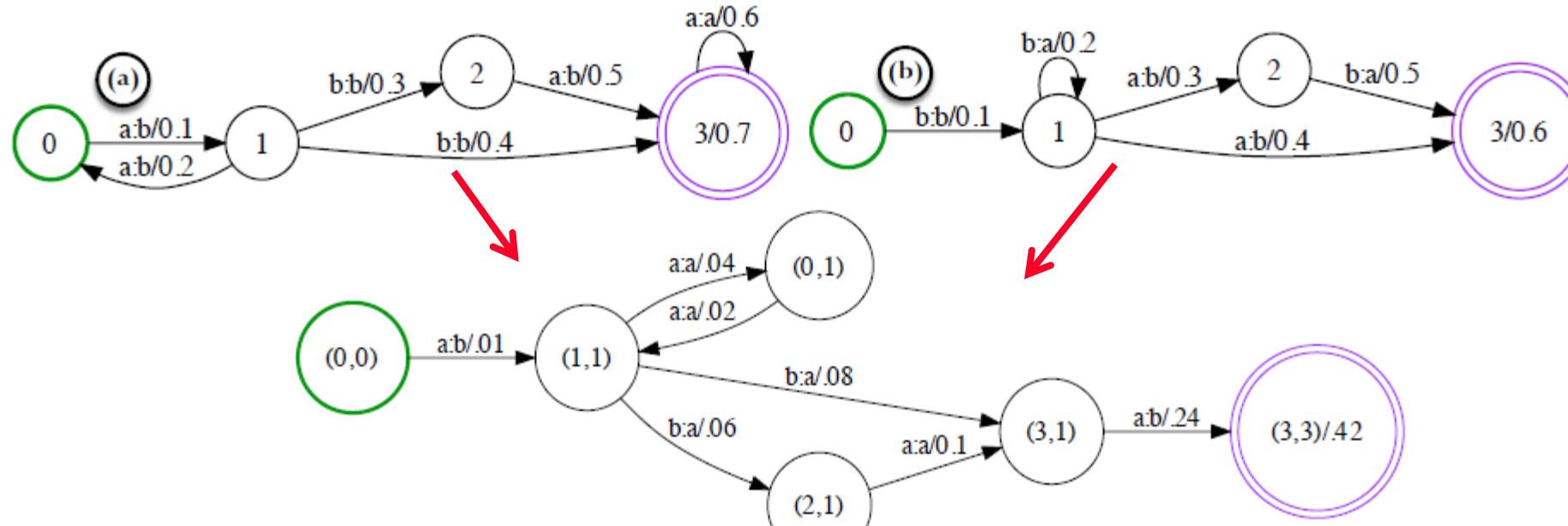
- $\min(\det(L \circ G))$
 - $L \circ G$
 - * L과 G의 composition
 - \det
 - * Determinization
 - \min
 - * Minimization

10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘

- $L \circ G$ 는 L composed with G를 의미
 - * $G \circ L \neq L \circ G$ if $L \neq G$
- L의 output symbol과 G의 input symbol이 일치하는 edge에 대해서 composition



<http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf>

10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘

- Pseudo code

Algorithm 1 Weighted-Composition(T_1, T_2)

```
1:  $Q \leftarrow I_1 \times I_2$  } Initialize the queue  
2:  $K \leftarrow I_1 \times I_2$  } and new WFST  
3: while  $K \neq \emptyset$  do  
4:    $q = (q_1, q_2) \leftarrow \text{Head}(K)$   
5:   Dequeue( $K$ )  
6:   if  $q \in I_1 \times I_2$  then } Initial state matches  
7:      $I \leftarrow I \cup \{q\}$   
8:      $\lambda(q) \leftarrow \lambda_1(q_1) \otimes \lambda_2(q_2)$   
9:   if  $q \in F_1 \times F_2$  then } Final state matches  
10:     $F \leftarrow F \cup \{q\}$   
11:     $p(q) \leftarrow p_1(q_1) \otimes p_2(q_2)$   
12:    for each  $(e_1, e_2) \in E[q_1] \times E[q_2]$  such that  $o[e_1] = i[e_2]$  do } Create new arcs  
13:      if  $(q') = (n[e_1], n[e_2]) \notin Q$  then for any matching labels, and add to the new WFST  
14:         $Q \leftarrow Q \cup \{q'\}$   
15:        Enqueue( $K, q'$ )  
16:         $E \leftarrow E \uplus \{(q, i[e_1], o[e_2], w[e_1] \otimes w[e_2], q')\}$   
17: return  $T$ 
```

Algorithm complexity	
Time	$O(V_1 \cdot V_2 \cdot D_1 \cdot (\log D_2 + M_2))$
Space	$O(V_1 \cdot V_2 \cdot V_1 \cdot M_2)$
$V_i = \# \text{states}, D_i = \text{max out-degree}, M_i = \text{max multiplicity of } i^{\text{th}} \text{ WFST. } \text{www.OpenFST.org}$	

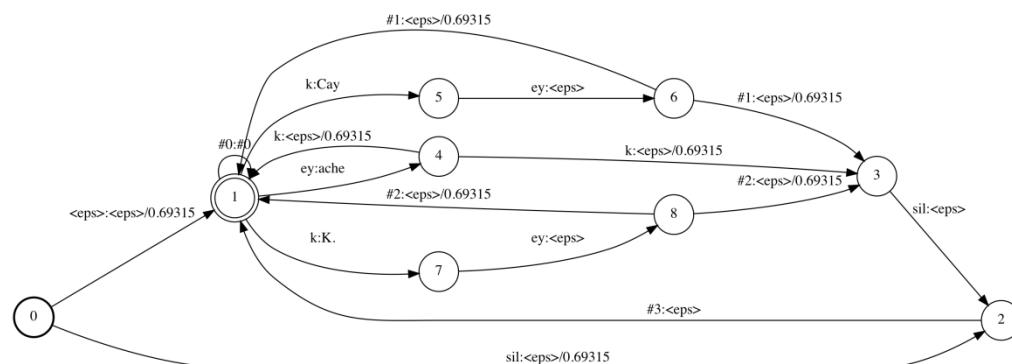
<http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf>

10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

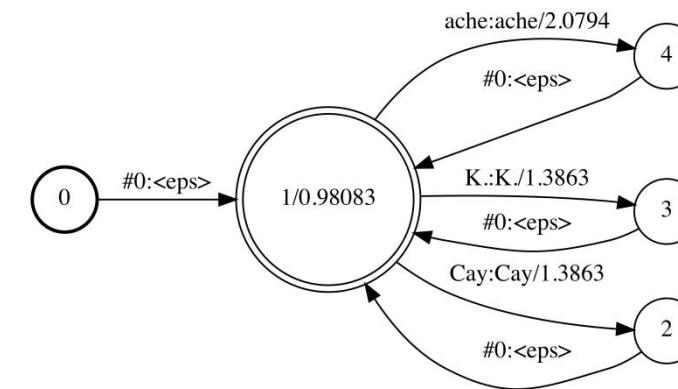
■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘
 - 예제

* 이전에 생성된 L과 G를 이용하여 composition 수행



(a) Lexicon에 대한 WFST (L)

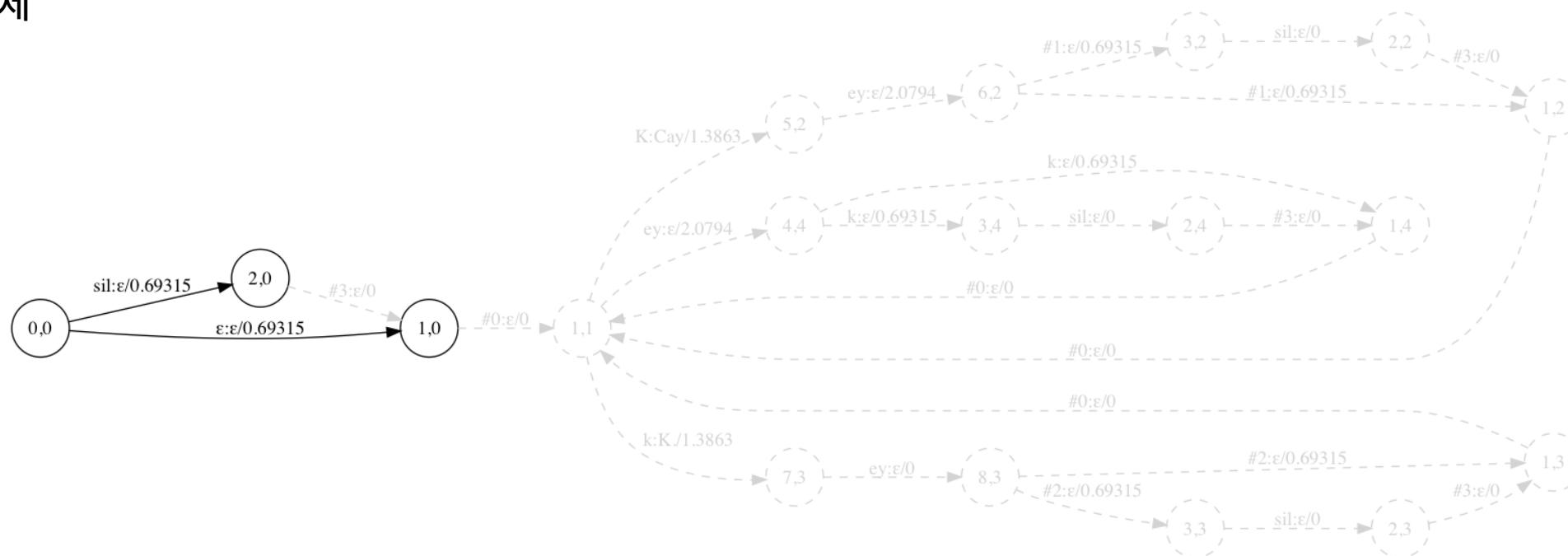


(b) Uni-gram 언어모델에 대한 WFST (G)

10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

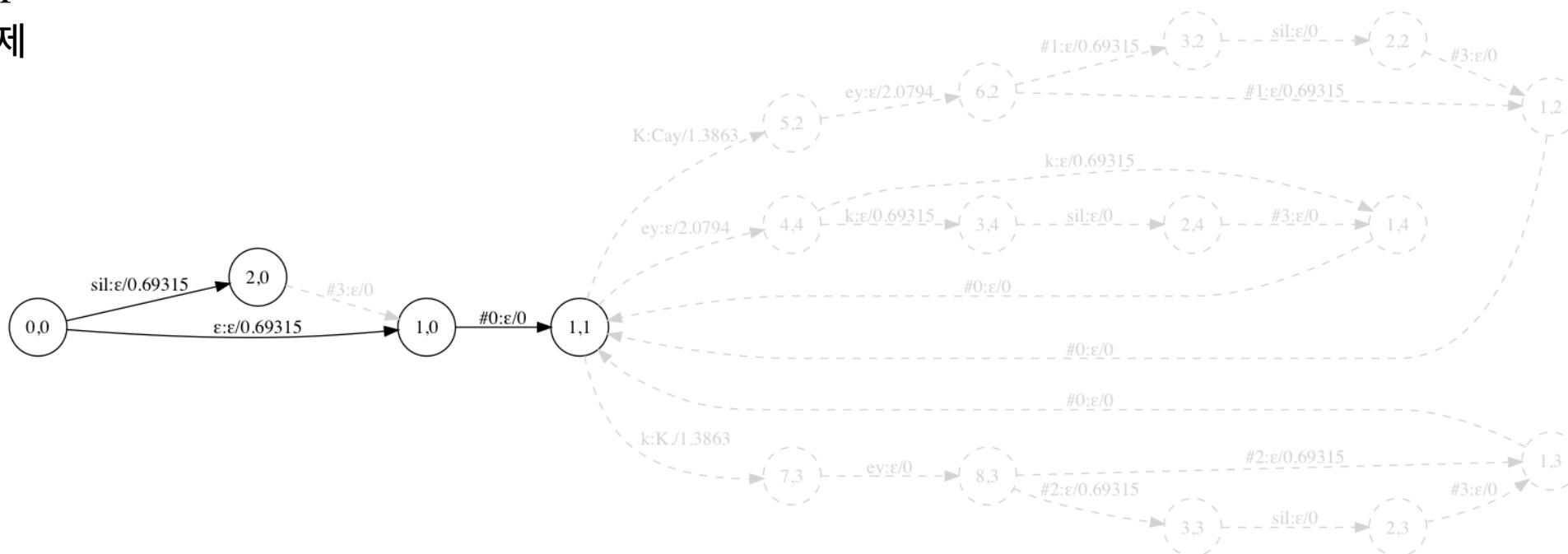
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

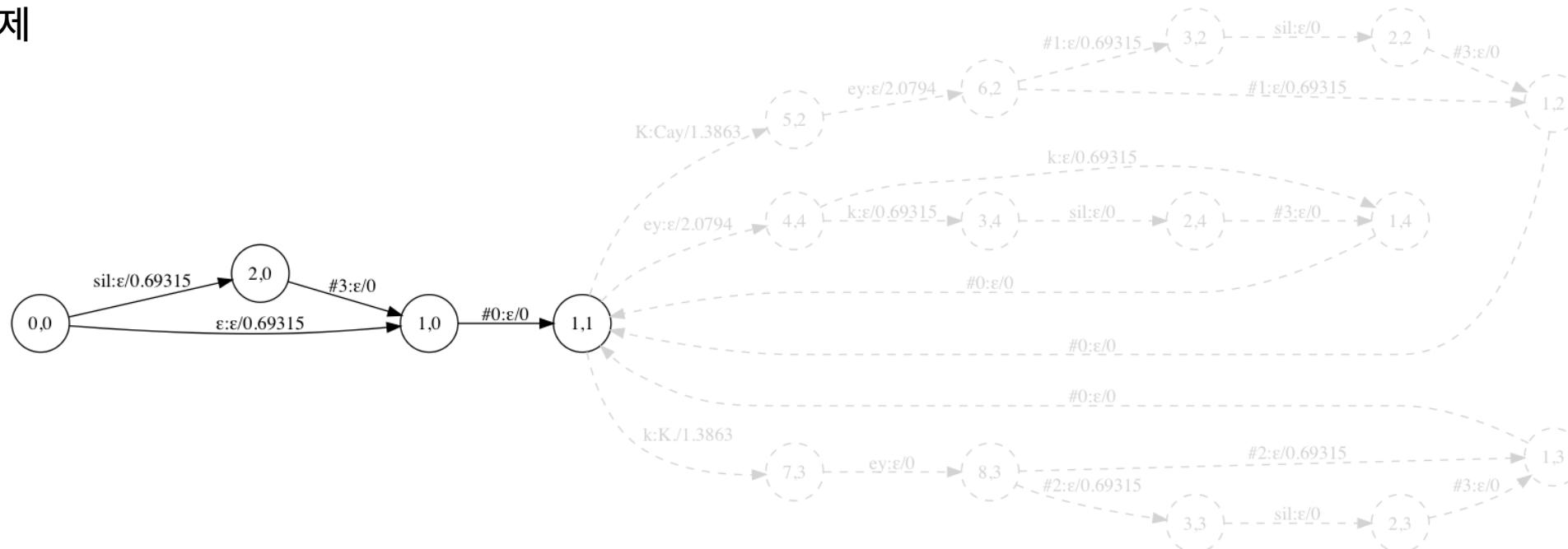
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

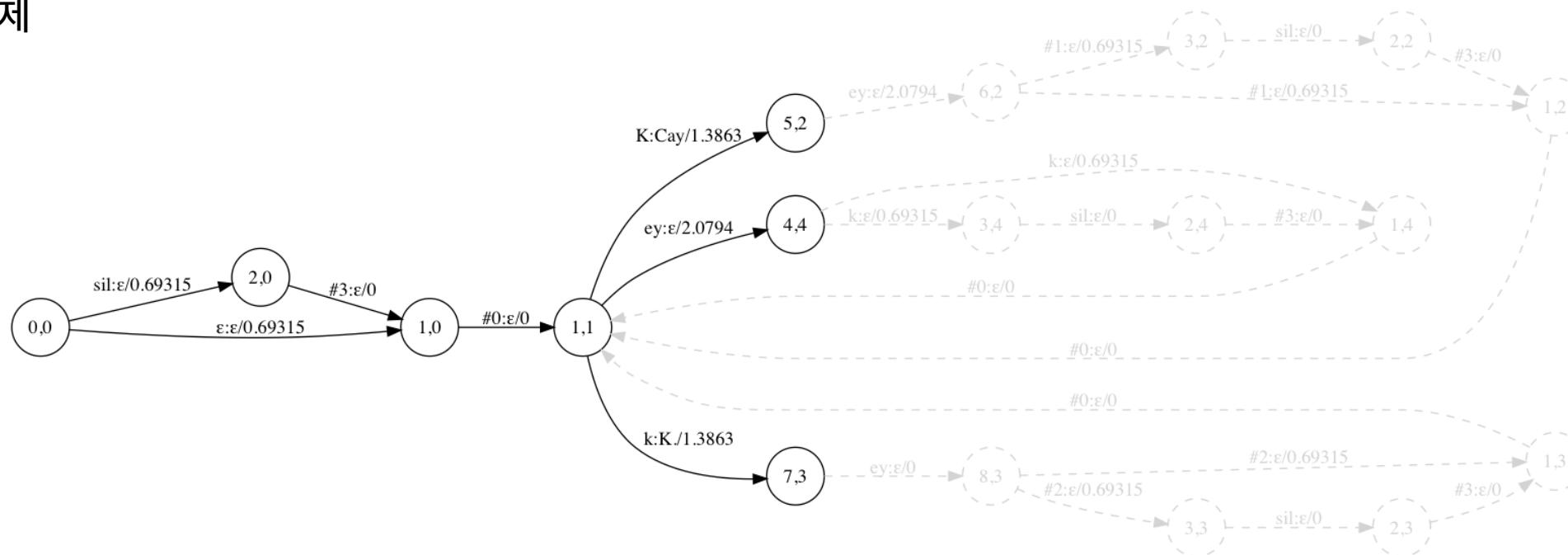
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

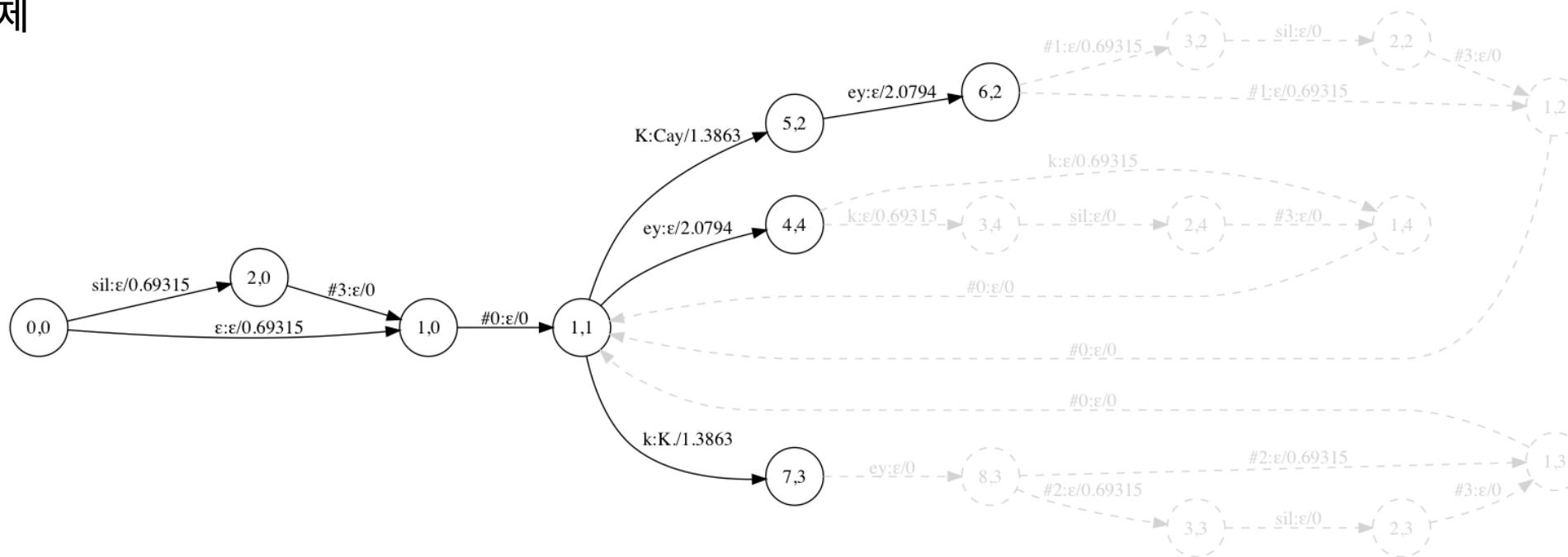
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

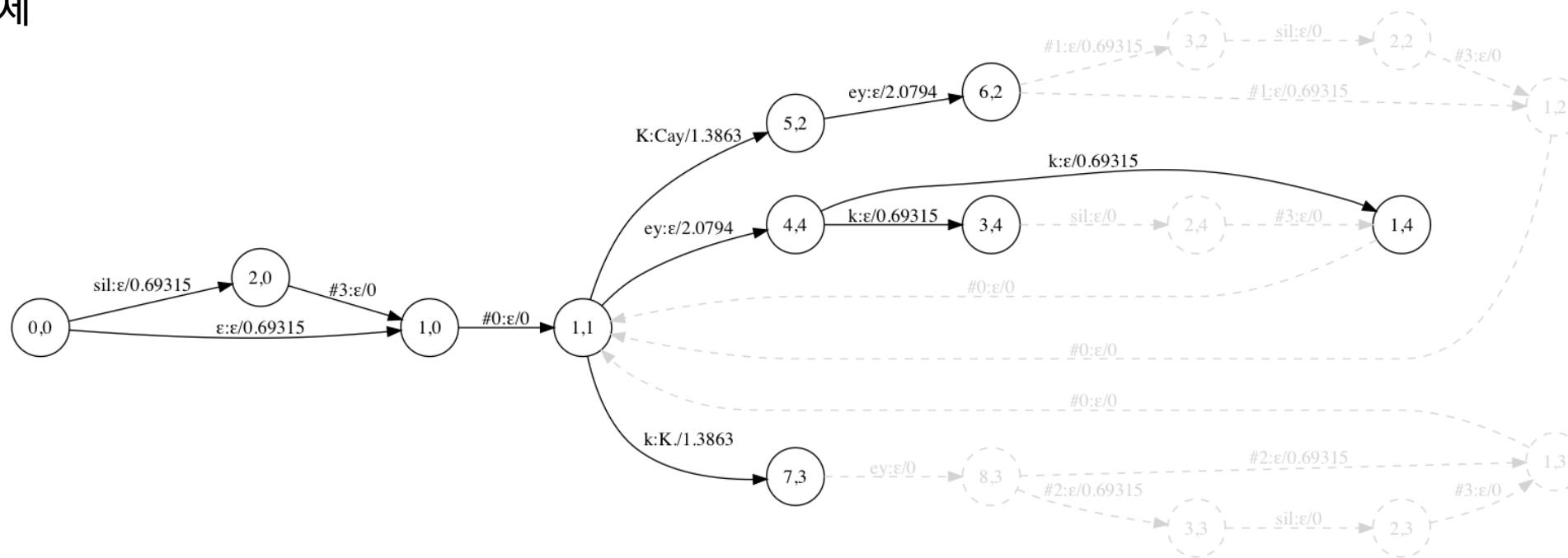
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

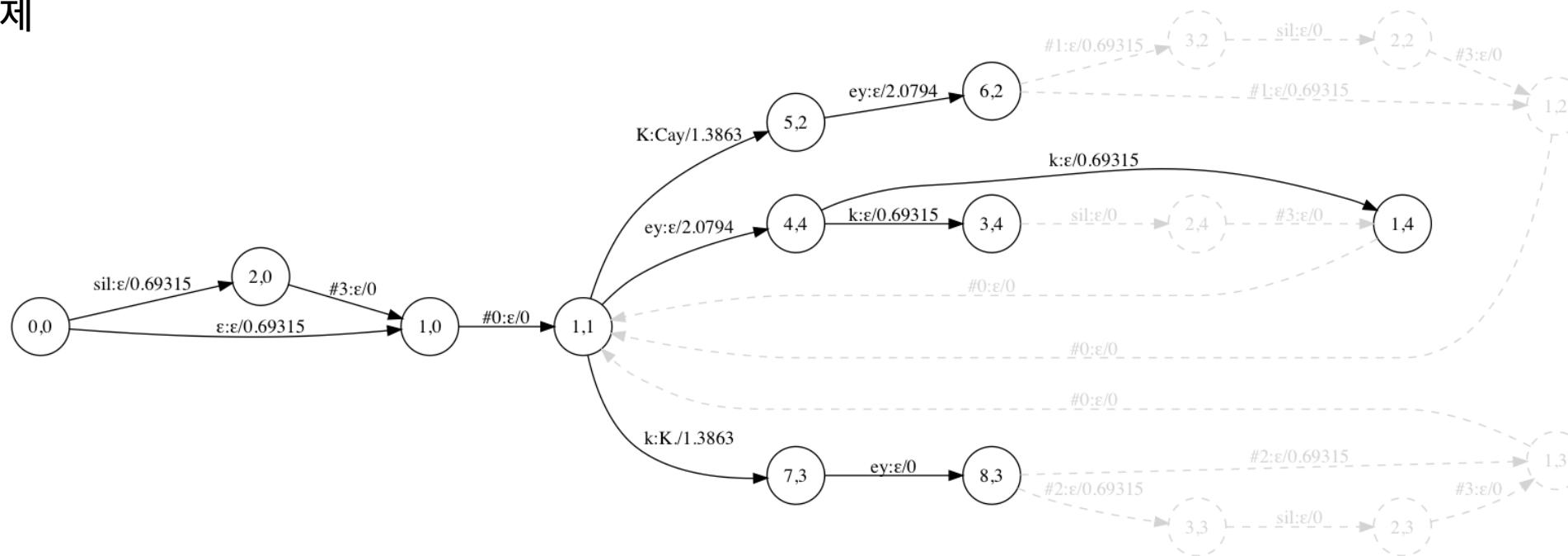
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

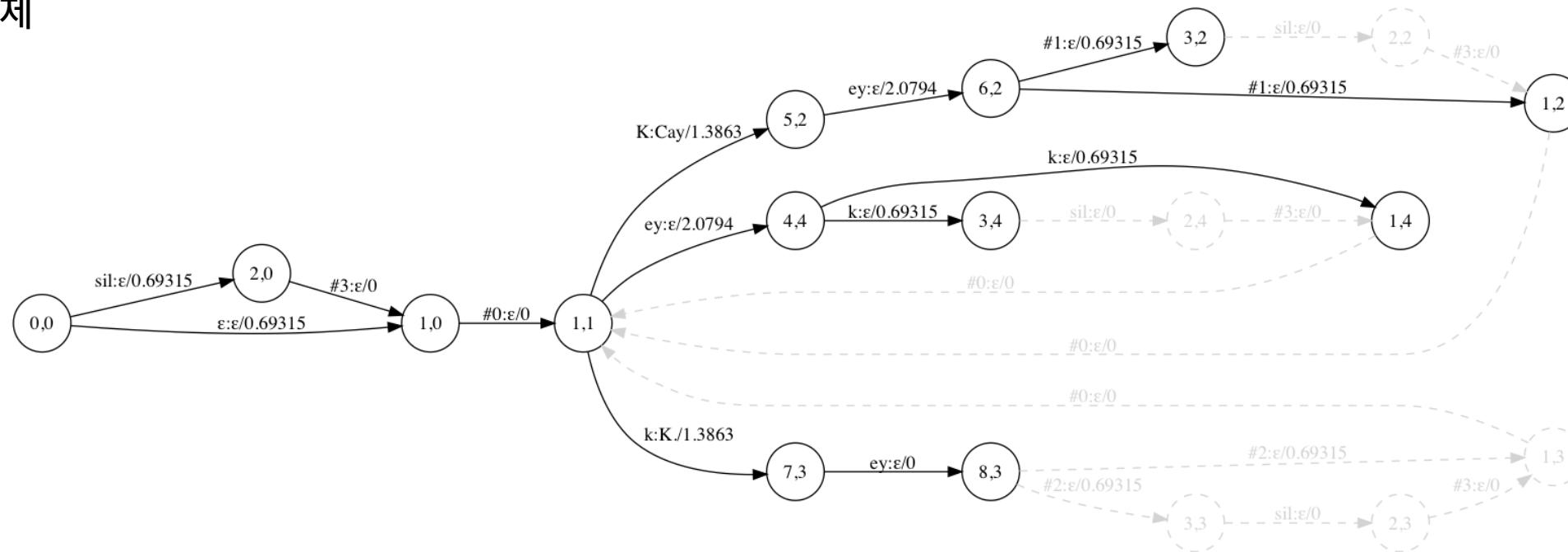
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

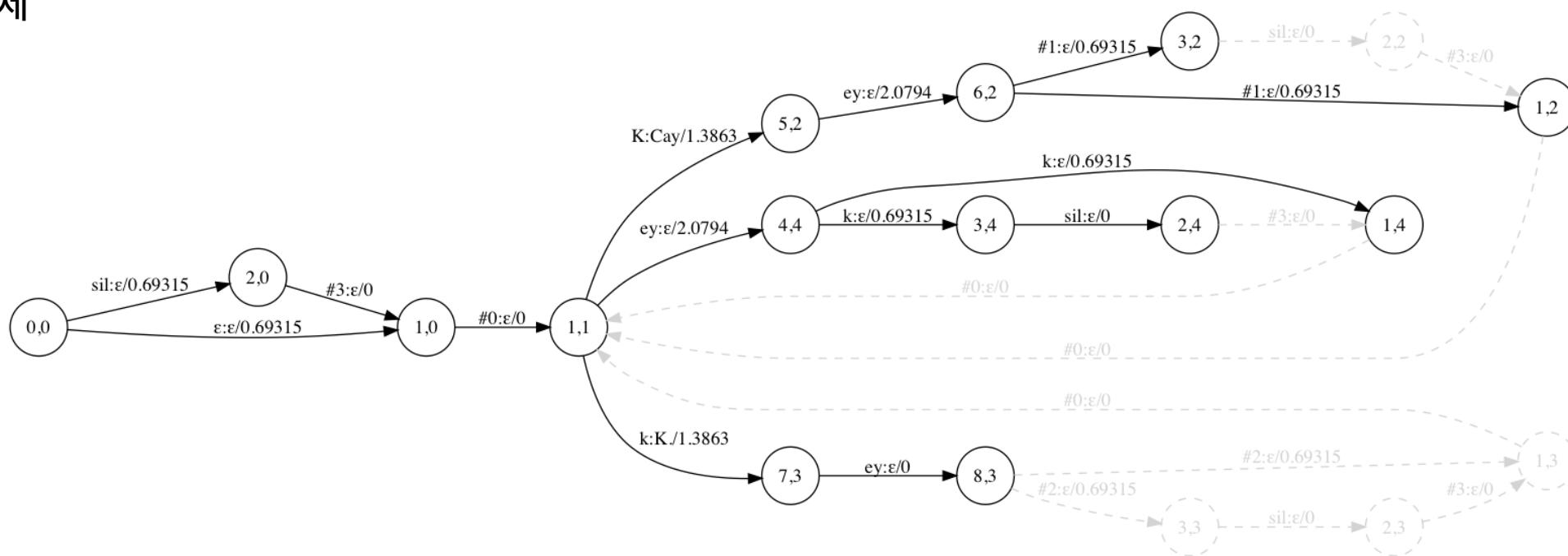
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

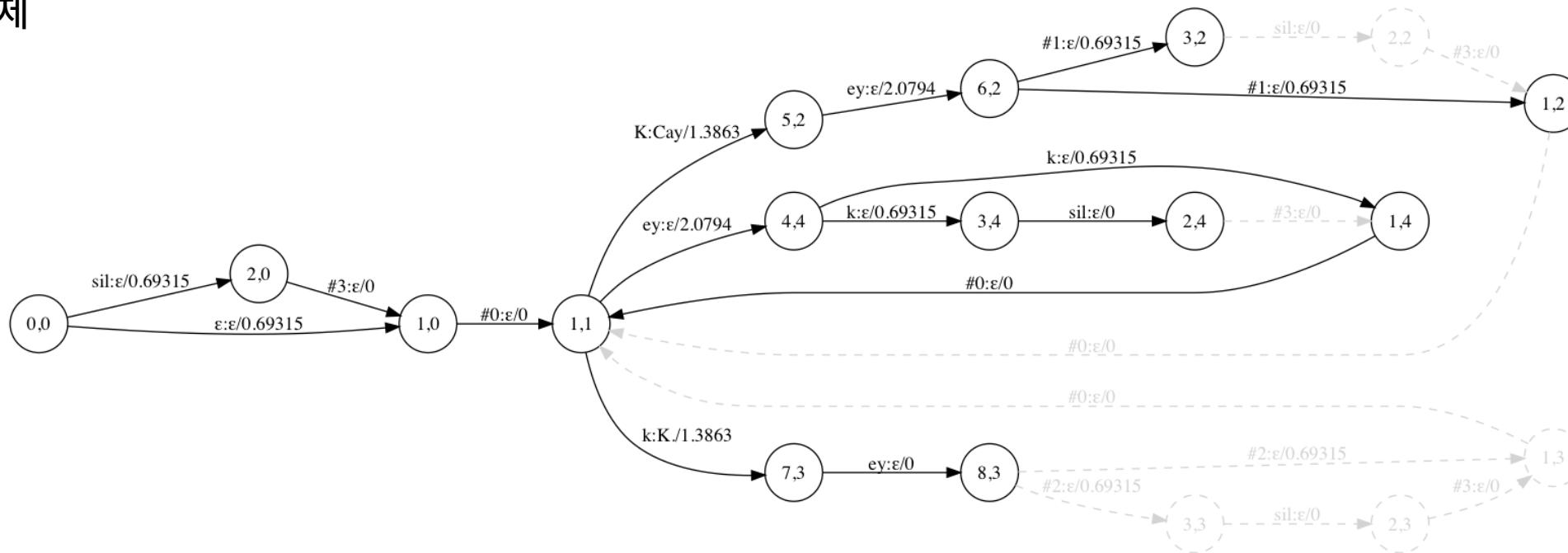
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

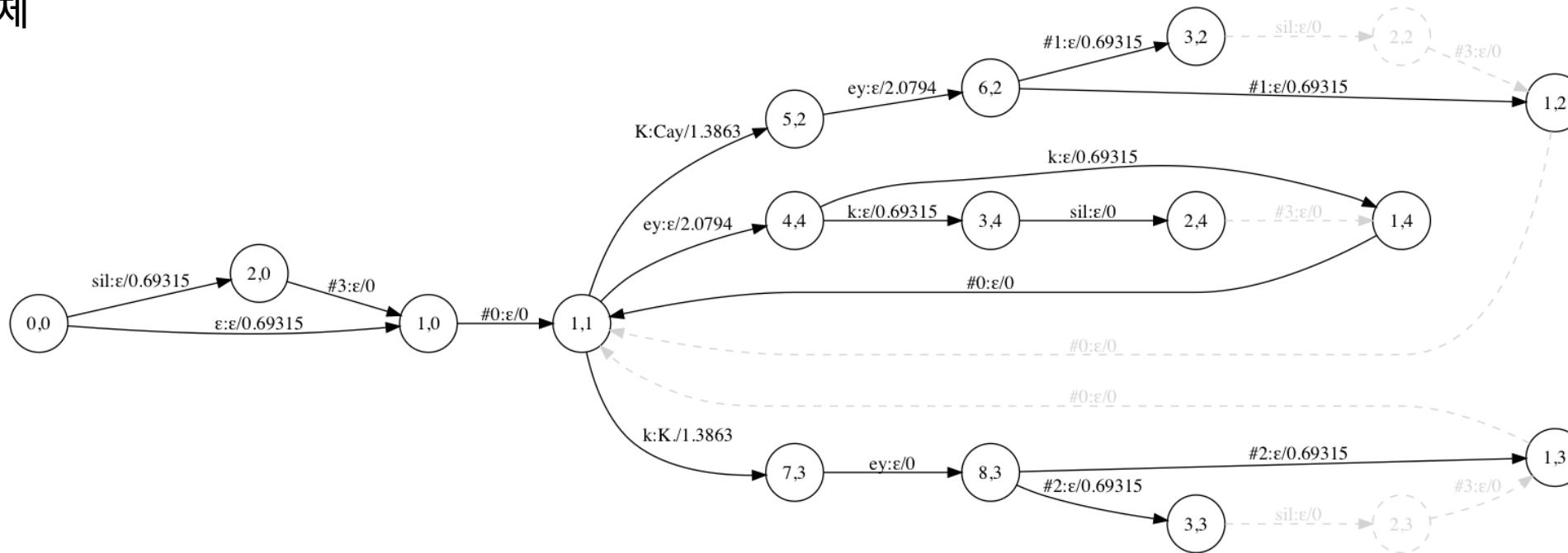
- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

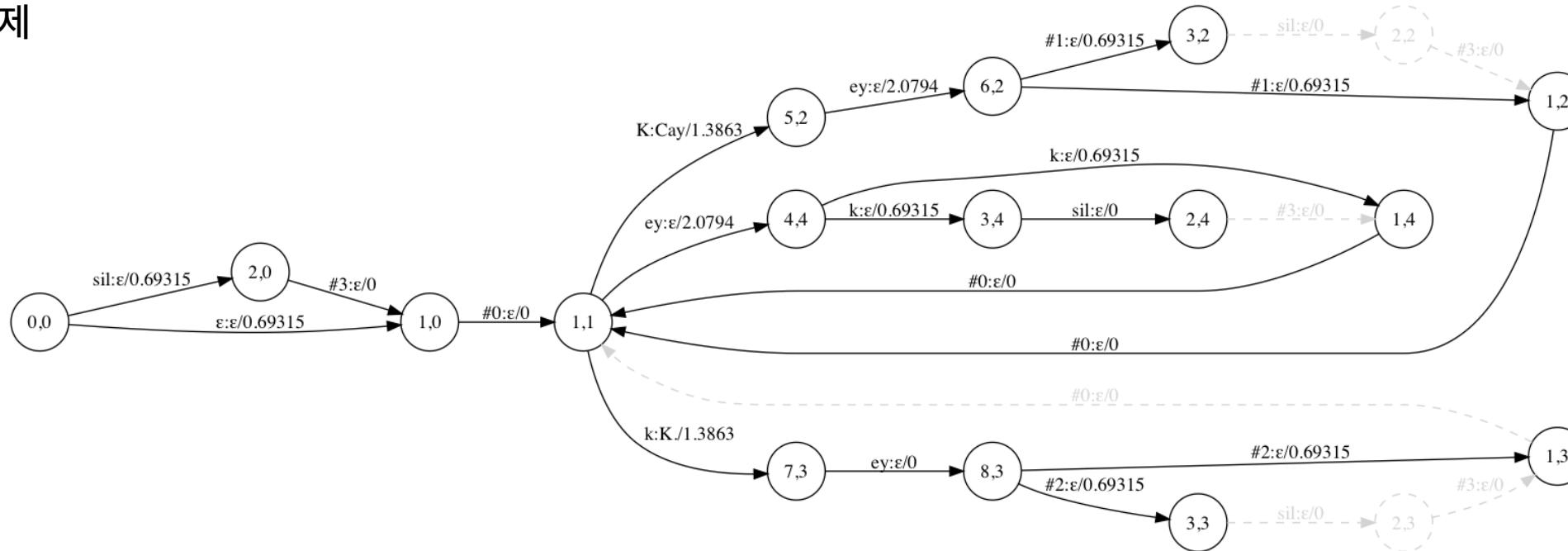
■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

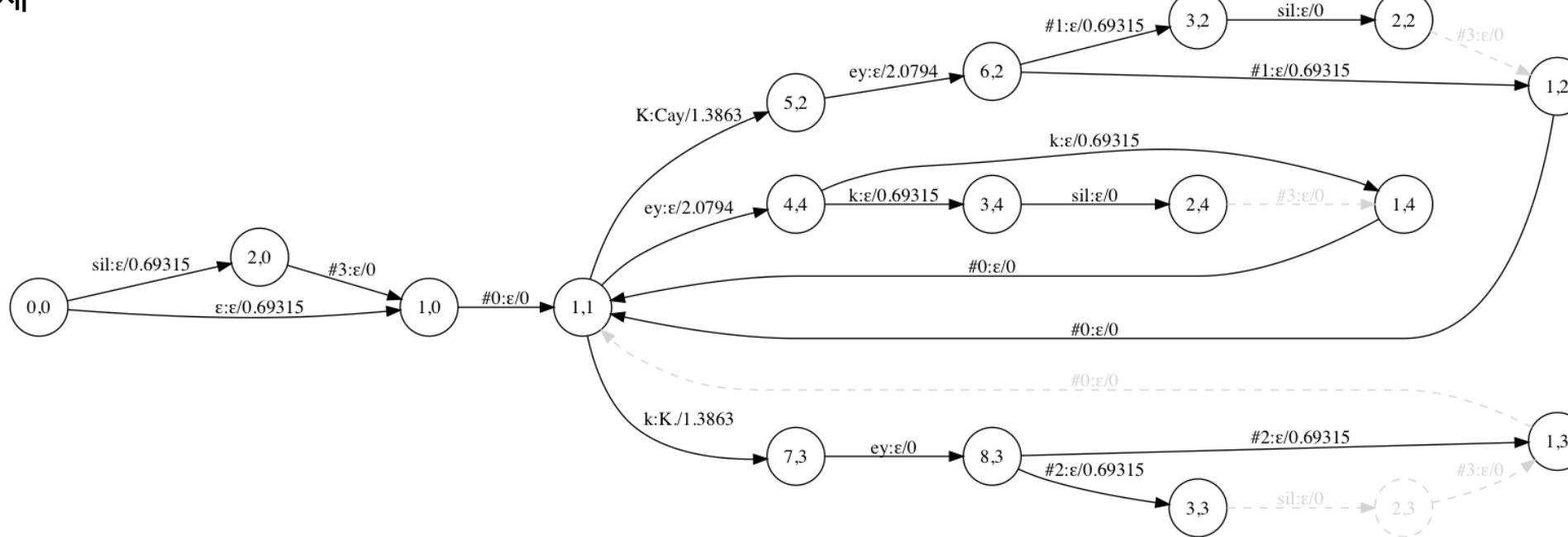
- $\min(\det(L \circ G))$ 생성 방법
 - Composition 알고리즘
 - 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘
 - 예제

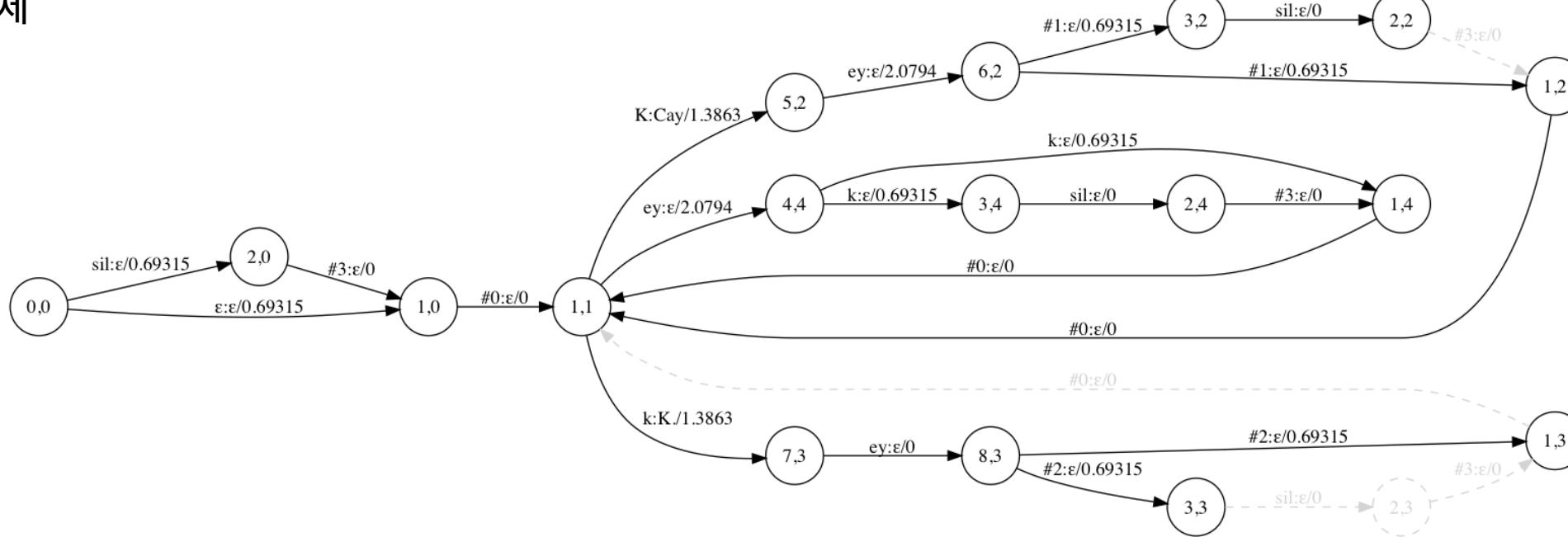


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘

- 예제

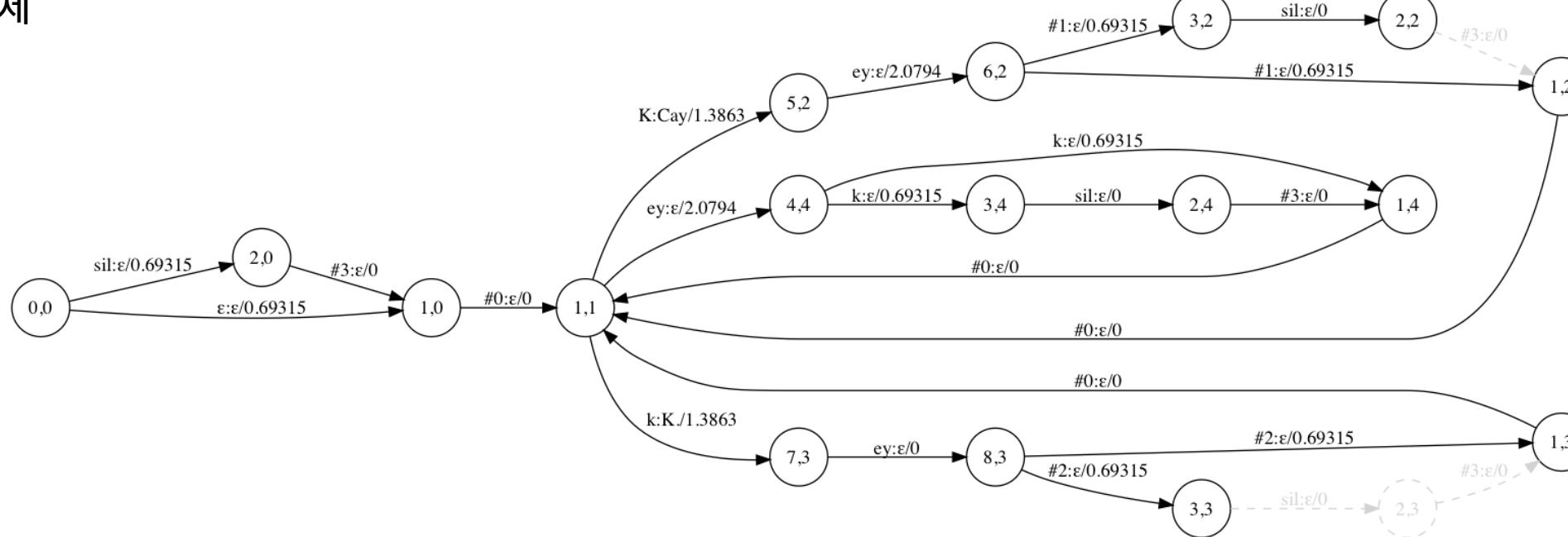


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘

- 예제

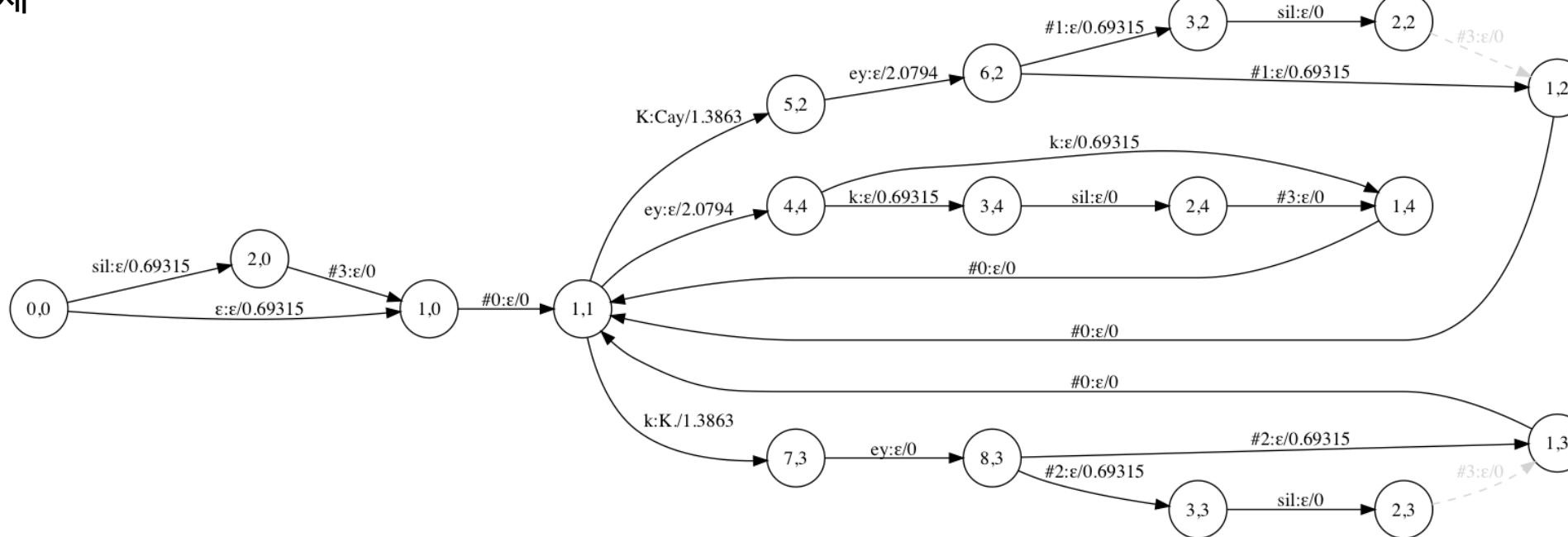


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

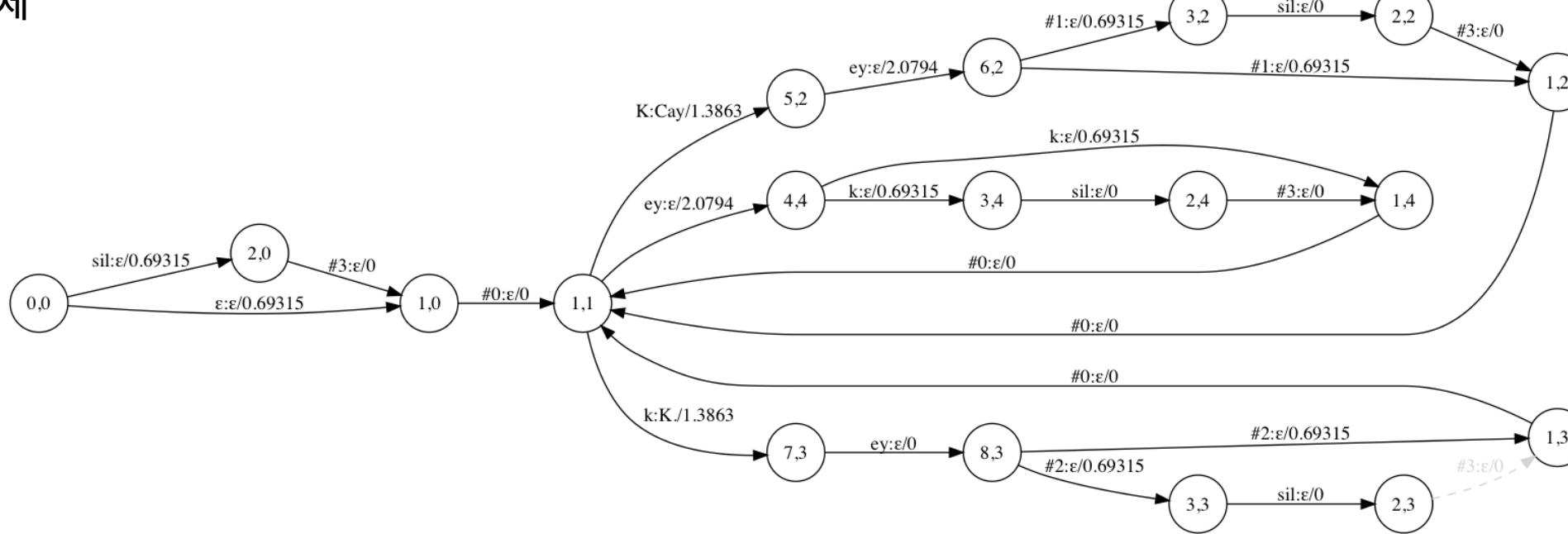
- Composition 알고리즘

- 예제



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

- $\min(\det(L \circ G))$ 생성 방법
 - Composition 알고리즘
 - 예제



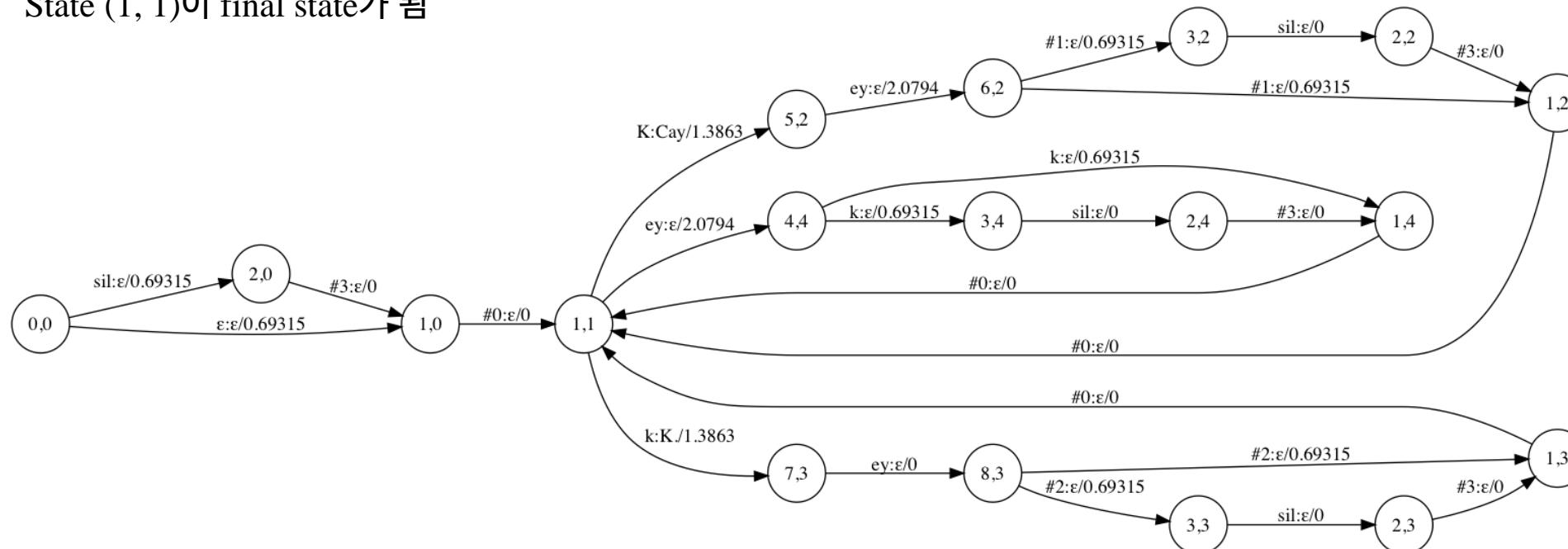
10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Composition 알고리즘

- 예제

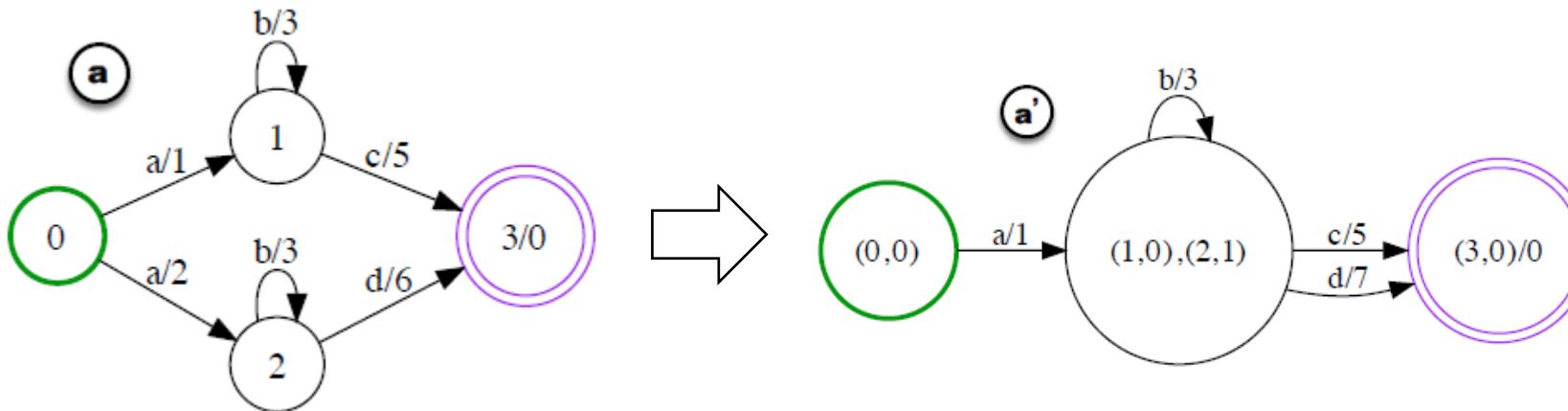
- * Composition $L \circ G$ 에 대한 WFST 결과는 아래와 같음
 - State $(0, 0)$ 이 initial state가 됨
 - State $(1, 1)$ 이 final state가 됨



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘
 - Eliminates ambiguity in the input paths
 - * Improves efficiency of downstream operations such as shortest-path
 - * Each state has at most one outgoing transition containing any particular input label



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- Pseudo-code

WEIGHTED-DETERMINIZATION(A)

```
1    $i' \leftarrow \{(i, \lambda(i)) : i \in I\}$ 
2    $\lambda'(i') \leftarrow \bar{1}$ 
3    $Q \leftarrow \{i'\}$ 
4   while  $Q \neq \emptyset$  do
5      $p \leftarrow \text{HEAD}(Q)$  Not the same!!
6     DEQUEUE( $Q$ )
7     for each  $x \in i[E[Q[p']]$  do
8        $w' \leftarrow \bigoplus\{v \otimes w : (p, v) \in p', (p, x, w, q) \in E\}$ 
9        $q' \leftarrow \{(q, \bigoplus\{w'^{-1} \otimes (v \otimes w) : (p, v) \in p', (p, x, w, q) \in E\}) :$ 
           $q = n[e], i[e] = x, e \in E[Q[p']]\}$ 
10       $E' \leftarrow E' \cup \{(p', x, w', q')\}$ 
11      if  $q' \notin Q'$  then
12         $Q' \leftarrow Q' \cup \{q'\}$ 
13        if  $Q[q'] \cap F \neq \emptyset$  then
14           $F' \leftarrow F' \cup \{q'\}$ 
15         $\rho'(q') \leftarrow \bigoplus\{v \otimes \rho(q) : (q, v) \in q', q \in F\}$  Compute new final weight, if necessary
16        ENQUEUE( $Q, q'$ )
17    return  $T'$ 
```

Initialize the queue and new WFST

Compute new arc weight

Compute new states and residual weights

Compute new final weight, if necessary

Algorithm complexity

Time	exponential
Space	exponential

<http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf>

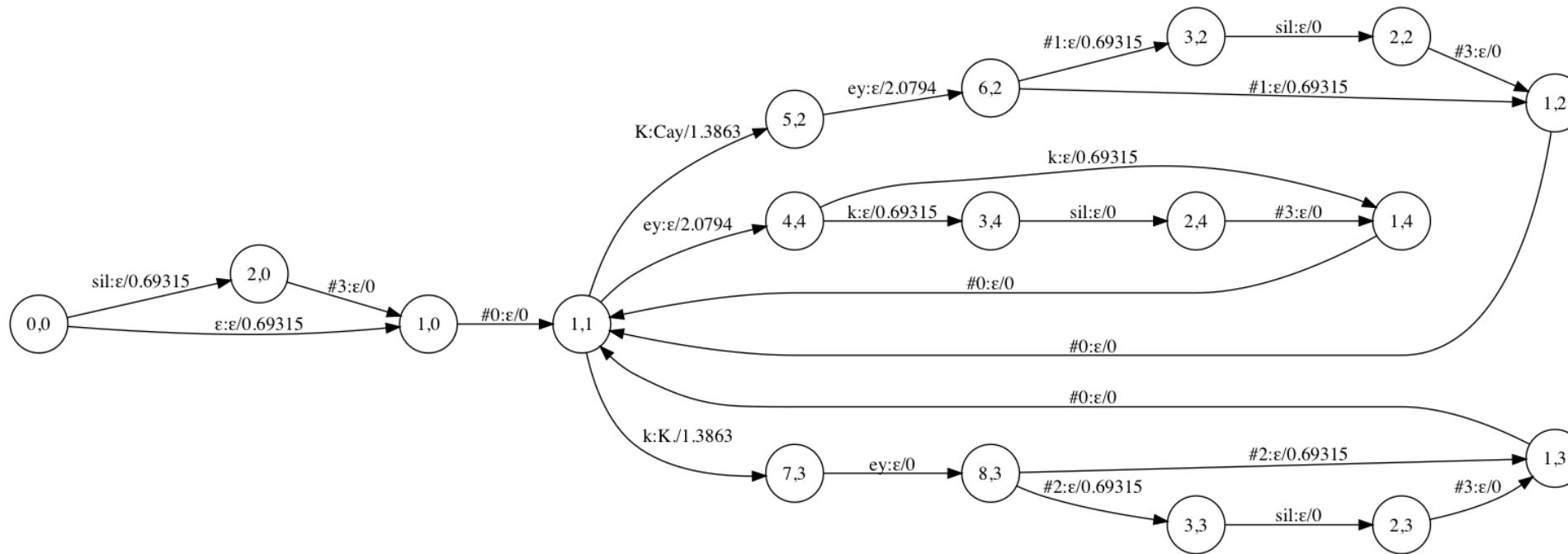
10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

- * 아래 $L \circ G$ 에 대한 WFST 결과를 이용하여 det (determinization) 수행



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

(Step 5)

$\oplus p' \leftarrow \text{HEAD}(S)$

$\Rightarrow p' \in \begin{cases} \text{state} : \epsilon/0.69315 \\ \text{sil} : \bar{\epsilon}/0.69315 \end{cases}$

② $\{ \text{edge } q' \}$ ~~제거~~ \Rightarrow

$\Rightarrow p'$ 은 ~~state~~ \Rightarrow state가 있음

input label은 sil과 ϵ .

(2-i) label = sil 일 때
input label이 sil인 모든 edge를 고려함

$w' \leftarrow \oplus \{ (U \otimes W) : (P, V) \in P', (P, X, W, Q) \in E \}$ 계산

$(P, V) \in (0, \bar{I}) = (0, 0)$ ~~는~~ 있음.

$(P, X, W, Q) \in (0, \text{sil}, \bar{I}, 2)$

$\therefore w' = \oplus \{ (\bar{I} \otimes 0.69315) \}$

$= \min((0 + 0.69315))$

$= 0.69315$

③ $\{ \text{edge } q' \}$ ~~제거~~ \Rightarrow

queue S

$(0, \bar{I}) \in (0, 0)$

$q' \leftarrow \{ (q, \oplus \{ W' \otimes (V \otimes W) : (P, V) \in P', (P, X, W, Q) \in E \}) \}$
 $: q = \text{node}, \text{edge} = \epsilon, e \in E \subseteq P'$ \Rightarrow 계산

$\therefore q' = \{ (2, \oplus \{ -0.69315 + (\bar{I} \otimes 0.69315) \ }) \}$

$= \{ (2, \min(-0.69315 + 0.69315)) \}$

$= \{ (2, 0) \}$

최종 $\Rightarrow \{ (0, 0), \text{sil}, 0.69315, \{ (2, 0) \} \}$

Create queue S' Queue S와 Q' 에 $(2, 0)$ 저장

S $\boxed{(2, 0)}$ $\boxed{(2, 0)}$ Q' $\boxed{(0, 0)}$ $\boxed{(2, 0)}$

(2-ii) label = ϵ 일 때 input label이 ϵ 인 모든 edge를 고려함

w' 계산 $\Rightarrow (P, V) \in (0, 0)$ 이므로, $(P, X, W, Q) \in (0, \epsilon, \bar{I}, 3)$

$\therefore w' = \oplus \{ (\bar{I} \otimes 0.69315) \}$

$= 0.69315$

$\therefore q' = \{ (3, \oplus \{ -0.69315 + (\bar{I} \otimes 0.69315) \ }) \}$

$= \{ (3, 0) \}$

최종 $\Rightarrow \{ (0, 0), \epsilon, 0.69315, \{ (3, 0) \} \}$

\Rightarrow \boxed{S} $\boxed{Q'}$

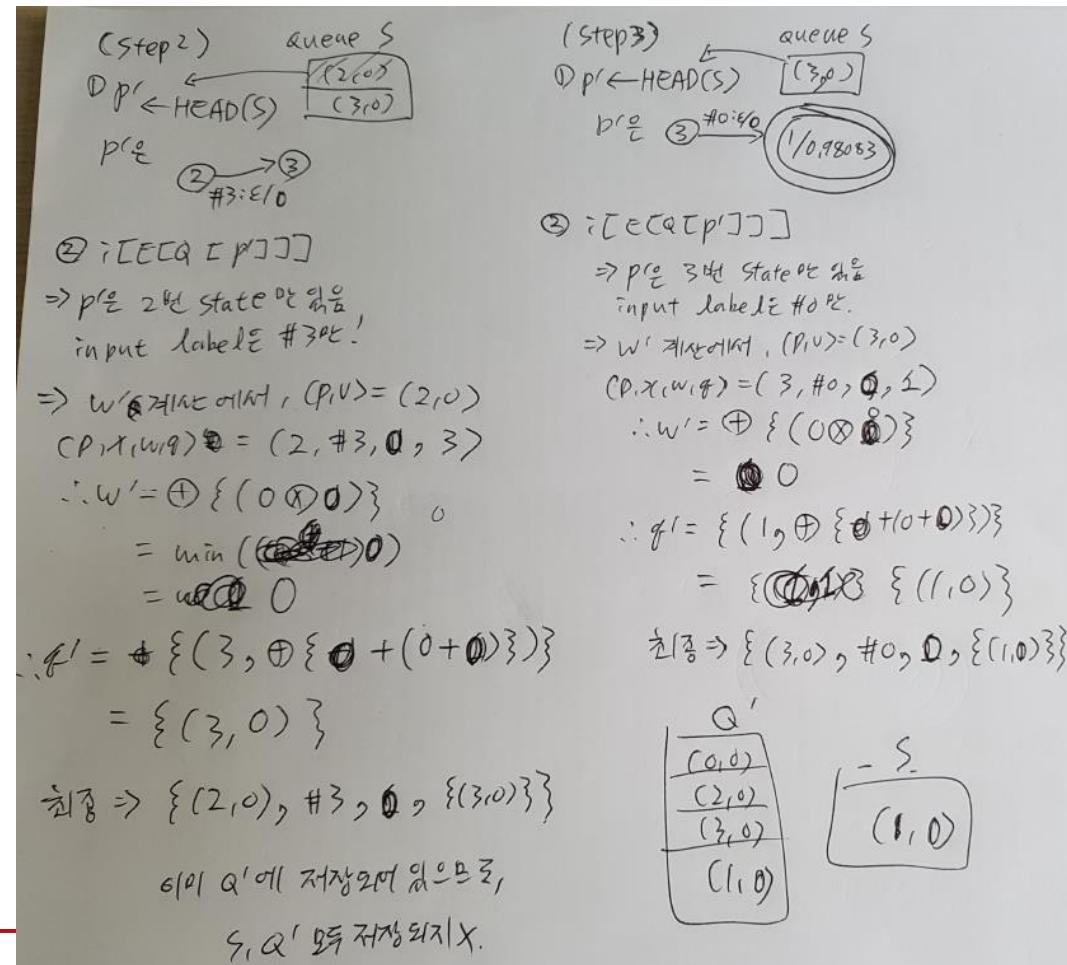
$\boxed{(2, 0)}$ $\boxed{(2, 0)}$ $\boxed{(3, 0)}$ $\boxed{(3, 0)}$

10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

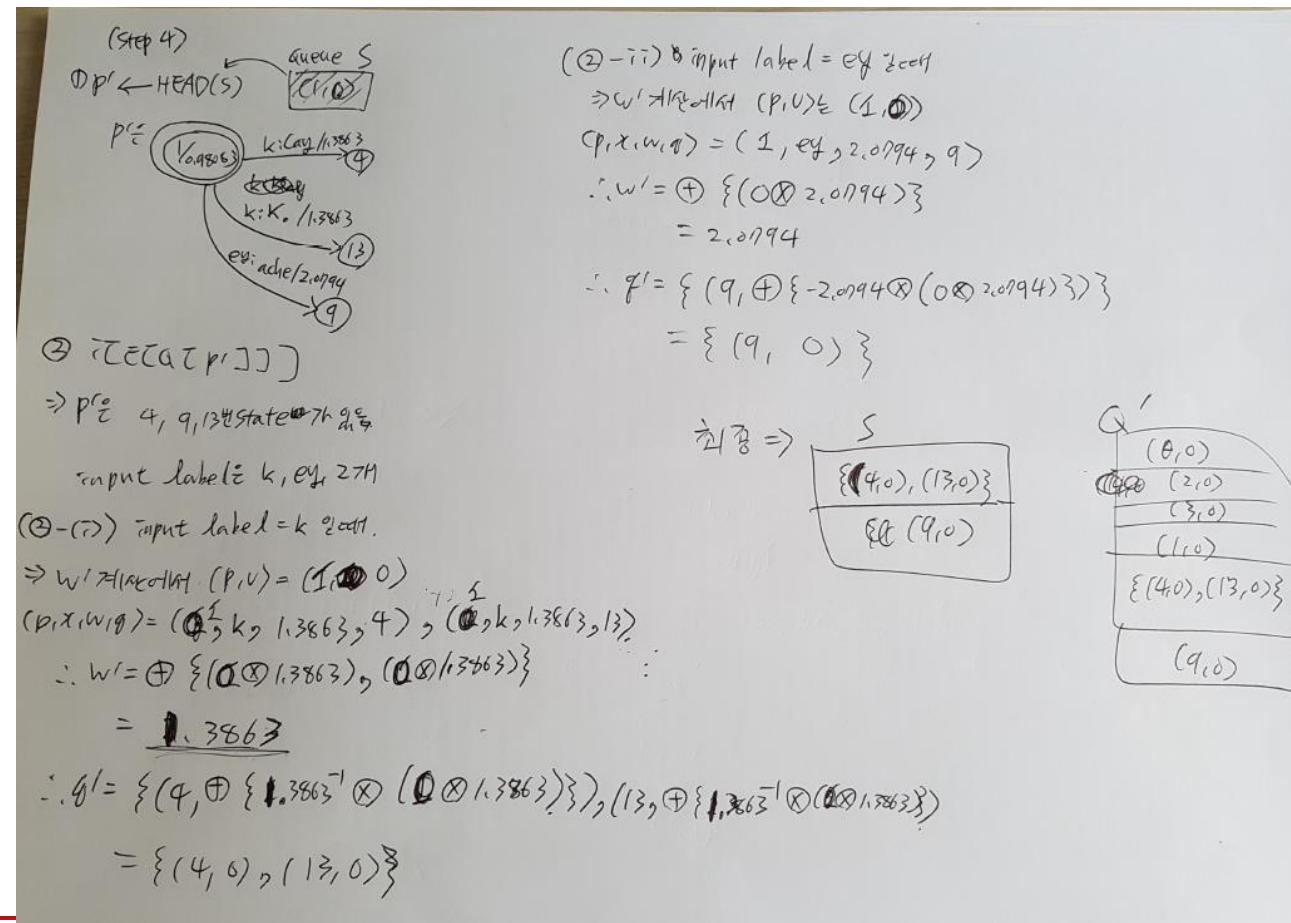


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

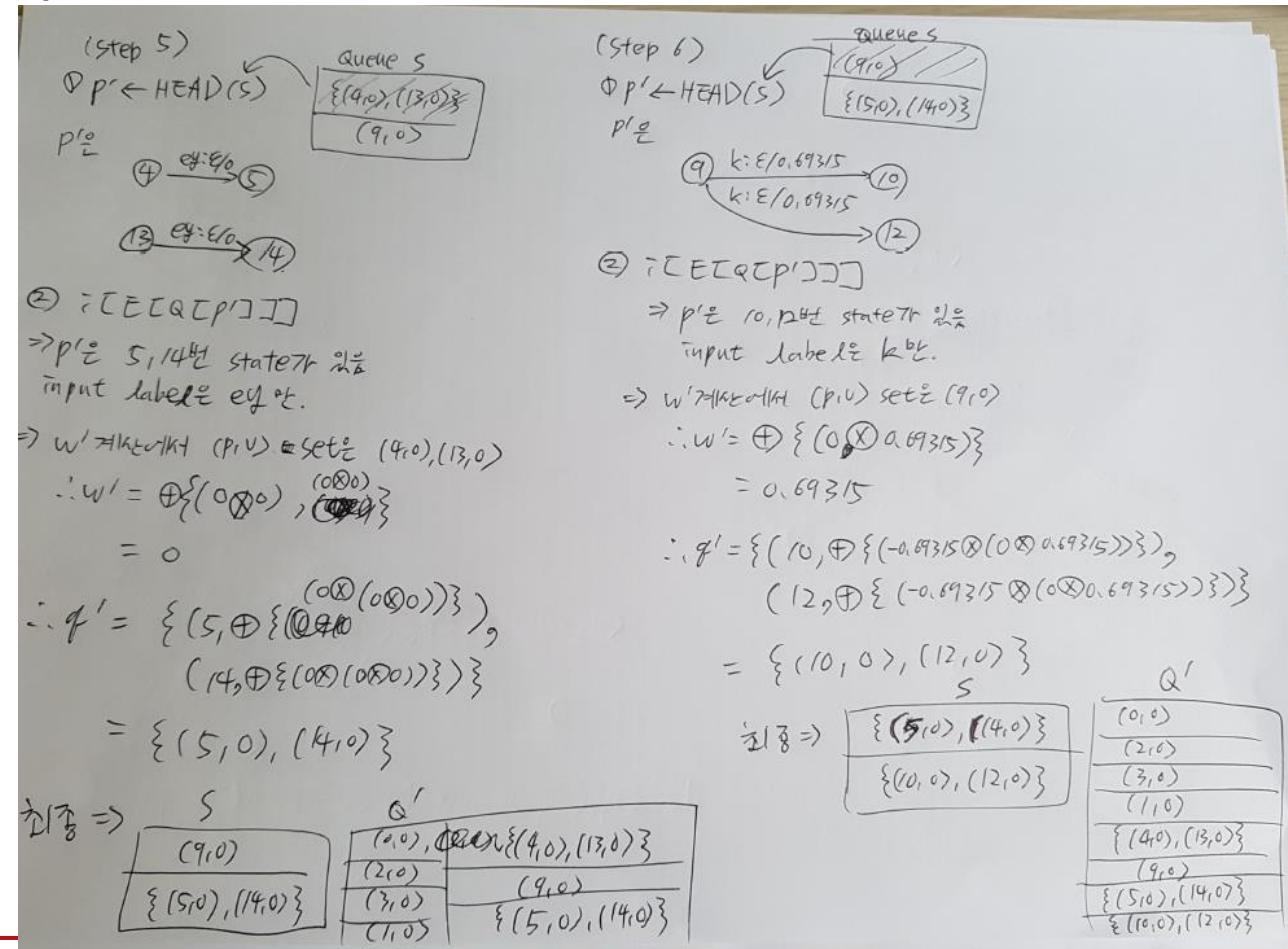


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

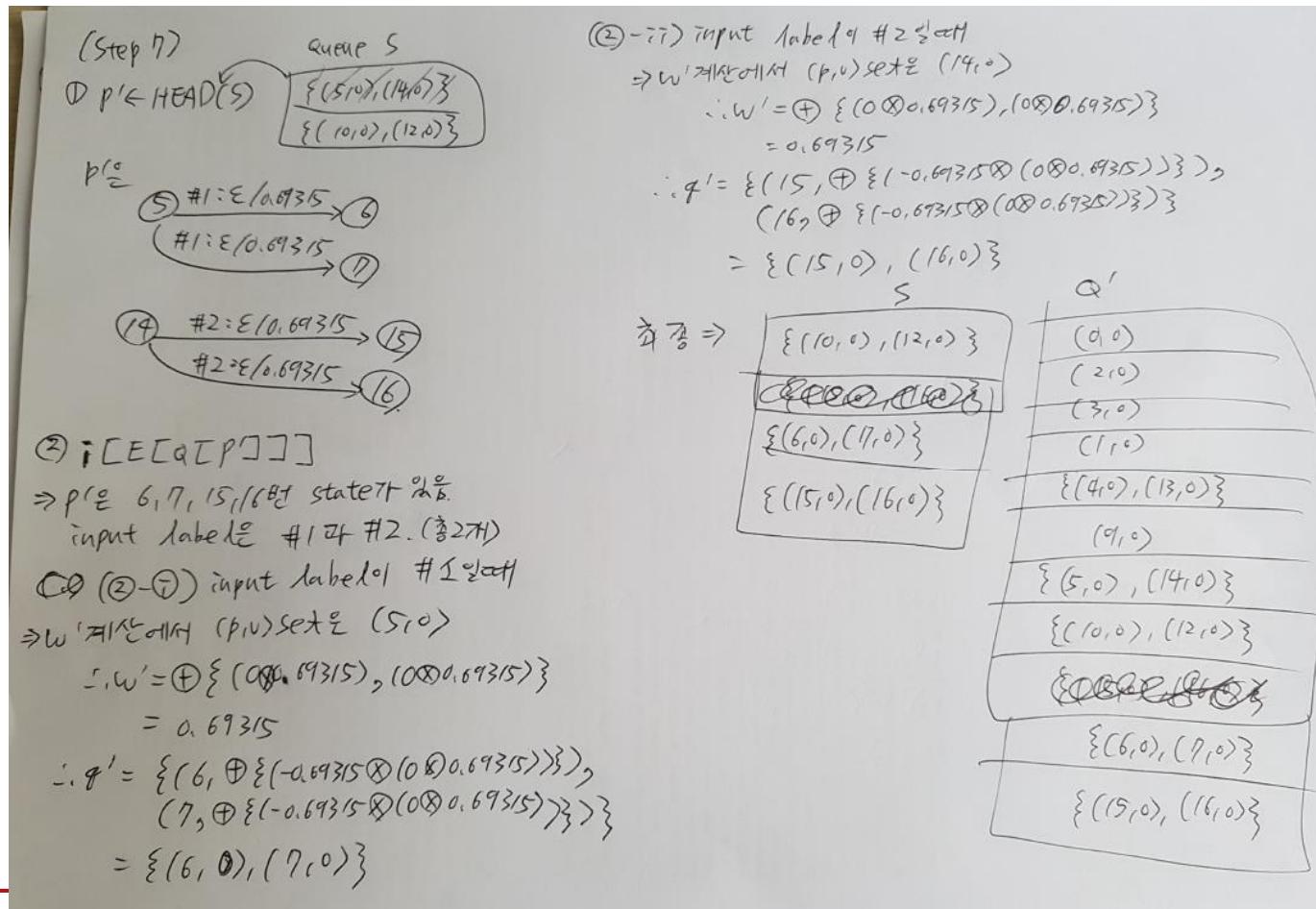


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

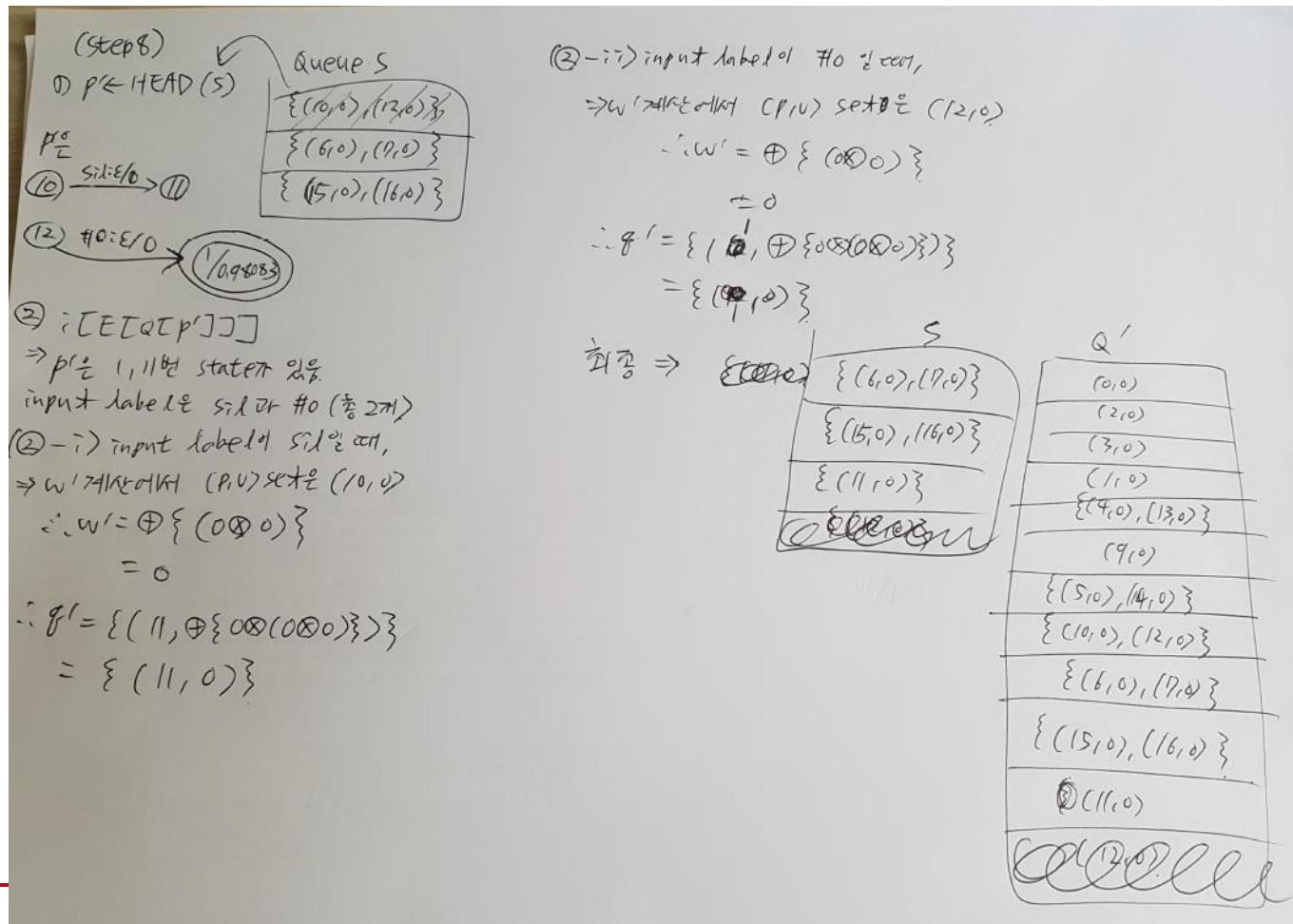


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

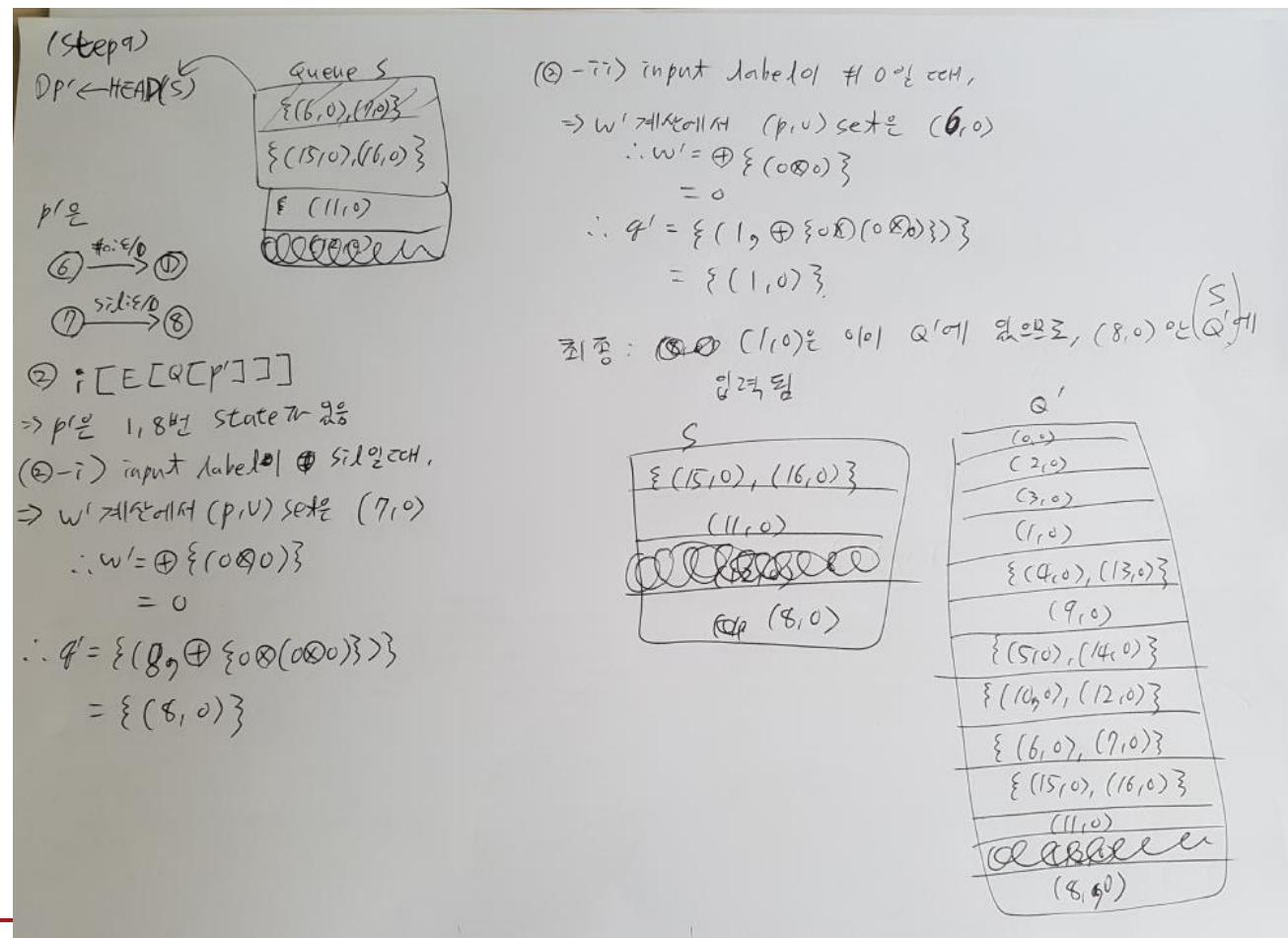


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

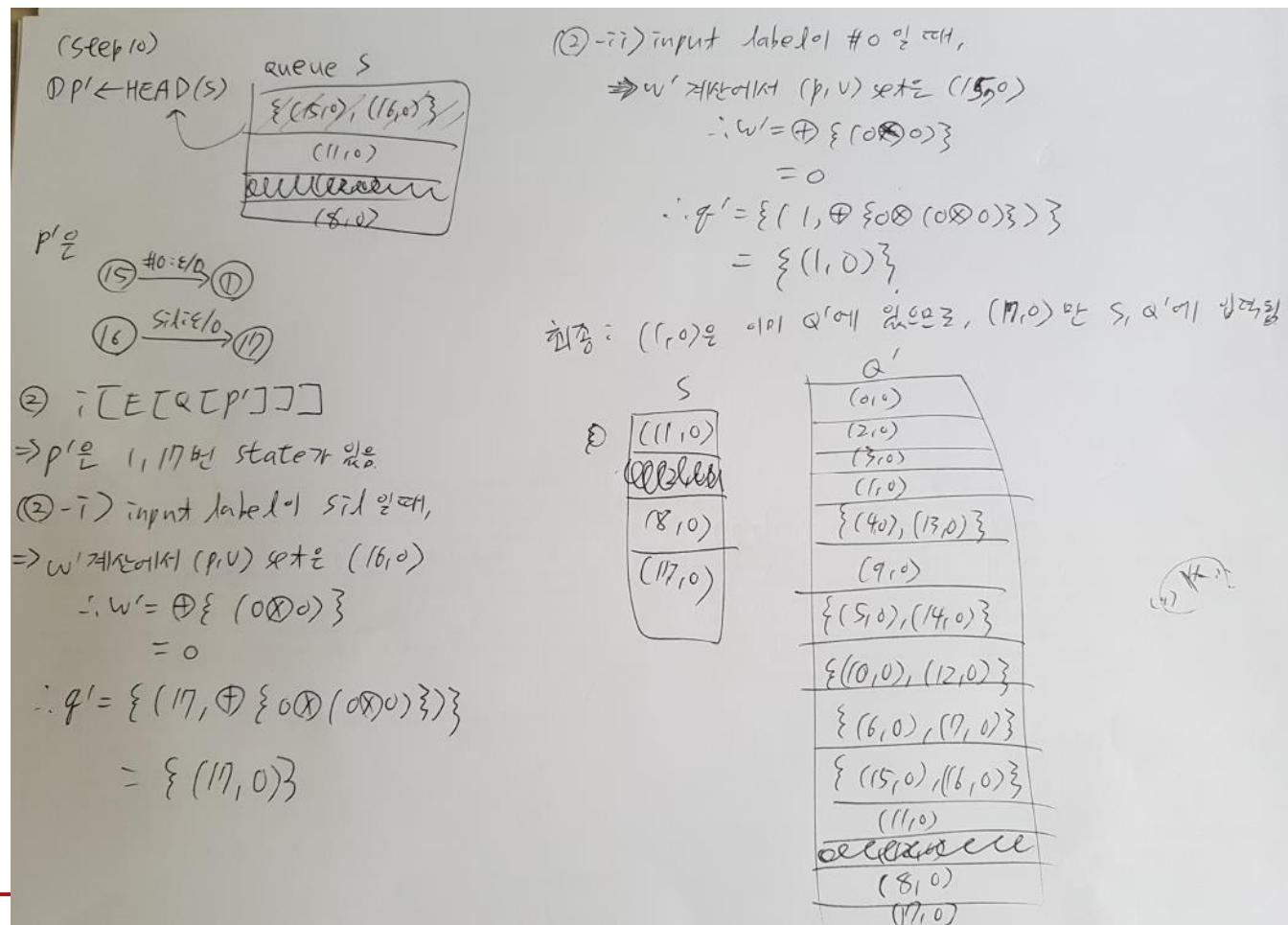


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

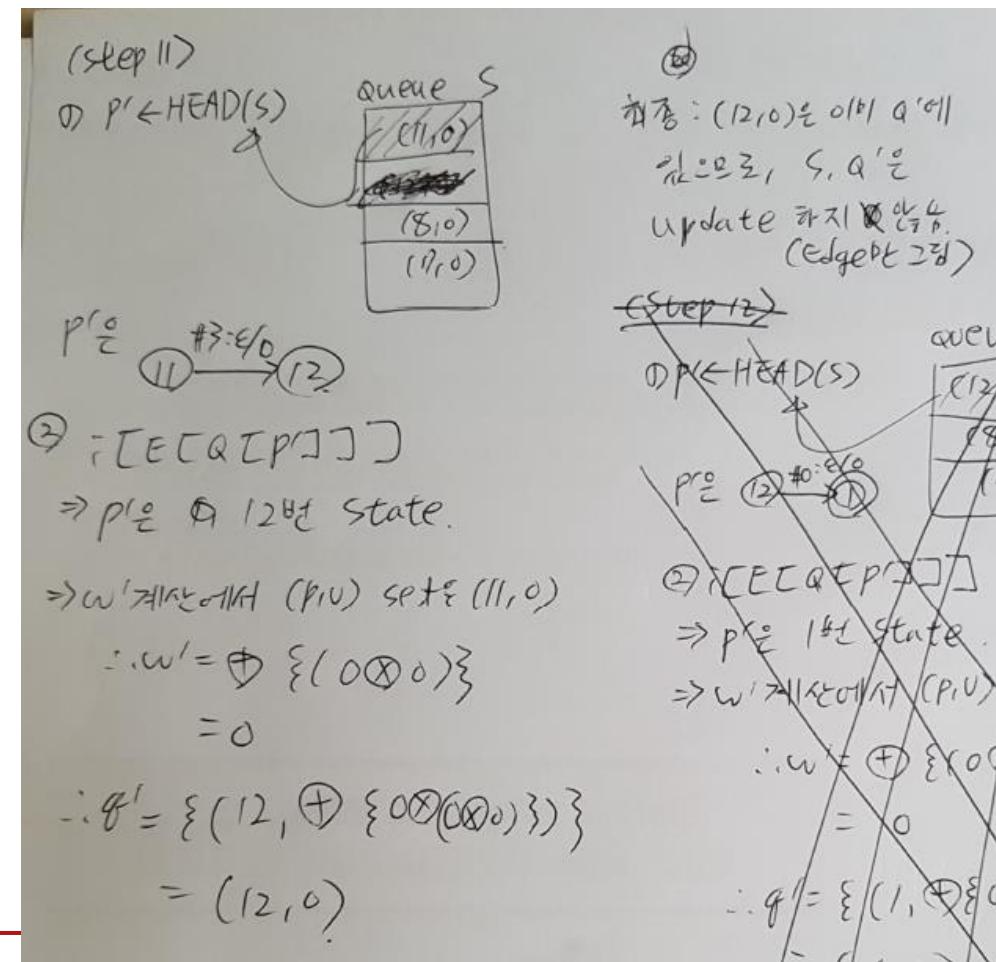


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

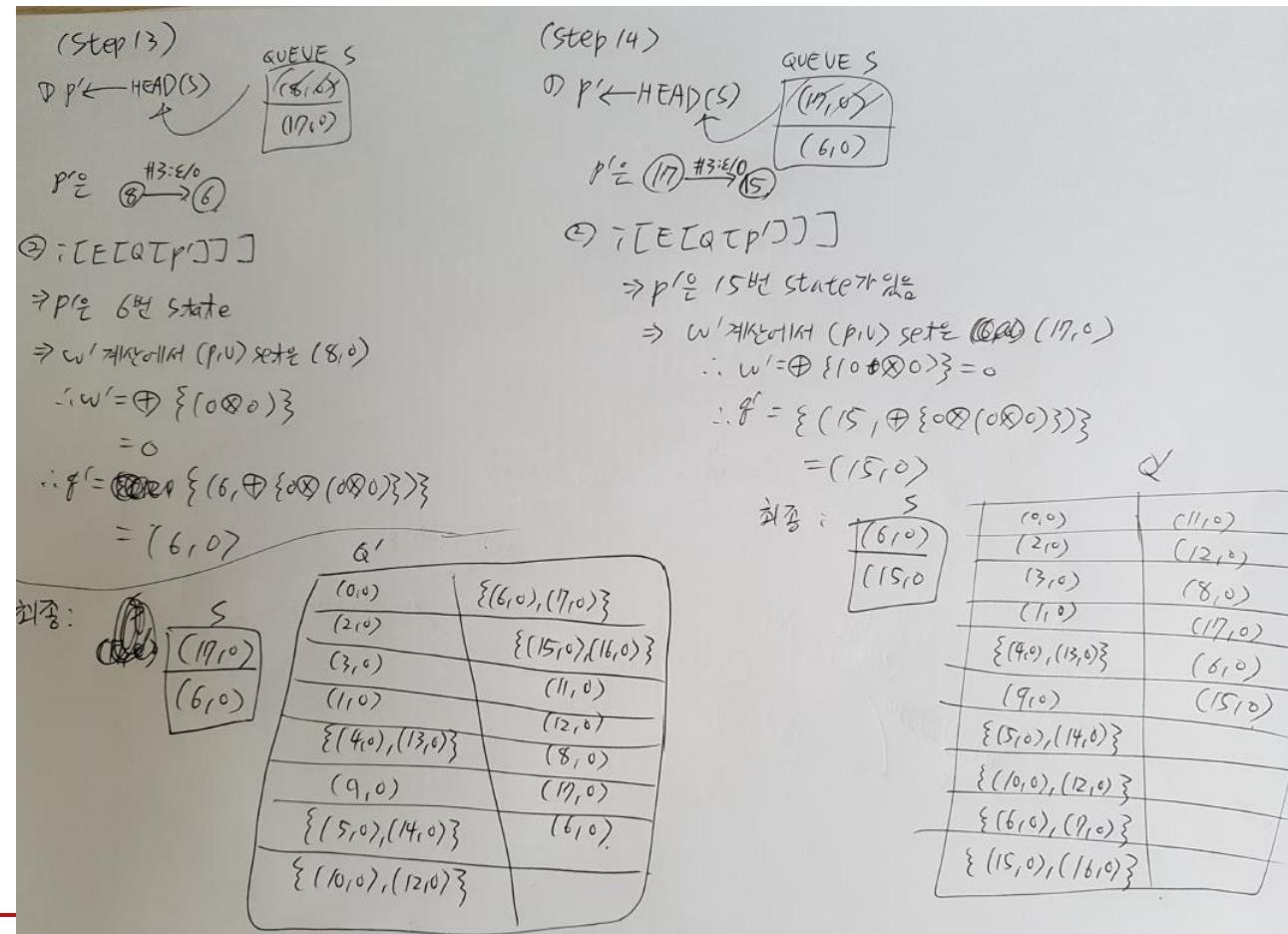


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

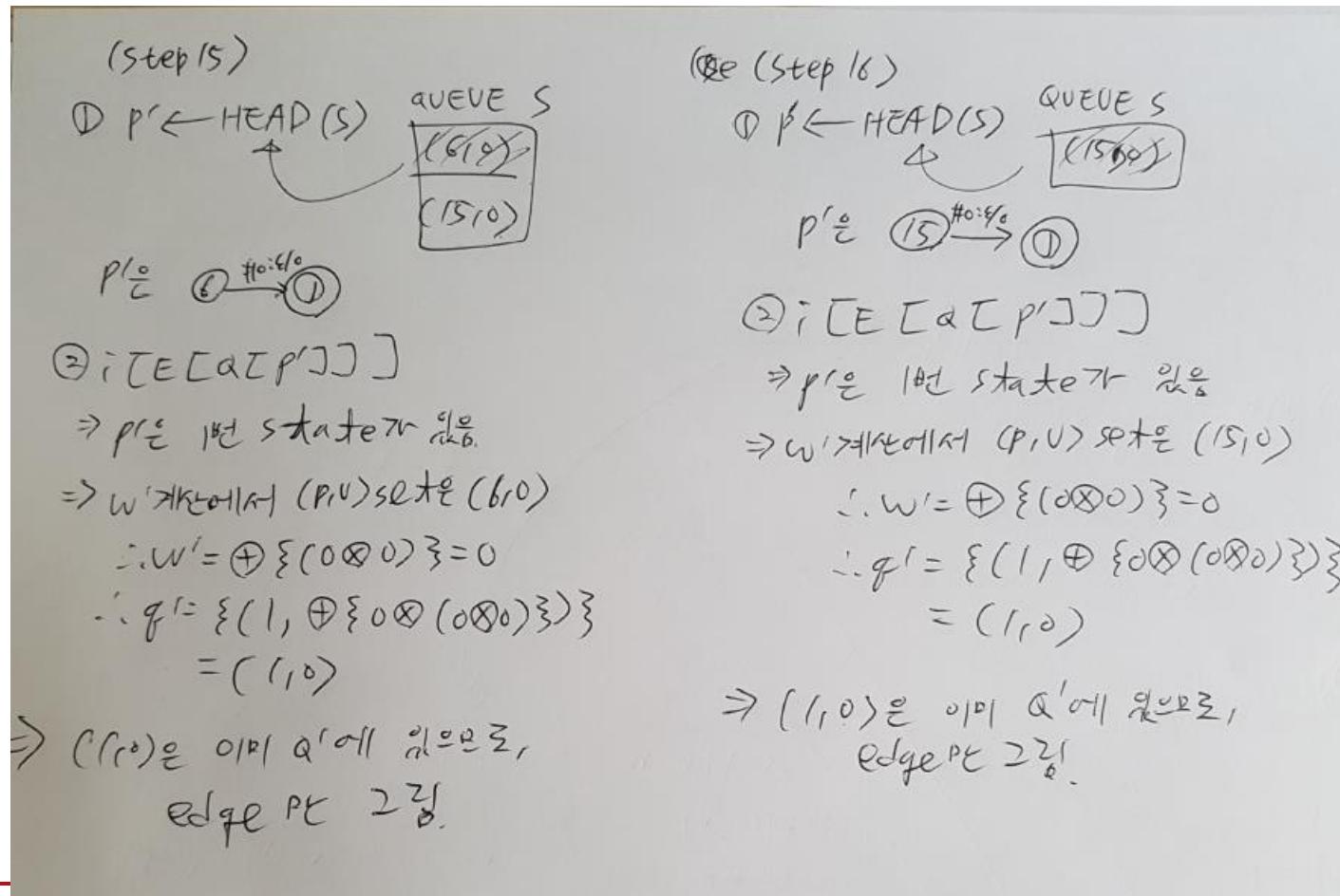


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Determinization 알고리즘

- 예제

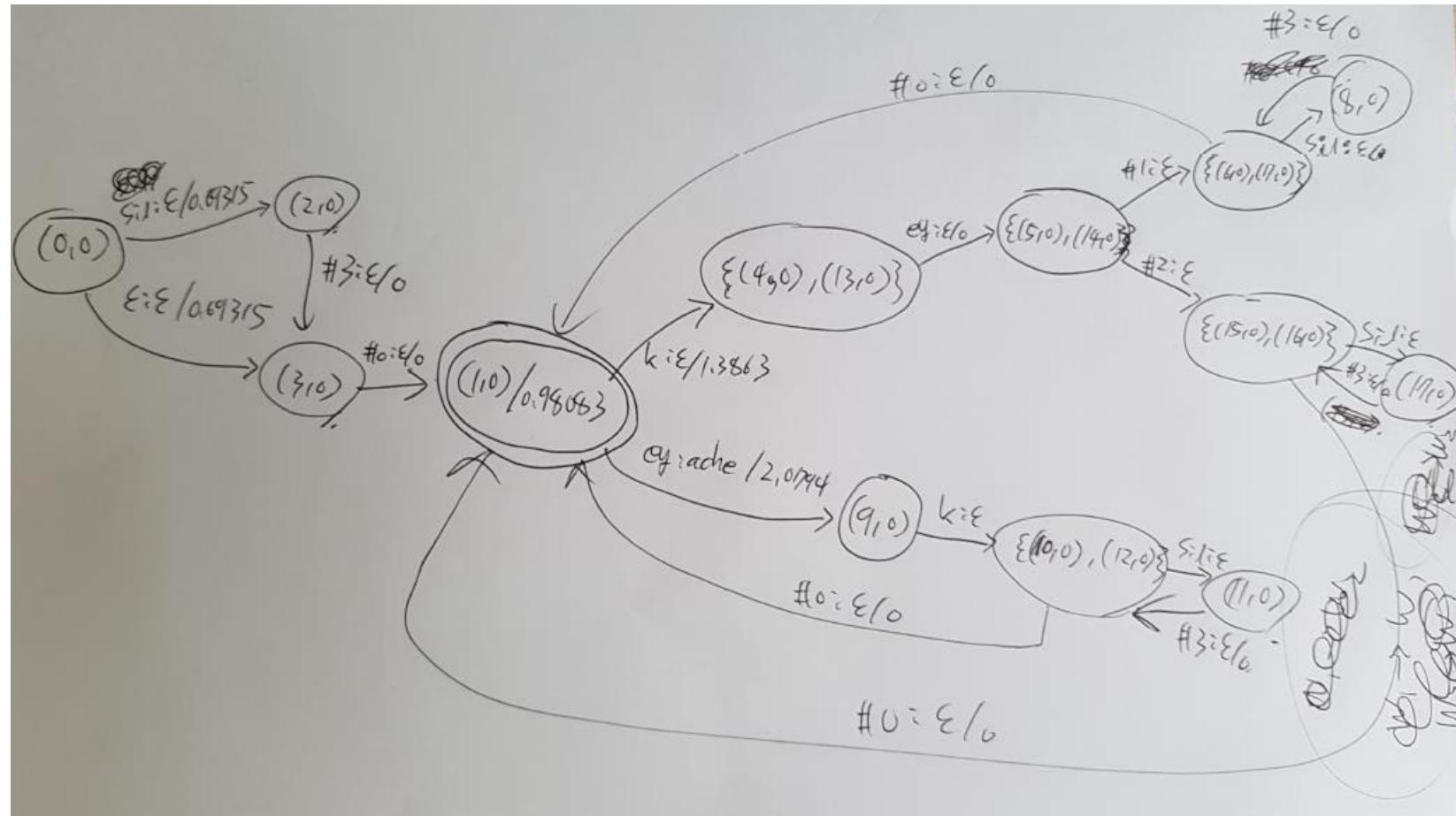


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

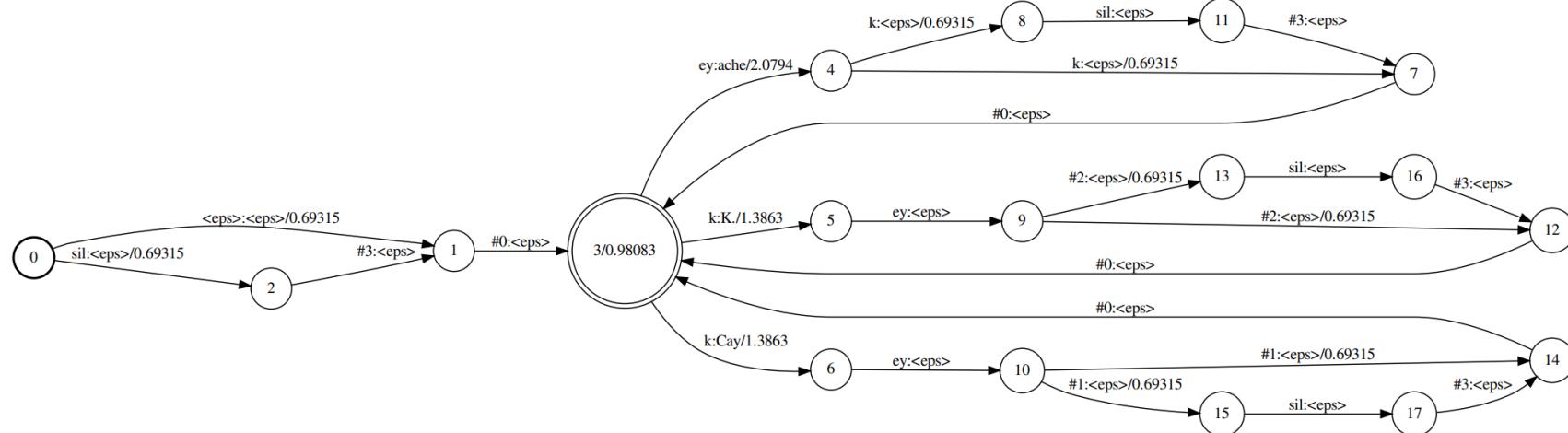
- Determinization 알고리즘

- 예제



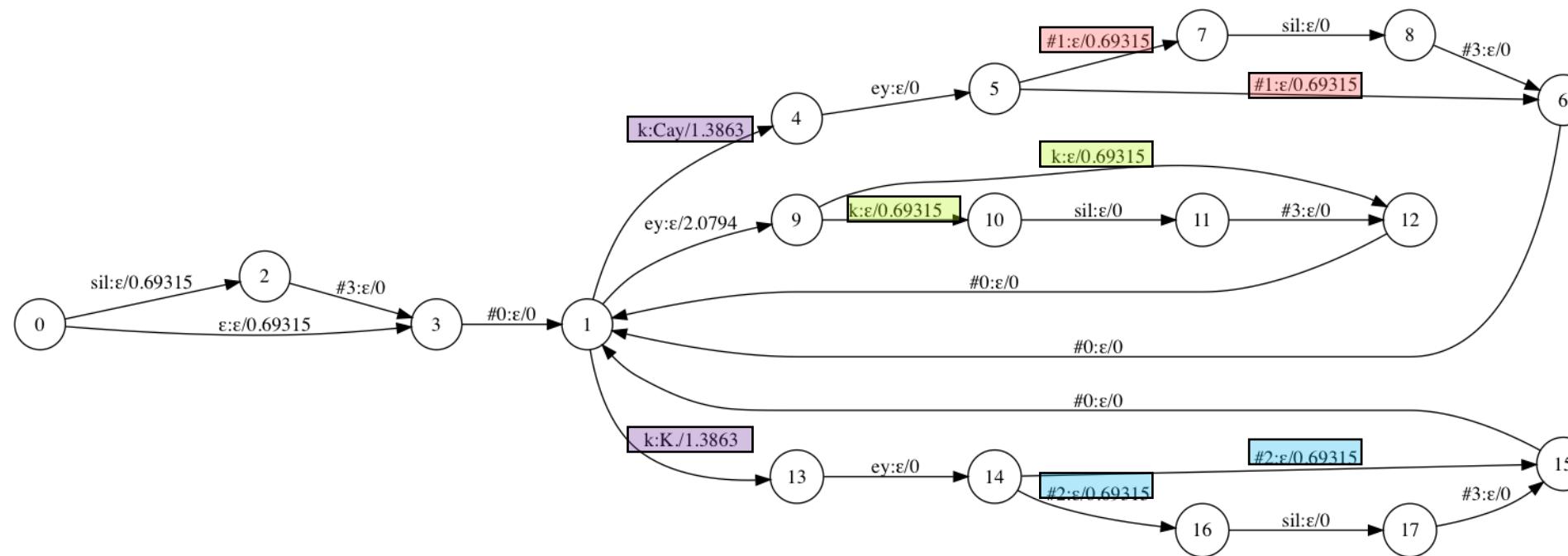
10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

- result of $L \circ G$



10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

- result of $\det(L \circ G)$

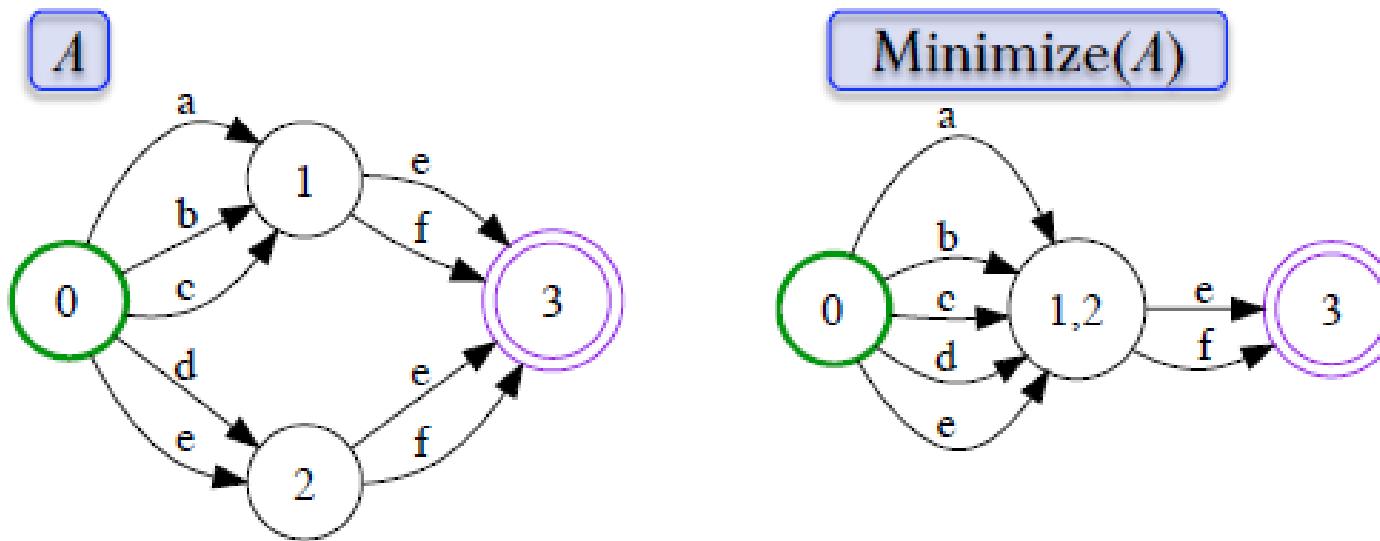


10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

- $\min(\det(L \circ G))$ 생성 방법

- Minimization 알고리즘

- Given an input WFST, produces a ‘minimal’ version which is guaranteed to have the smallest possible number of states while preserving the input language and weight/path properties of the original



<http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf>

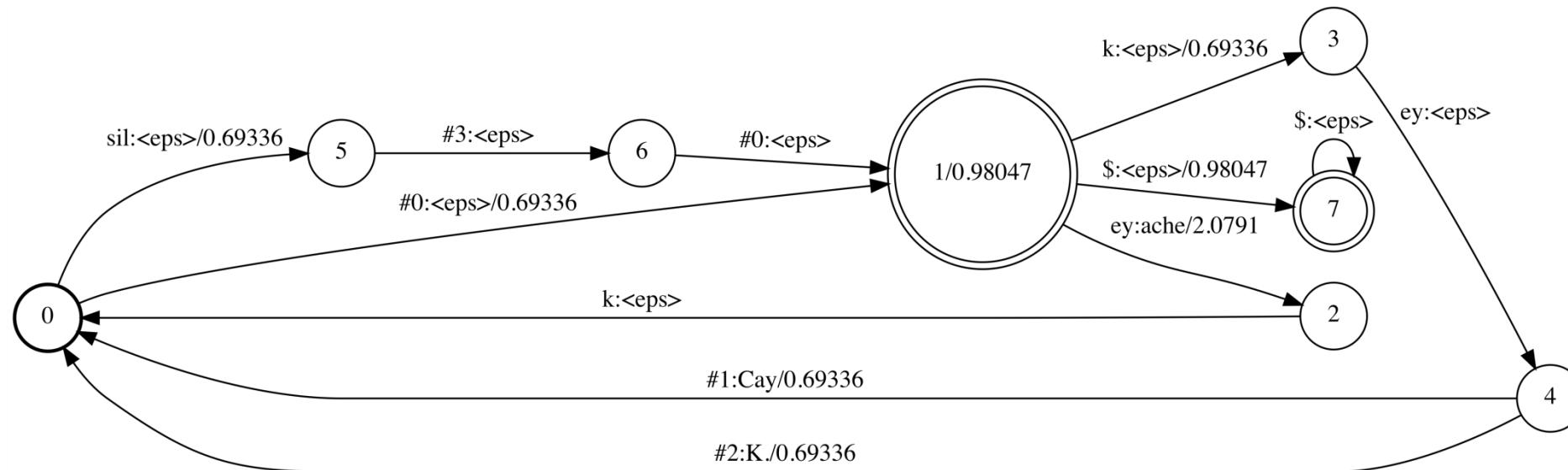
10.2.3 WFST 기반 효율적 디코딩에 필요한 연산

■ $\min(\det(L \circ G))$ 생성 방법

- Minimization 알고리즘

- 예제

- * Minimization까지 수행했을 때, 최종 $\min(\det(L \circ G))$ 의 WFST는 아래와 같음



<http://vpanayotov.blogspot.kr/2012/06/kaldi-decoding-graph-construction.html>