Weighted Tree Automata and Transducers for Syntactic Natural Language Processing

Jonathan May
Thesis Defense
April 20, 2010

How do we view natural language?

As a string?

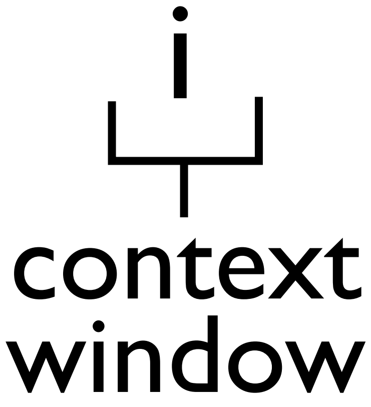
How do we view natural language?

As a string?

i

How do we view natural language?

As a string?

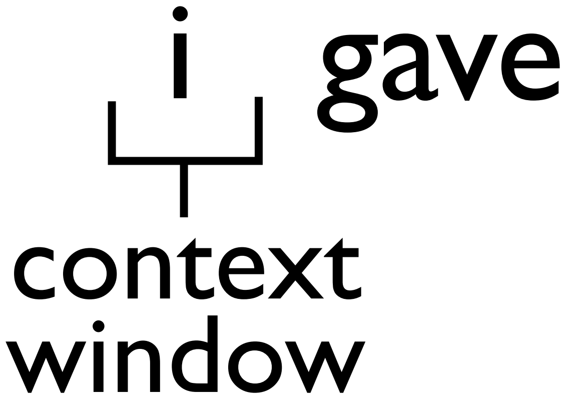


i

context window

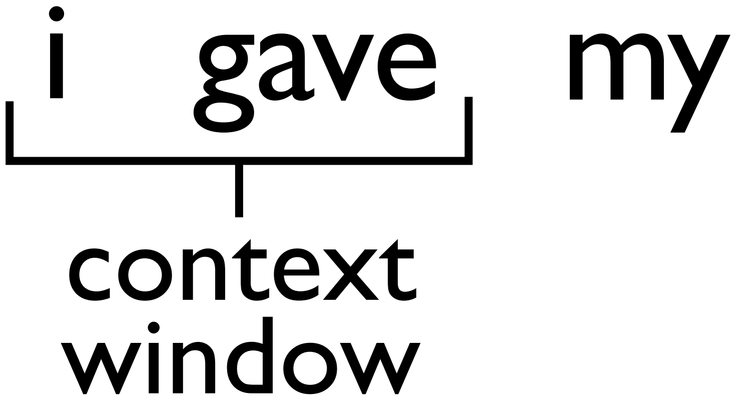
How do we view natural language?

As a string?



How do we view natural language?

As a string?

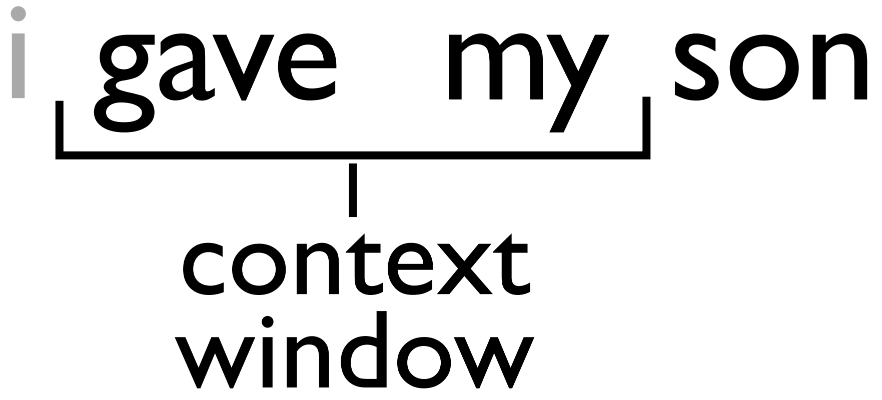


i gave my

context window

How do we view natural language?

As a string?

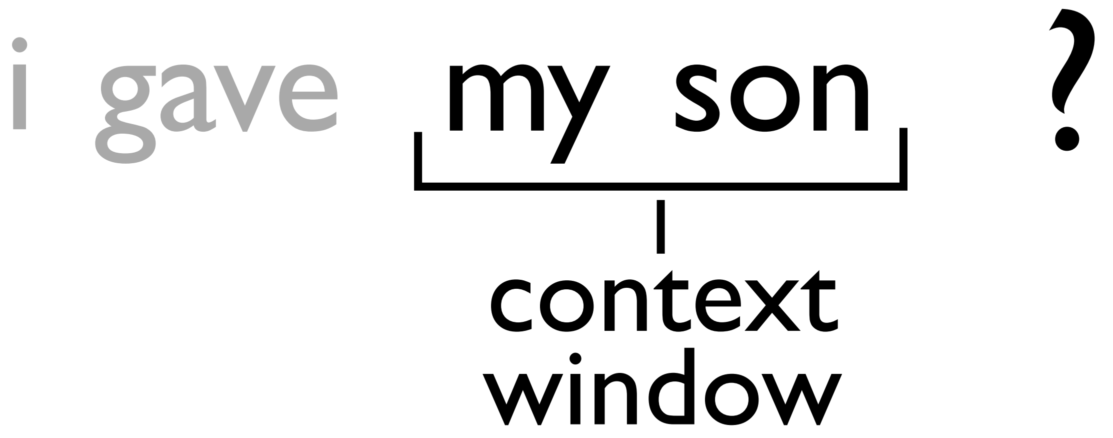


i gave my son

context window

How do we view natural language?

As a string?

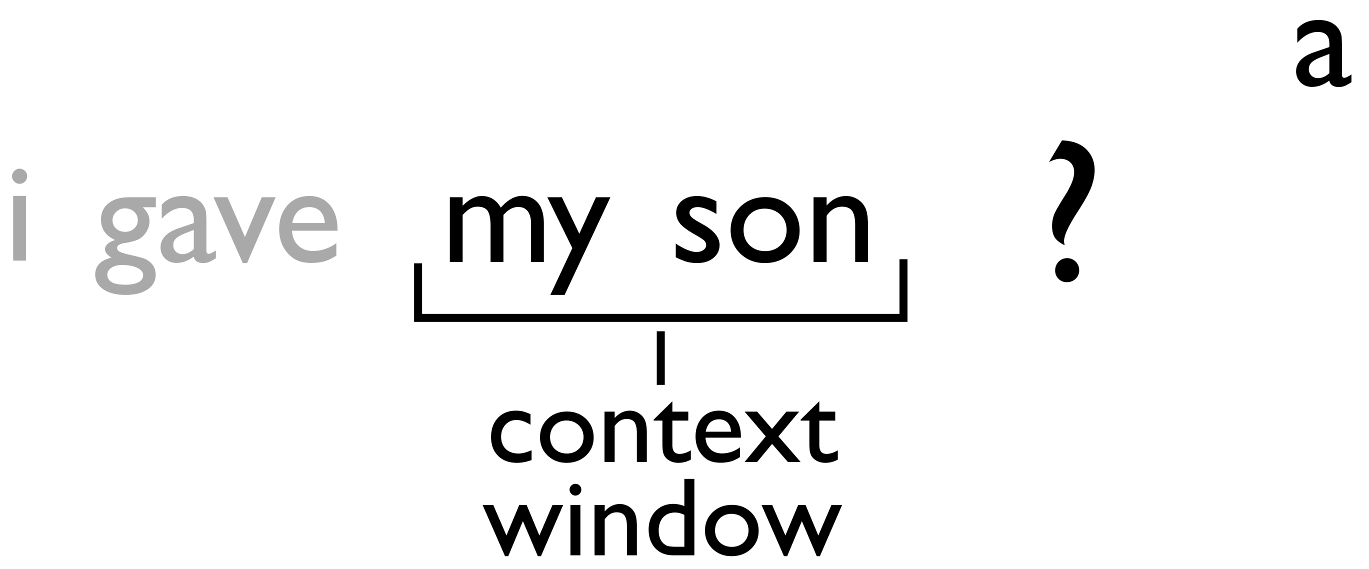


i gave my son ?

context window

How do we view natural language?

As a string?



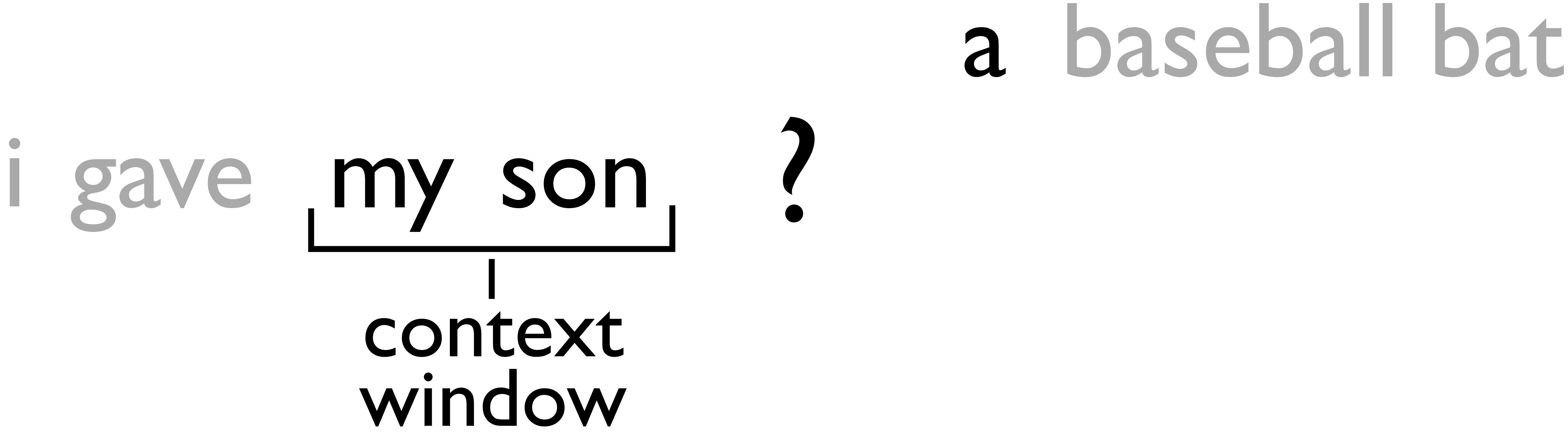
a

i gave my son ?

context window

How do we view natural language?

As a string?



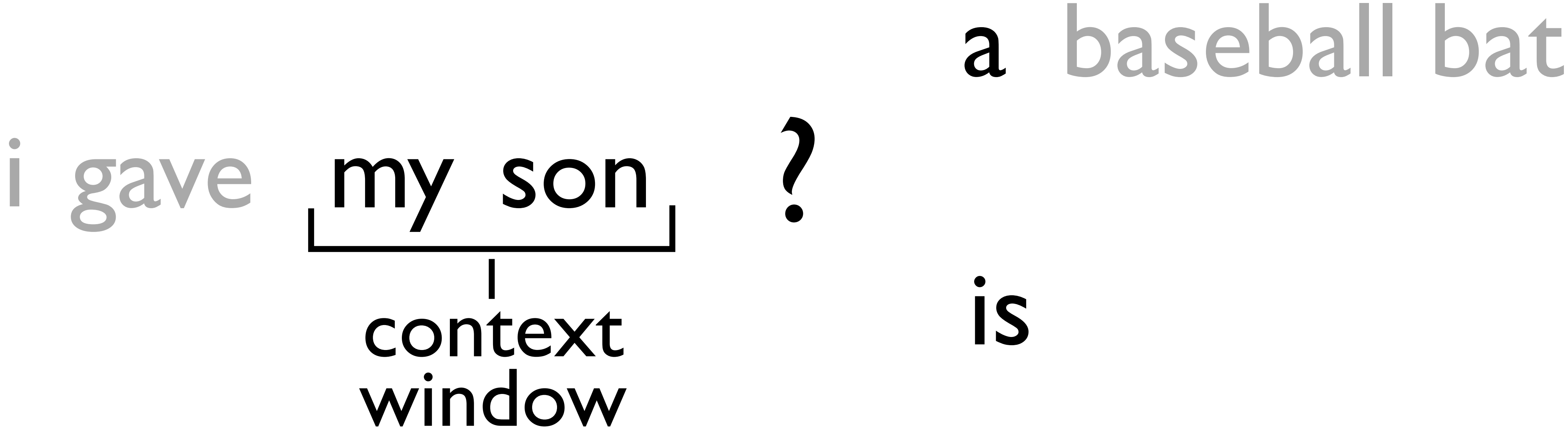
a baseball bat

i gave my son ?

context window

How do we view natural language?

As a string?



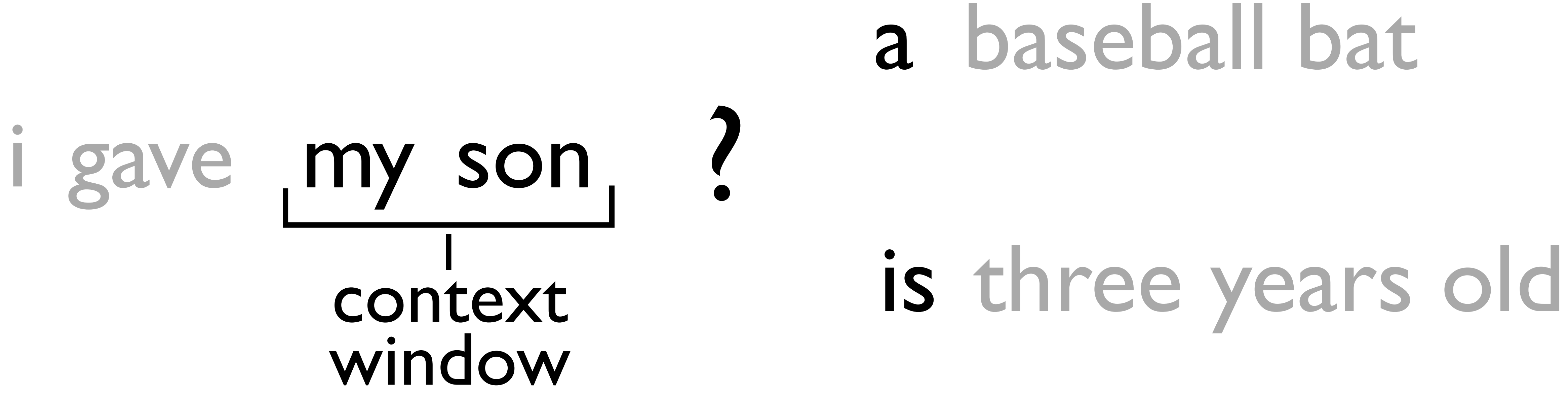
i gave my son ? a baseball bat

is

context window

How do we view natural language?

As a string?



i gave my son ?

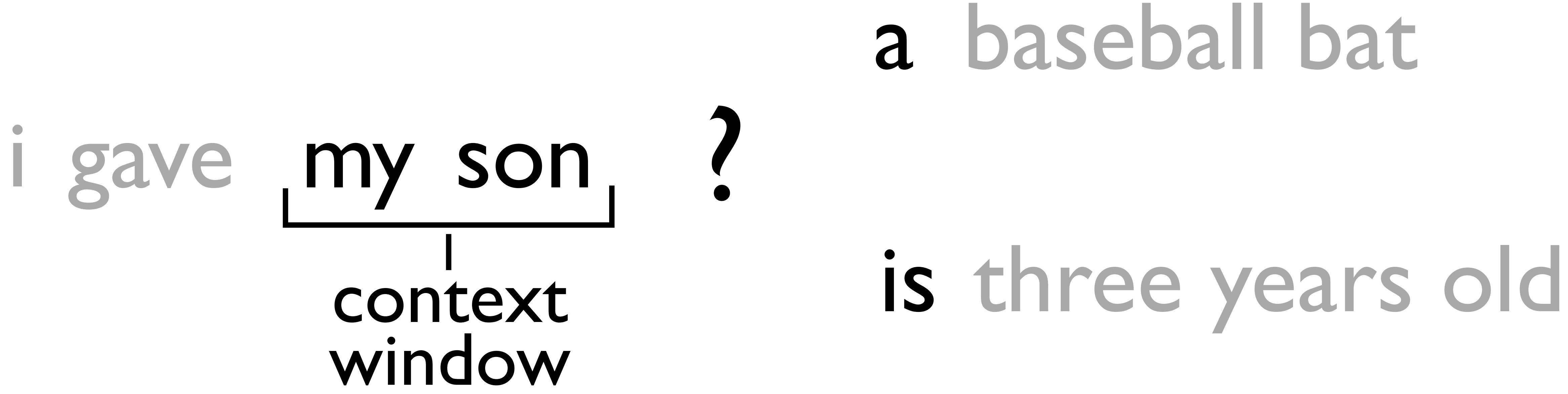
context window

a baseball bat

is three years old

How do we view natural language?

As a string?



a baseball bat

i gave my son ?

context window

is three years old

Language is more hierarchical than this!

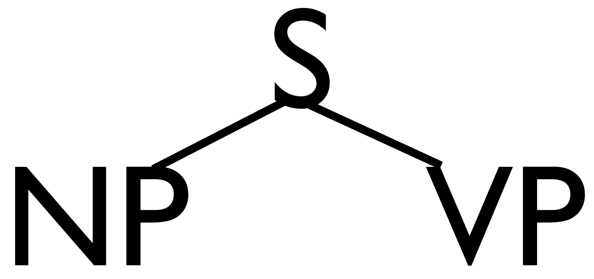
How do we view natural language?

Or as a tree?

S

How do we view natural language?

Or as a tree?

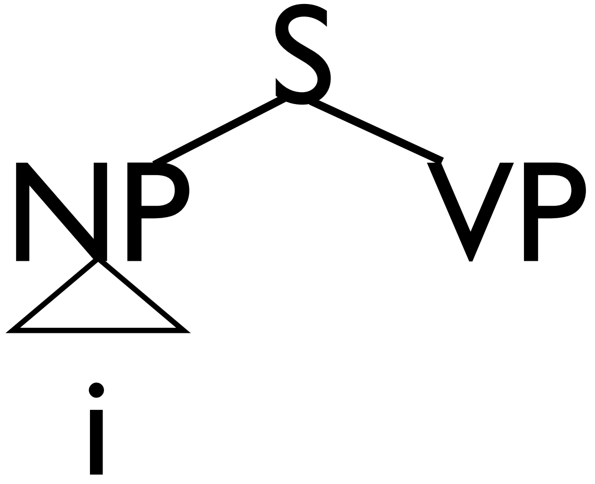

S

NP

VP

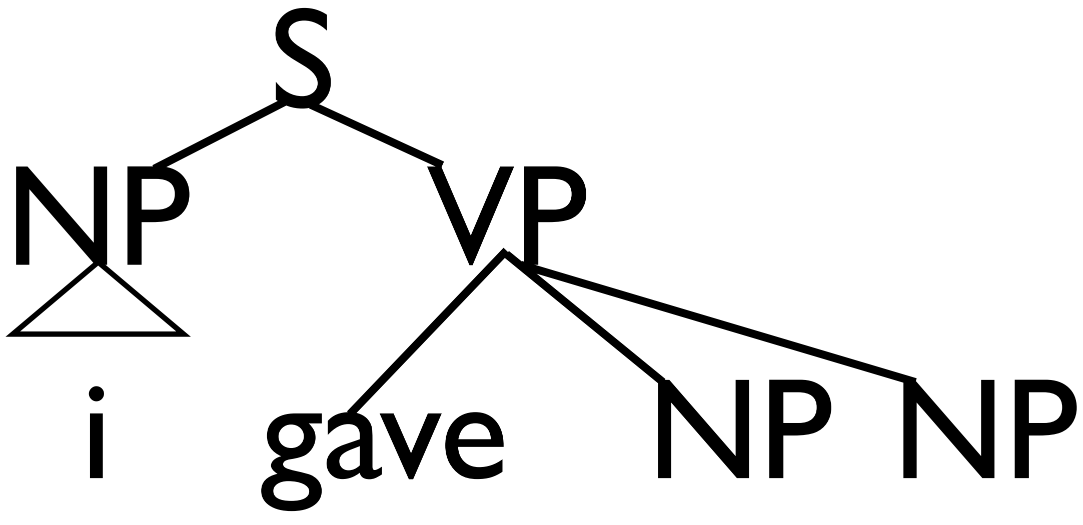
How do we view natural language?

Or as a tree?



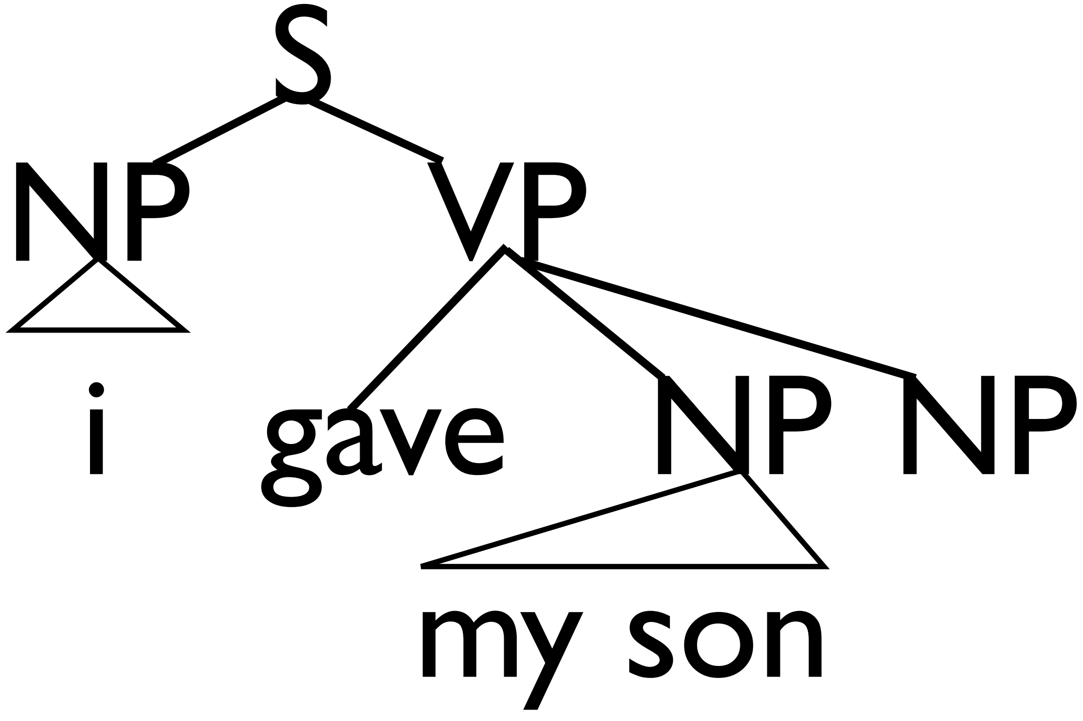
How do we view natural language?

Or as a tree?



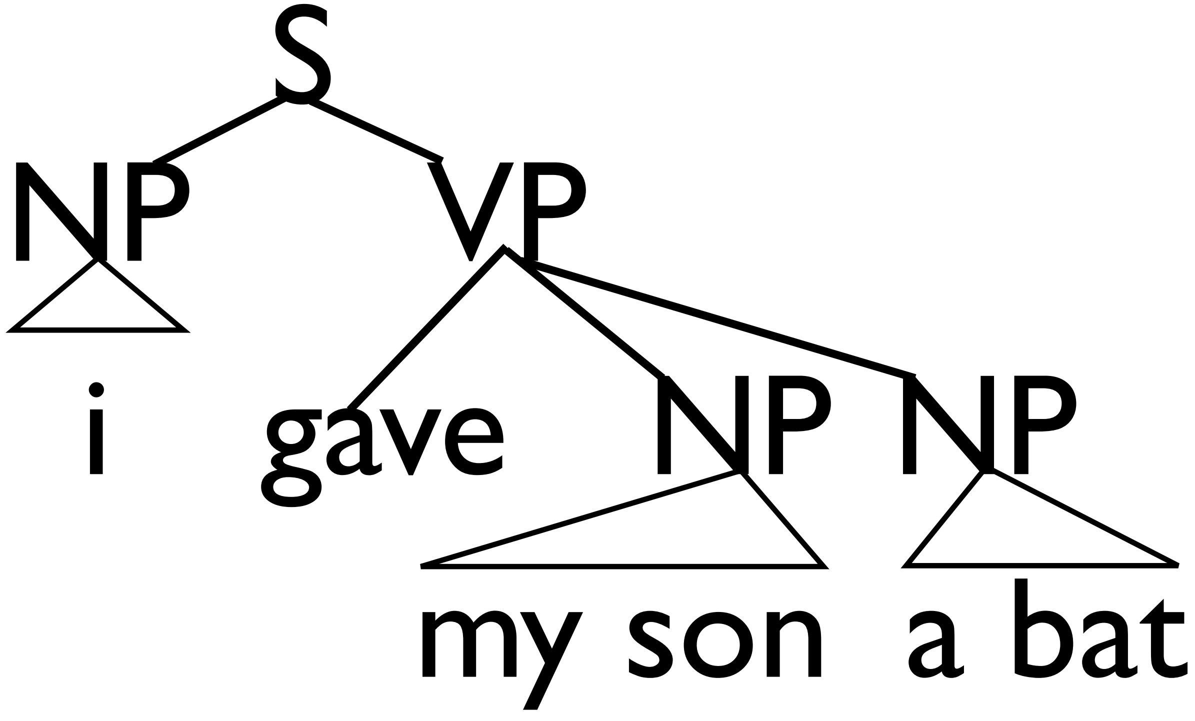
How do we view natural language?

Or as a tree?



How do we view natural language?

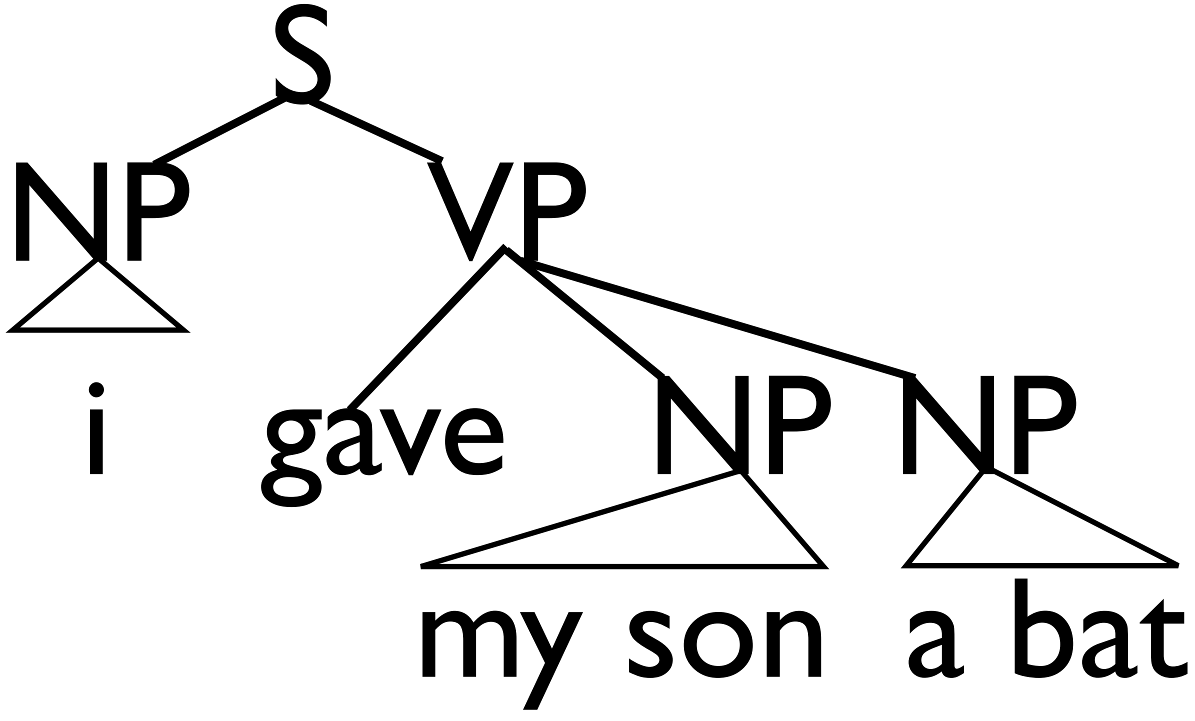
Or as a tree?



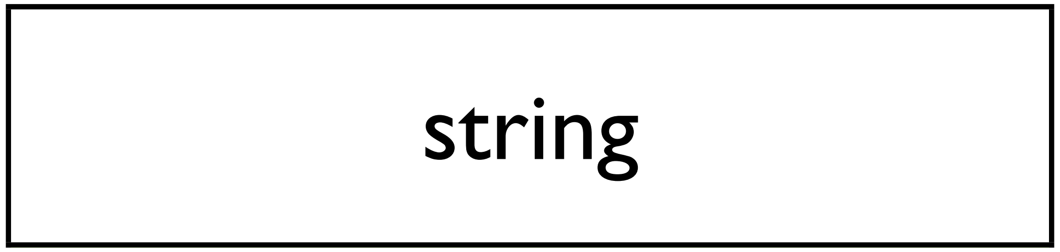
How do we view natural language?

Or as a tree?

Trees provide syntactic context!



String World vs Tree World



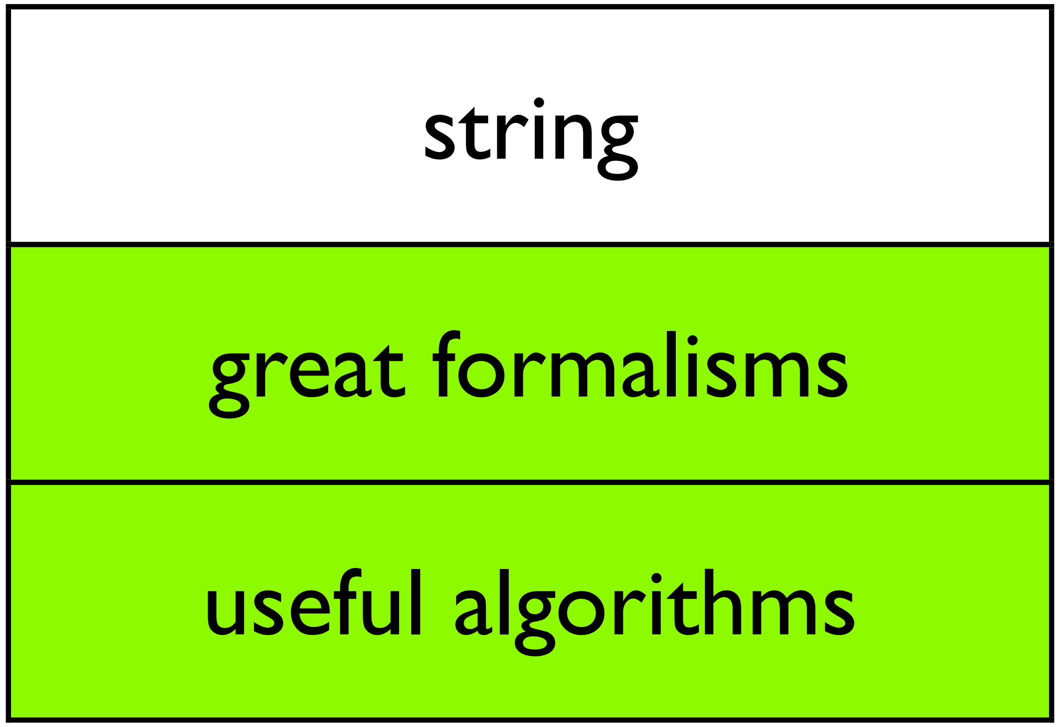
string

String World vs Tree World

string

great formalisms

String World vs Tree World

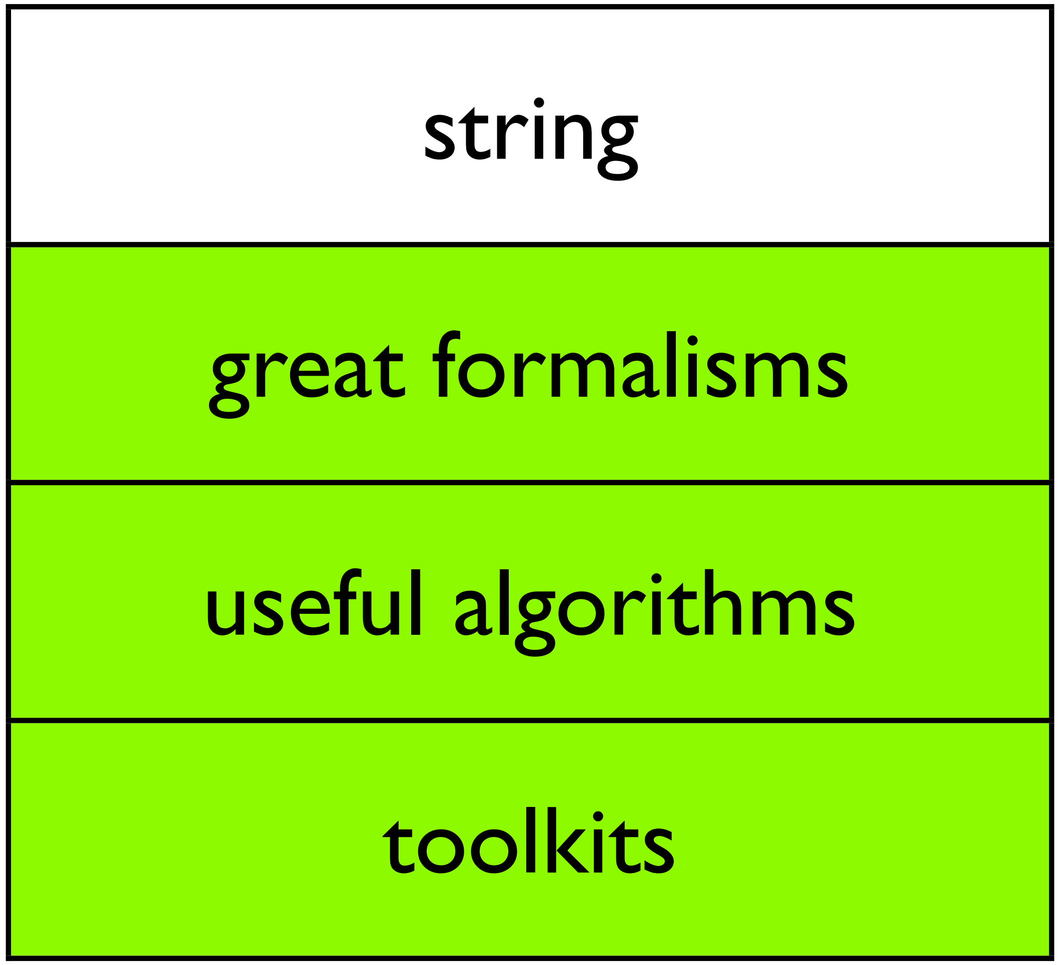


string

great formalisms

useful algorithms

String World vs Tree World



string

great formalisms

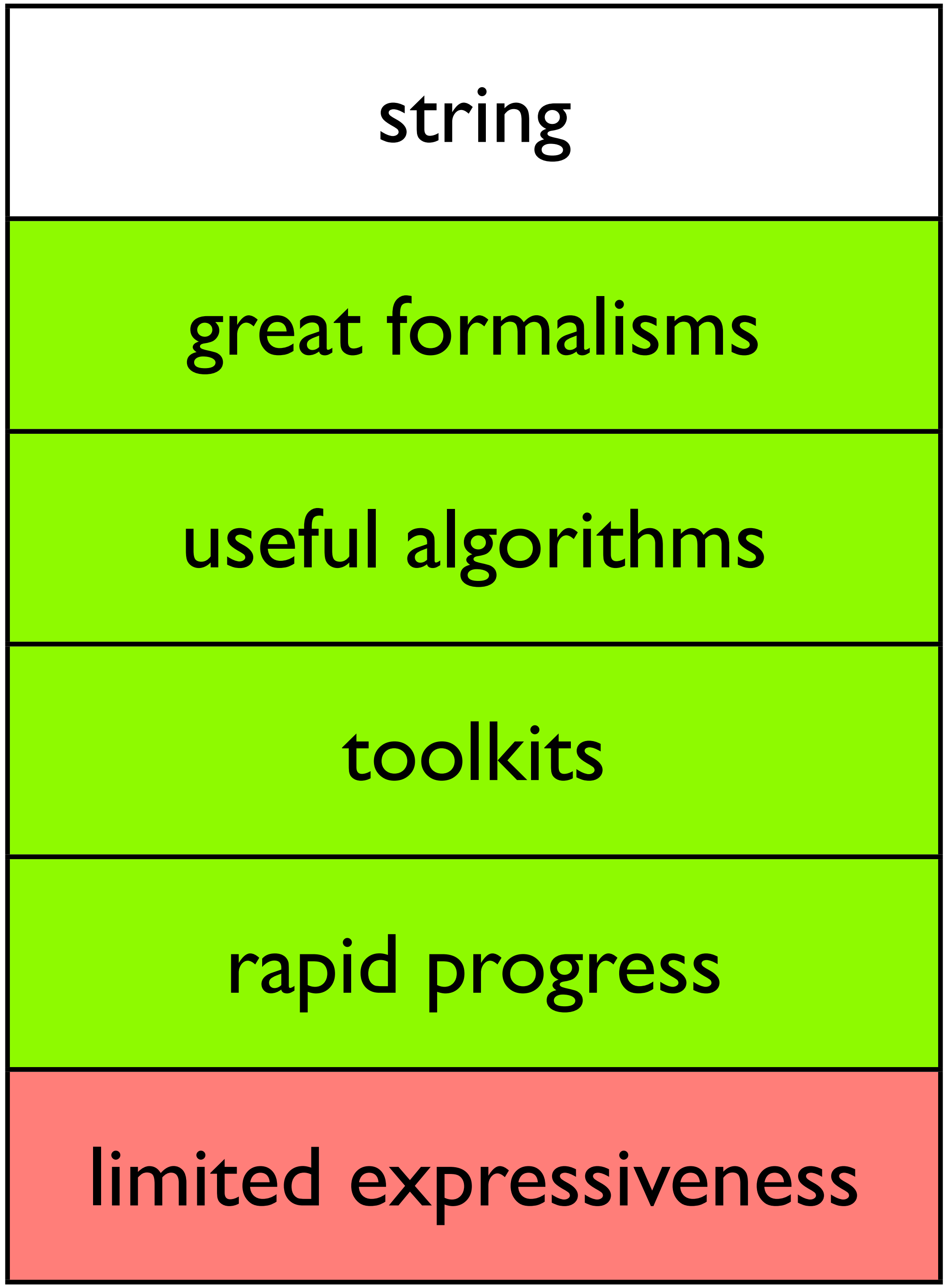
useful algorithms

toolkits

String World vs Tree World

string
great formalisms
useful algorithms
toolkits
rapid progress

String World vs Tree World



string

great formalisms

useful algorithms

toolkits

rapid progress

limited expressiveness

String World vs Tree World

string	tree
great formalisms	
useful algorithms	
toolkits	
rapid progress	
limited expressiveness	

String World vs Tree World

string	tree
great formalisms	great formalisms
useful algorithms	
toolkits	
rapid progress	
limited expressiveness	

String World vs Tree World

string	tree
great formalisms	great formalisms
useful algorithms	
toolkits	
rapid progress	
limited expressiveness	powerful expressiveness

String World vs Tree World

string	tree
great formalisms	great formalisms
useful algorithms	few algorithms
toolkits	
rapid progress	
limited expressiveness	powerful expressiveness

String World vs Tree World

string	tree
great formalisms	great formalisms
useful algorithms	few algorithms
toolkits	no toolkits
rapid progress	
limited expressiveness	powerful expressiveness

String World vs Tree World

string	tree
great formalisms	great formalisms
useful algorithms	few algorithms
toolkits	no toolkits
rapid progress	slow progress
limited expressiveness	powerful expressiveness

Contributions

string	tree
great formalisms	great formalisms
useful algorithms	few algorithms
toolkits	no toolkits
rapid progress	slow progress
limited expressiveness	powerful expressiveness

Contributions

string	tree
great formalisms	great formalisms
useful algorithms	**new algorithms!**
toolkits	no toolkits
rapid progress	slow progress
limited expressiveness	powerful expressiveness

Contributions

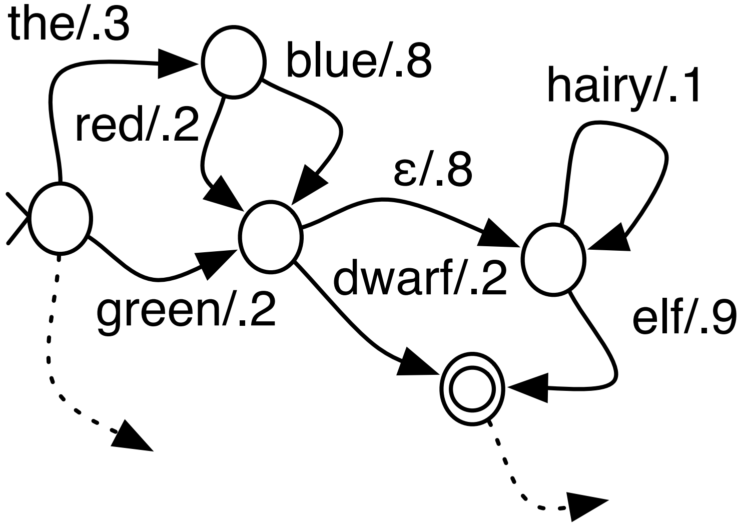
string	tree
great formalisms	great formalisms
useful algorithms	**new algorithms!**
toolkits	**new toolkit!**
rapid progress	slow progress
limited expressiveness	powerful expressiveness

Contributions

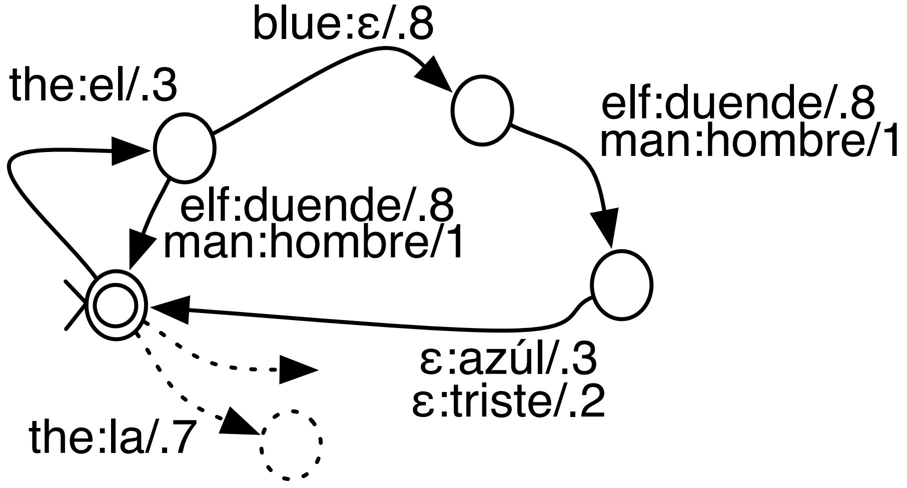
string	tree
great formalisms	great formalisms
useful algorithms	**new algorithms!**
toolkits	**new toolkit!**
rapid progress	**rapid progress!**
limited expressiveness	powerful expressiveness

Weighted finite-state string machines

Acceptor

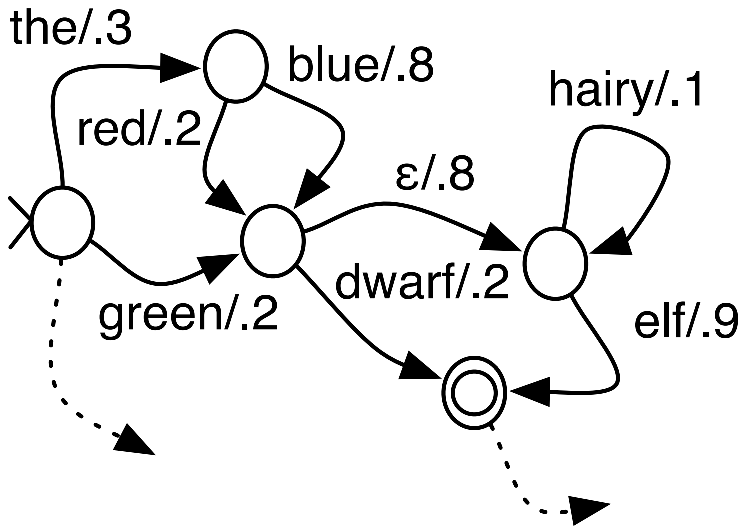


Transducer



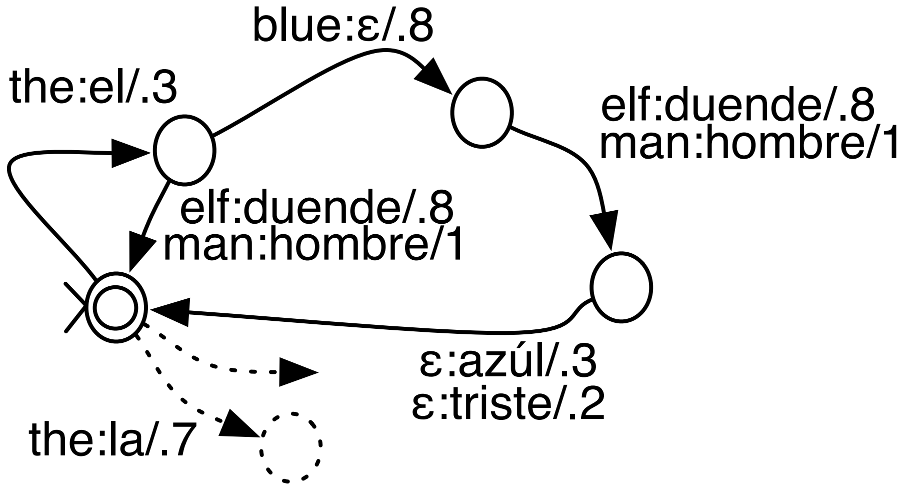
Weighted finite-state string machines

Acceptor



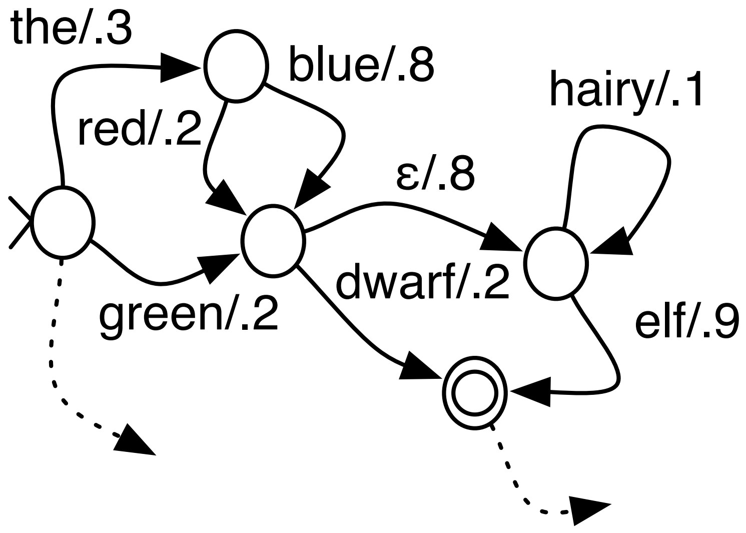
the blue dwarf/.048
green hairy elf/.0144
the red hairy hairy elf/.000432
...

Transducer



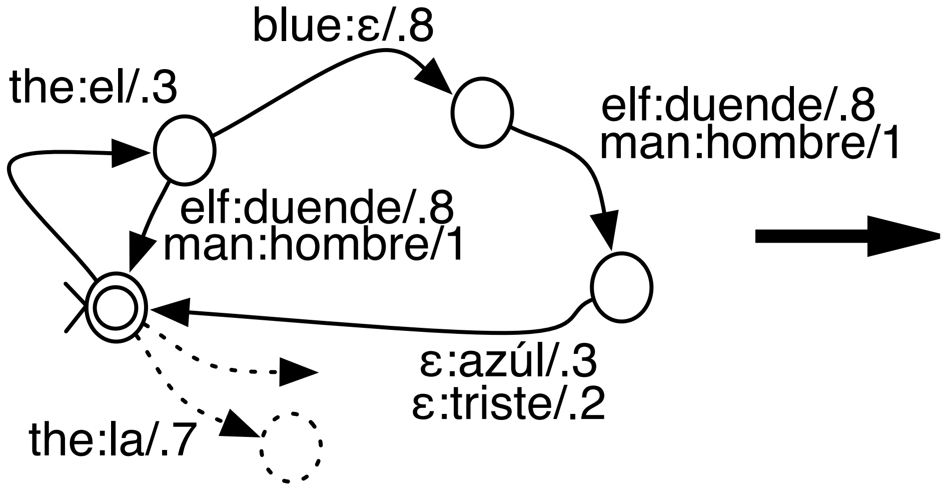
Weighted finite-state string machines

Acceptor



the blue dwarf/.048
green hairy elf/.0144
the red hairy hairy elf/.000432
...

Transducer



the blue elf : el duende azúl/.0576
the blue man : el duende triste/.048
...

Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf

Using WFSTs for NLP

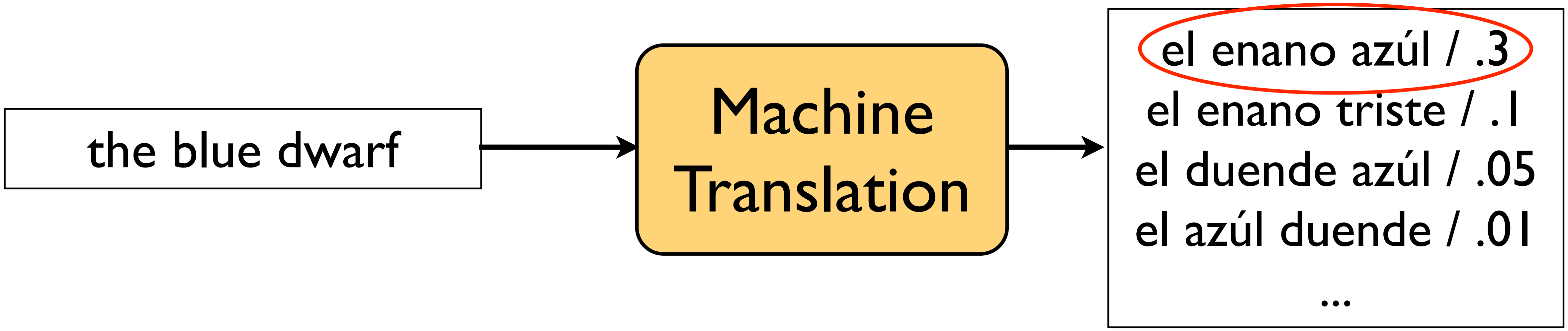
Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf

Machine Translation

Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

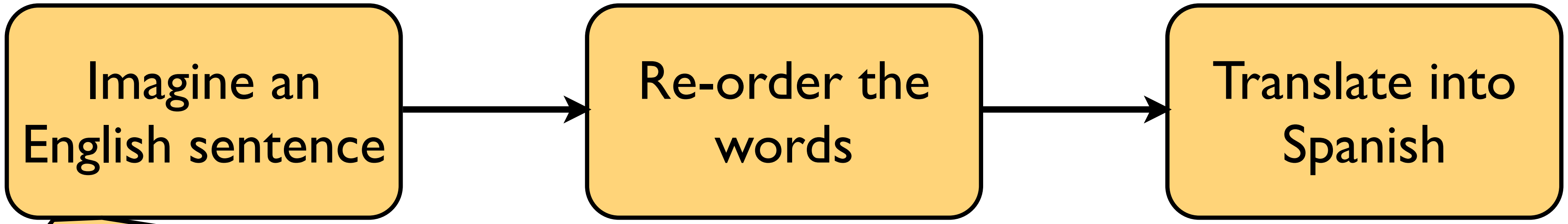


the blue dwarf

Machine Translation

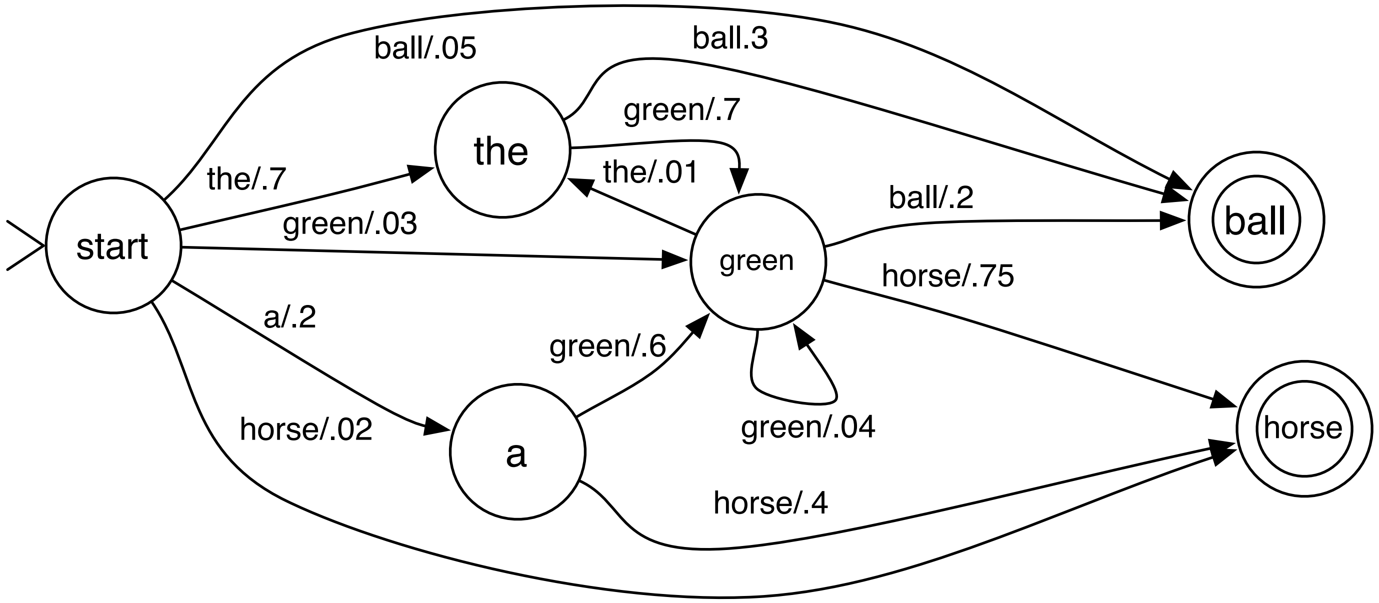
el enano azúl / .3
el enano triste / .1
el duende azúl / .05
el azúl duende / .01
...

MT as weighted transducers


Imagine an English sentence

Re-order the words

Translate into Spanish

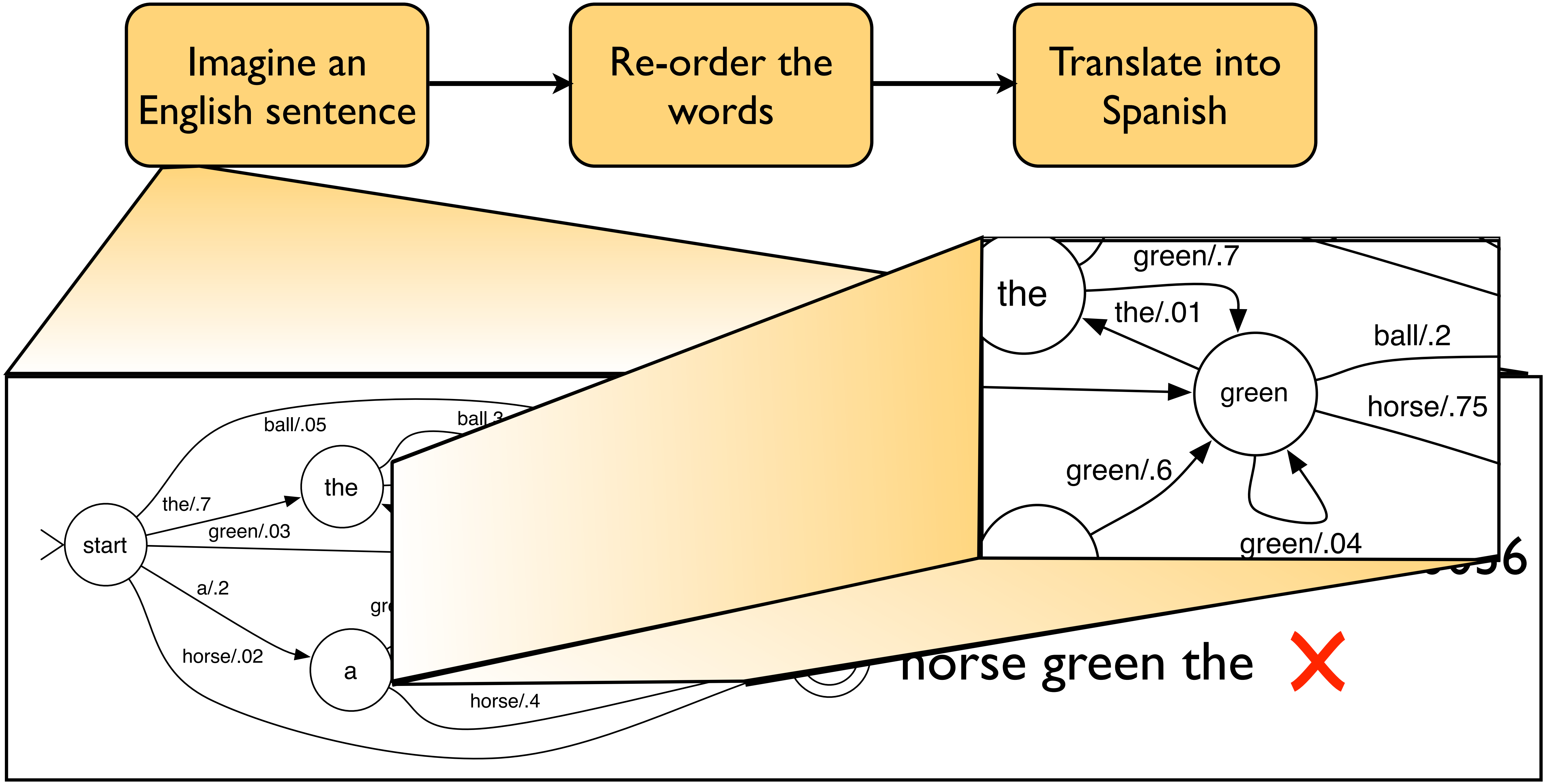


the green ball = .098

a green green horse = .0036

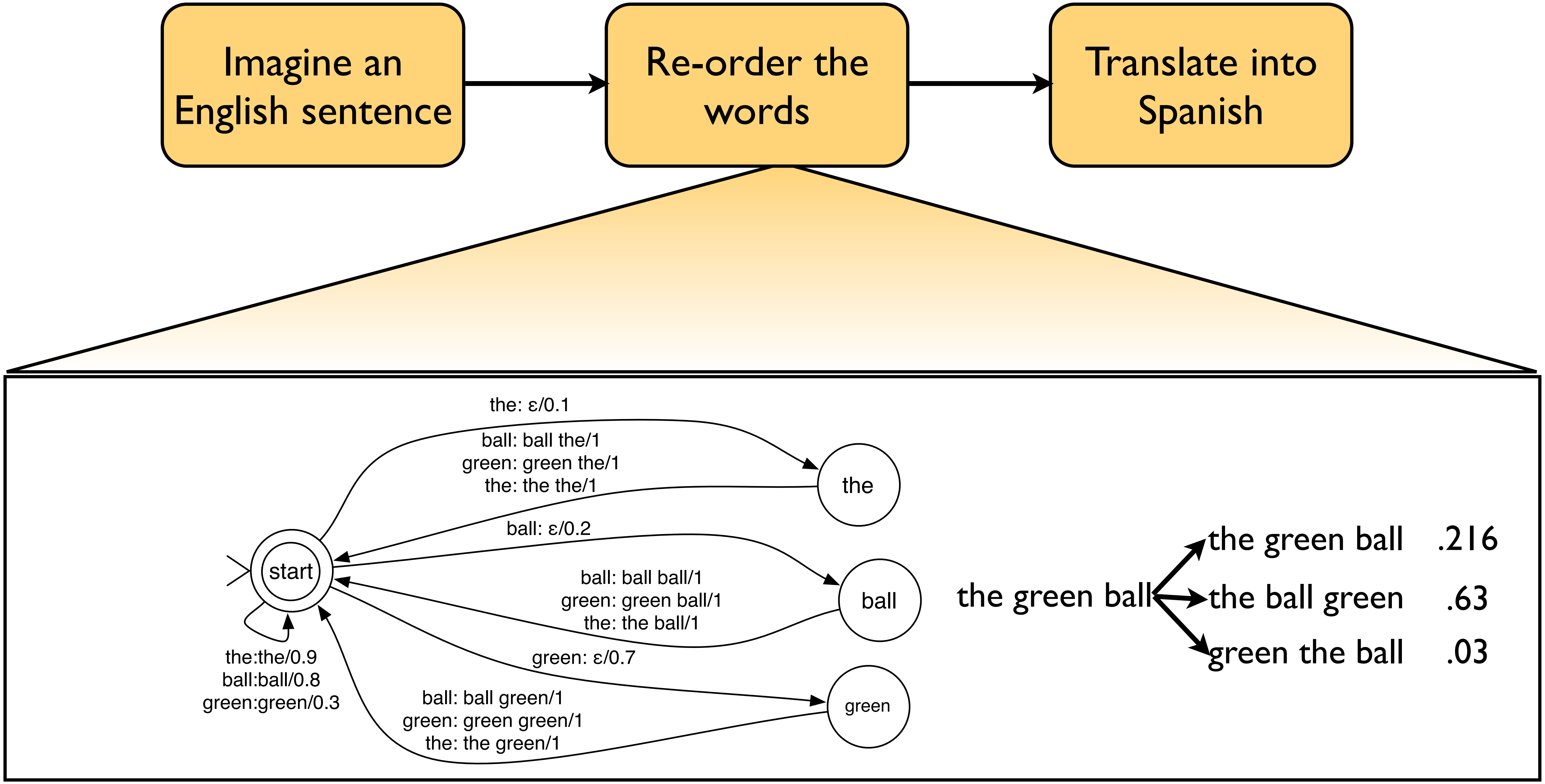
horse green the ✗

MT as weighted transducers



Imagine an English sentence → Re-order the words → Translate into Spanish

MT as weighted transducers



Imagine an English sentence

Re-order the words

Translate into Spanish

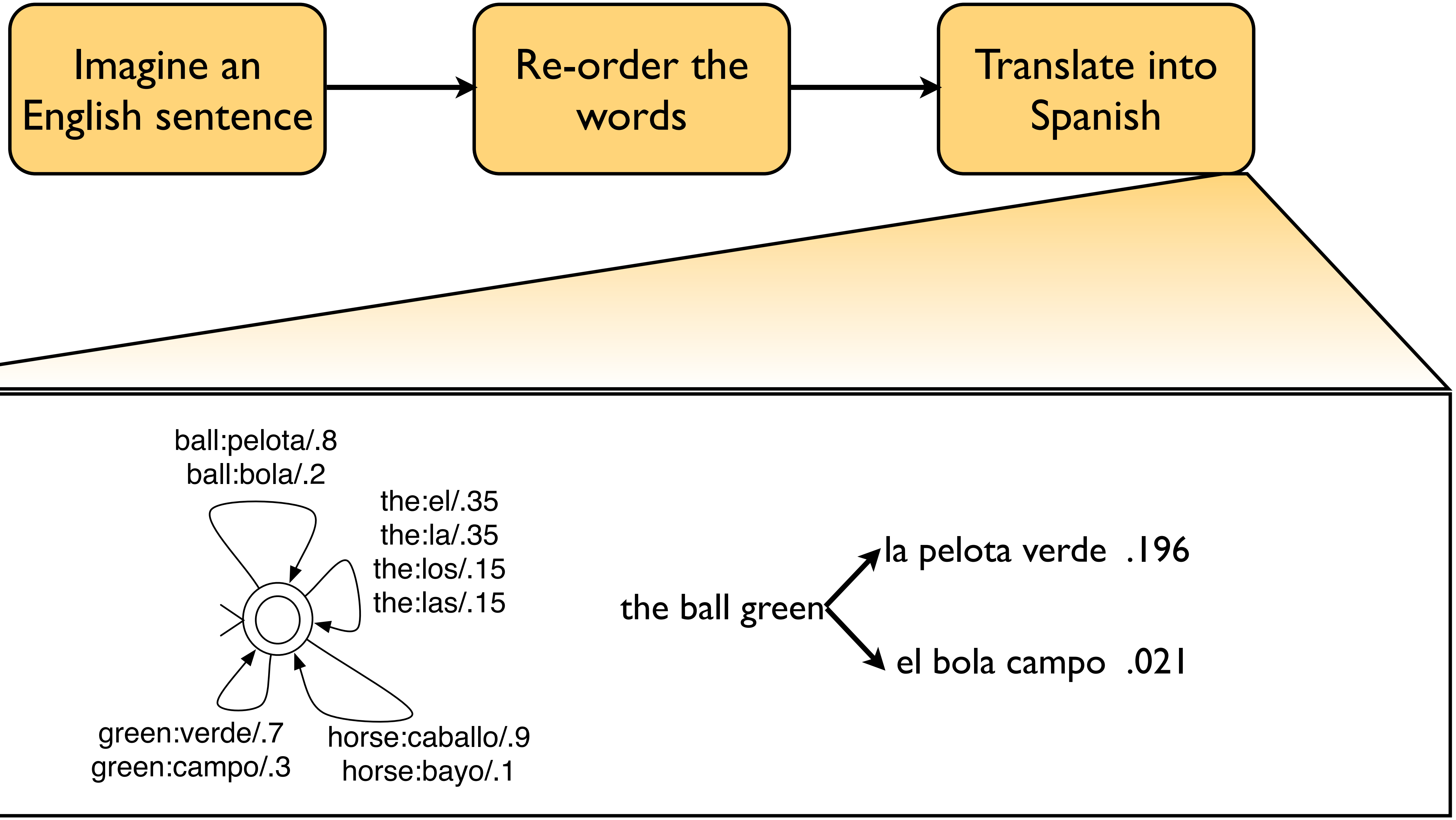
the green ball

the green ball .216

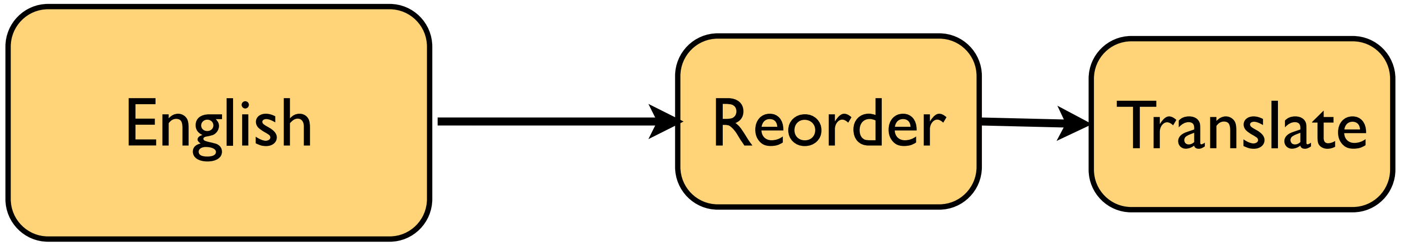
the ball green .63

green the ball .03

MT as weighted transducers

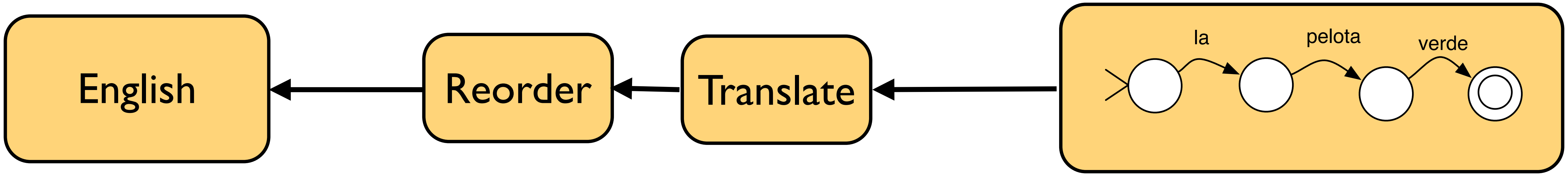


MT as weighted transducers



Generative story: we corrupt good English into (possibly bad) Spanish

MT as weighted transducers



English

Reorder

Translate

Decoding story: given some good Spanish, determine the best good English that could produce it

Secret weapons

- WFST toolkits do this calculation for us:
    - AT&T FSM[1] / Google OpenFst[2]
    - USC/ISI Carmel[3]
- Generic operations for manipulation, combination, inference, training

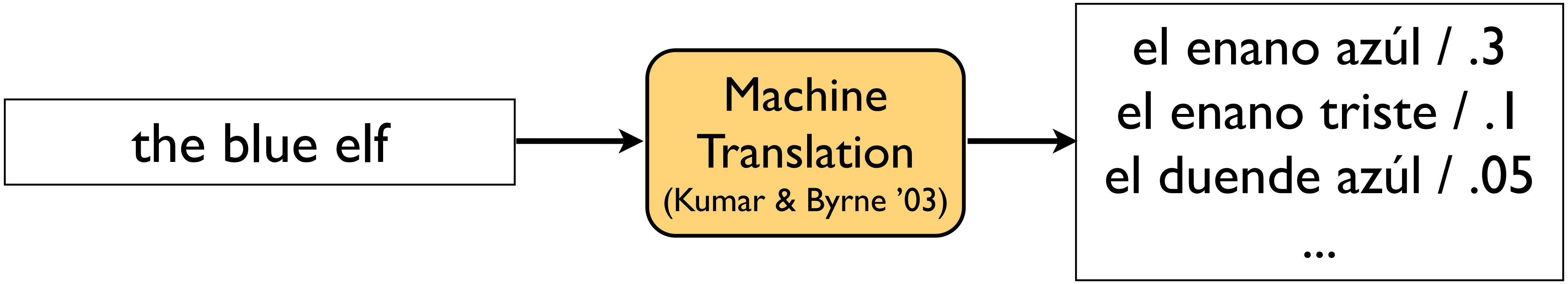
WFST toolkit operations
k-best
em training
determinization
composition
pipeline inference
on-the-fly inference

[1]: Mohri, Pereira, Riley, ’98

[2]: Allauzen et al., ’07

[3]: Graehl, ’97

Widely applicable!

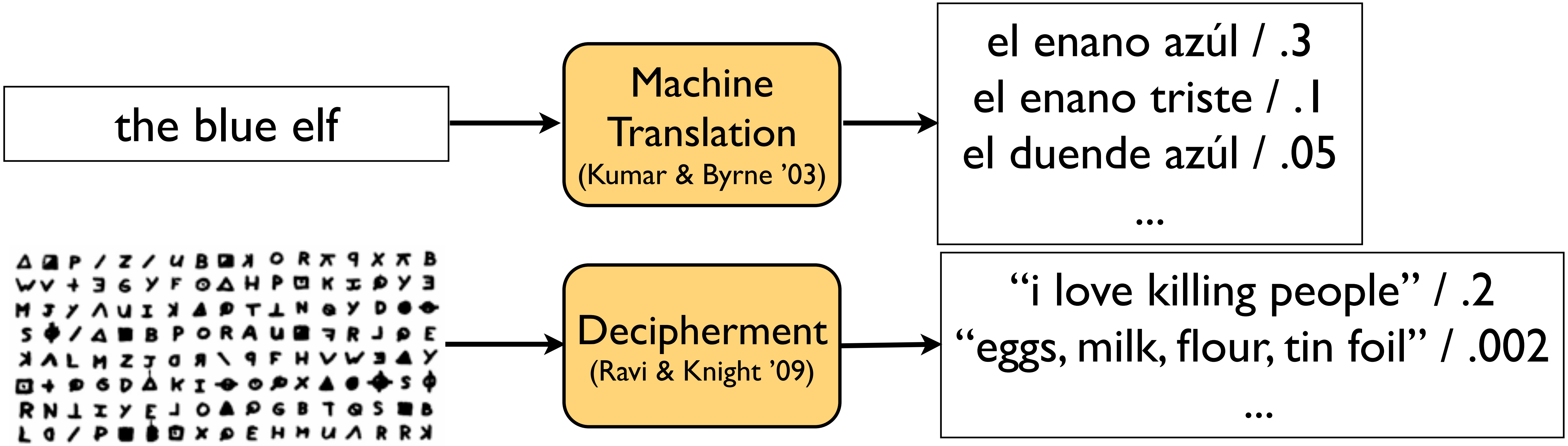


the blue elf

Machine Translation (Kumar & Byrne ’03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Widely applicable!



the blue elf

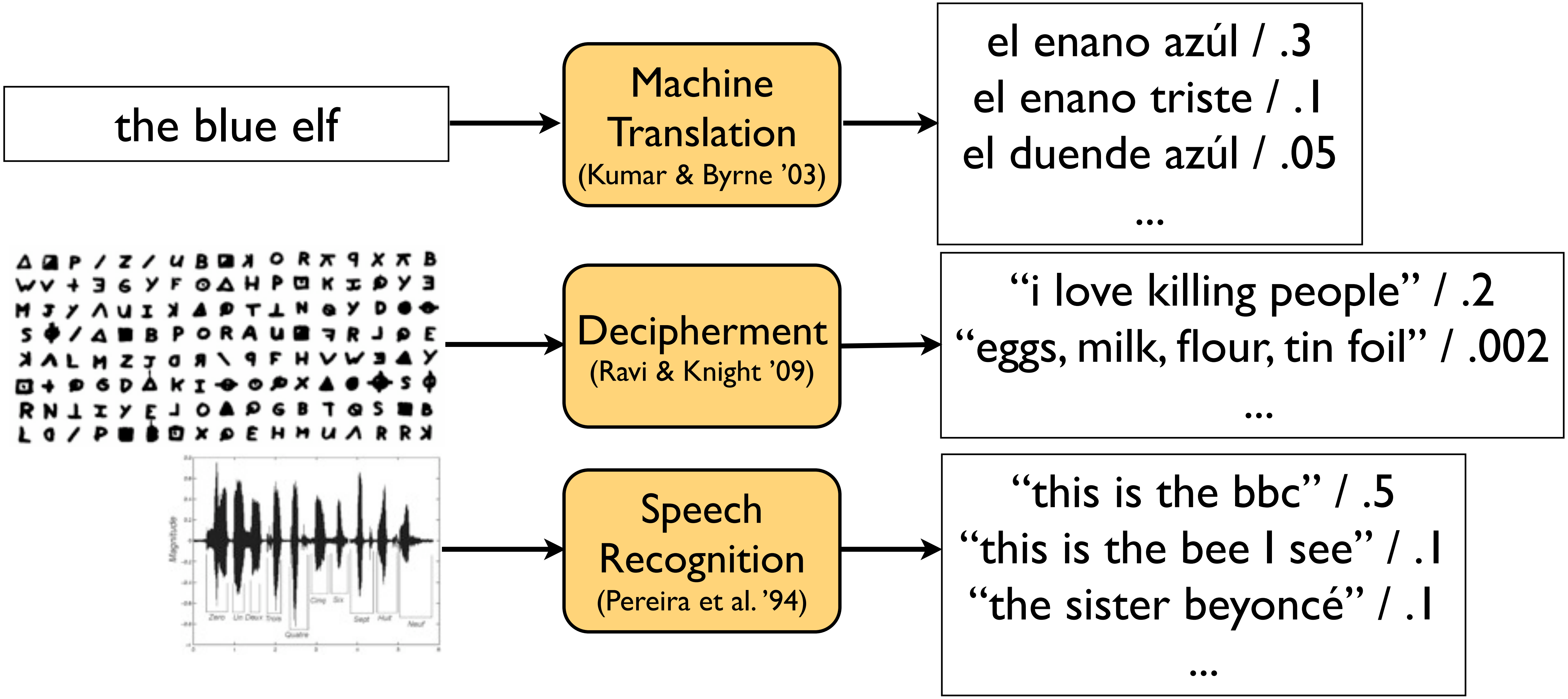
Machine Translation
(Kumar & Byrne ’03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Decipherment
(Ravi & Knight ’09)

“i love killing people” / .2
“eggs, milk, flour, tin foil” / .002
...

Widely applicable!



the blue elf

Machine Translation (Kumar & Byrne '03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

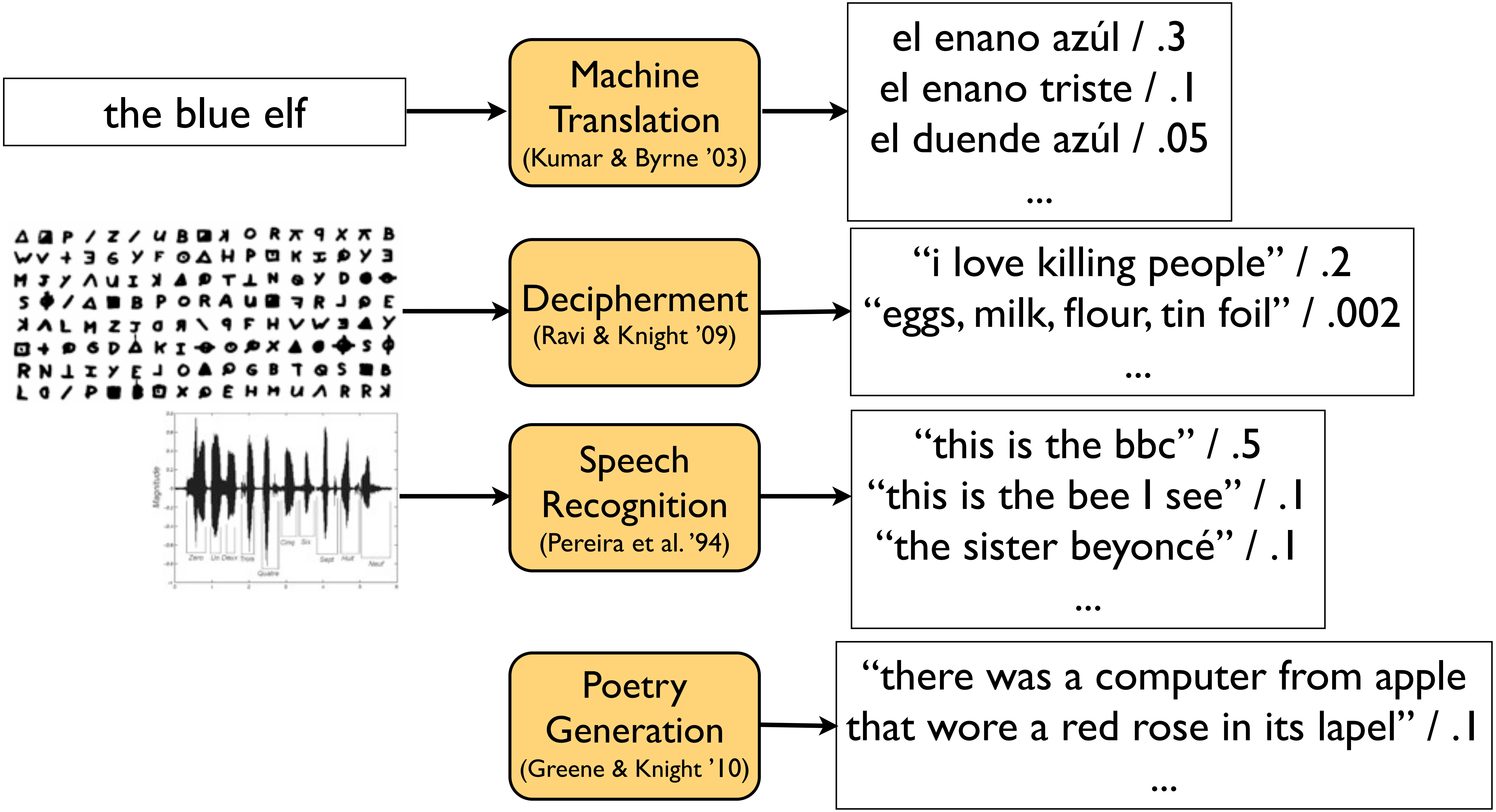
Decipherment (Ravi & Knight '09)

"i love killing people" / .2
"eggs, milk, flour, tin foil" / .002
...

Speech Recognition (Pereira et al. '94)

"this is the bbc" / .5
"this is the bee I see" / .1
"the sister beyoncé" / .1
...

Widely applicable!



the blue elf

Machine Translation
(Kumar & Byrne '03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Decipherment
(Ravi & Knight '09)

"i love killing people" / .2
"eggs, milk, flour, tin foil" / .002
...

Speech Recognition
(Pereira et al. '94)

"this is the bbc" / .5
"this is the bee I see" / .1
"the sister beyoncé" / .1
...

Poetry Generation
(Greene & Knight '10)

"there was a computer from apple
that wore a red rose in its lapel" / .1
...

NLP work using WFSTs

Translation
(Kumar & Byrne ’03)

Decipherment
(Ravi & Knight ’09)

Speech Recognition
(Pereira et al. ’94)

Poetry Generation
(Greene & Knight ’10)

OCR
(Kolak et al. ’03)

Morphology
(Karttunen et al. ’92)

POS Tagging
(Church ’88)

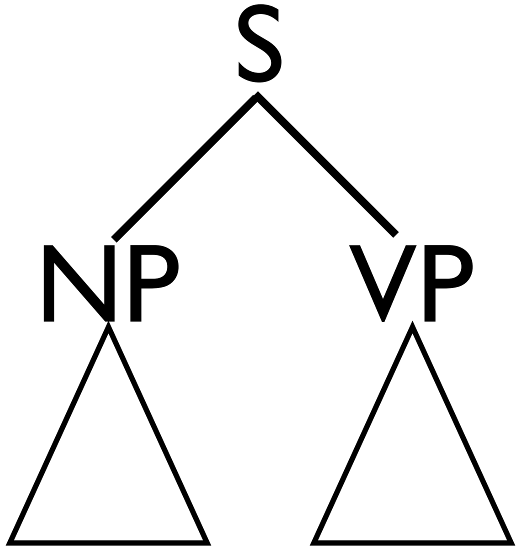
Spelling Correction
(Boyd ’09)

Transliteration
(Knight & Graehl ’98)

Also see summary: book chapter of *Handbook of Weighted Automata* (Knight & May ’08)

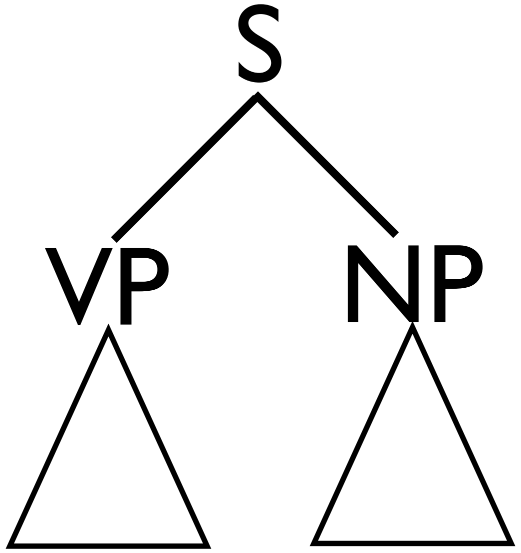
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information



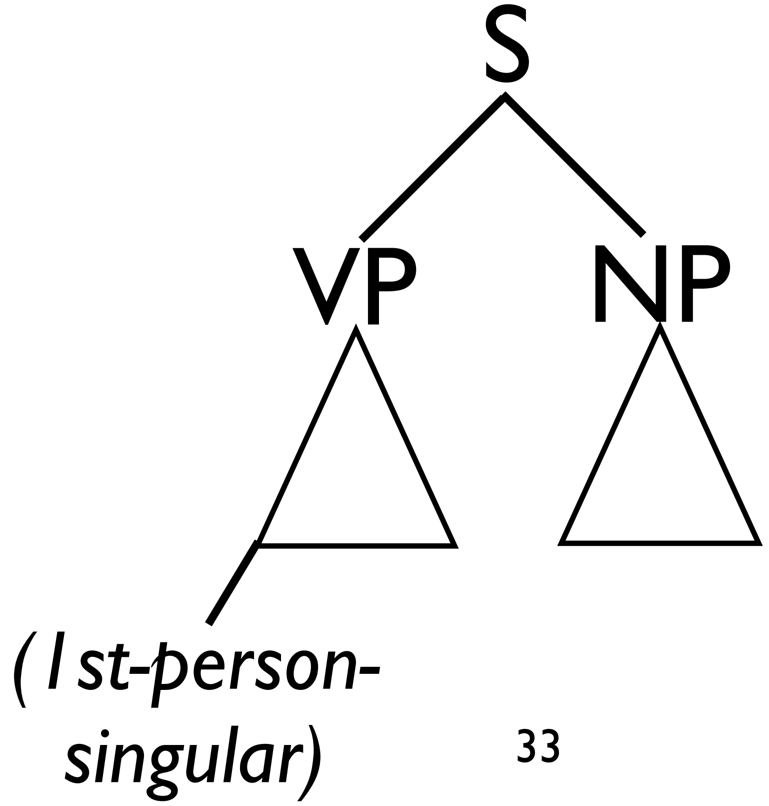
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information



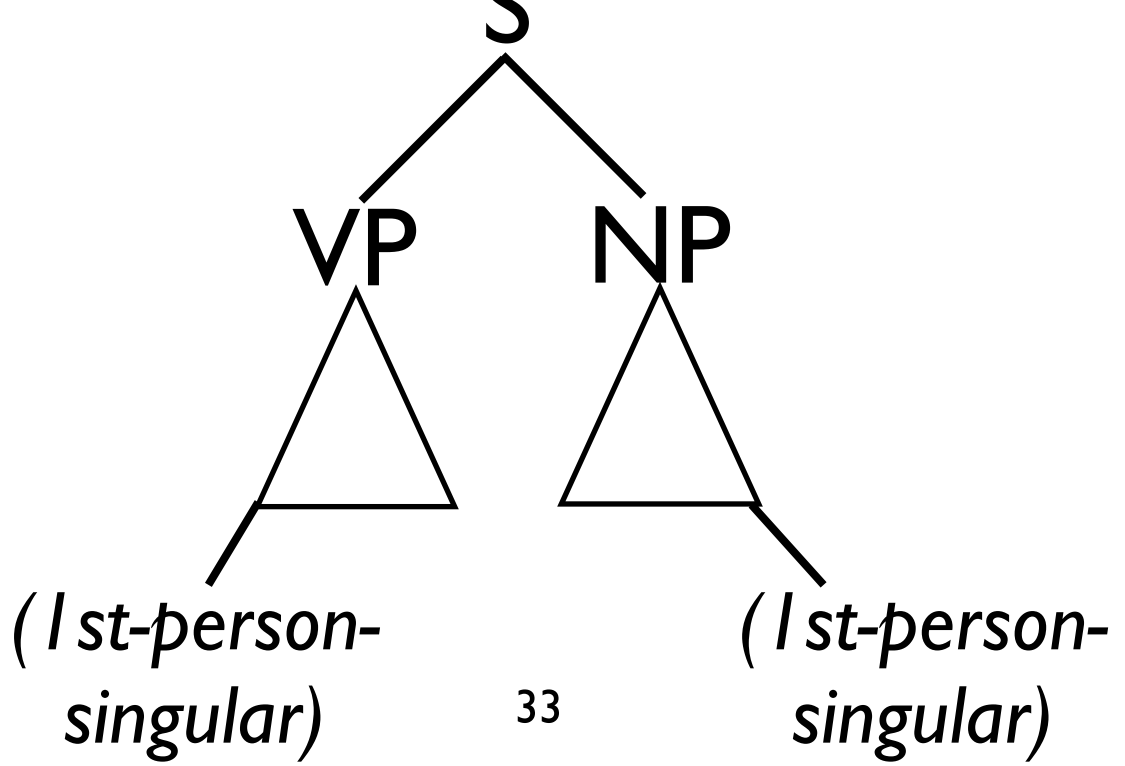
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information



Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information



Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!

Limitations of strings

- Can't do arbitrary long-distance reordering
- Can't maintain arbitrary long-distance dependencies
- Can't naturally integrate syntax information

But that's what we want!

Parsing
(Collins '97)

Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!

Parsing	Question Answering
(Collins ’97)	(Echihabi & Marcu ’03)

Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!

Parsing
(Collins ’97)

Question Answering
(Echihabi & Marcu ’03)

Language Modeling
(Charniak ’01)

Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!

Parsing (Collins ’97)

Question Answering (Echihabi & Marcu ’03)

Language Modeling (Charniak ’01)

Summarization (Knight & Marcu ’03)

Limitations of strings

- Can't do arbitrary long-distance reordering
- Can't maintain arbitrary long-distance dependencies
- Can't naturally integrate syntax information

But that's what we want!

Parsing
(Collins '97)

Question Answering
(Echihabi & Marcu '03)

Machine Translation
(Yamada & Knight '01)
(Galley et al. '04)
(Mi et al. '08)
(Zhang et al. '08)

Language Modeling
(Charniak '01)

Summarization
(Knight & Marcu '03)

Lots of work with tree models, but NO tree toolkit!

Parsing
(Collins ’97)

Language Modeling
(Charniak ’01)

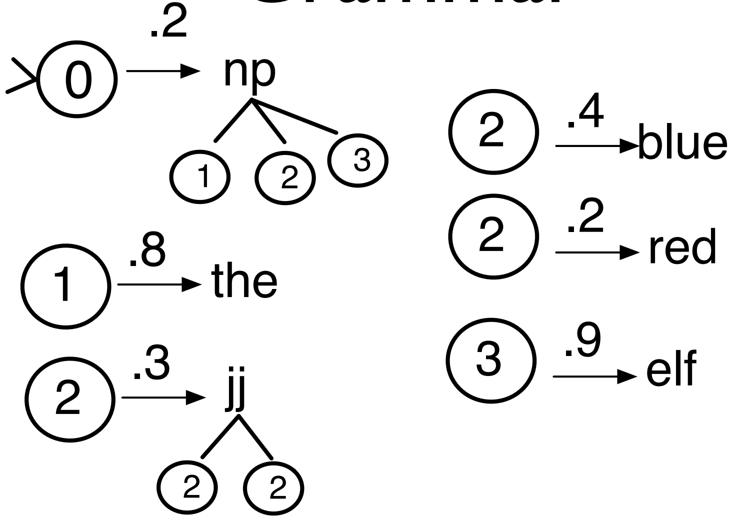
Question Answering
(Echihabi & Marcu ’03)

Summarization
(Knight & Marcu ’03)

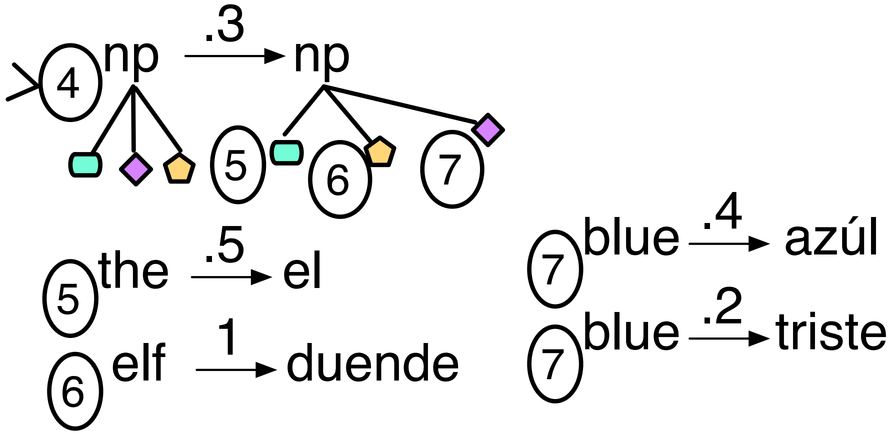
Machine Translation
(Yamada & Knight ’01)
(Galley et al. ’04)
(Mi et al. ’08)
(Zhang et al. ’08)

Weighted finite-state tree machines

Grammar

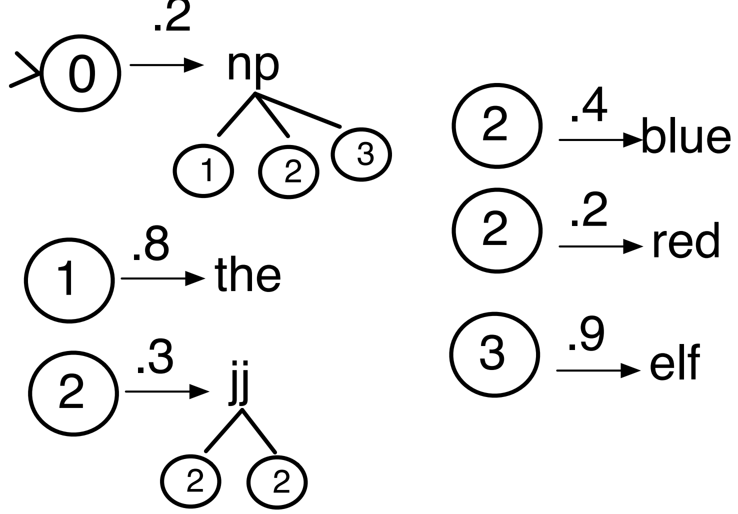


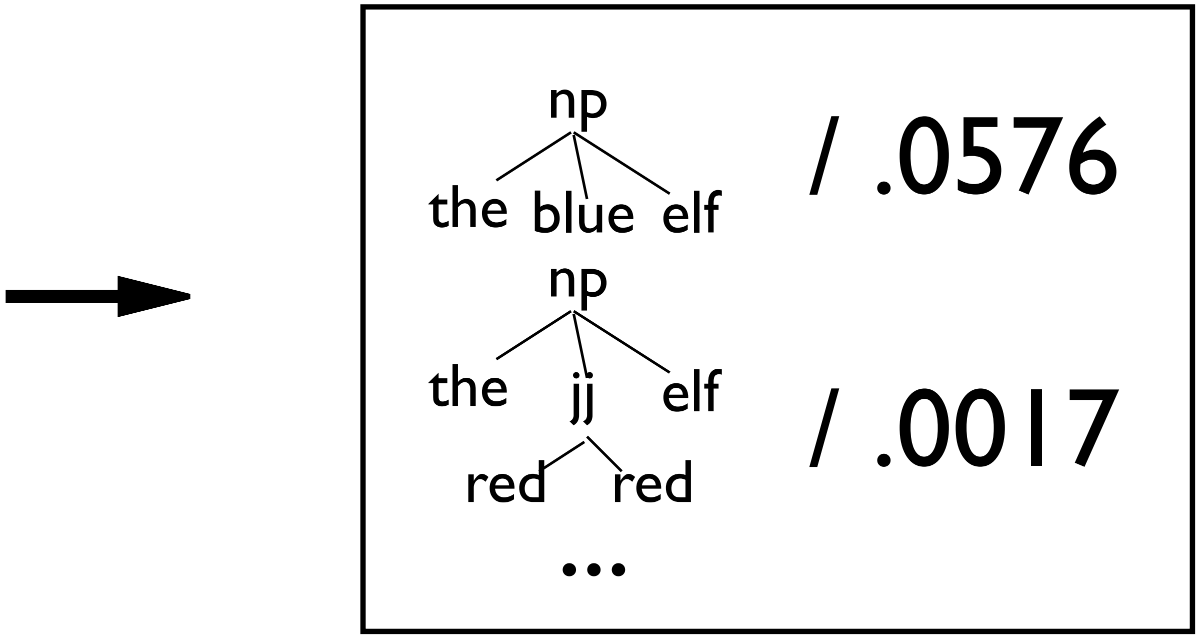
Transducer



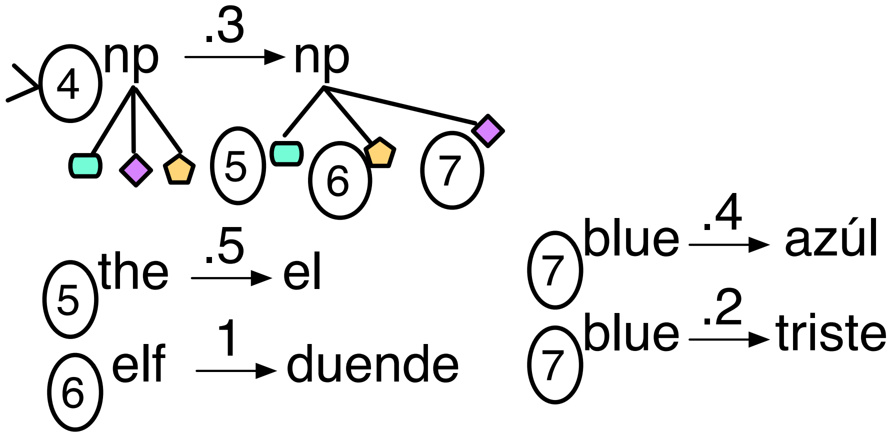
Weighted finite-state tree machines

Grammar

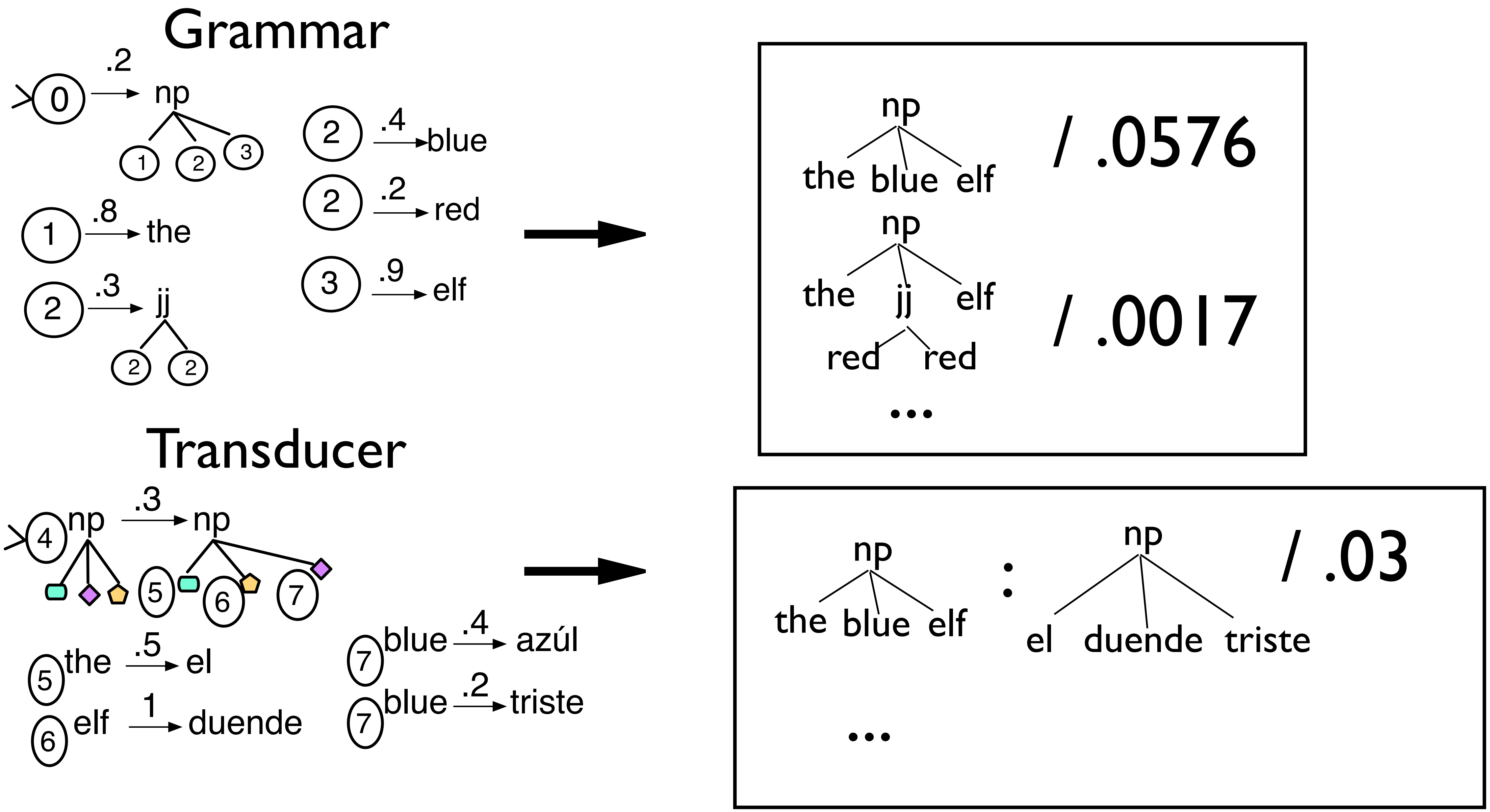




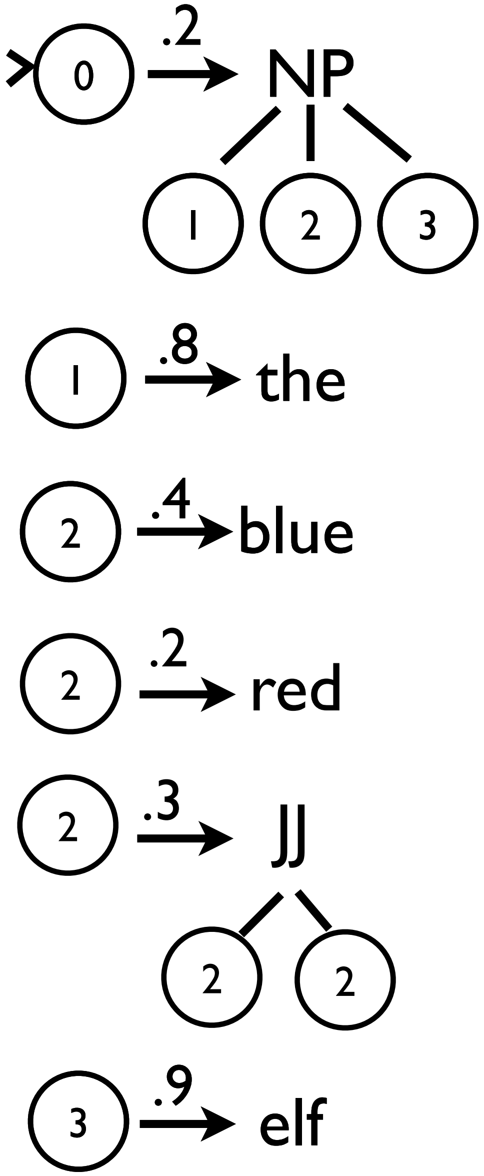
Transducer



Weighted finite-state tree machines



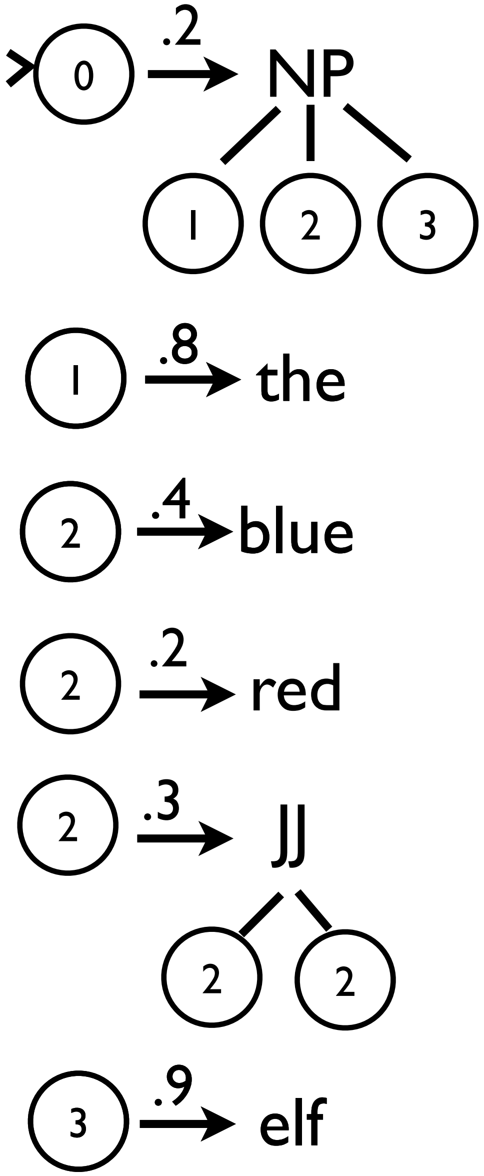
Weighted regular tree grammars

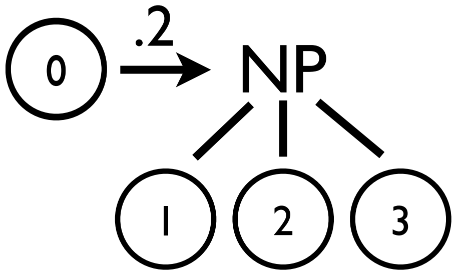


Tree	Weight
0	1

(Berstel & Reutenauer, 1982)

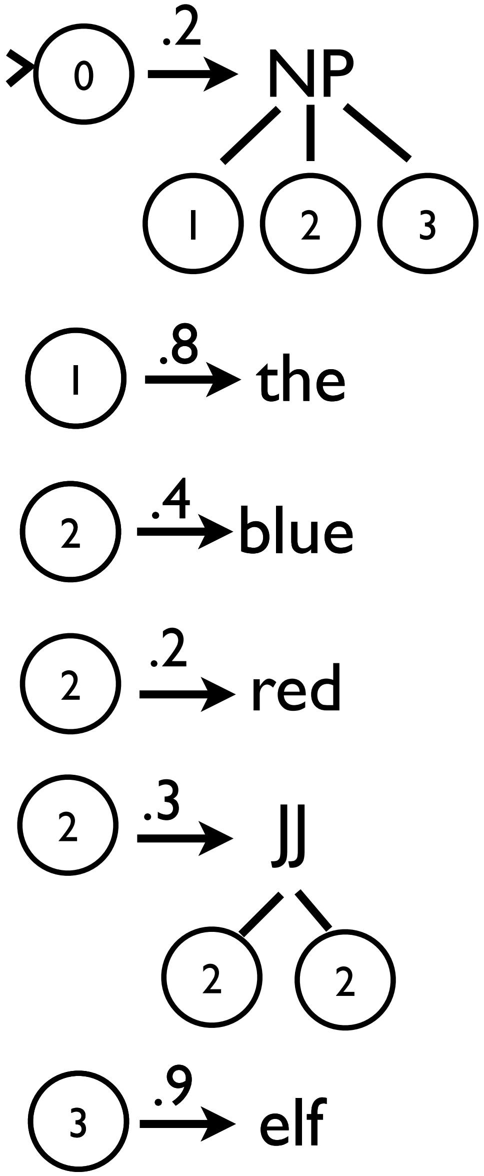
Weighted regular tree grammars

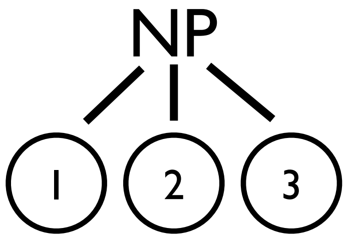


Tree	Weight
	1

(Berstel & Reutenauer, 1982)

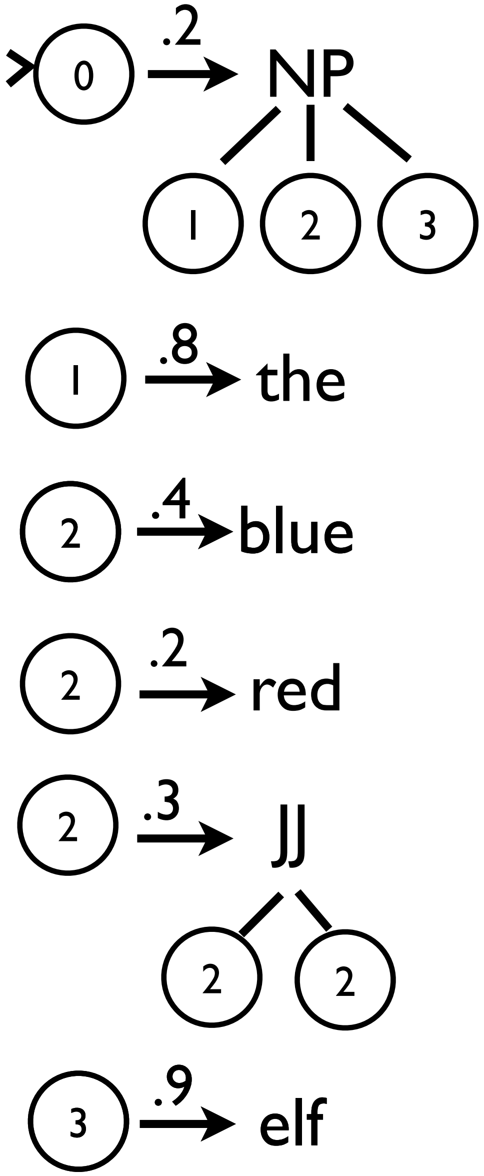
Weighted regular tree grammars



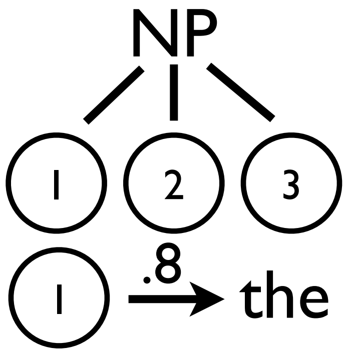
Tree	Weight
	.2

(Berstel & Reutenauer, 1982)

Weighted regular tree grammars



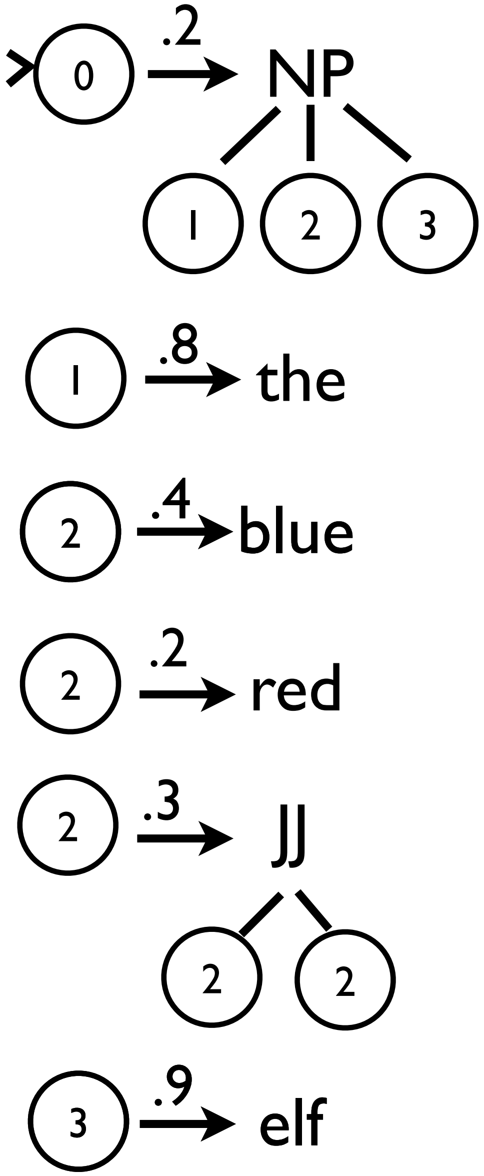
Tree Weight

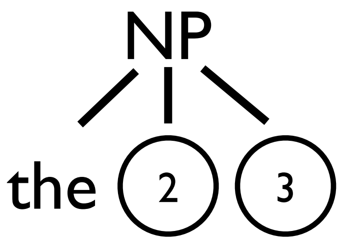


.2

(Berstel & Reutenauer, 1982)

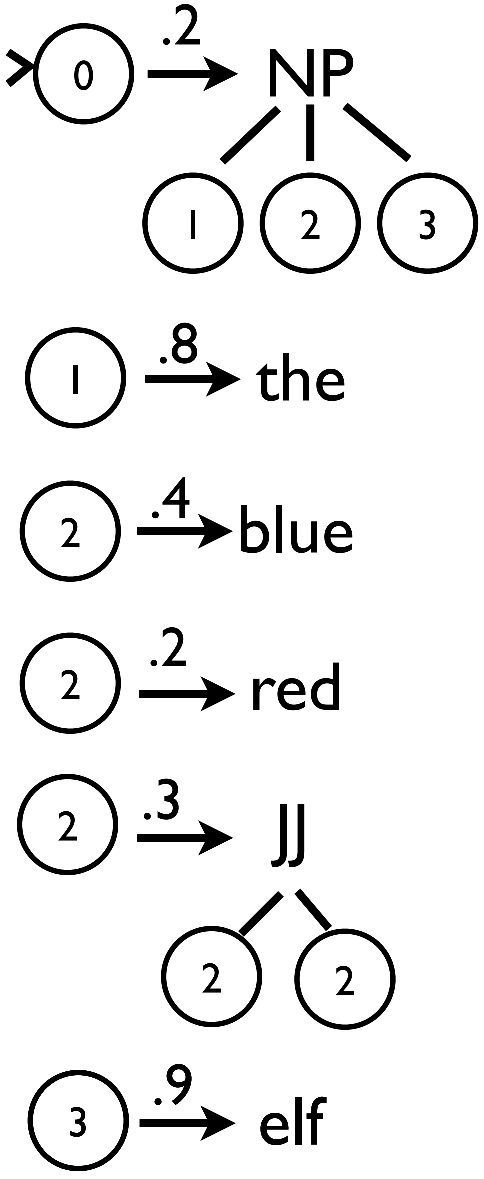
Weighted regular tree grammars

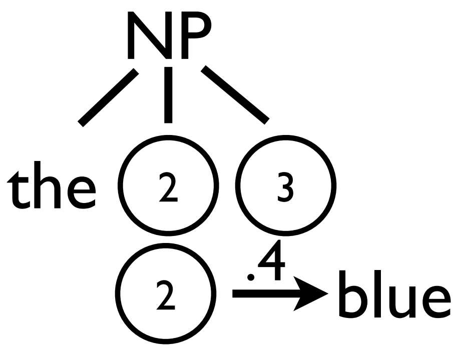


Tree	Weight
	.16

(Berstel & Reutenauer, 1982)

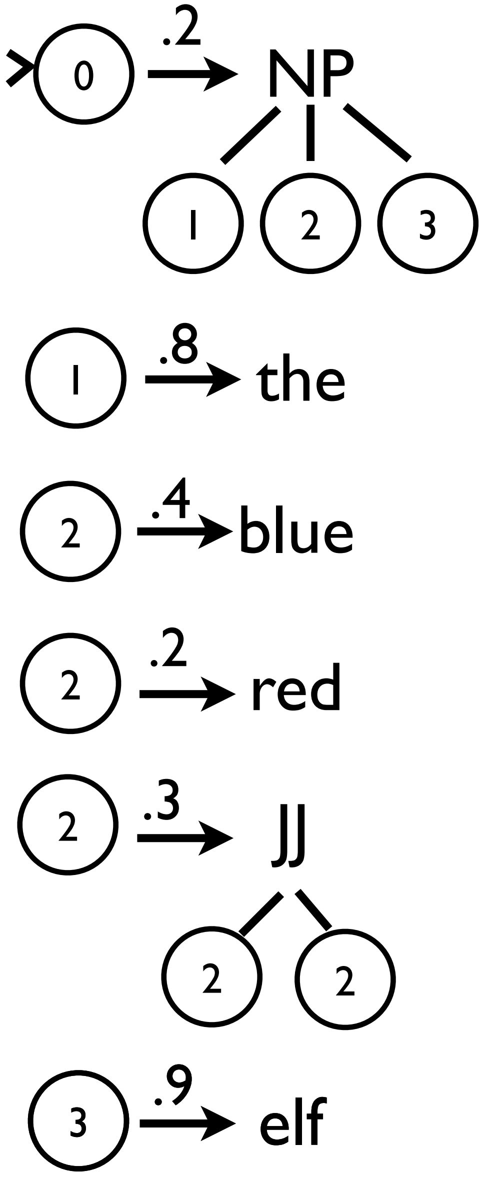
Weighted regular tree grammars

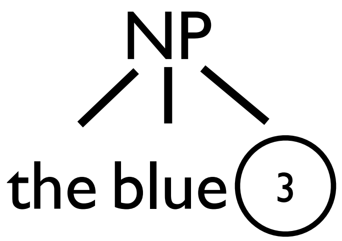


Tree	Weight
	.16

(Berstel & Reutenauer, 1982)

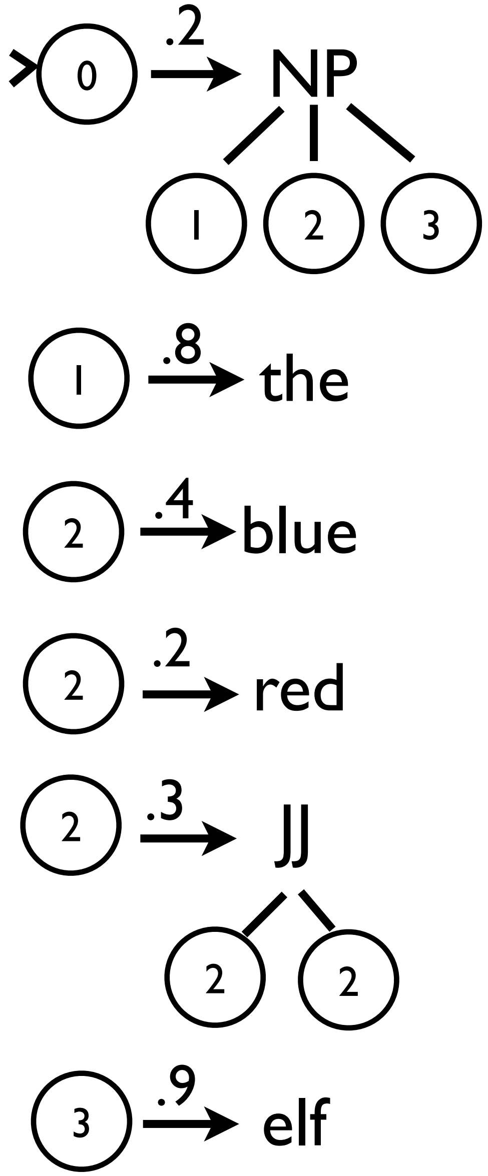
Weighted regular tree grammars

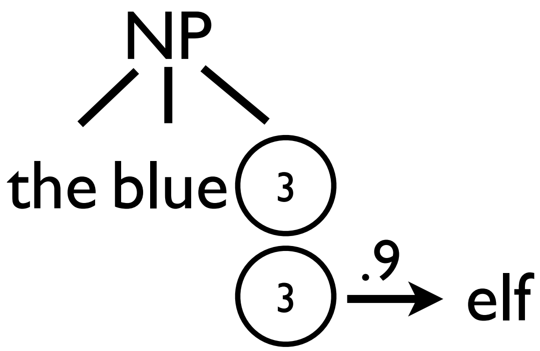


Tree	Weight
	.064

(Berstel & Reutenauer, 1982)

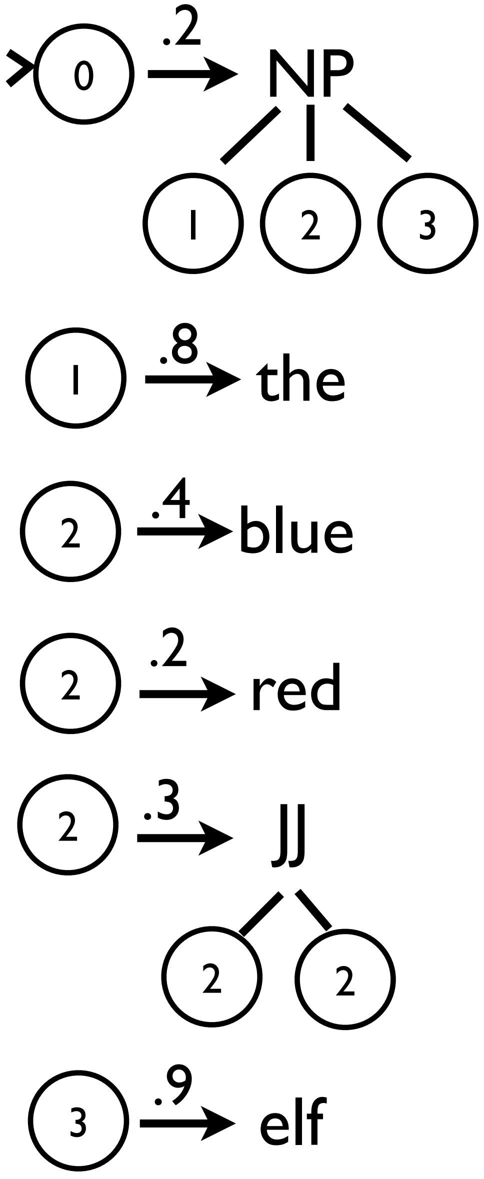
Weighted regular tree grammars

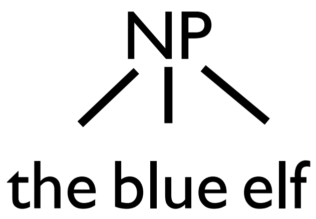


Tree	Weight
	.064

(Berstel & Reutenauer, 1982)

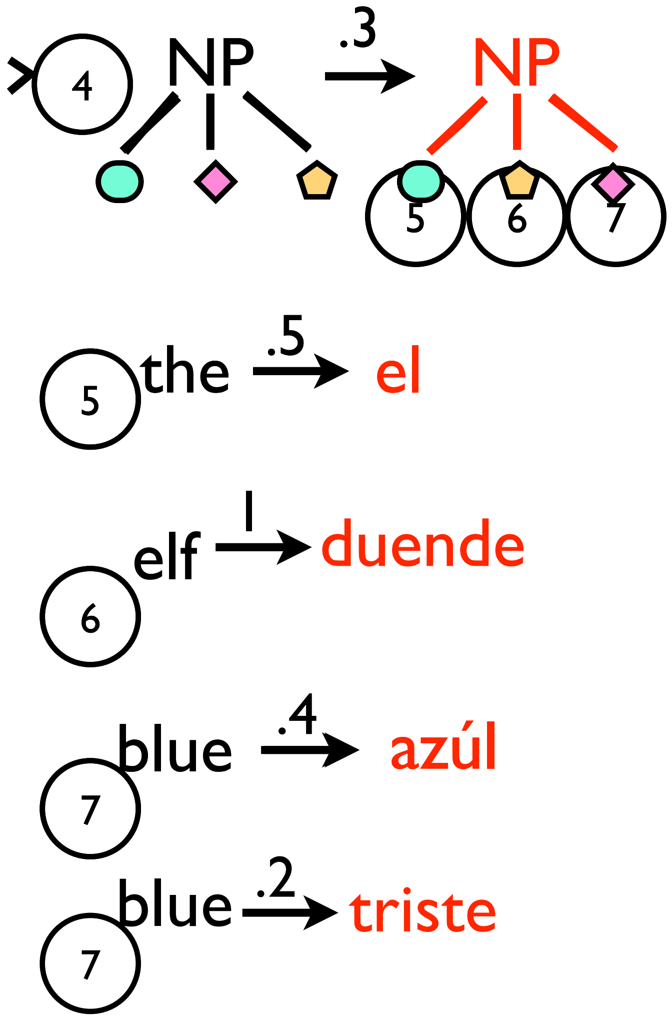
Weighted regular tree grammars



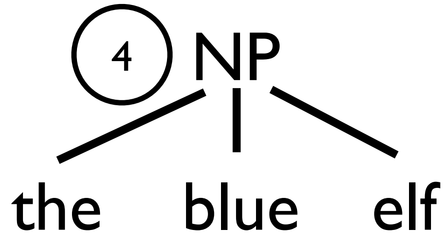
Tree	Weight
	.0576

(Berstel & Reutenauer, 1982)

Weighted tree transducers



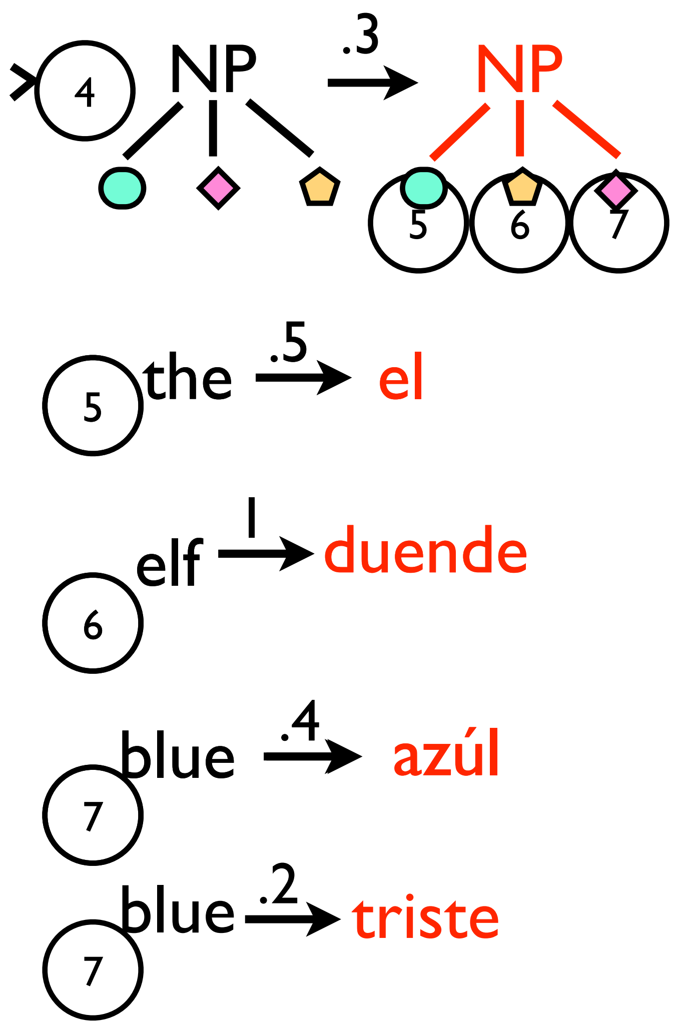
Tree Weight



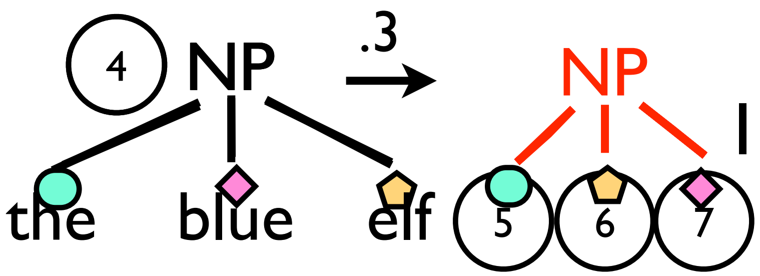
1

(Kuich, 1998)

Weighted tree transducers

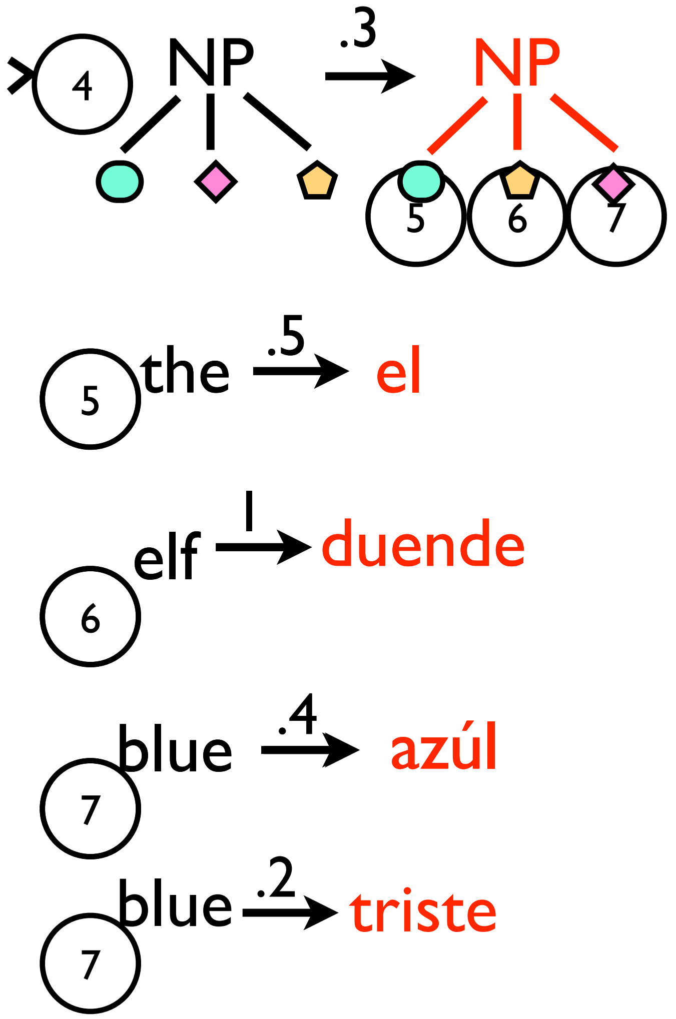


Tree Weight

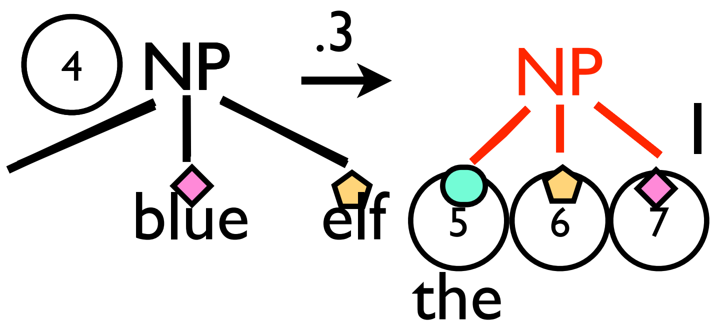


(Kuich, 1998)

Weighted tree transducers

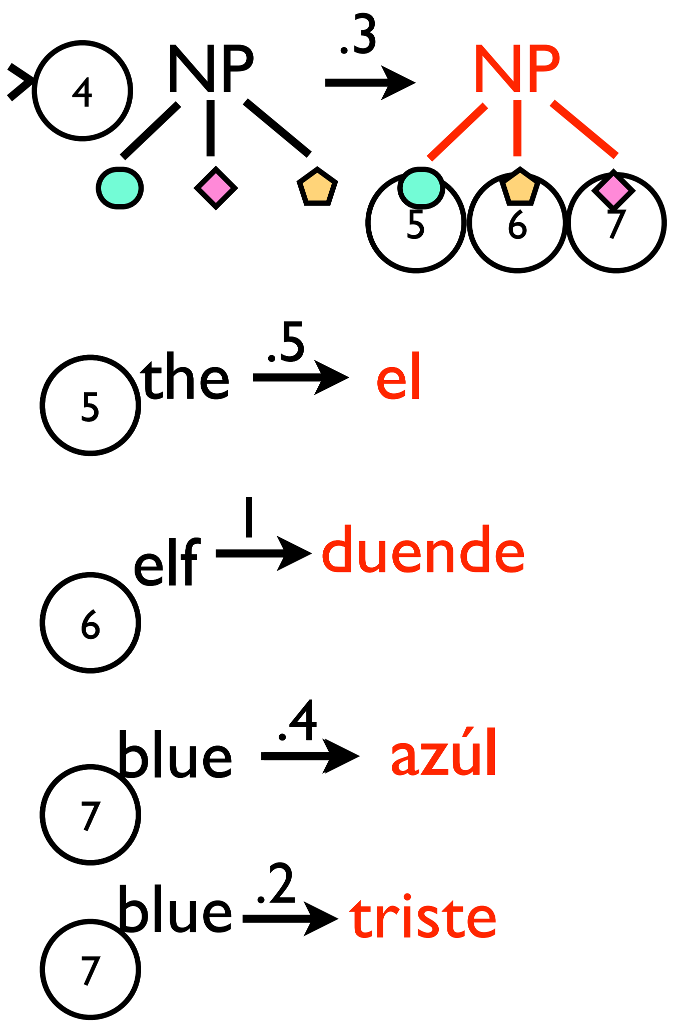


Tree Weight

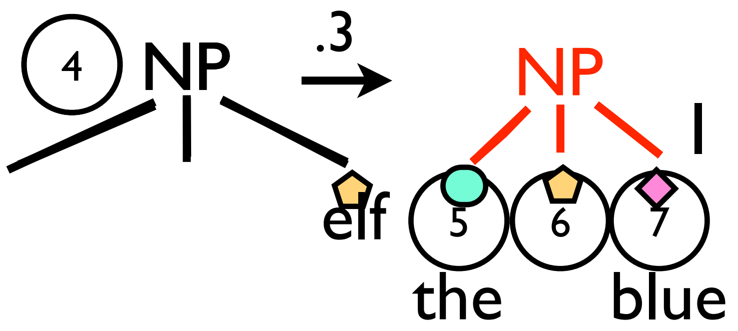


(Kuich, 1998)

Weighted tree transducers

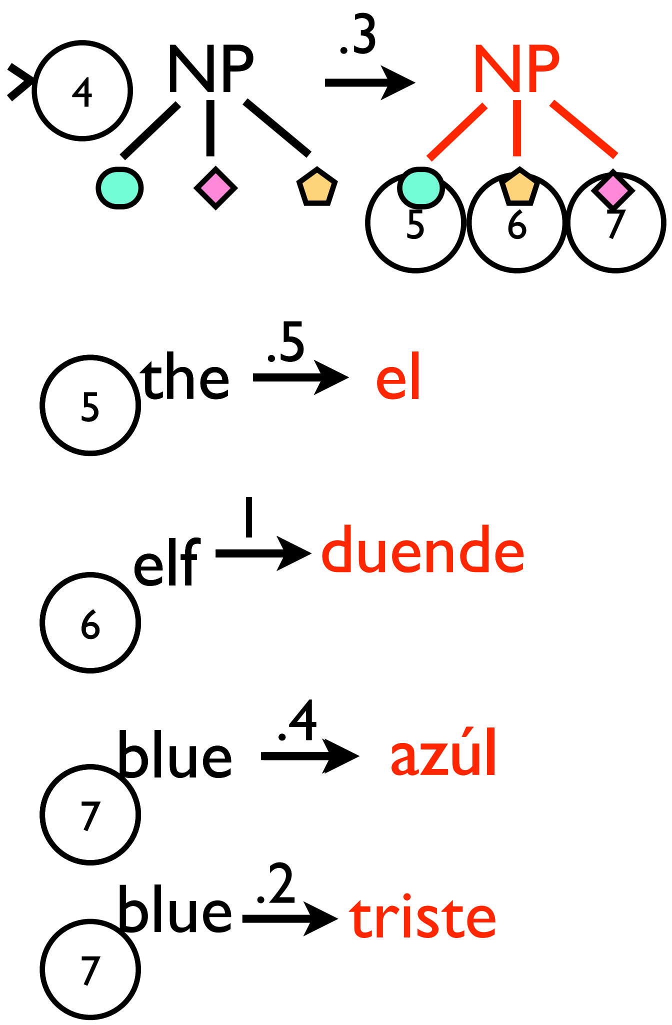


Tree Weight



(Kuich, 1998)

Weighted tree transducers



4 NP .3 → NP

5 6 7

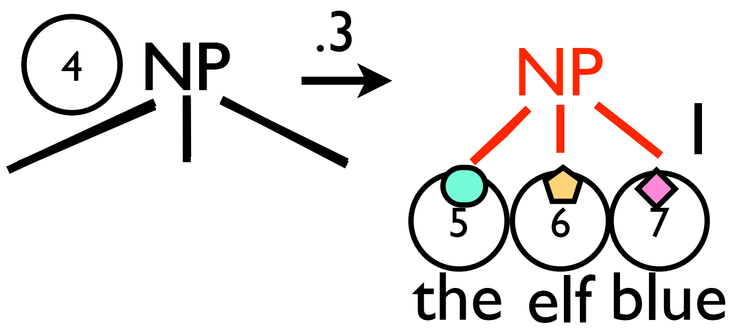
5 the .5 → el

6 elf 1 → duende

7 blue .4 → azúl

7 blue .2 → triste

Tree Weight



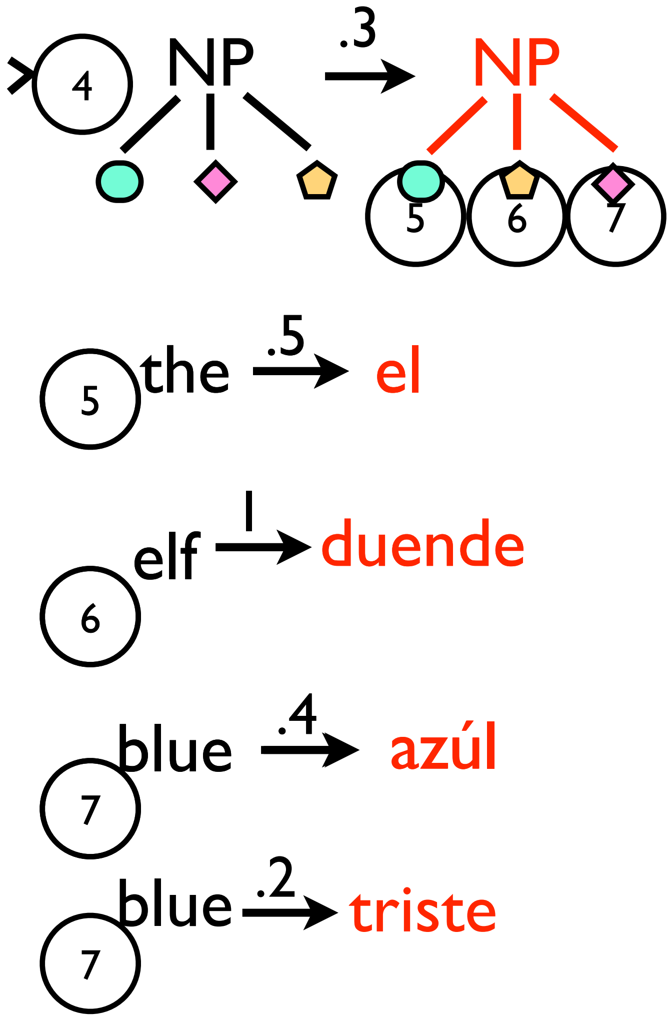
4 NP .3 → NP 1

5 6 7

the elf blue

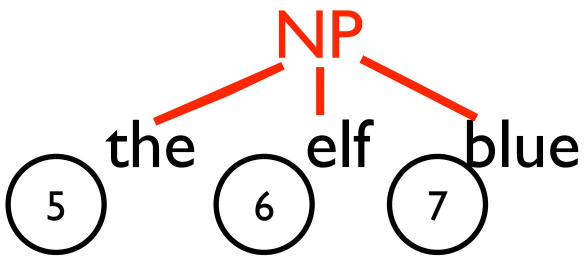
(Kuich, 1998)

Weighted tree transducers



Tree

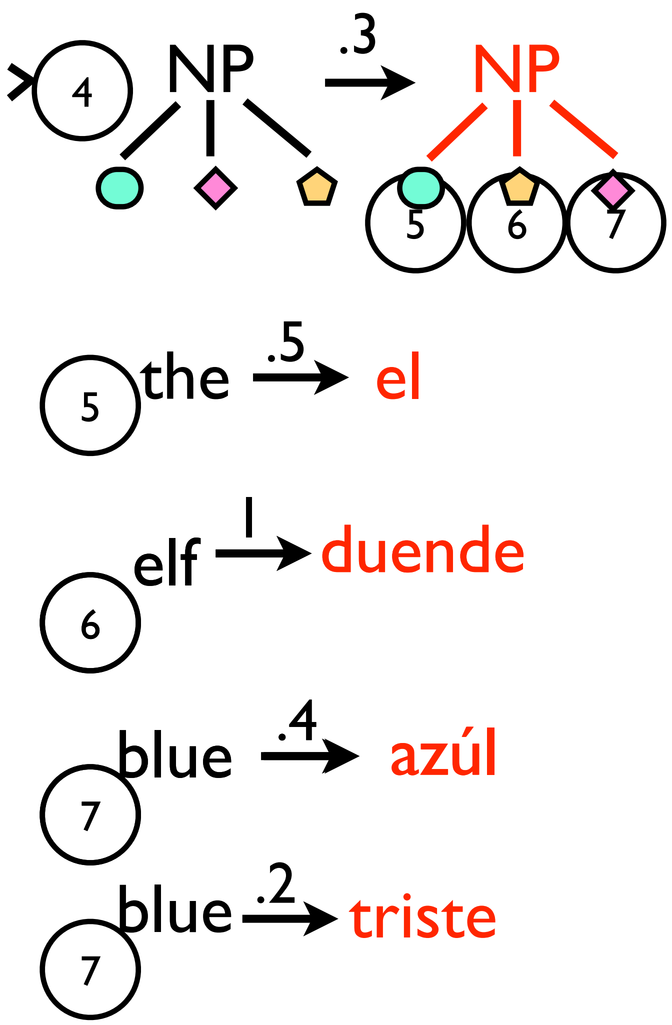
Weight



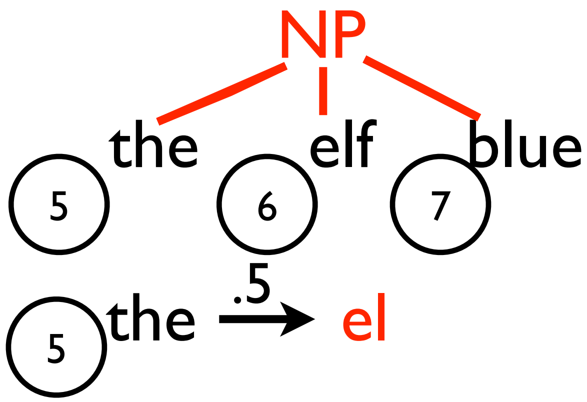
.3

(Kuich, 1998)

Weighted tree transducers



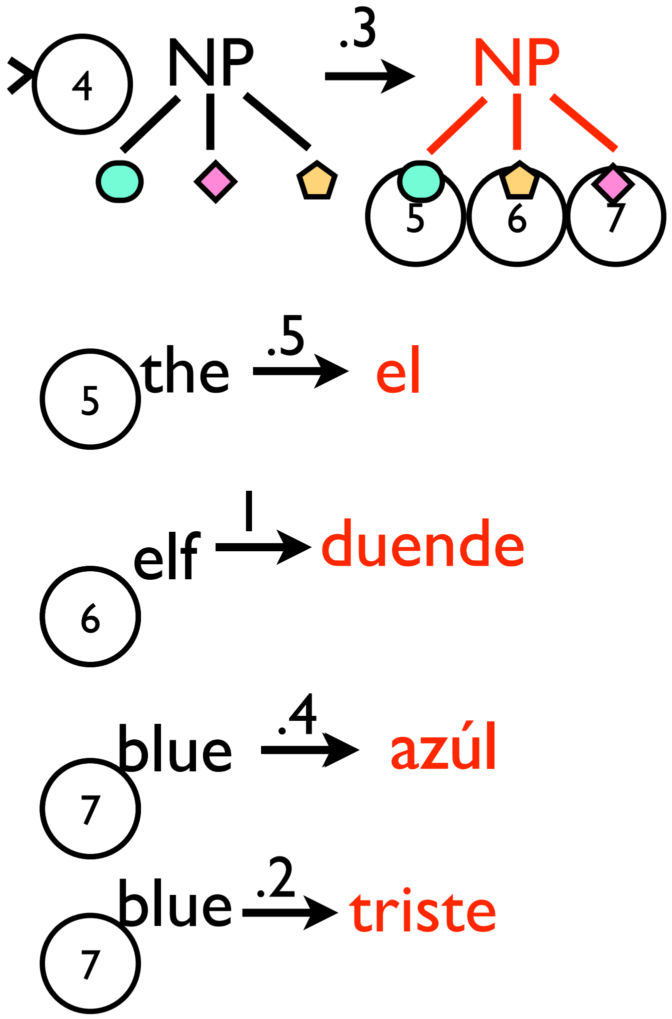
Tree Weight



.3

(Kuich, 1998)

Weighted tree transducers



4 NP .3 → NP 5 6 7

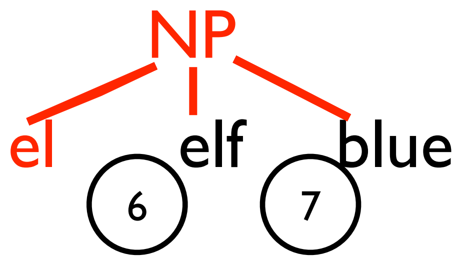
5 the .5 → el

6 elf 1 → duende

7 blue .4 → azúl

7 blue .2 → triste

Tree Weight

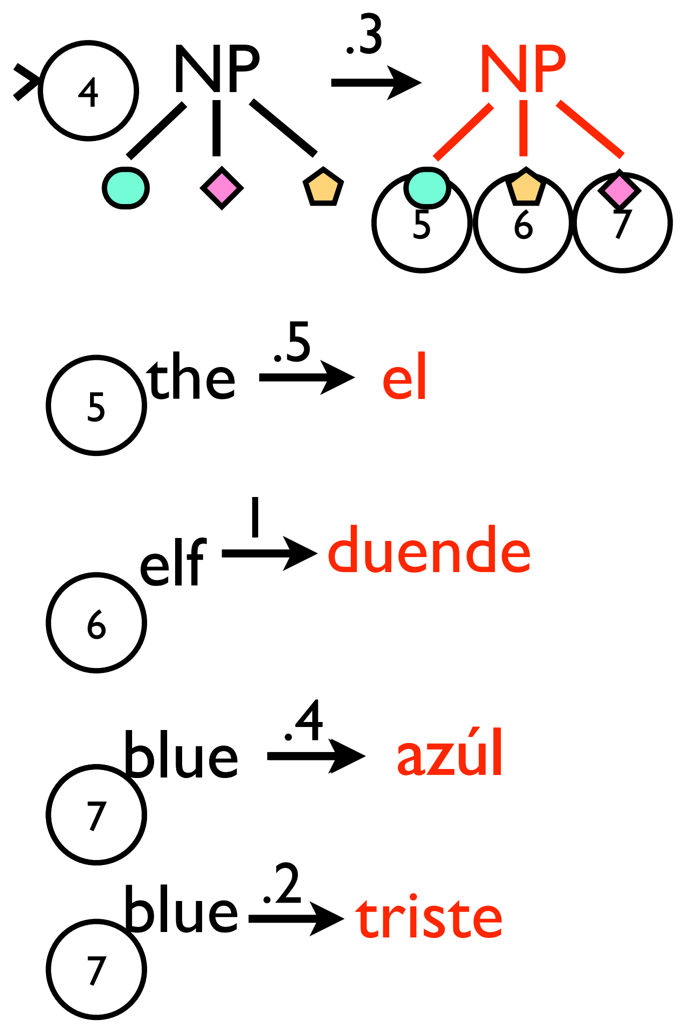


NP el elf blue 6 7

.15

(Kuich, 1998)

Weighted tree transducers



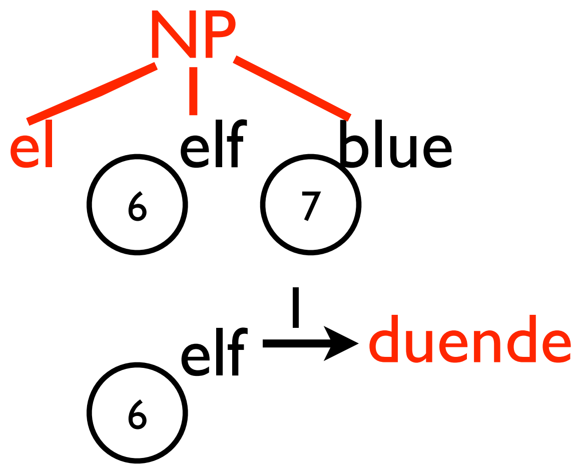
4 NP $\xrightarrow{.3}$ NP 5 6 7

5 the $\xrightarrow{.5}$ el

6 elf $\xrightarrow{1}$ duende

7 blue $\xrightarrow{.4}$ azúl

7 blue $\xrightarrow{.2}$ triste

Tree	Weight
	.15

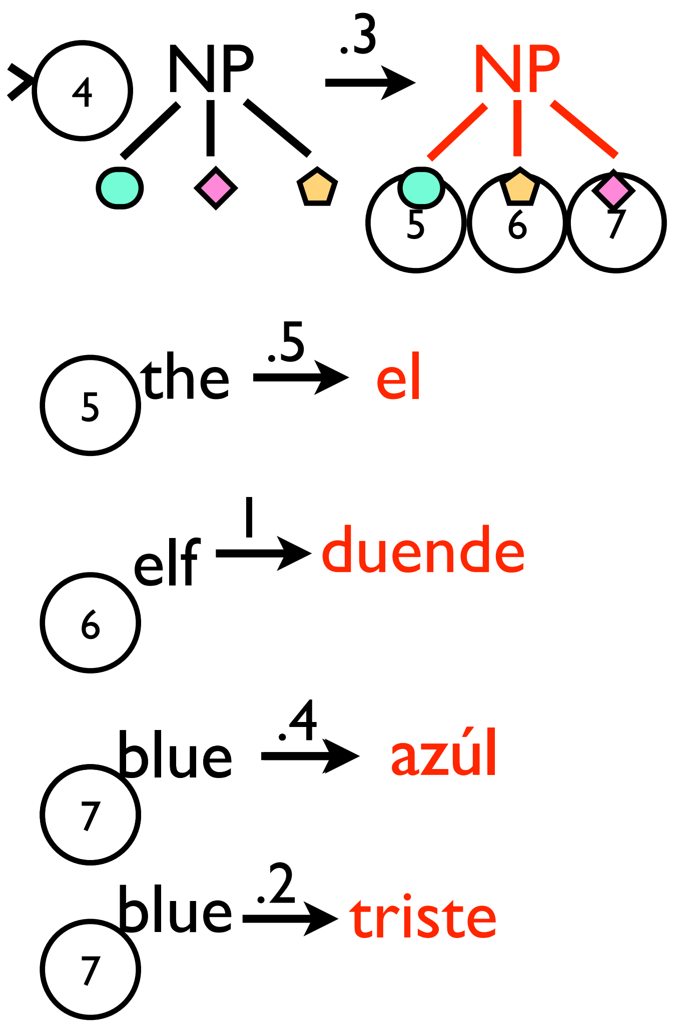
NP

el 6 elf 7 blue

6 elf $\xrightarrow{1}$ duende

(Kuich, 1998)

Weighted tree transducers



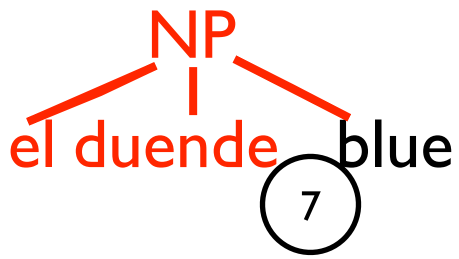
4 NP $\xrightarrow{.3}$ NP 5 6 7

5 the $\xrightarrow{.5}$ el

6 elf $\xrightarrow{1}$ duende

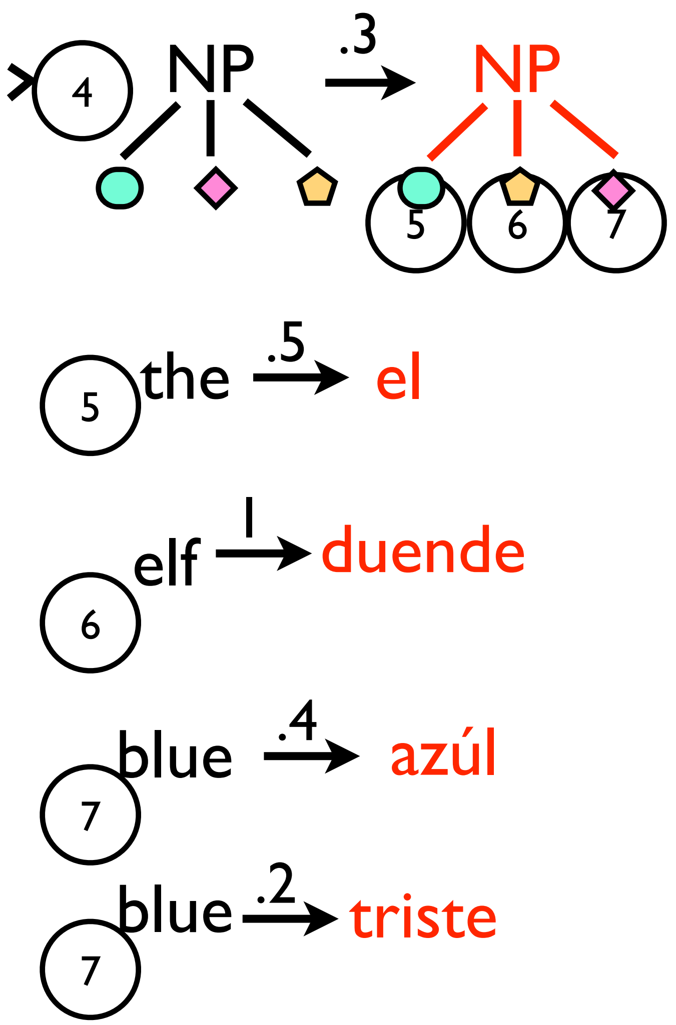
7 blue $\xrightarrow{.4}$ azúl

7 blue $\xrightarrow{.2}$ triste

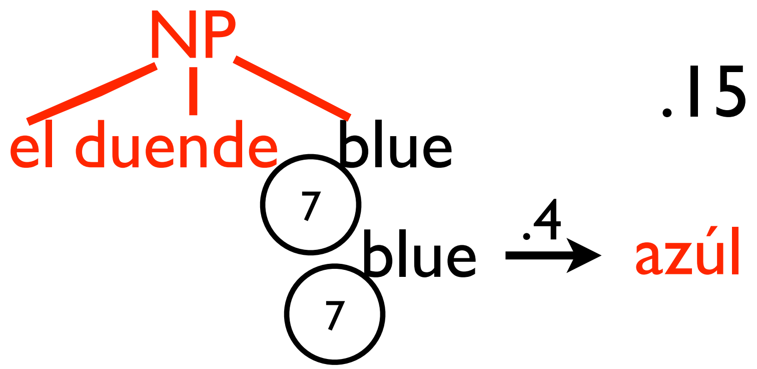
Tree	Weight
 NP el duende 7 blue	.15

(Kuich, 1998)

Weighted tree transducers



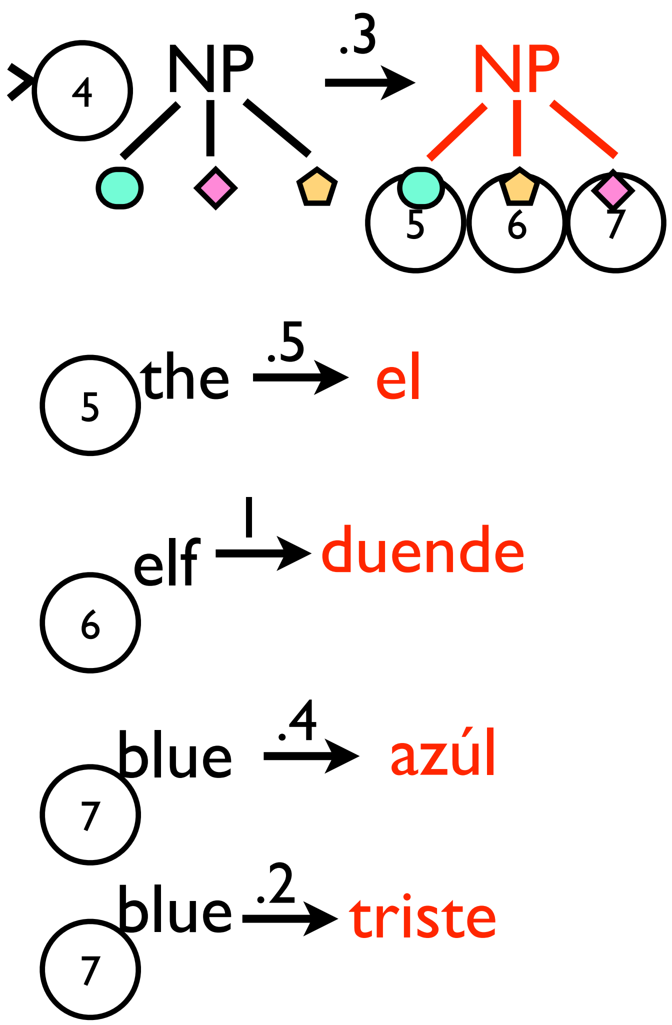
Tree | Weight

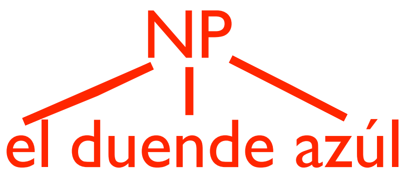


.15

(Kuich, 1998)

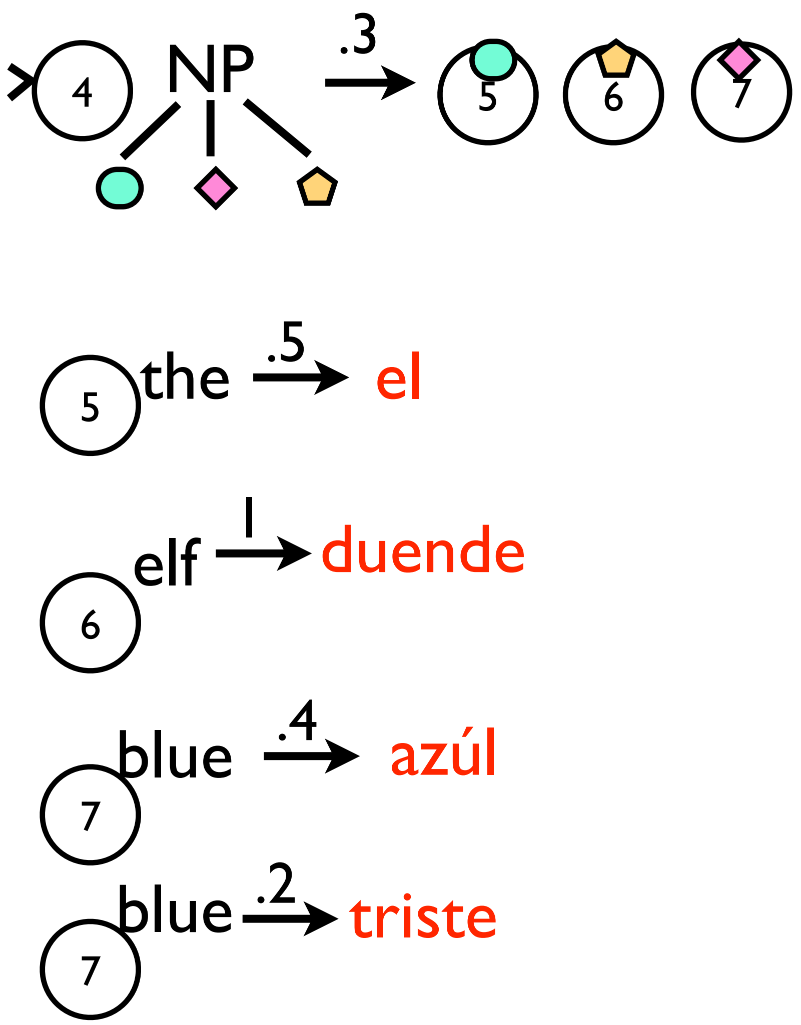
Weighted tree transducers

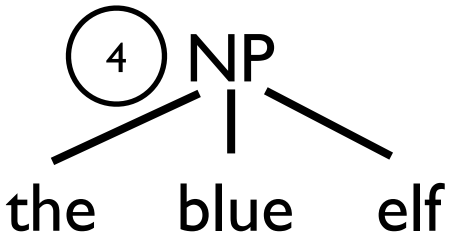


Tree	Weight
	.06

(Kuich, 1998)

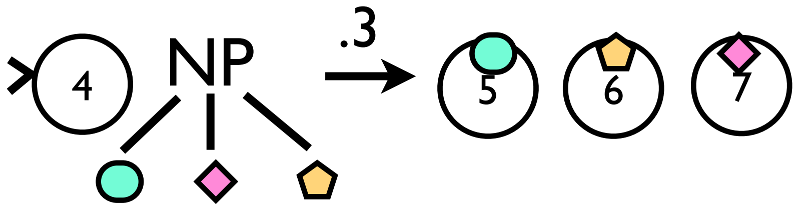
Weighted tree-string transducers



Tree	Weight
	1

(Kuich, 1998)

Weighted tree-string transducers



5 the $\xrightarrow{.5}$ el

6 elf $\xrightarrow{1}$ duende

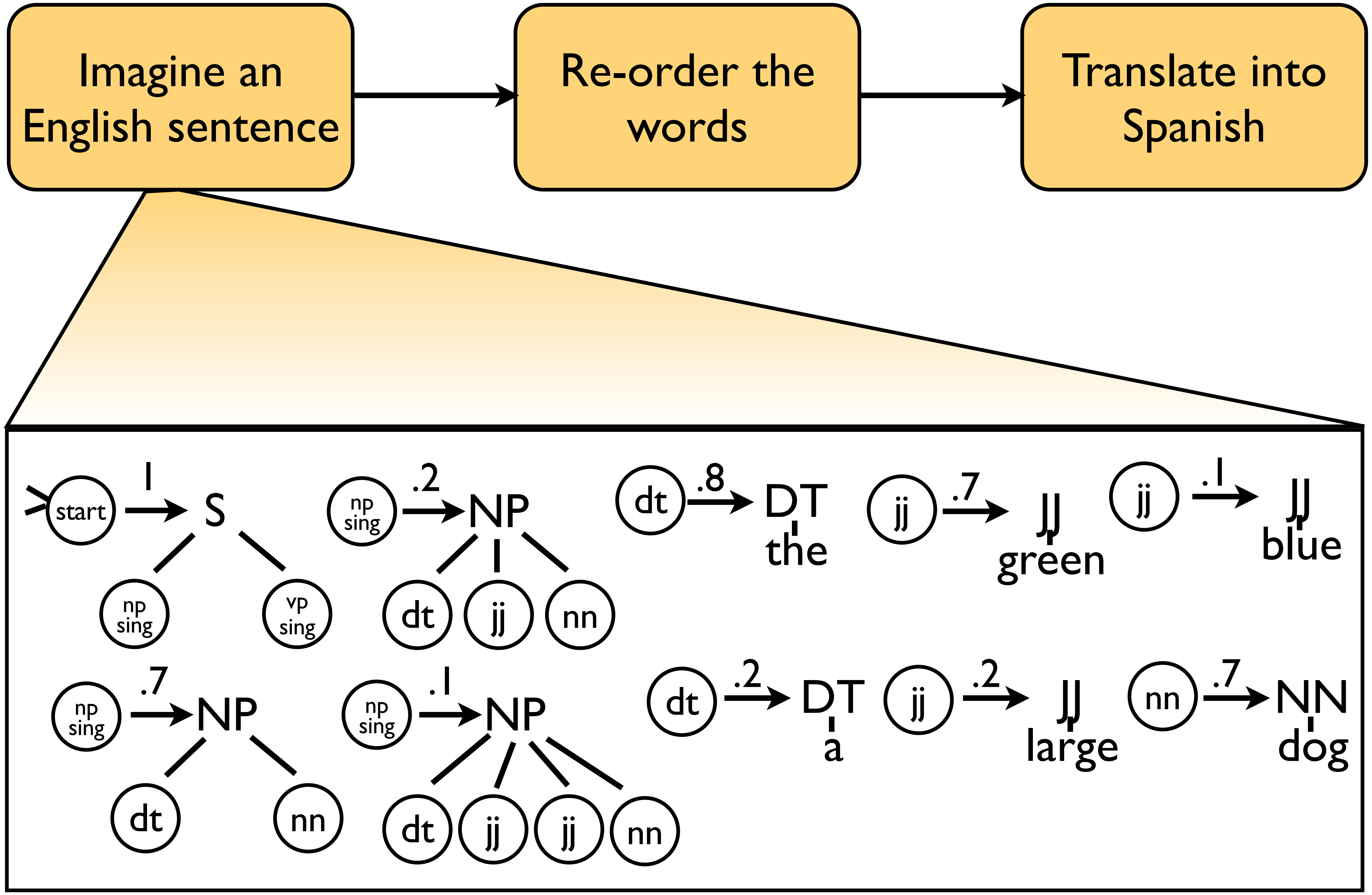
7 blue $\xrightarrow{.4}$ azúl

7 blue $\xrightarrow{.2}$ triste

String	Weight
el duende azúl	.06

(Kuich, 1998)

MT as weighted tree transducers

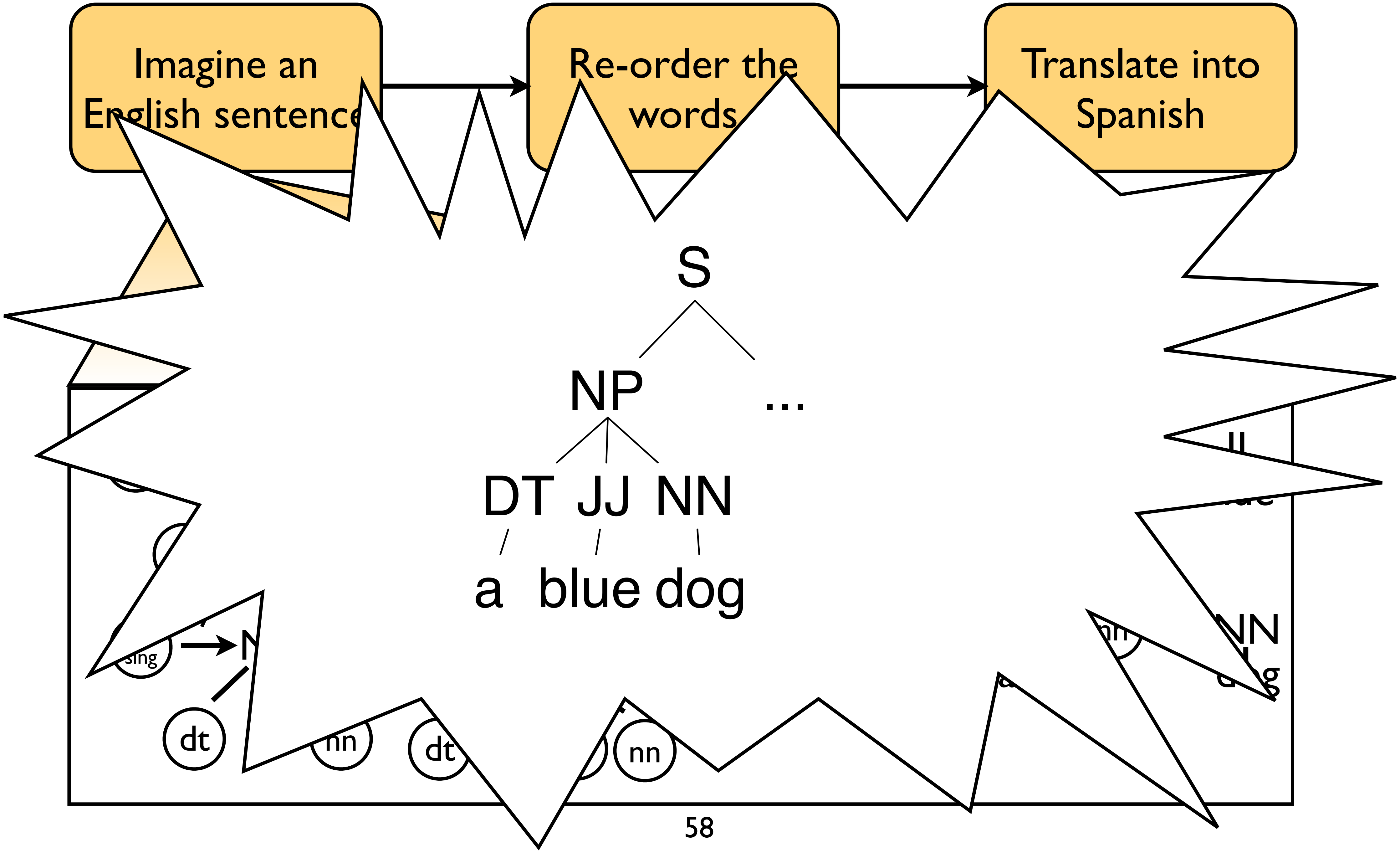


Imagine an English sentence

Re-order the words

Translate into Spanish

MT as weighted tree transducers

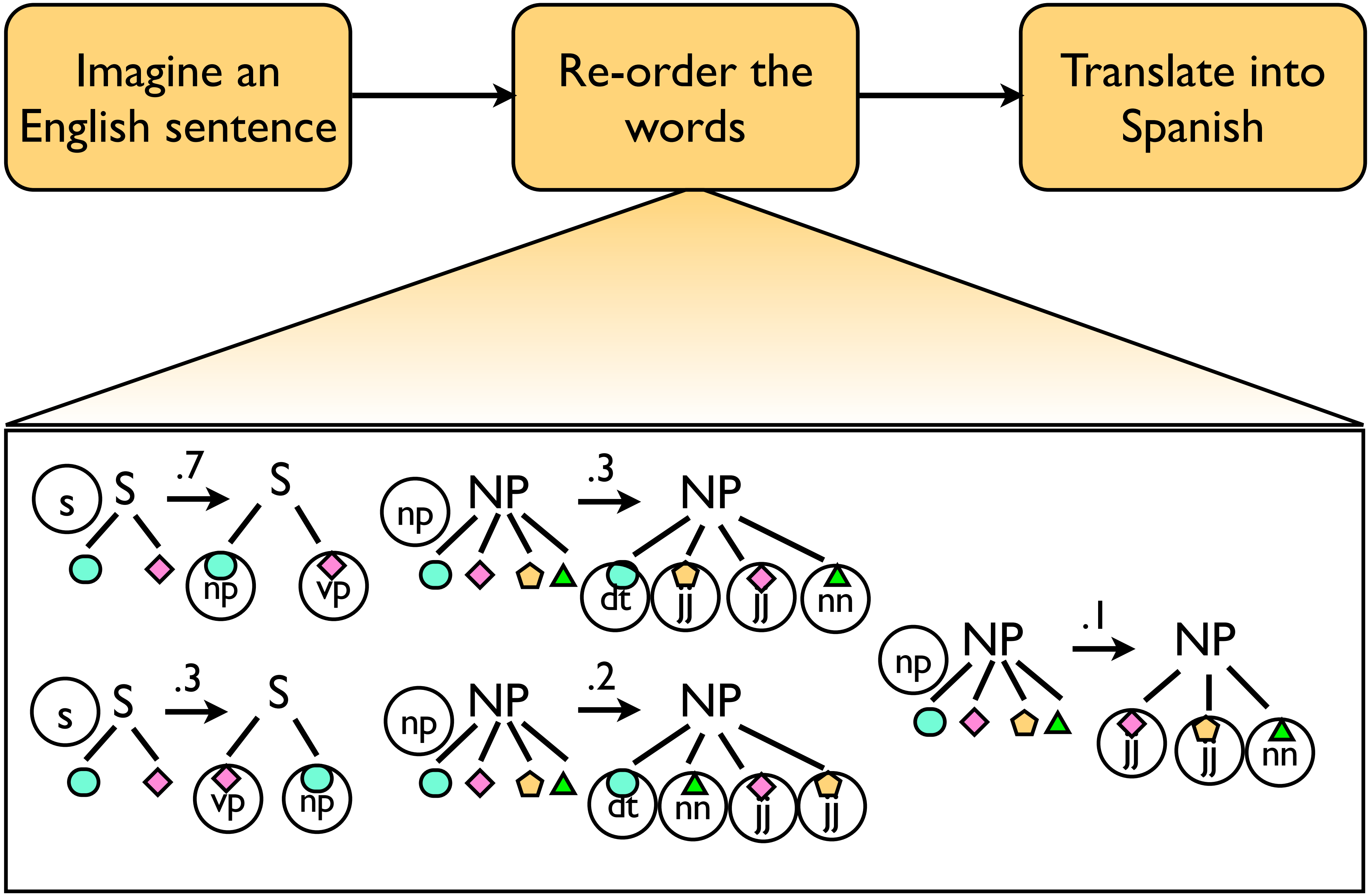


Imagine an English sentence

Re-order the words

Translate into Spanish

MT as weighted tree transducers

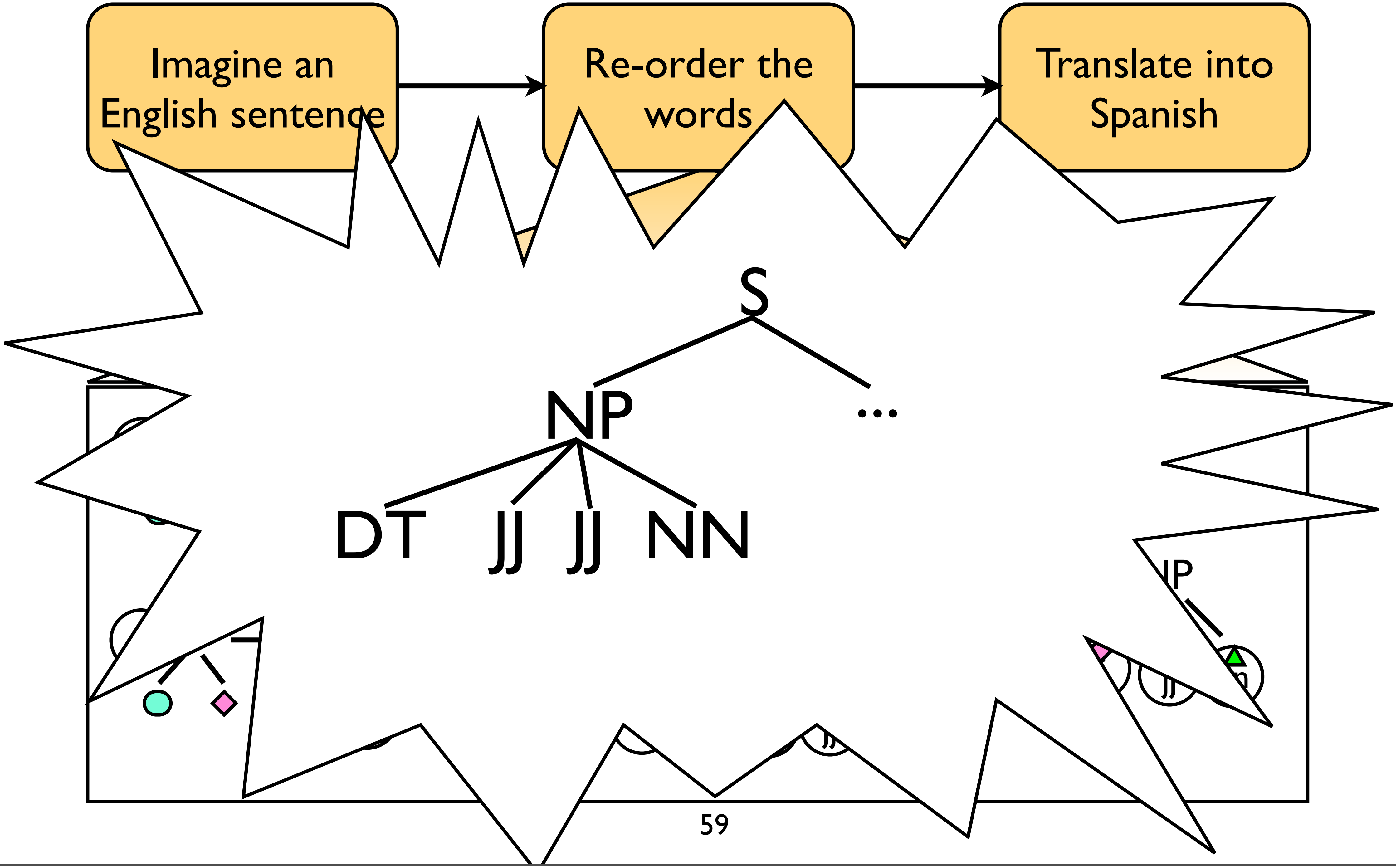


Imagine an English sentence

Re-order the words

Translate into Spanish

MT as weighted tree transducers

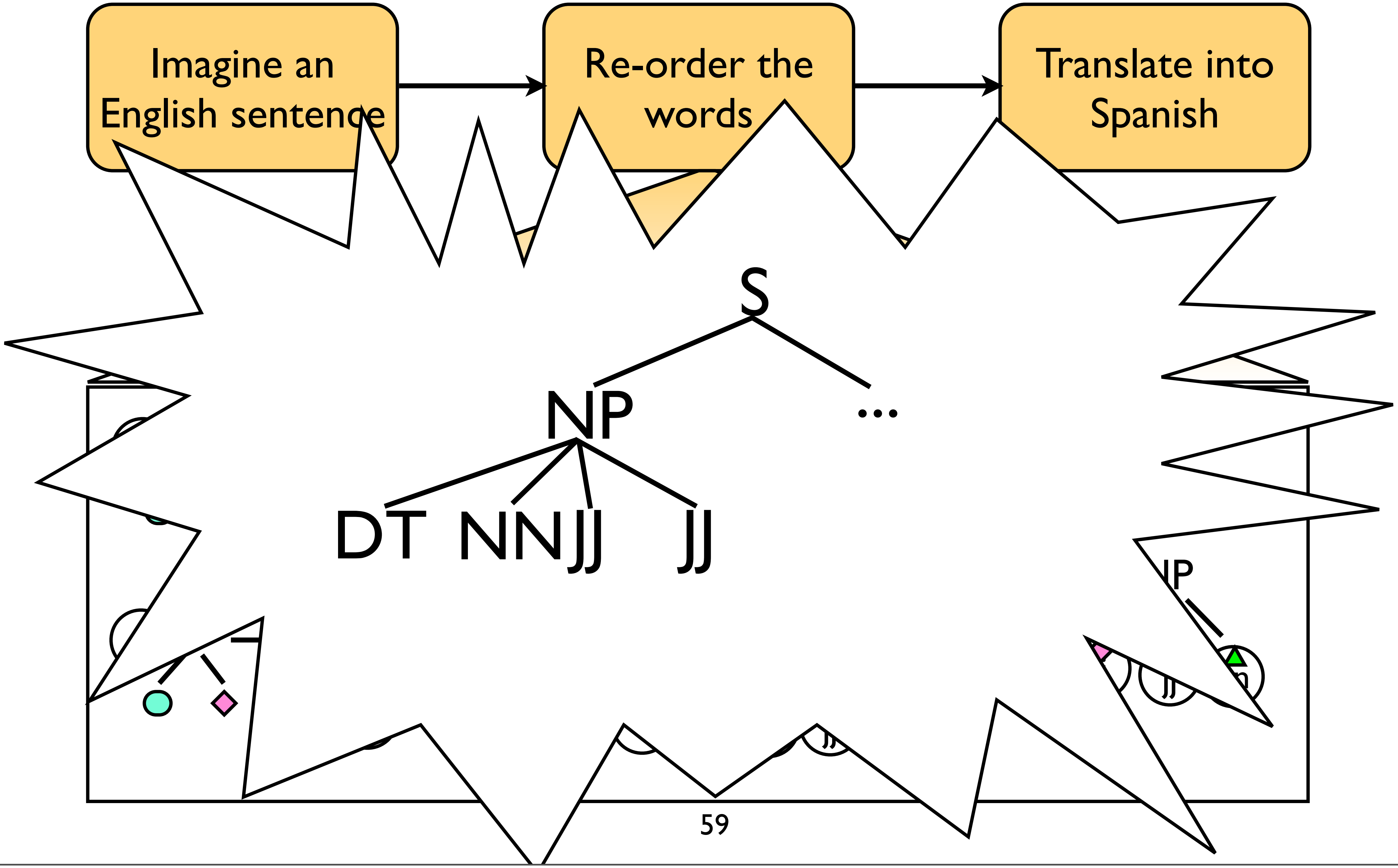


Imagine an English sentence

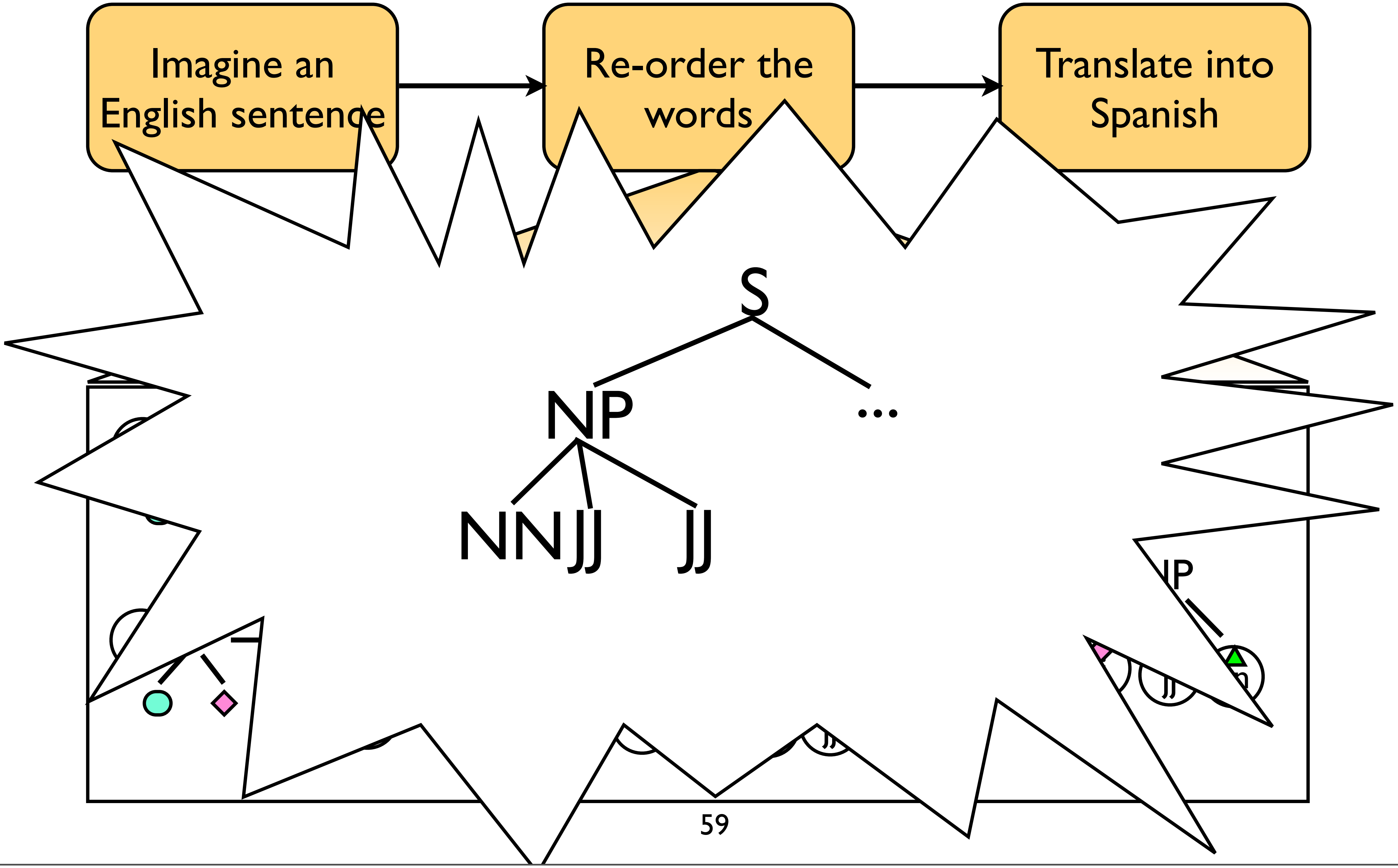
Re-order the words

Translate into Spanish

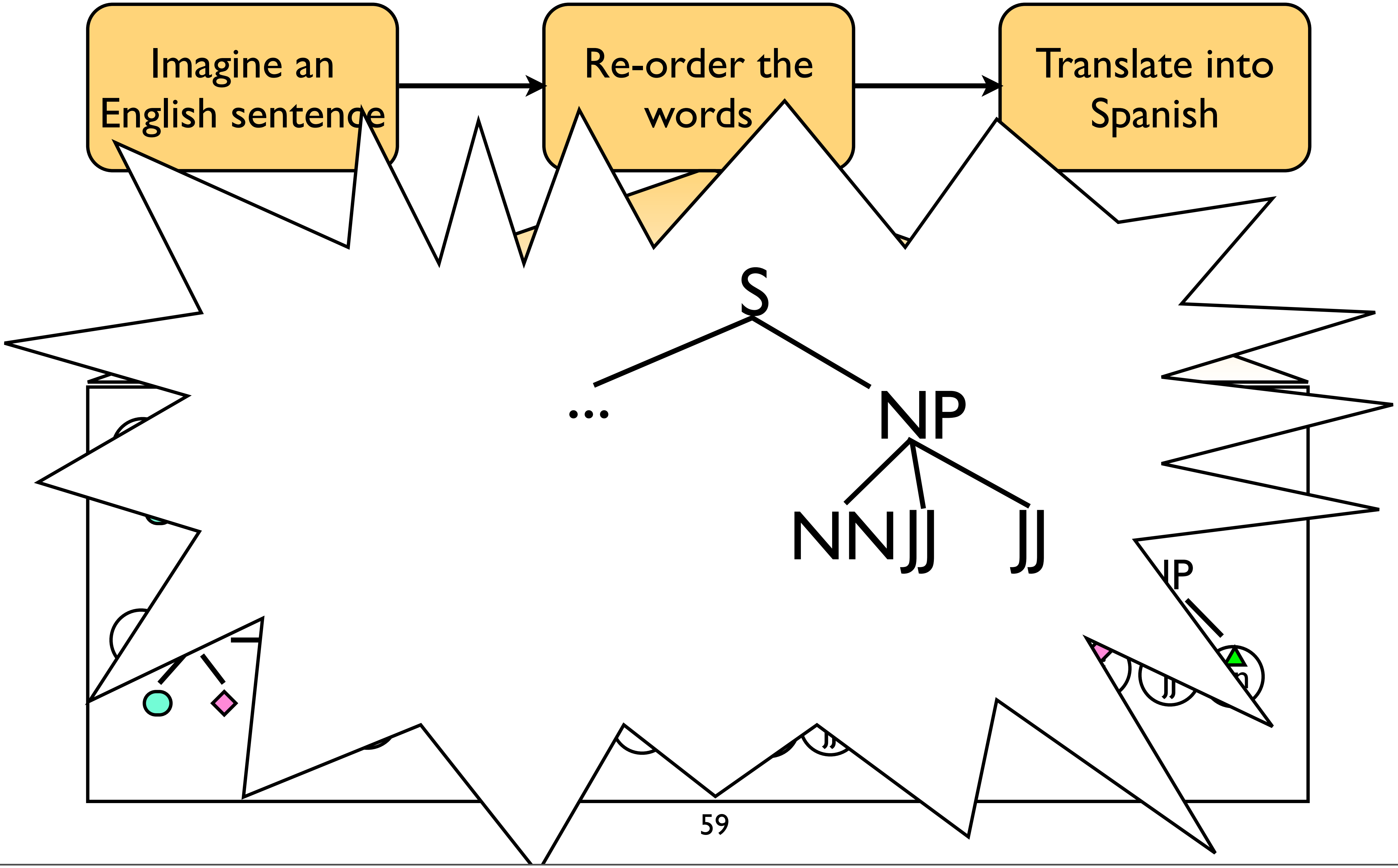
MT as weighted tree transducers



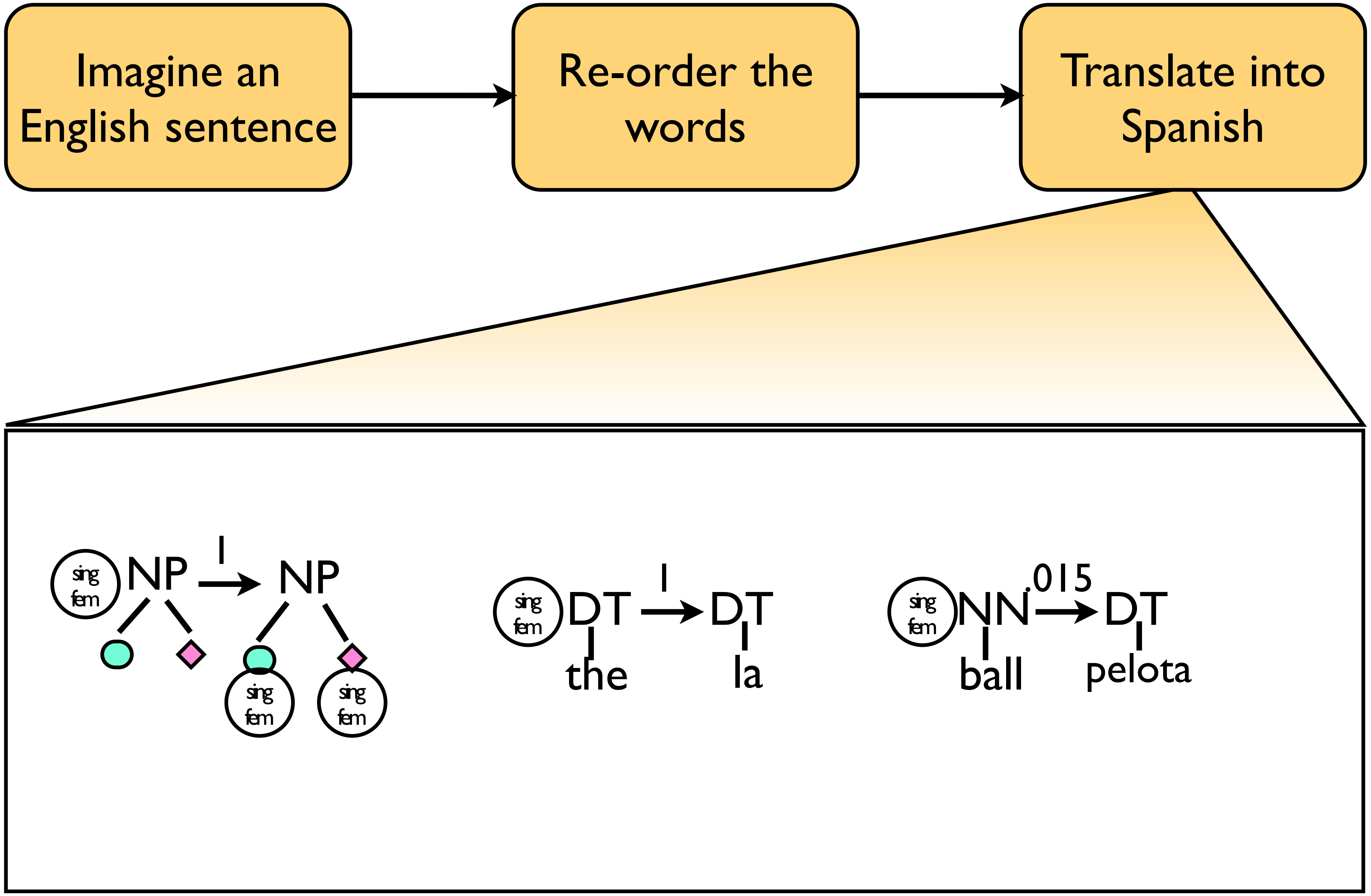
MT as weighted tree transducers



MT as weighted tree transducers



MT as weighted tree transducers

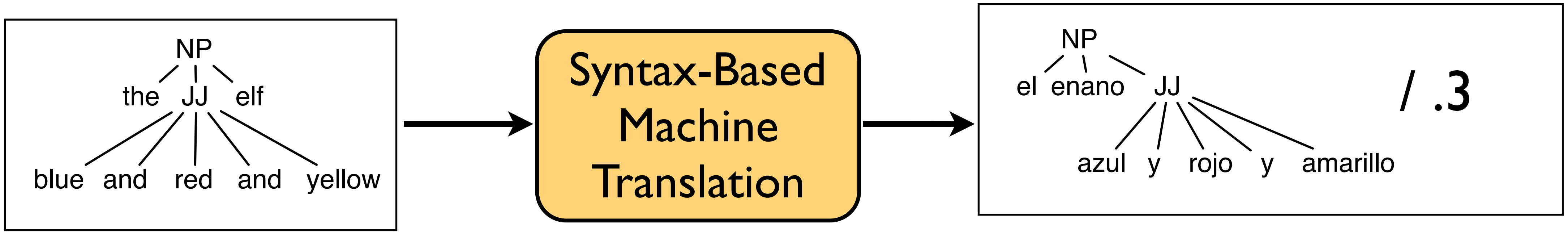


Imagine an English sentence

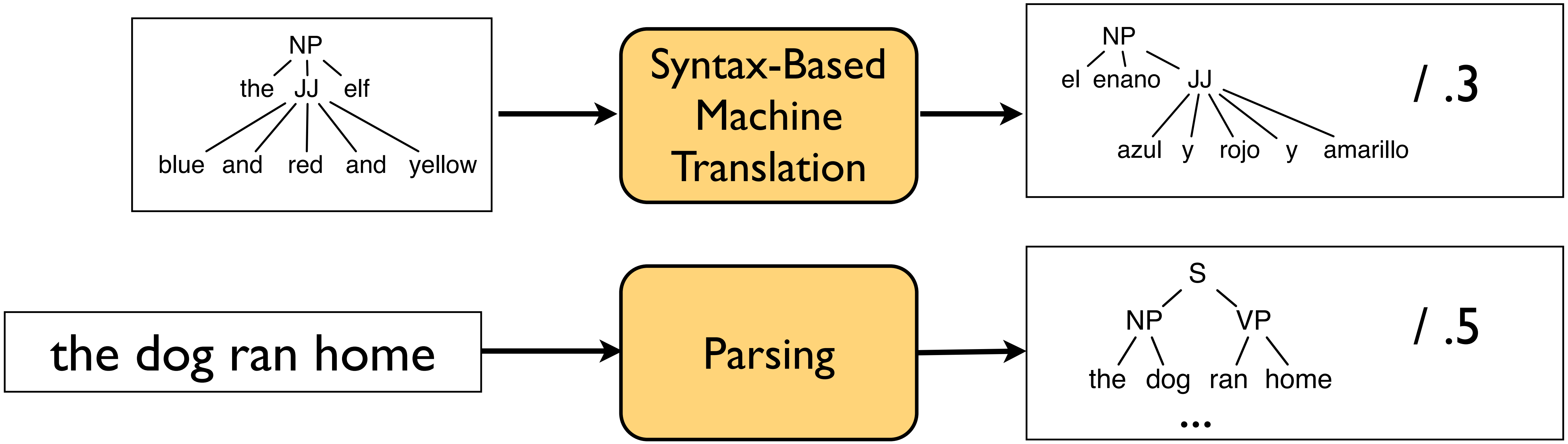
Re-order the words

Translate into Spanish

Great, so now we can solve harder problems!



Great, so now we can solve harder problems!

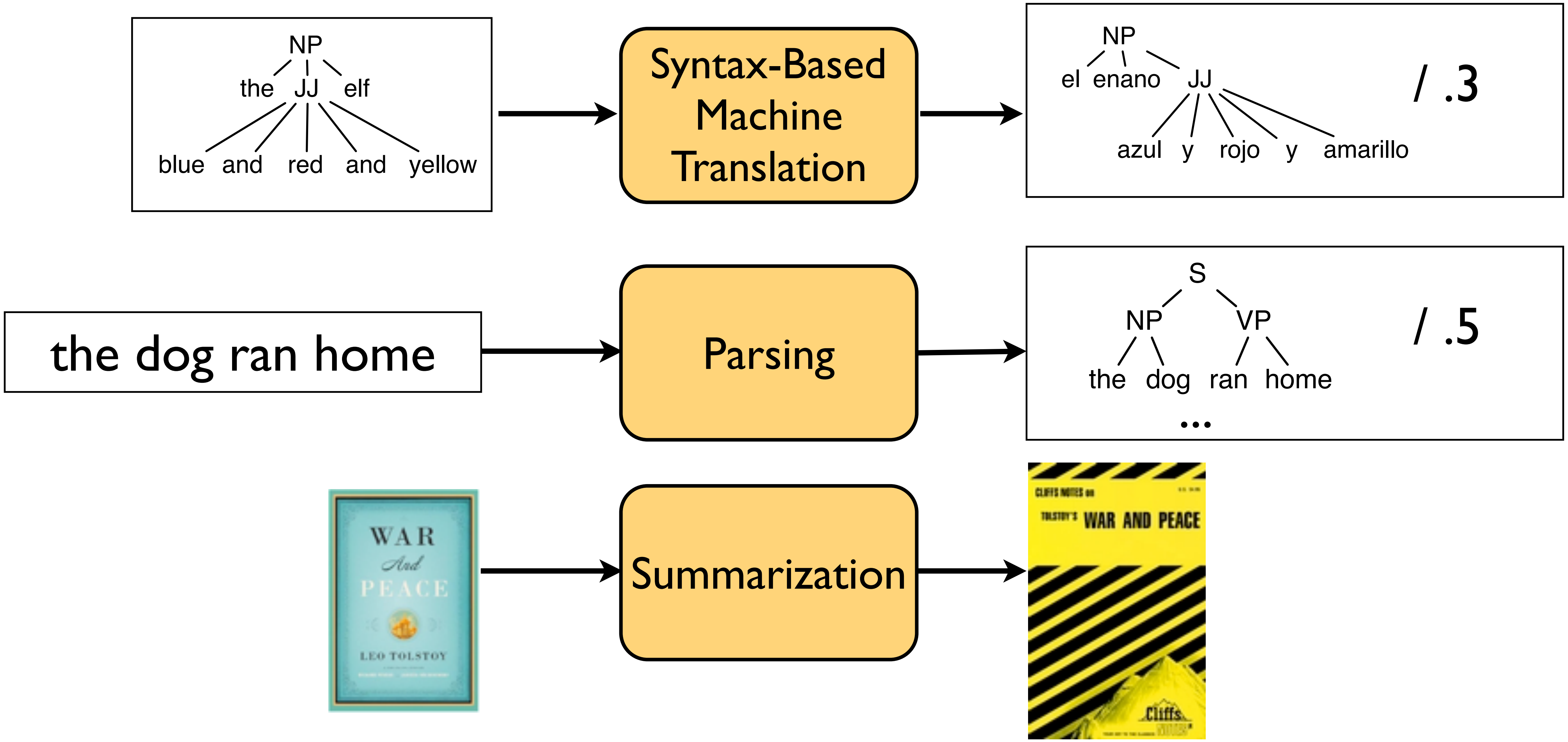


the dog ran home

Syntax-Based Machine Translation

Parsing

Great, so now we can solve harder problems!



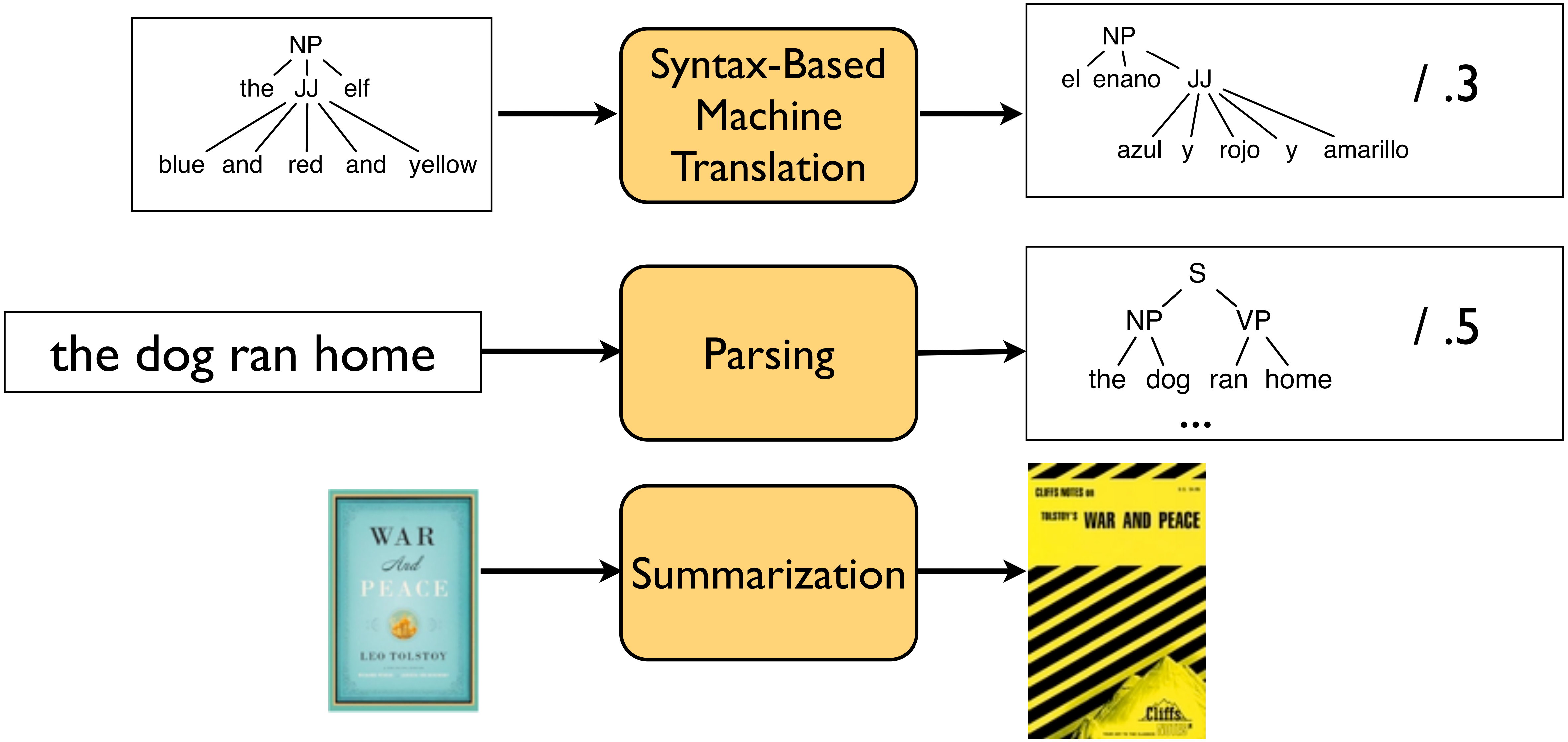
Syntax-Based Machine Translation

the dog ran home

Parsing

Summarization

Great, so now we can solve harder problems!



the dog ran home

Syntax-Based Machine Translation

Parsing

Summarization

Not so fast!

String world has many more available operations than tree world!

Operation	String	Tree
k-best	yes	alg[1]
em training	yes	alg[2]
determinization	yes	no
composition	yes	proof of concept[3]
pipeline inference	yes	proof of concept[4]
on-the-fly inference	yes	no

[1] Huang & Chiang, 2005

[2] Graehl & Knight, 2004

[3] Maletti, 2006

[4] Fülöp, Maletti, Vogler, 2010

Algorithmic contribution I: weighted determinization

Operation	String	Tree
k-best	yes	alg
em training	yes	alg
determinization	yes	alg
composition	yes	proof of concept[3]
pipeline inference	yes	proof of concept[4]
on-the-fly inference	yes	no

Algorithmic I

Algorithmic contribution II: efficient inference

Operation	String	Tree
k-best	yes	alg
em training	yes	alg
determinization	yes	alg
composition	yes	alg
pipeline inference	yes	alg
on-the-fly inference	yes	alg

Algorithmic I

Algorithmic II

Practical contribution I: weighted tree transducer toolkit

Practical I

Operation	String	Tree
k-best	yes	yes
em training	yes	yes
determinization	yes	yes
composition	yes	yes
pipeline inference	yes	yes
on-the-fly inference	yes	yes

Algorithmic I

Algorithmic II

Practical contribution II: syntactic re-alignment

Practical I

Practical II

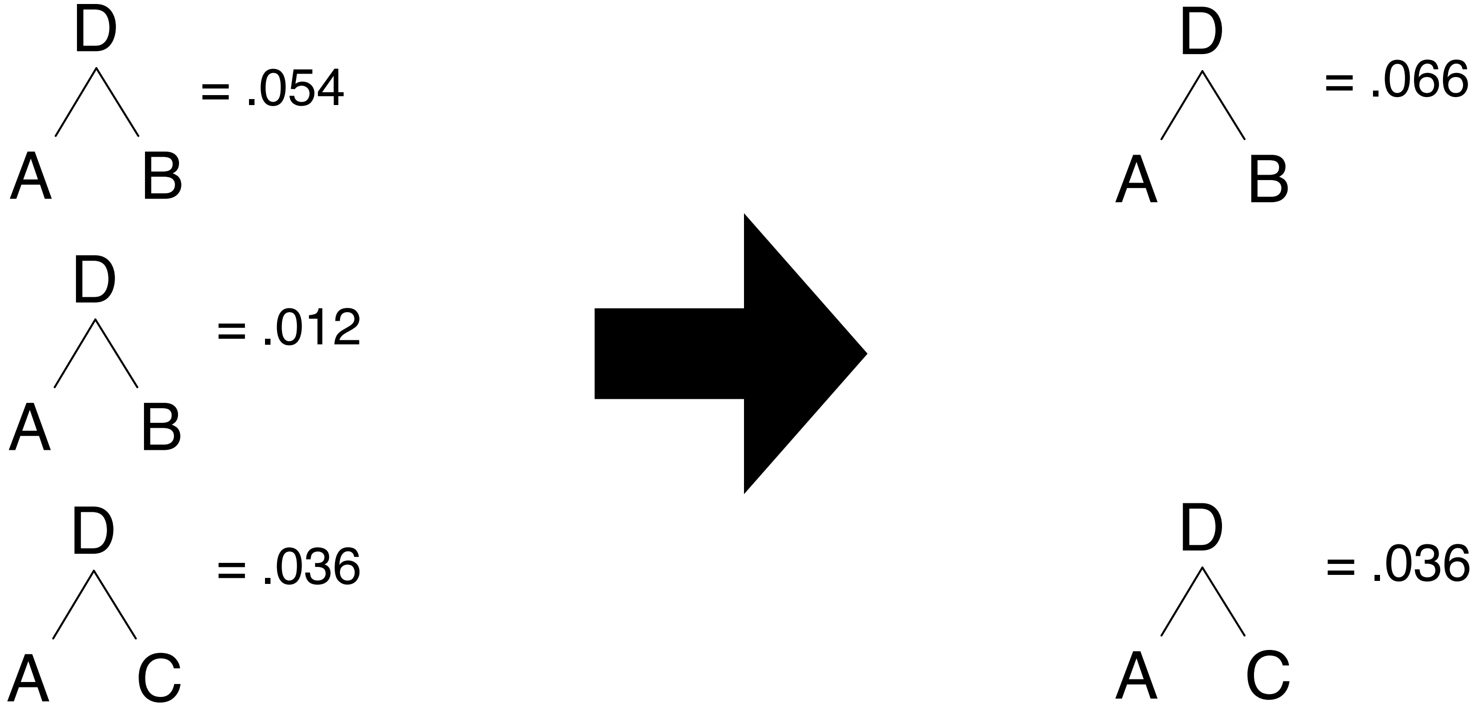
Algorithmic I

Algorithmic II

Operation	String	Tree
k-best	yes	yes
em training	yes	yes
determinization	yes	yes
composition	yes	yes
pipeline inference	yes	yes
on-the-fly inference	yes	yes

Determinization of weighted tree automata

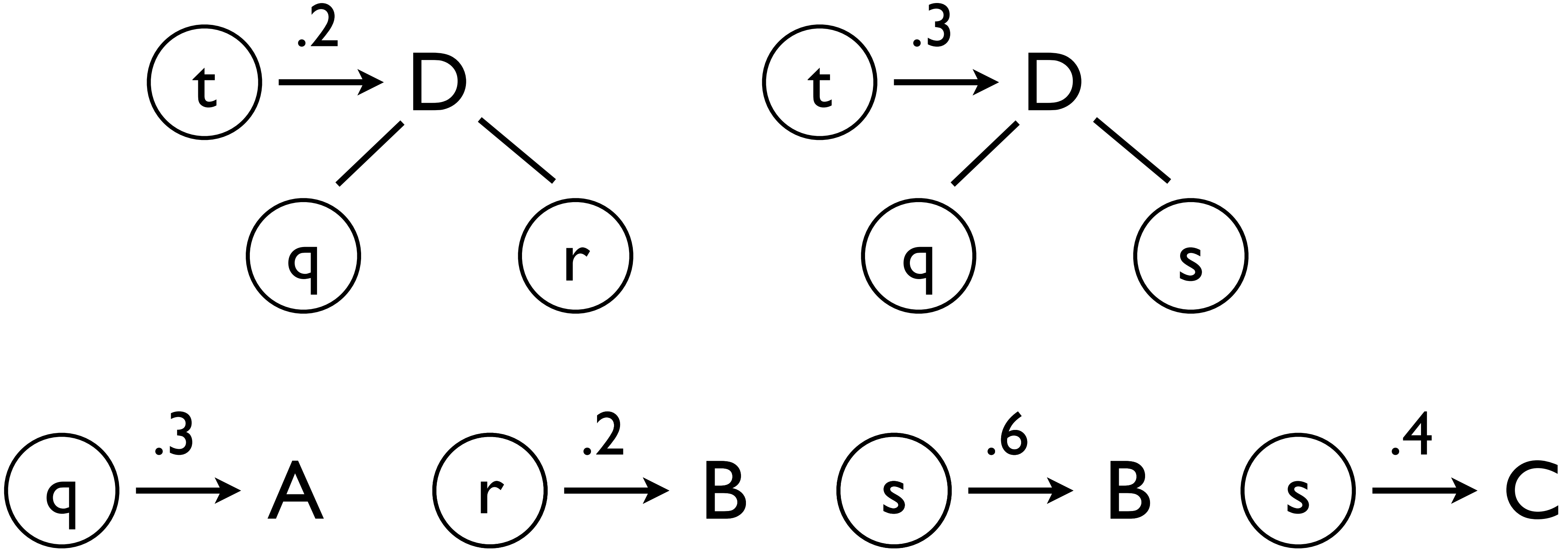
(May & Knight, HLT-NAACL '06)
(Büchse, May, Vogler, FSMNLP '09)



Elevated Mohri algorithm ('97) to tree automata
Demonstrated empirical gains in parsing and MT

Algorithmic Contribution I: WTA Determinization

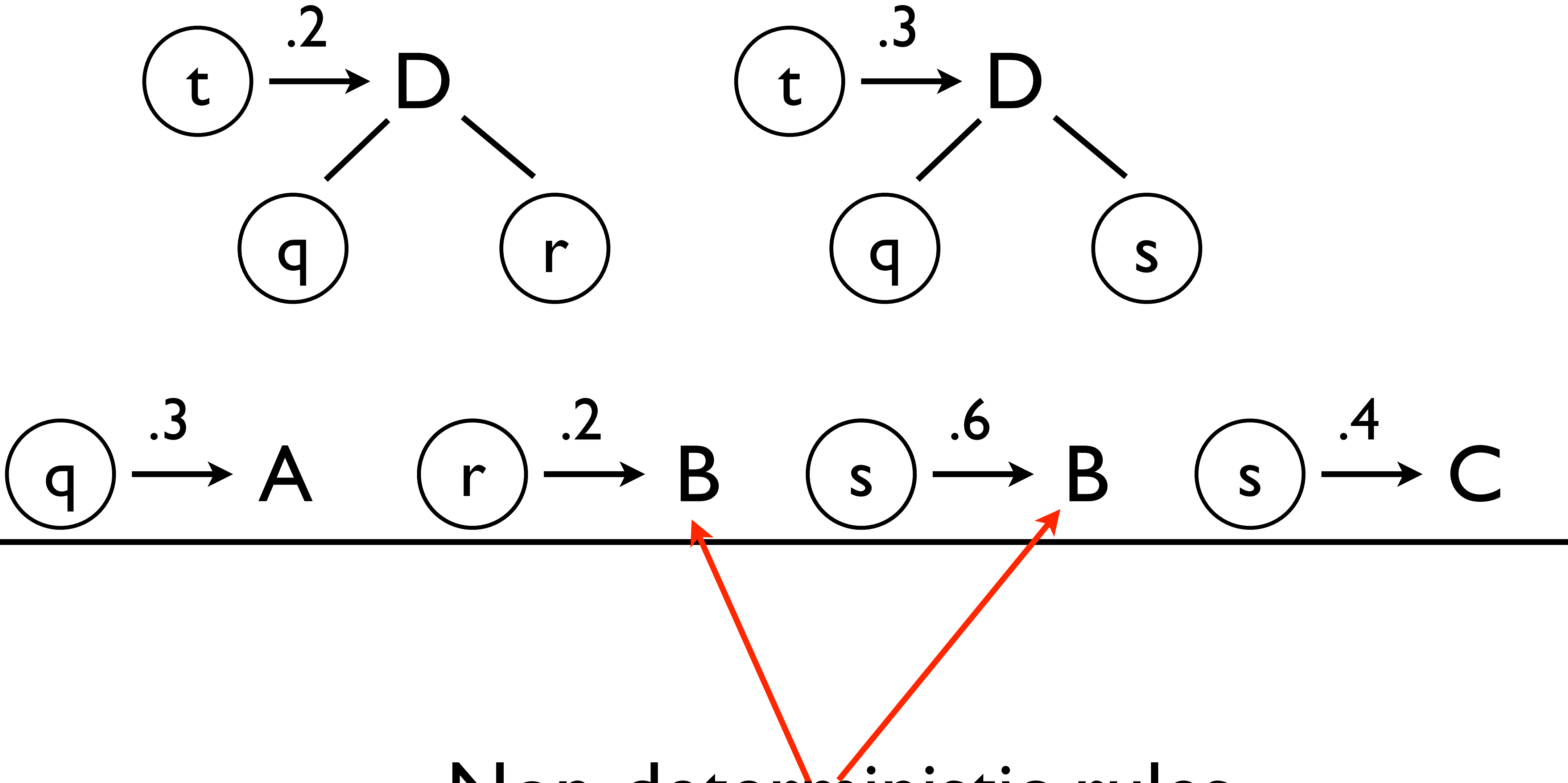
BEFORE



AFTER

Algorithmic Contribution I: WTA Determinization

BEFORE

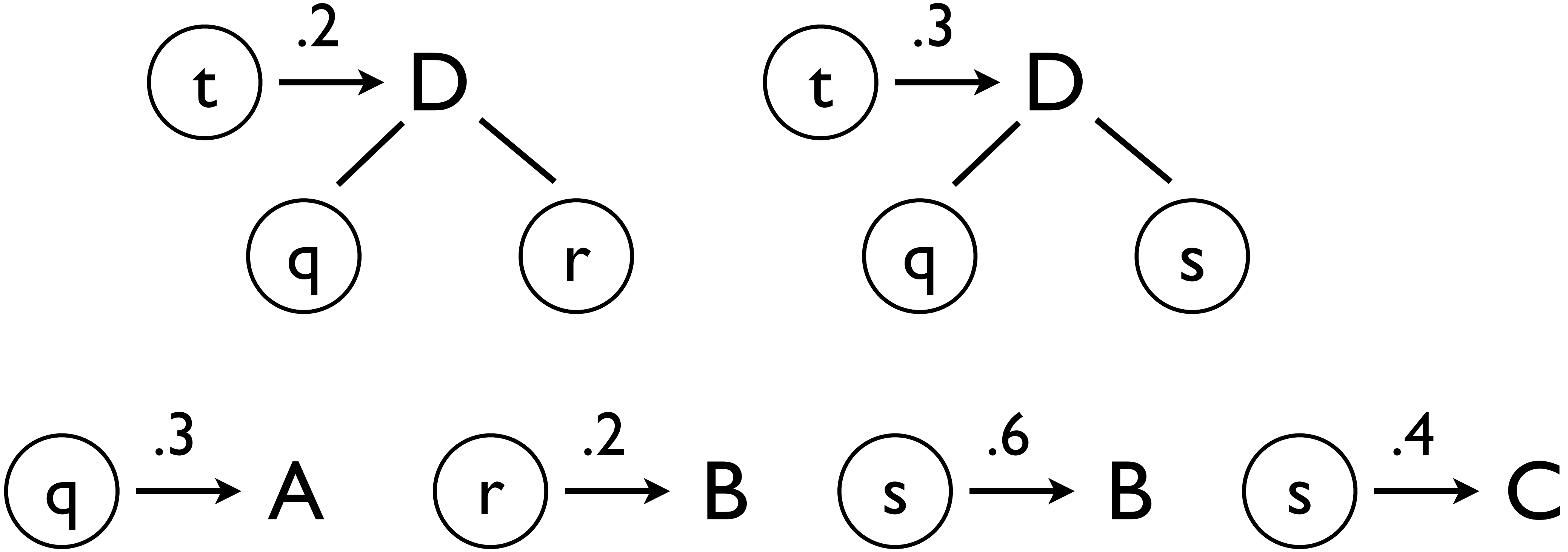


Non-deterministic rules
(treating grammar as bottom-up acceptor)

AFTER

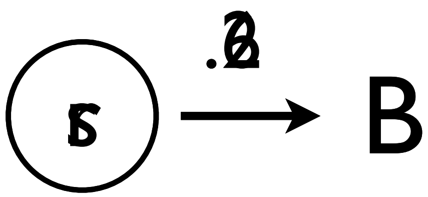
Algorithmic Contribution I: WTA Determinization

BEFORE



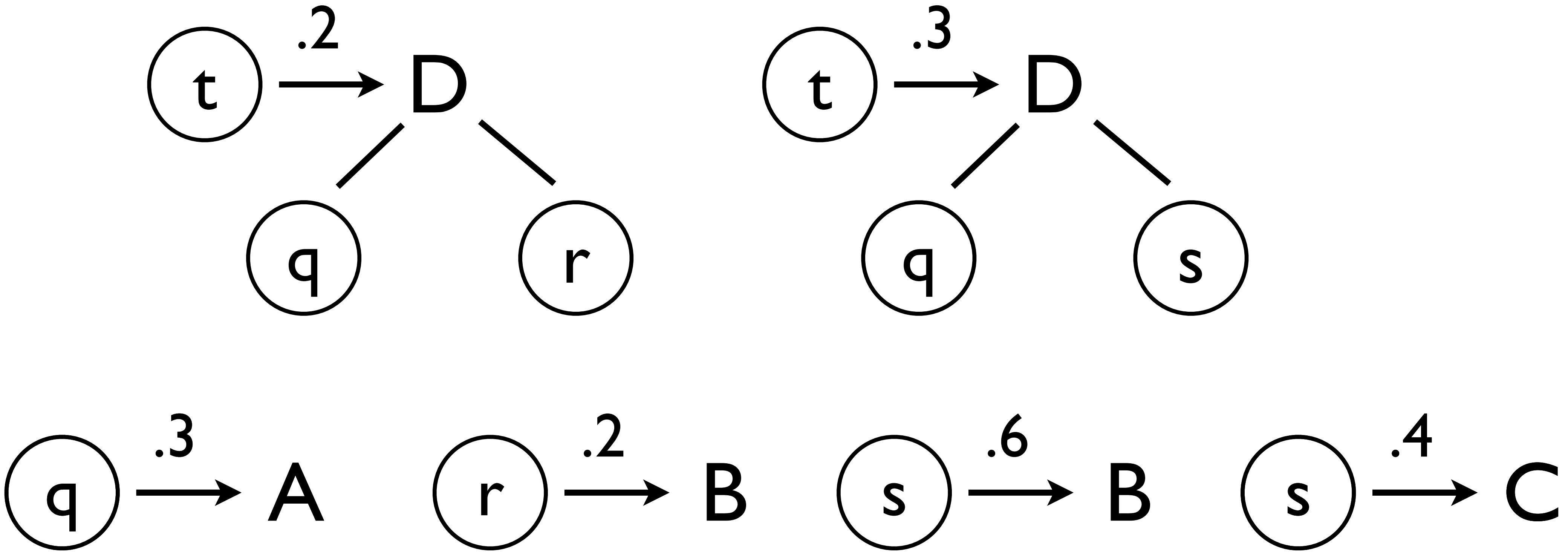
AFTER

Merge terminal rules with same right sides



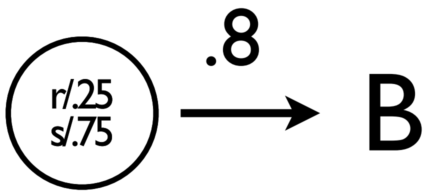
Algorithmic Contribution I: WTA Determinization

BEFORE

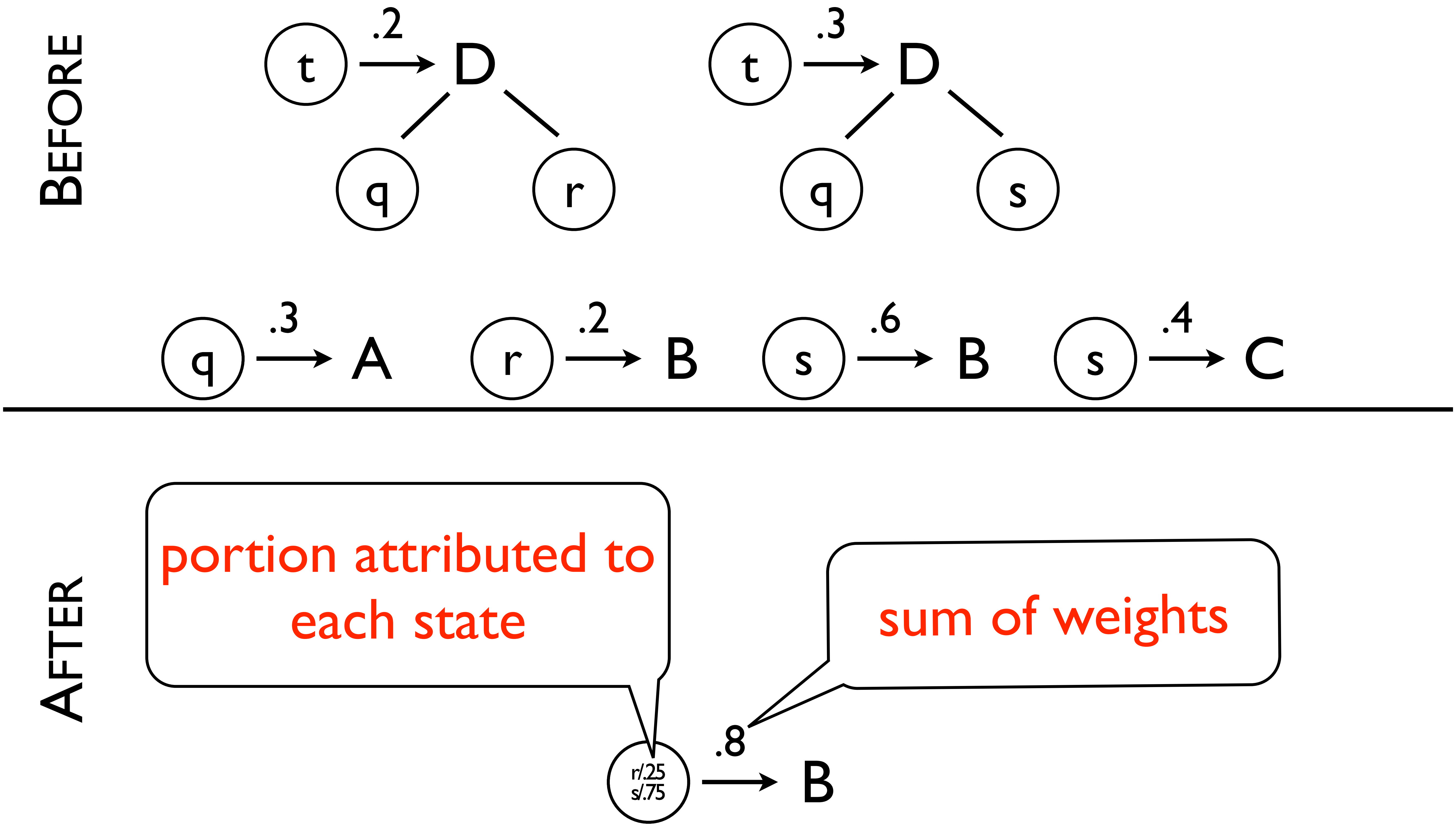


AFTER

Merge terminal rules with same right sides



Algorithmic Contribution I: WTA Determinization

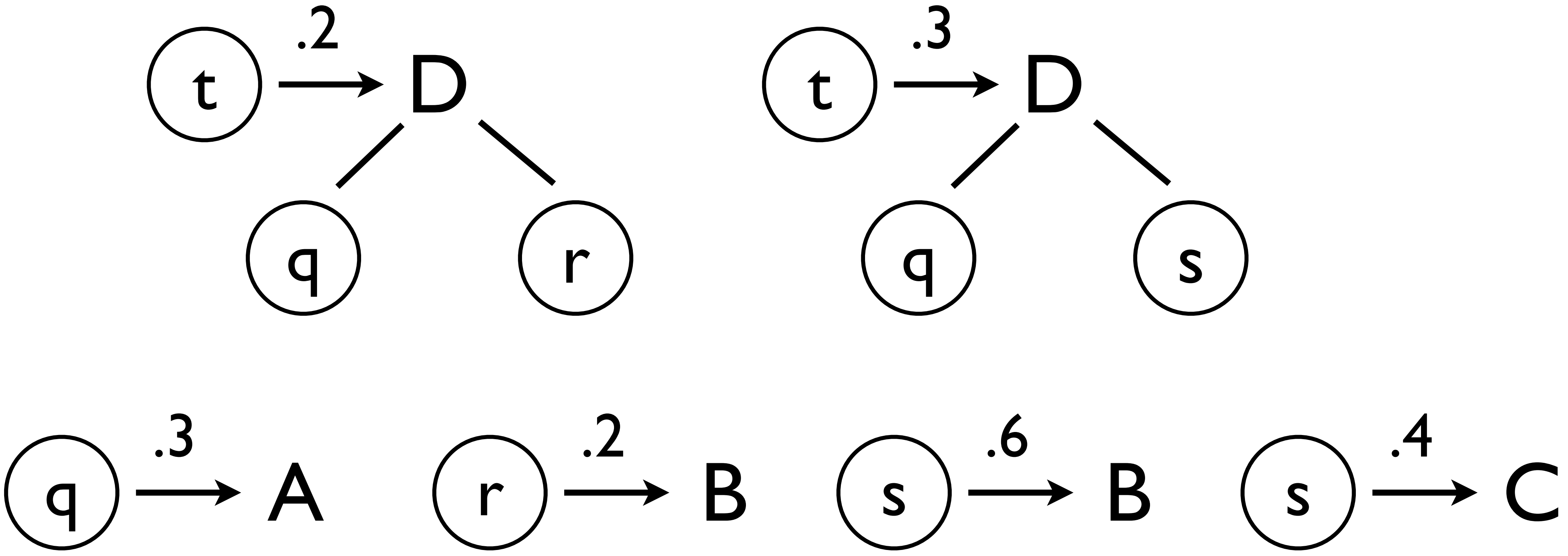


portion attributed to each state

sum of weights

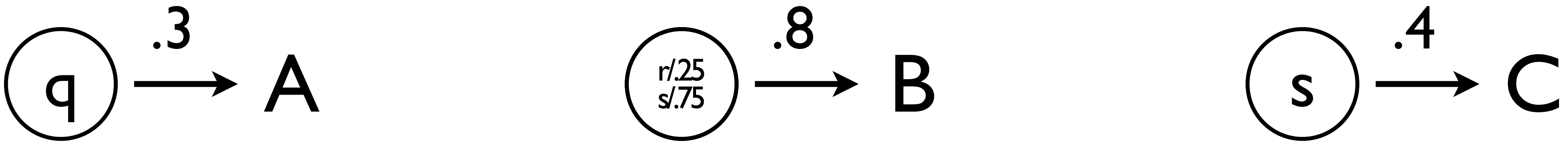
Algorithmic Contribution I: WTA Determinization

BEFORE



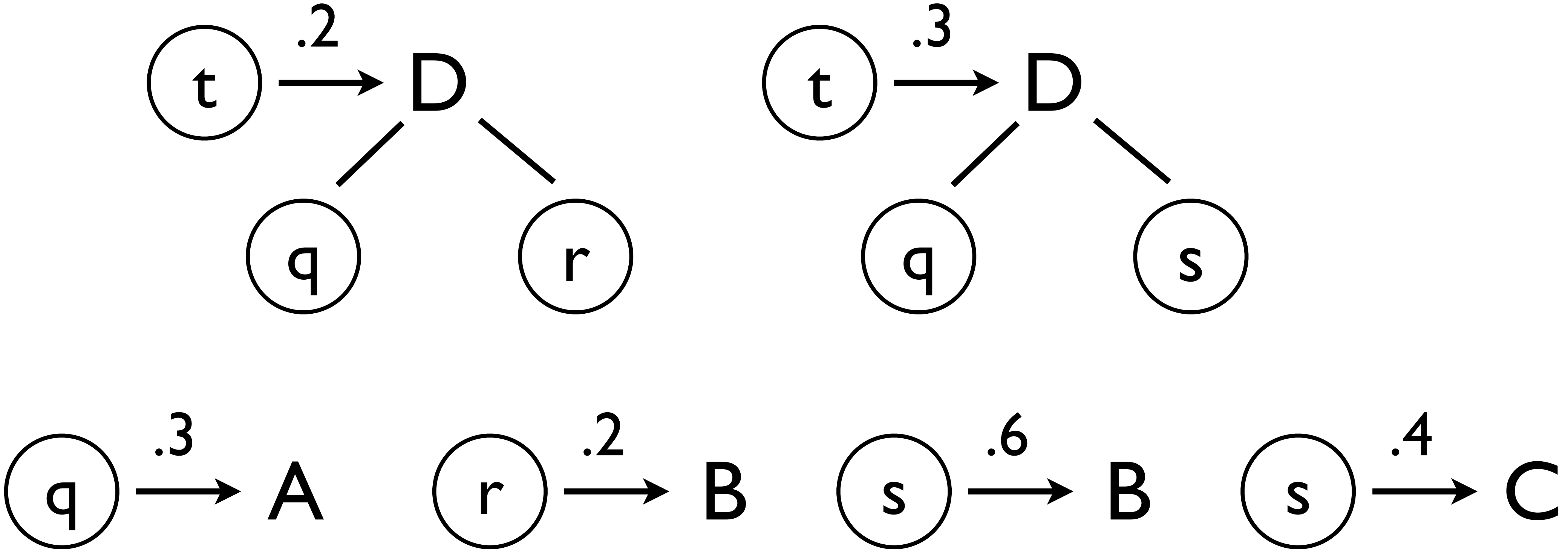
AFTER

Process the other terminal rules



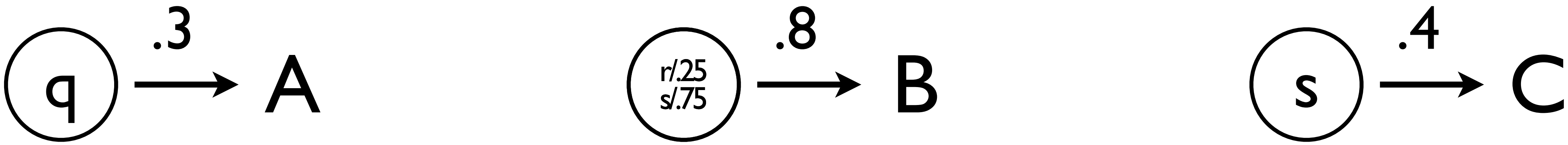
Algorithmic Contribution I: WTA Determinization

BEFORE



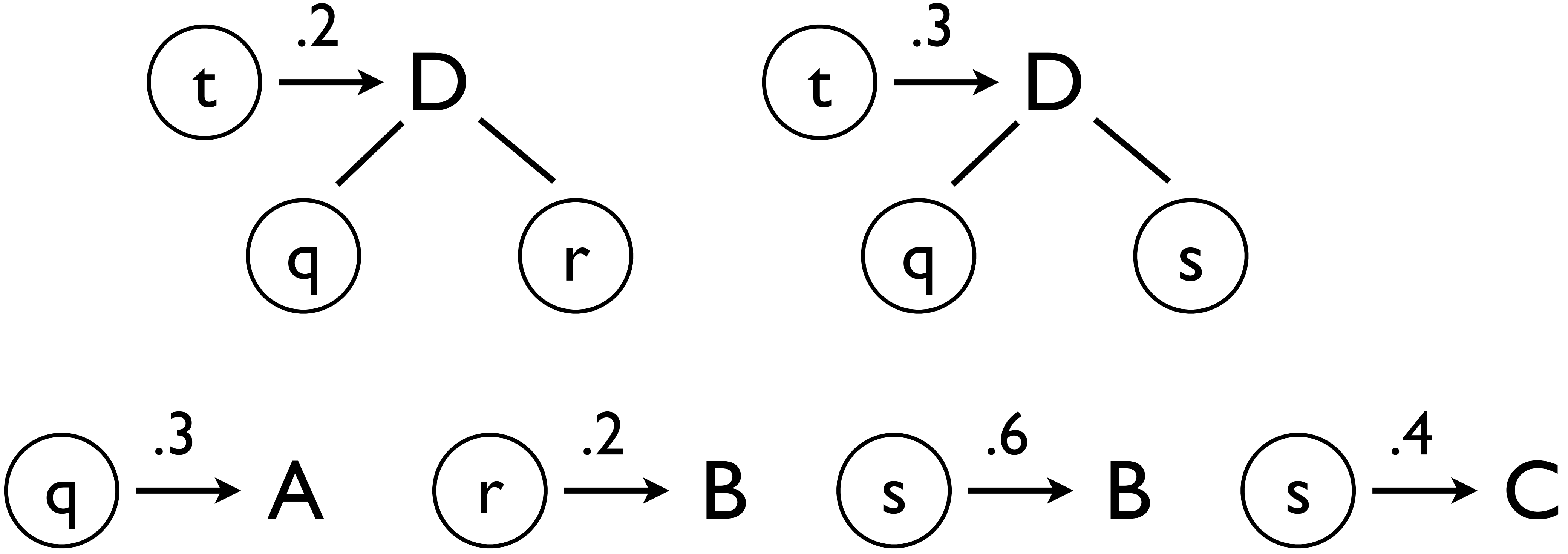
AFTER

Process the other terminal rules



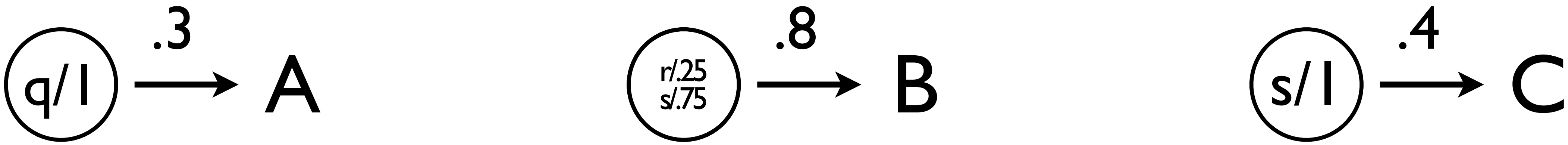
Algorithmic Contribution I: WTA Determinization

BEFORE



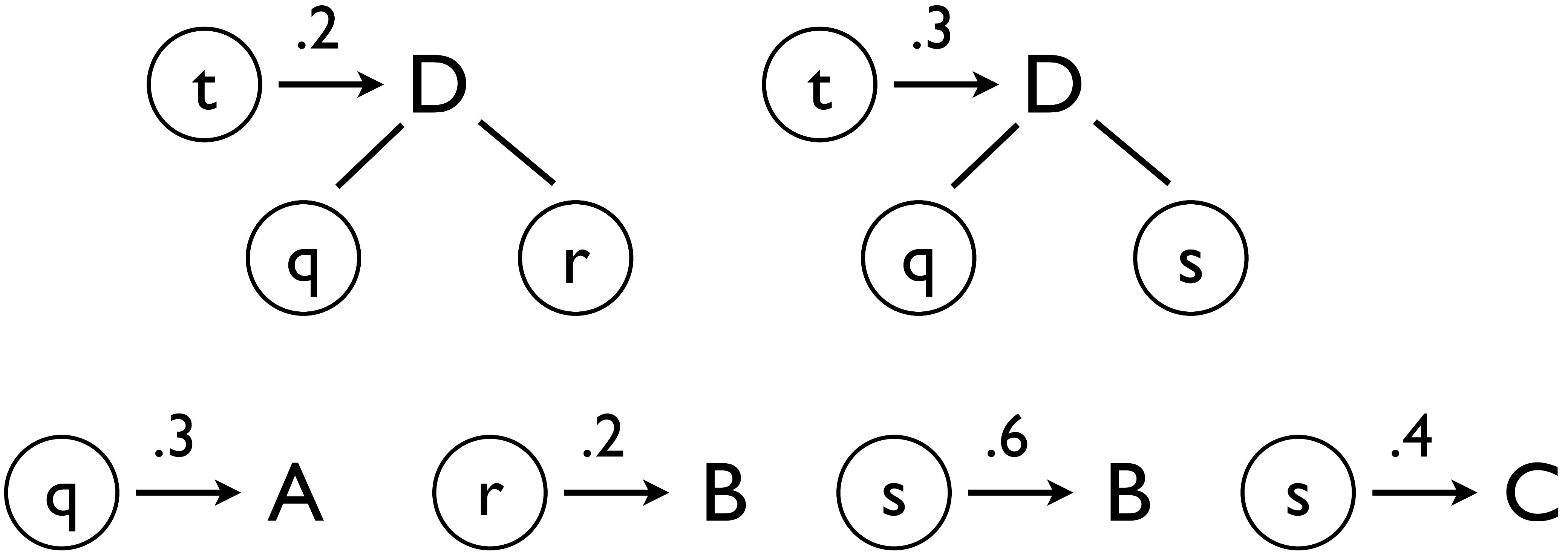
AFTER

Process the other terminal rules



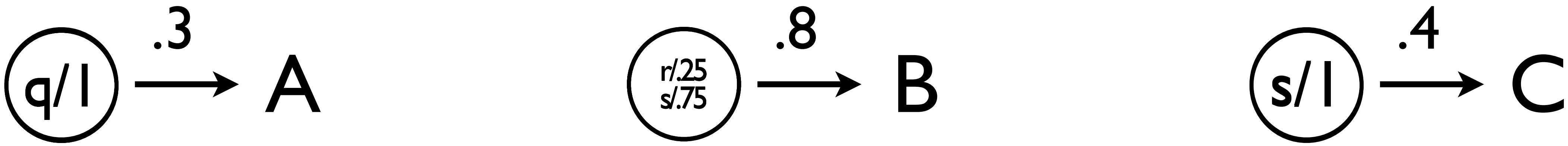
Algorithmic Contribution I: WTA Determinization

BEFORE

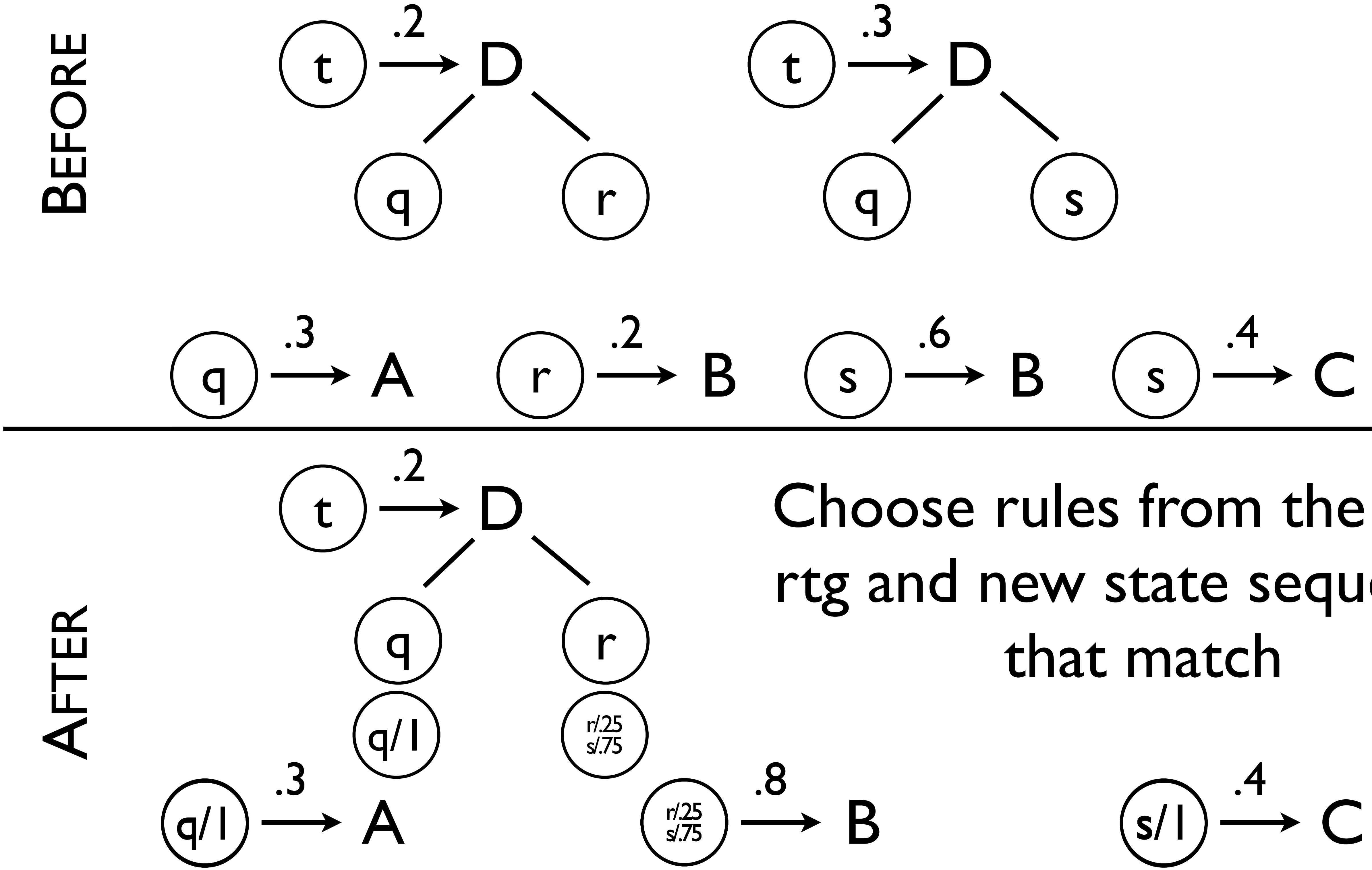


AFTER

Process the other terminal rules

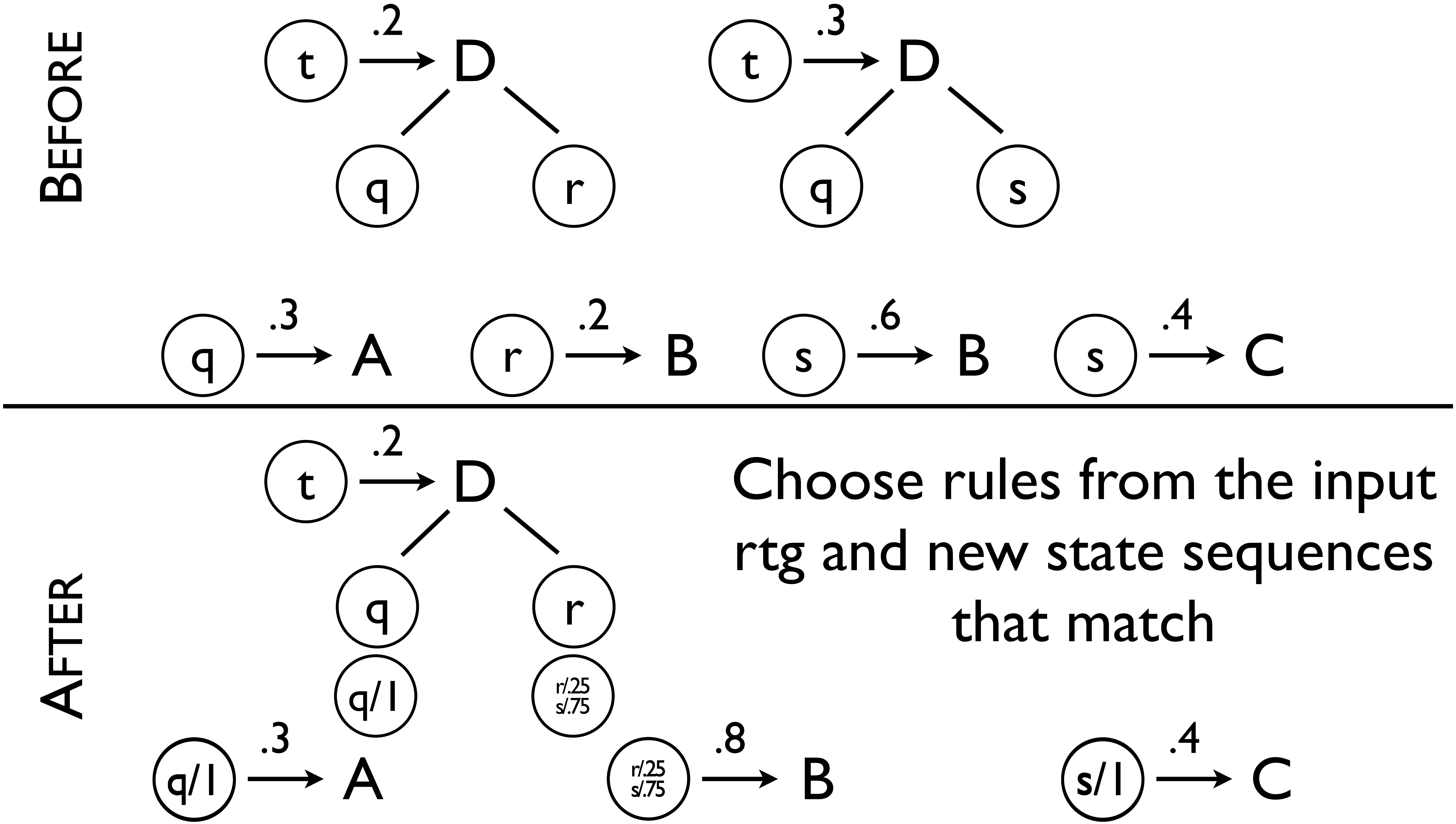


Algorithmic Contribution I: WTA Determinization



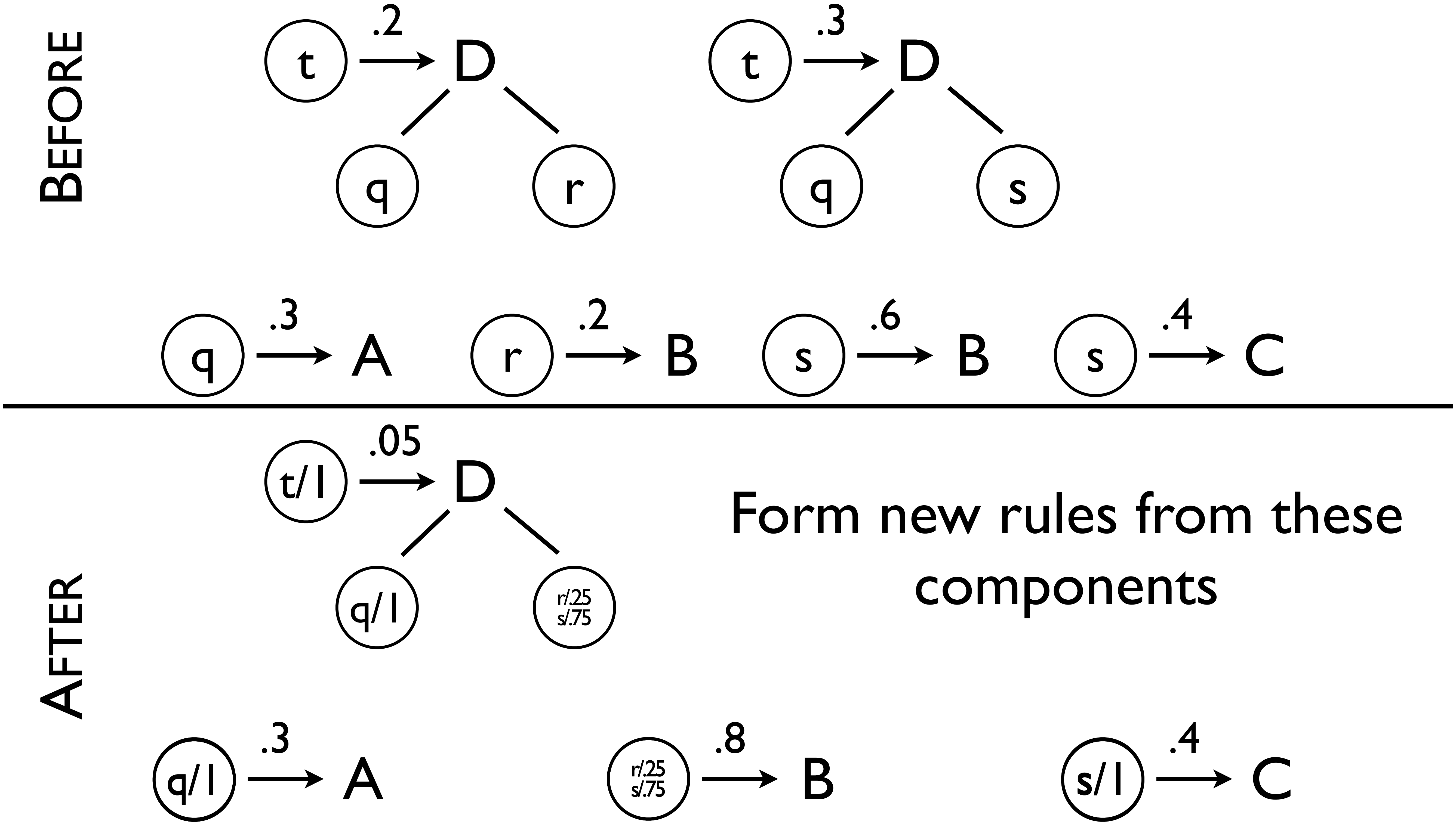
Choose rules from the input rtg and new state sequences that match

Algorithmic Contribution I: WTA Determinization



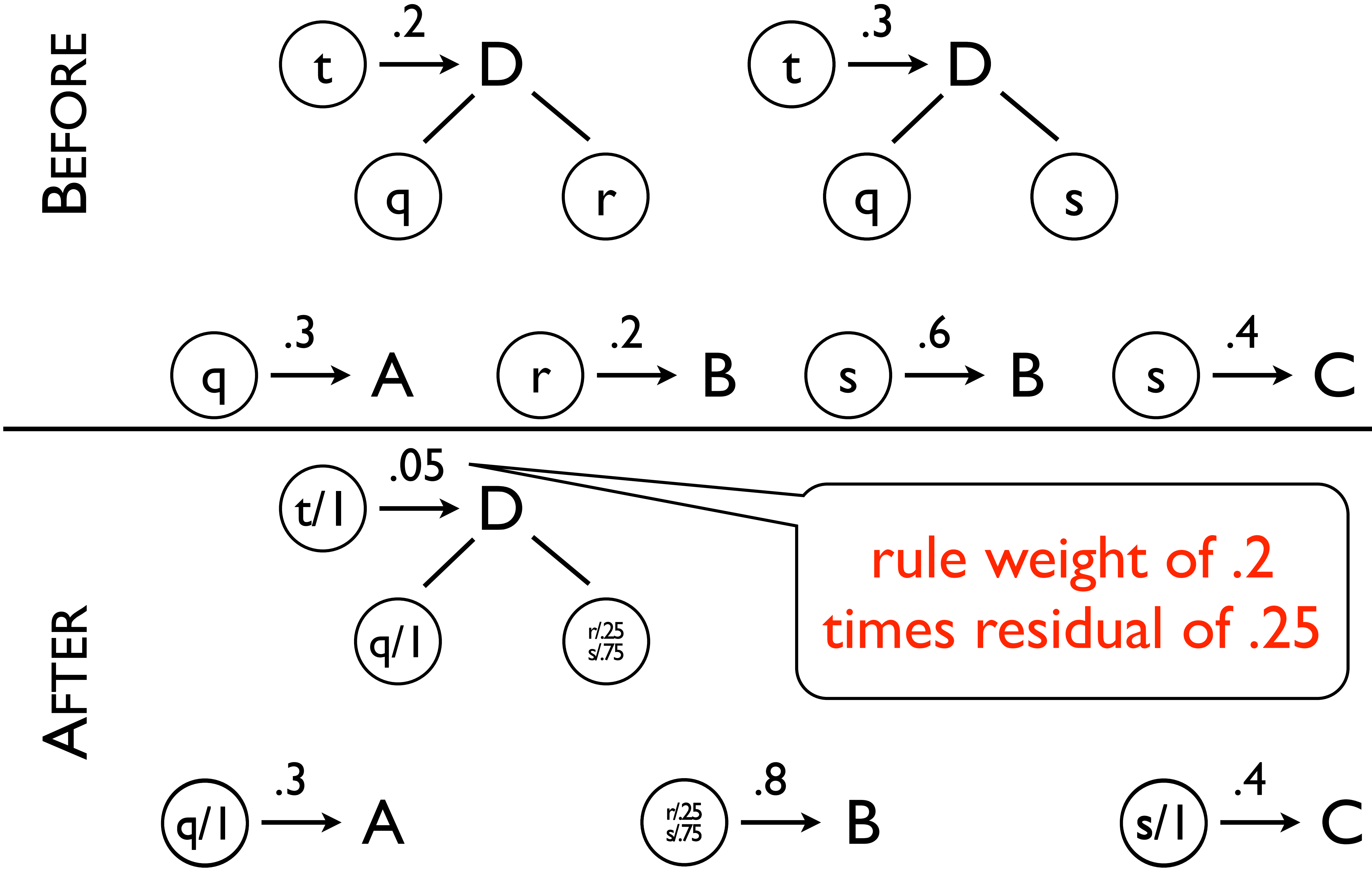
Choose rules from the input rtg and new state sequences that match

Algorithmic Contribution I: WTA Determinization



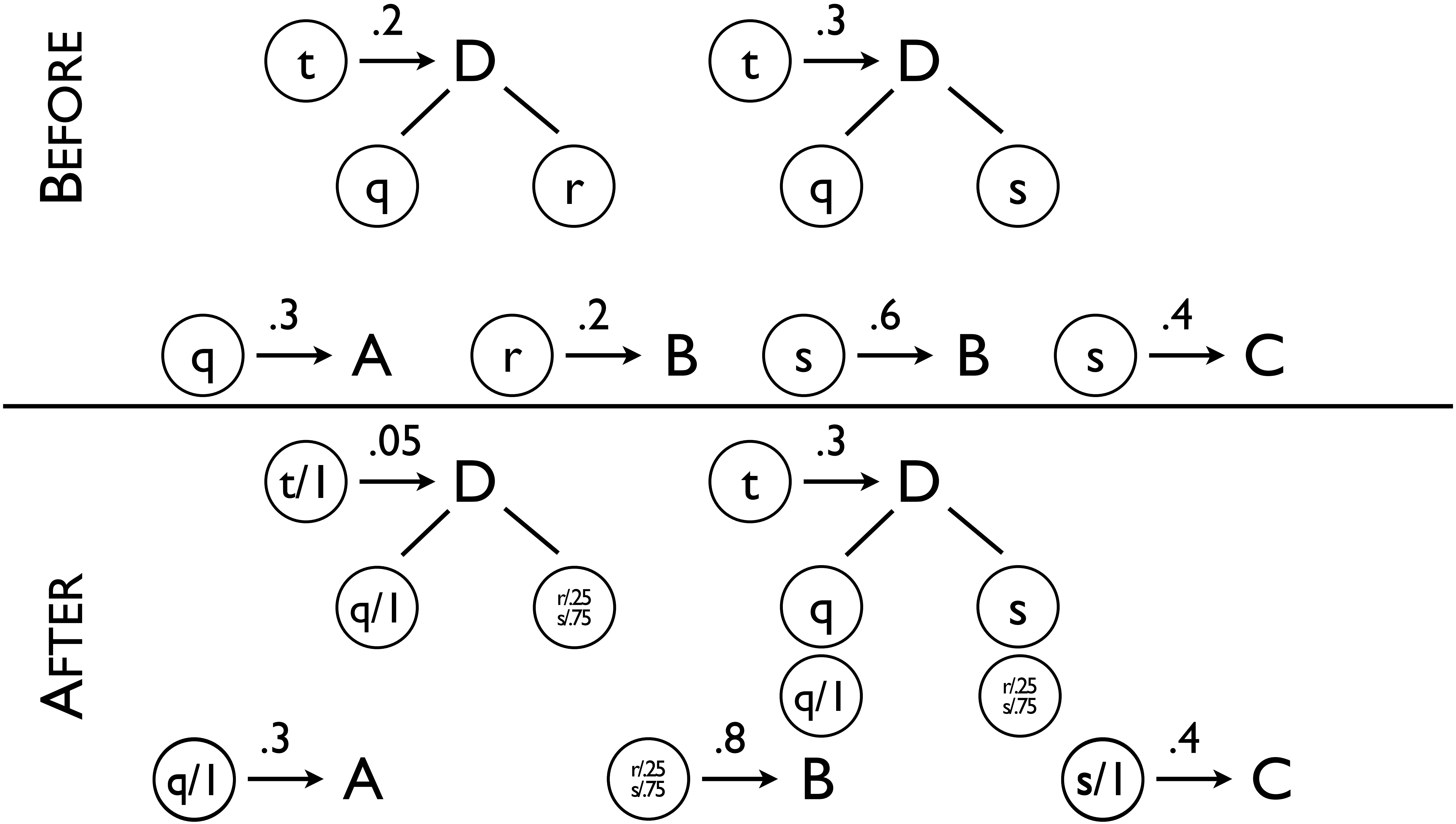
Form new rules from these components

Algorithmic Contribution I: WTA Determinization

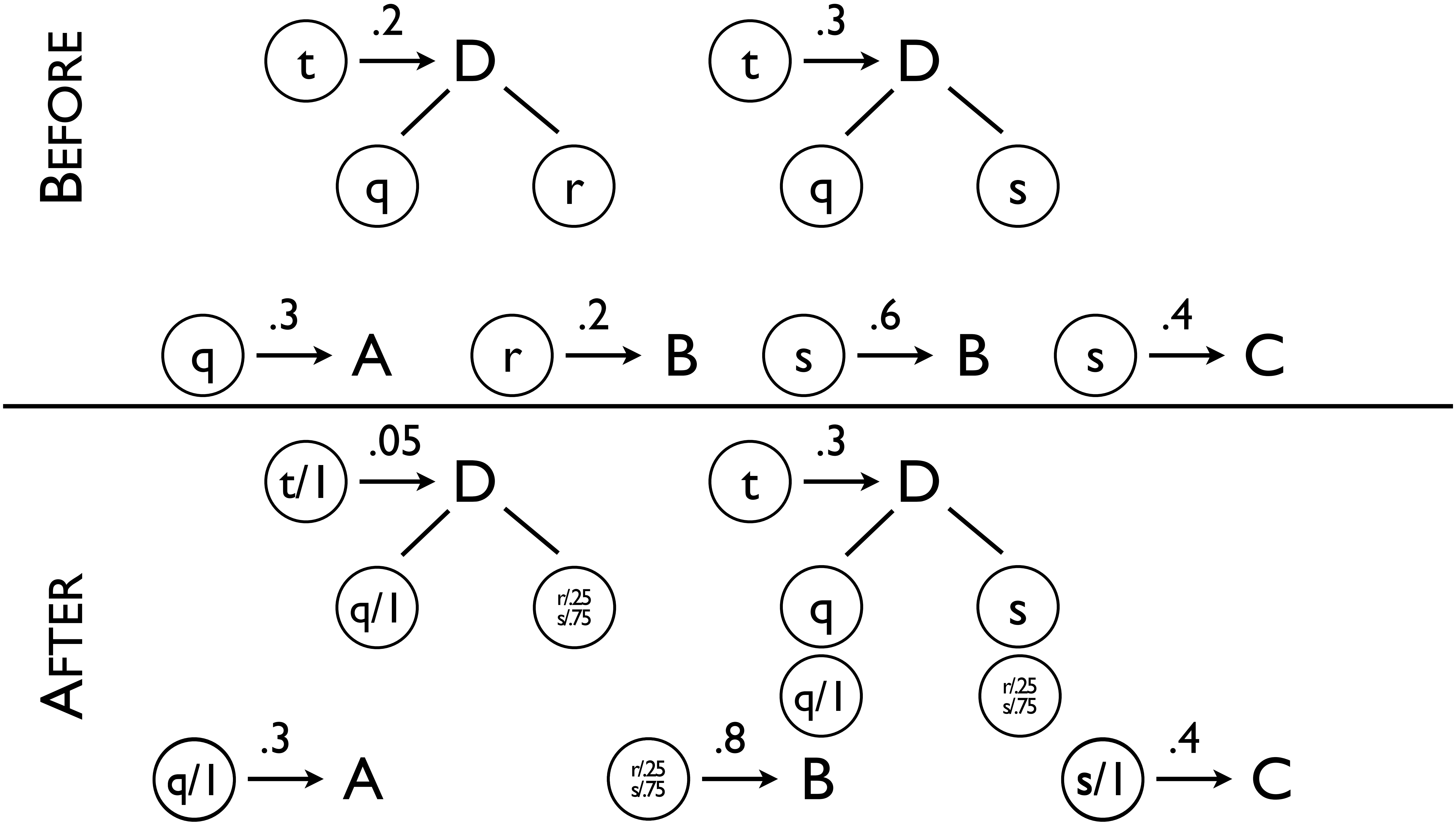


rule weight of .2 times residual of .25

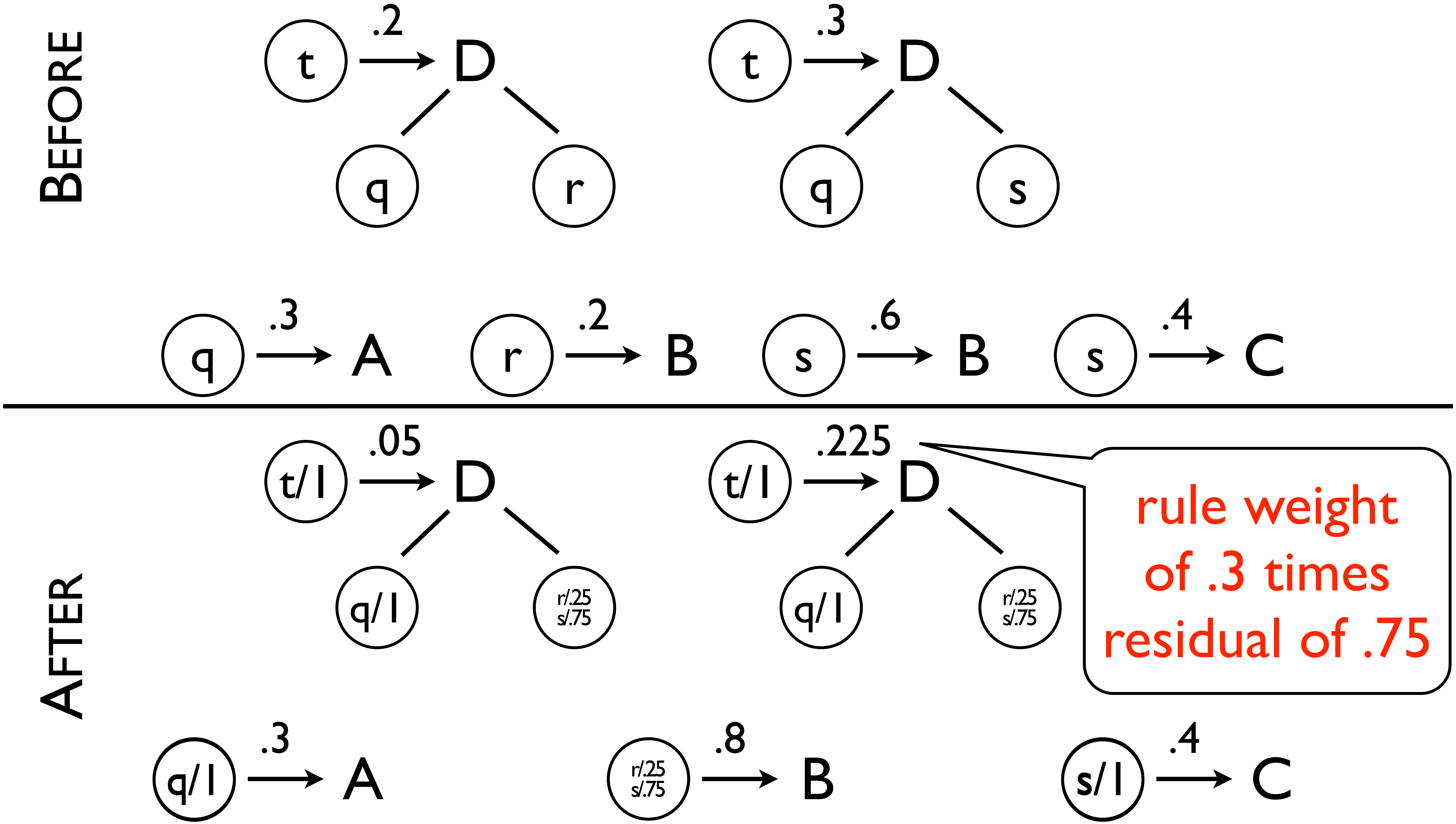
Algorithmic Contribution I: WTA Determinization



Algorithmic Contribution I: WTA Determinization

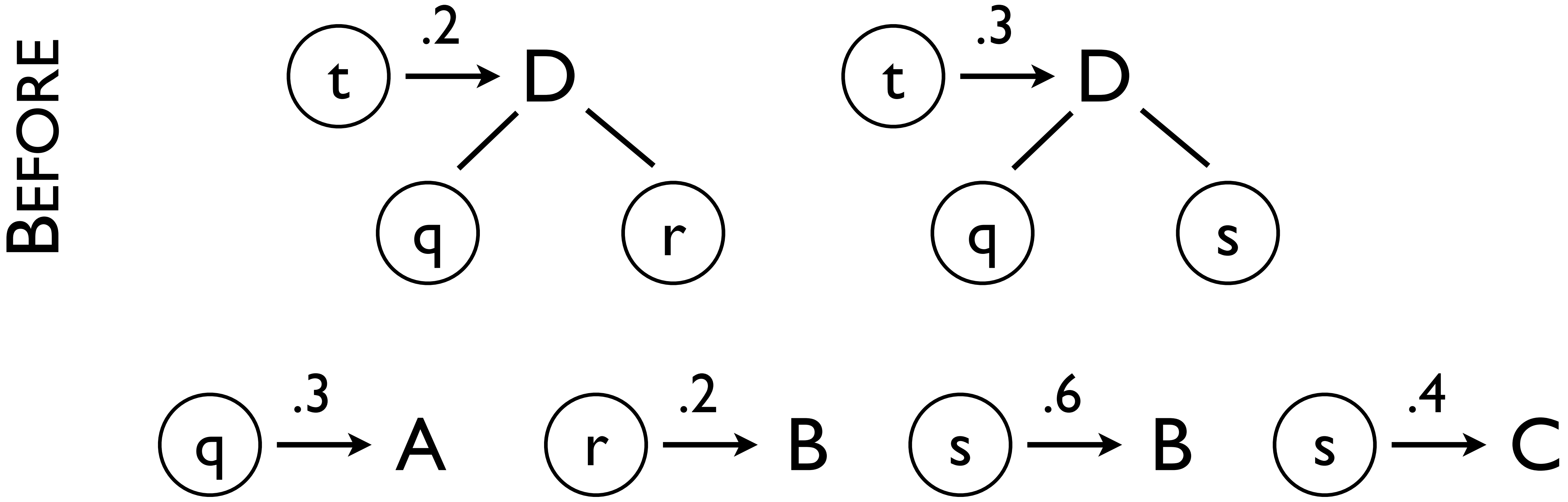


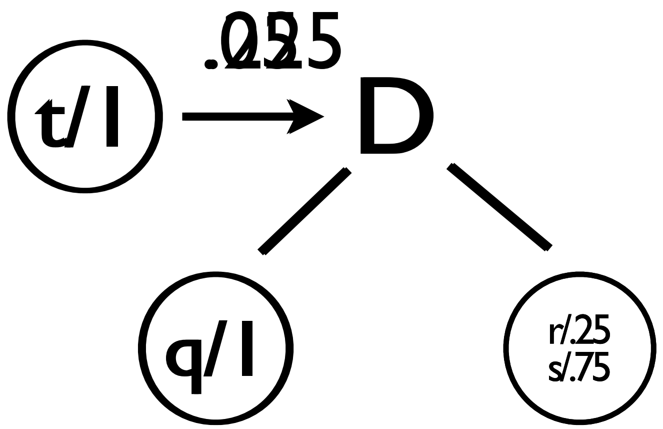
Algorithmic Contribution I: WTA Determinization



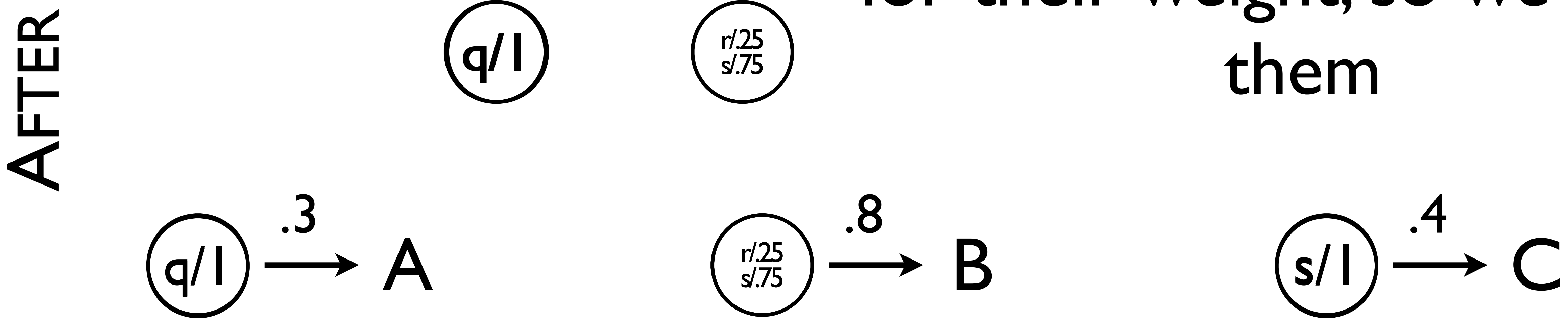
rule weight of .3 times residual of .75

Algorithmic Contribution I: WTA Determinization

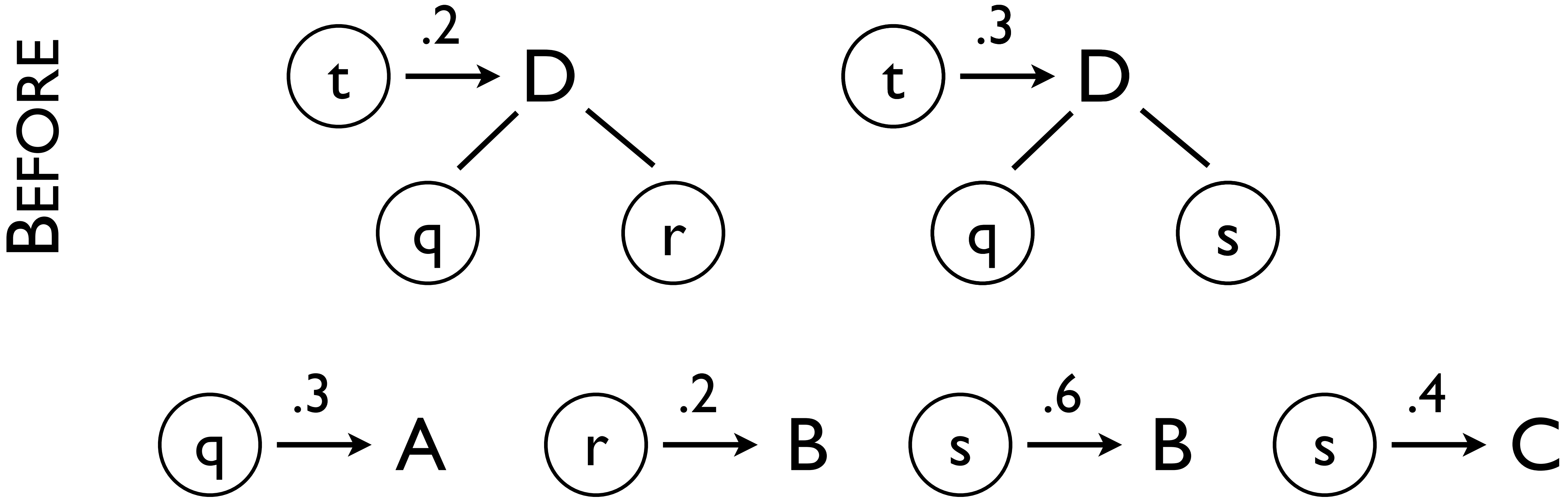


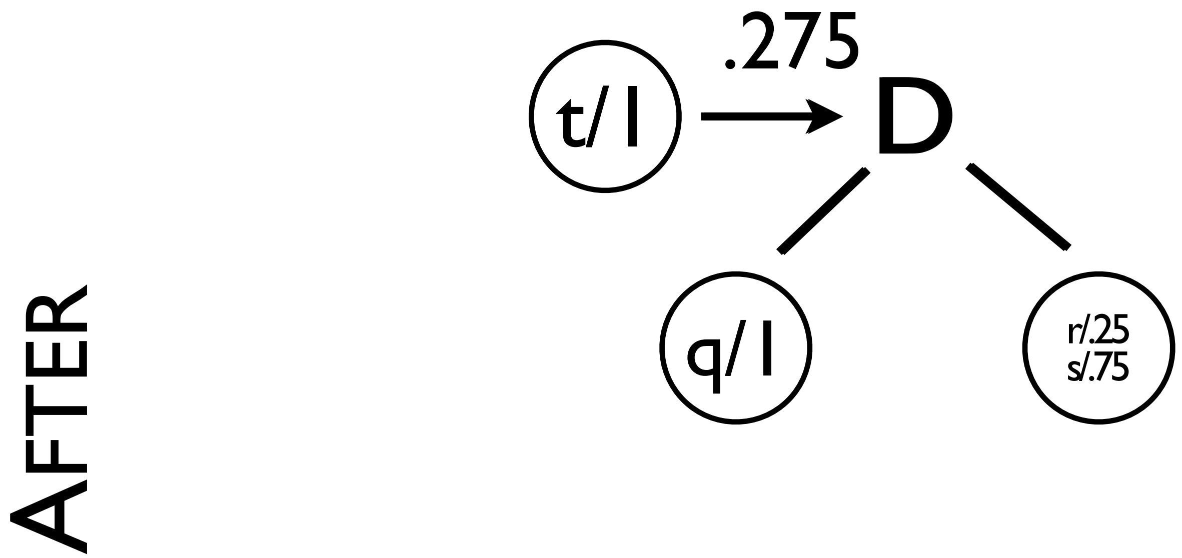


These rules are identical except for their weight, so we'll sum them

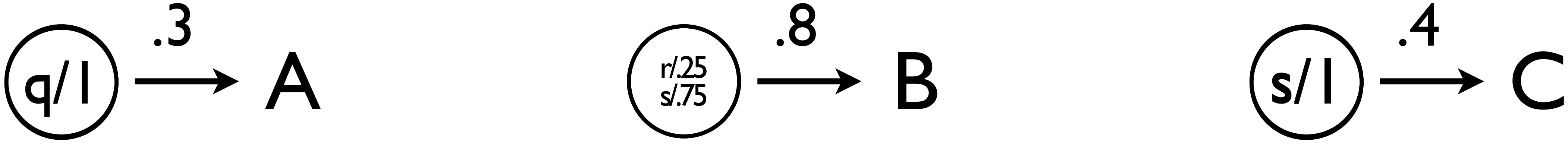


Algorithmic Contribution I: WTA Determinization

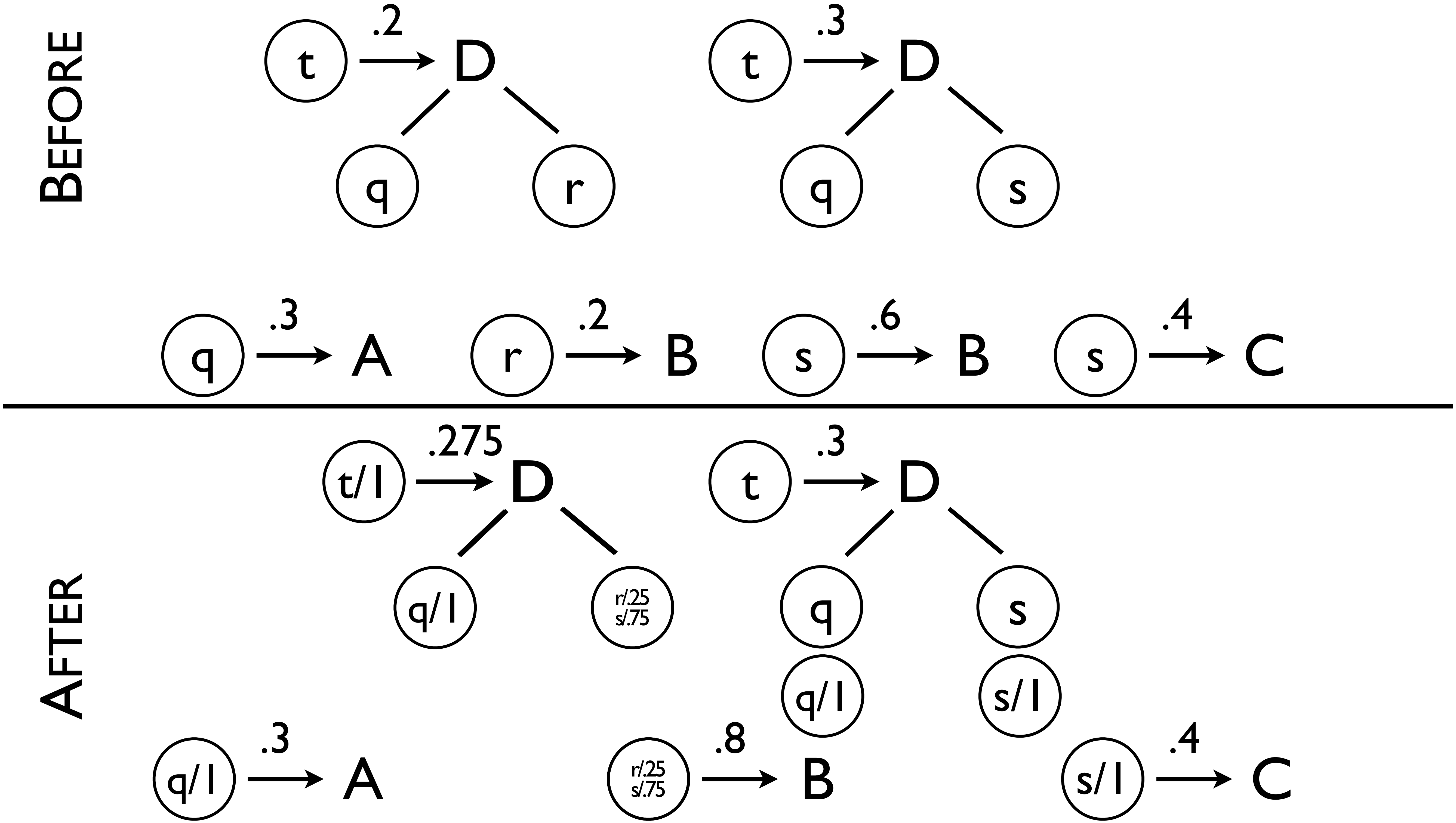




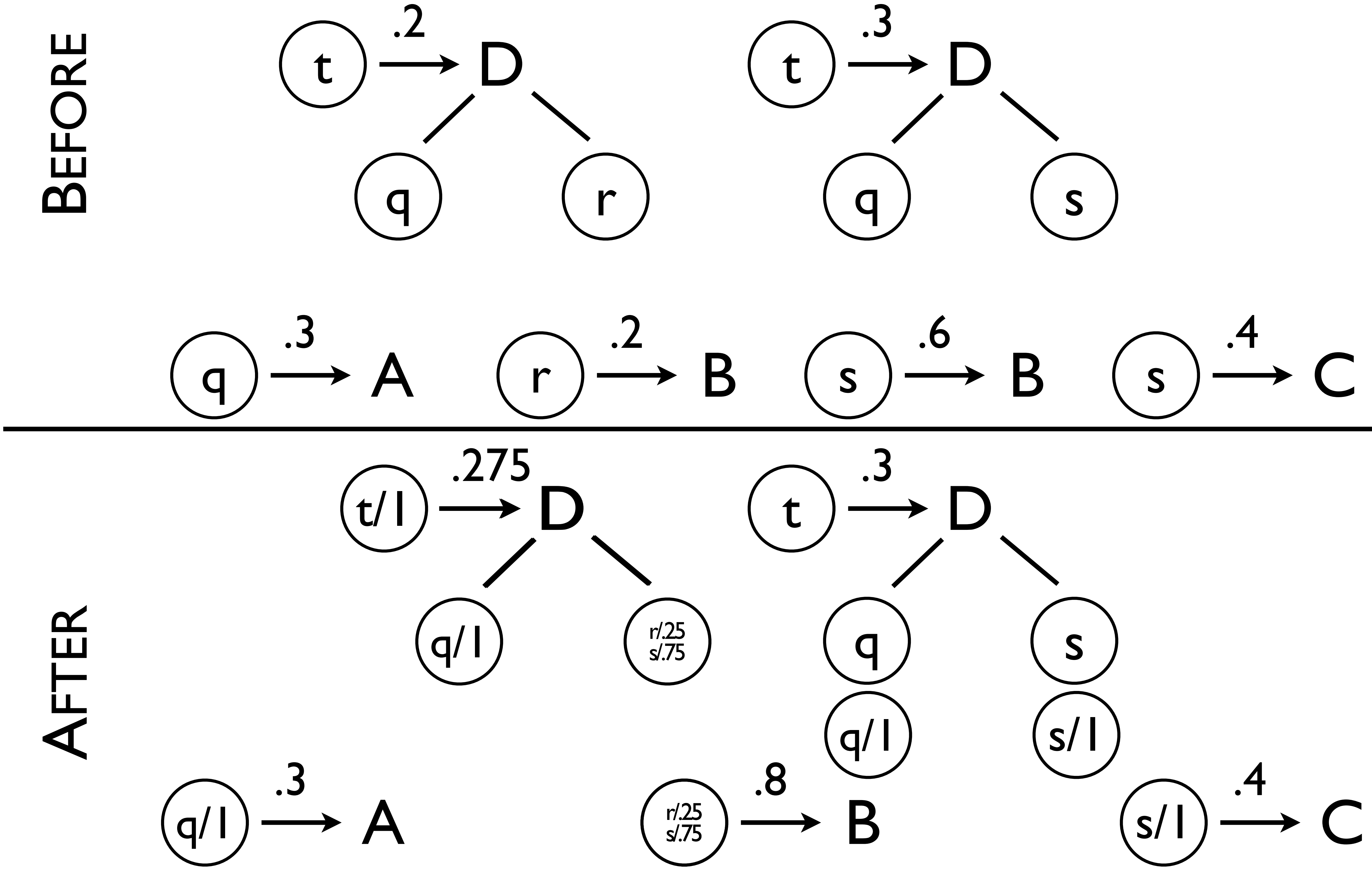
These rules are identical except for their weight, so we'll sum them



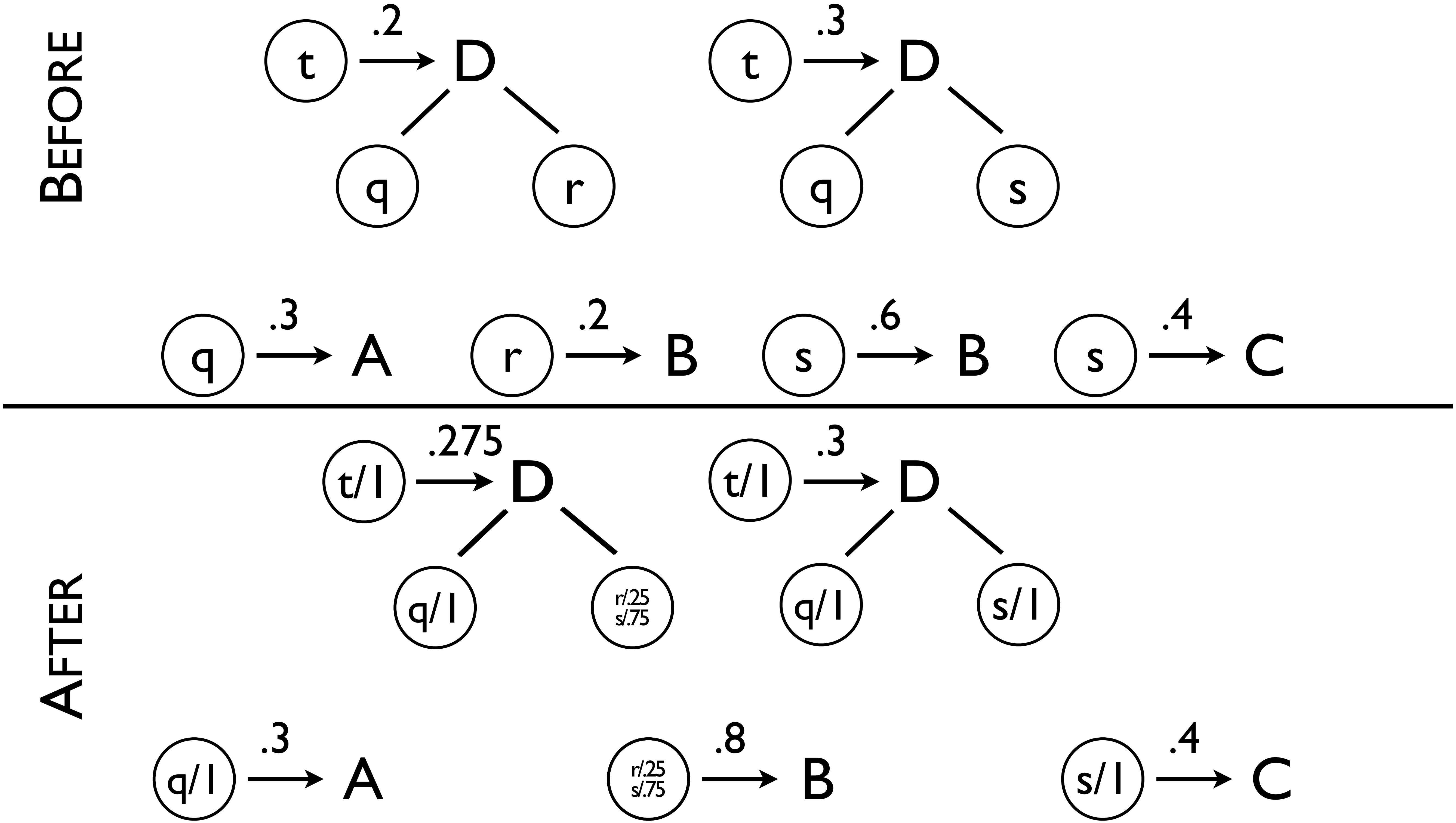
Algorithmic Contribution I: WTA Determinization



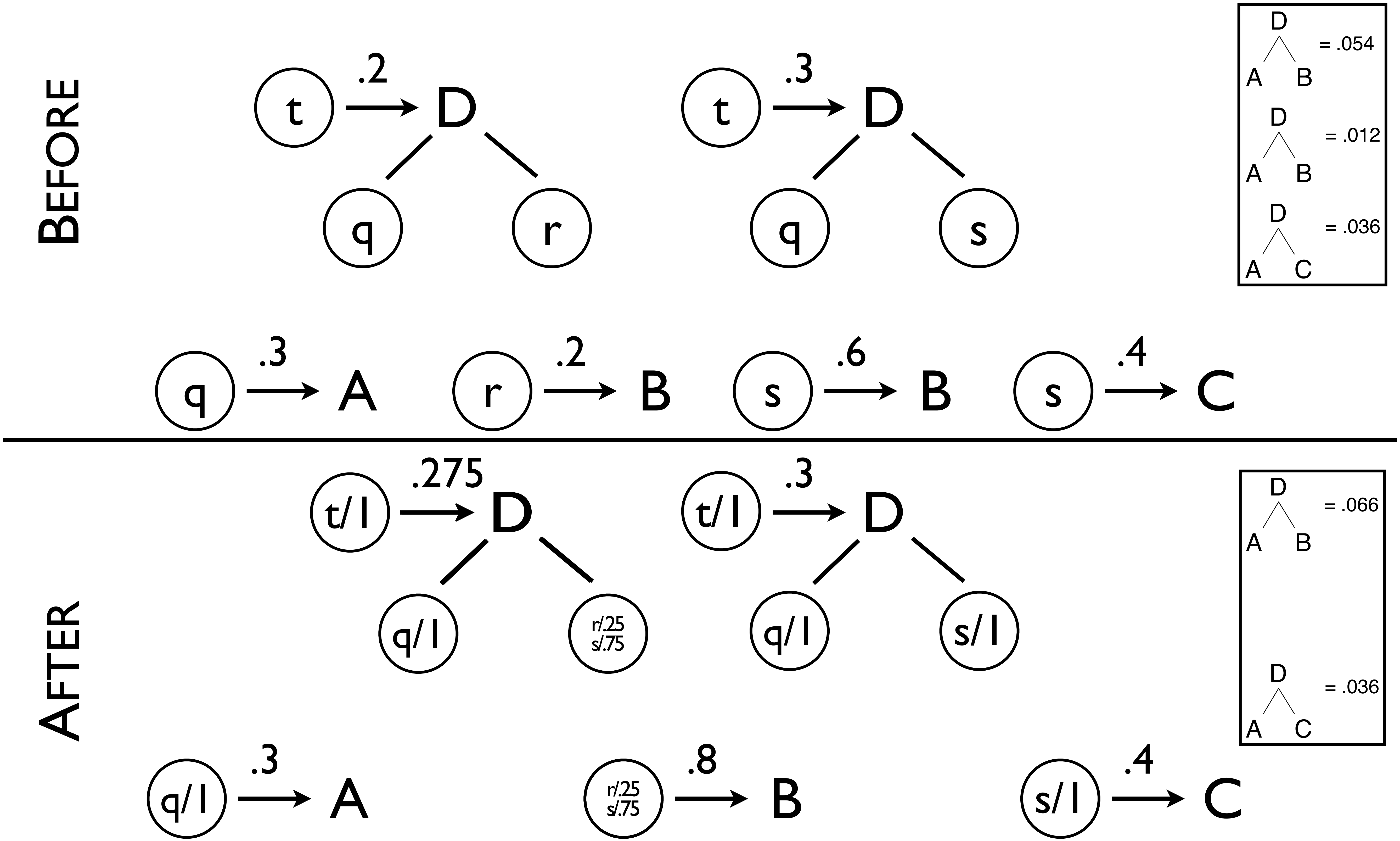
Algorithmic Contribution I: WTA Determinization



Algorithmic Contribution I: WTA Determinization



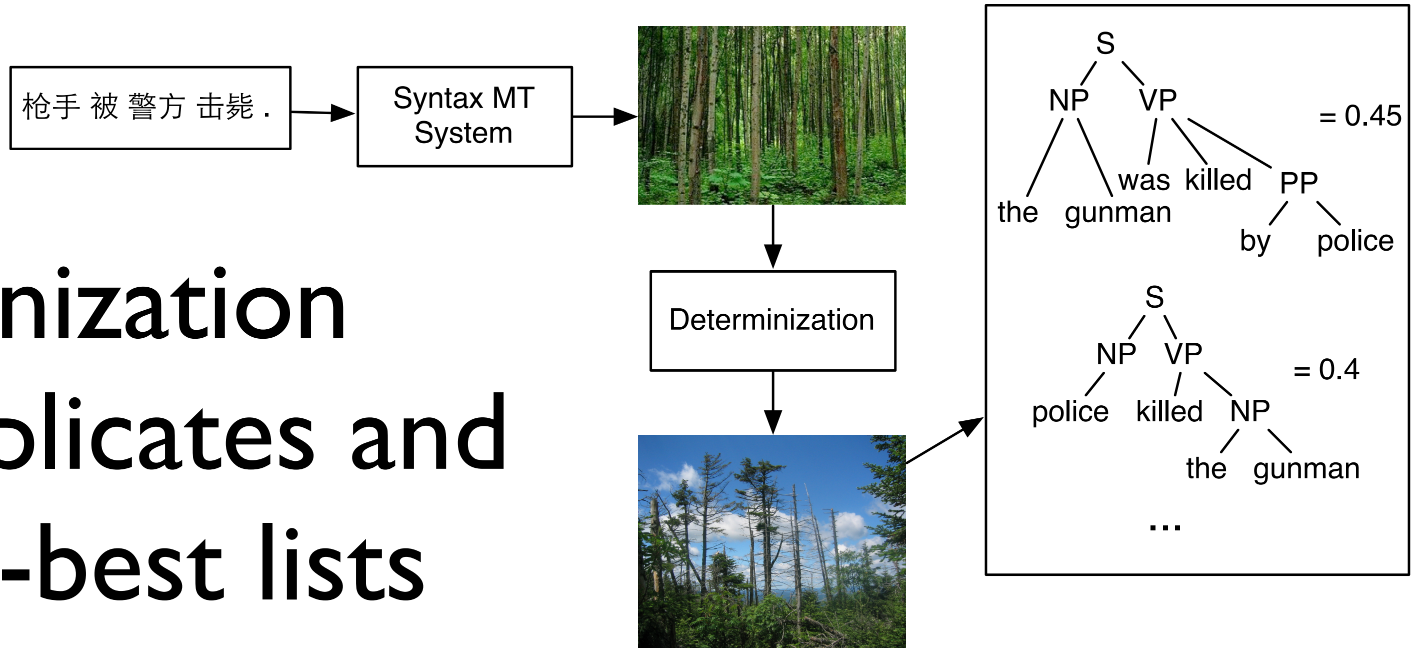
Algorithmic Contribution I: WTA Determinization



Empirical experiments

Machine translation (Galley et al. '04, '06)

Determinization removes duplicates and re-ranks n-best lists


枪手 被 警方 击毙 .

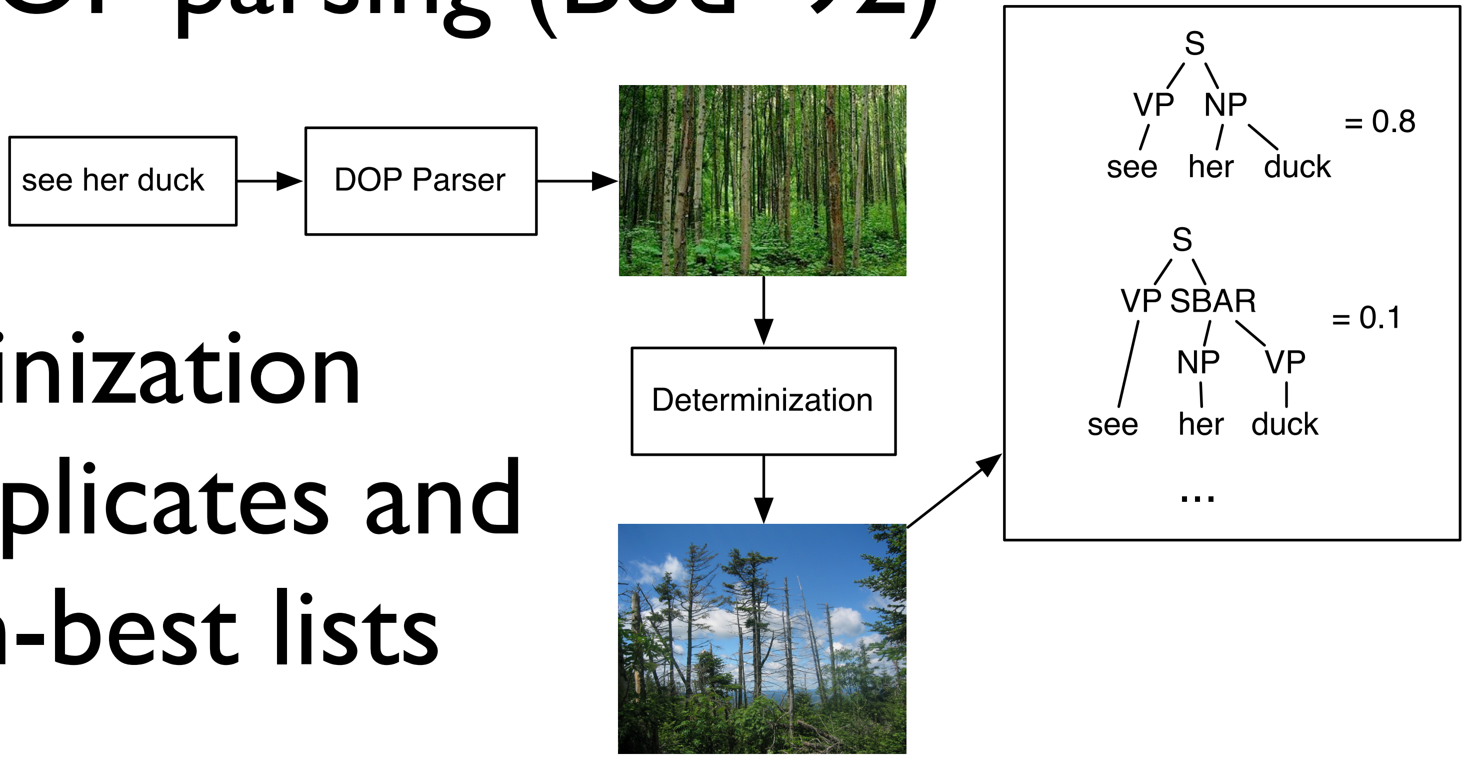
Syntax MT System

Determinization

Method	**Bleu**
Undeterminized	21.87
Top-500 "crunching"	23.33
Determinized	24.17

Empirical experiments

DOP parsing (Bod '92)



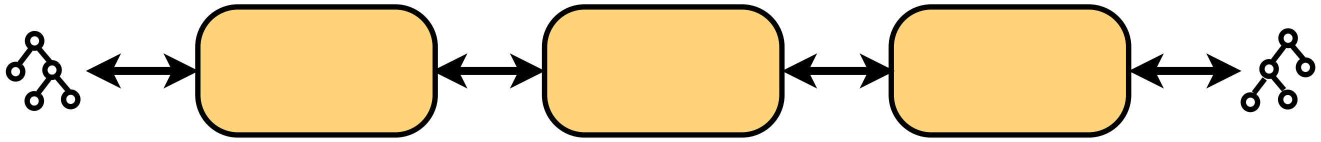
Determinization removes duplicates and re-ranks n-best lists

Method	Precision	Recall	F
Undeterminized	80.23	80.18	80.20
Top-500 “crunching”	80.48	80.29	80.39
Determinized	81.09	79.72	80.40

Efficient inference through cascades of weighted tree transducers

(May, Knight, Vogler, Submitted)

- First presentation of algorithms for inference through weighted extended tree transducer cascades
- On-the-fly approach significantly outperforms "classic" approach



Inference through *string* transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

Inference through *string* transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf

Inference through *string* transducers

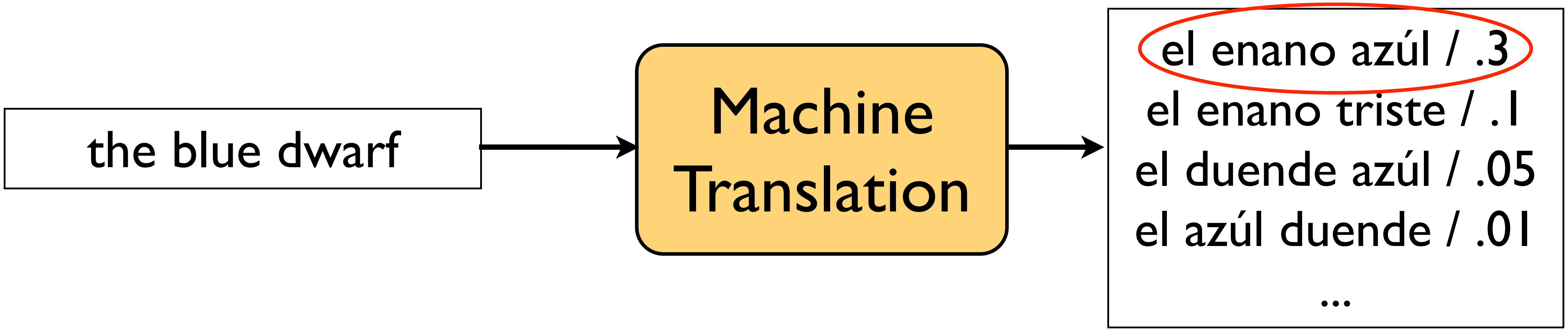
Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf

Machine Translation

Inference through *string* transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer



Inference through string *cascades*

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

(Pereira & Riley, 1997)

Inference through string *cascades*

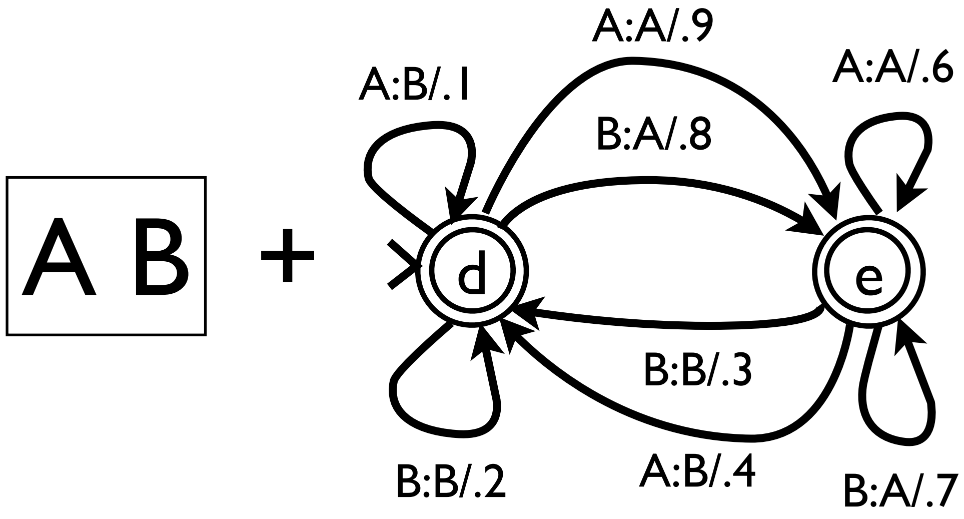
Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

A B

(Pereira & Riley, 1997)

Inference through string *cascades*

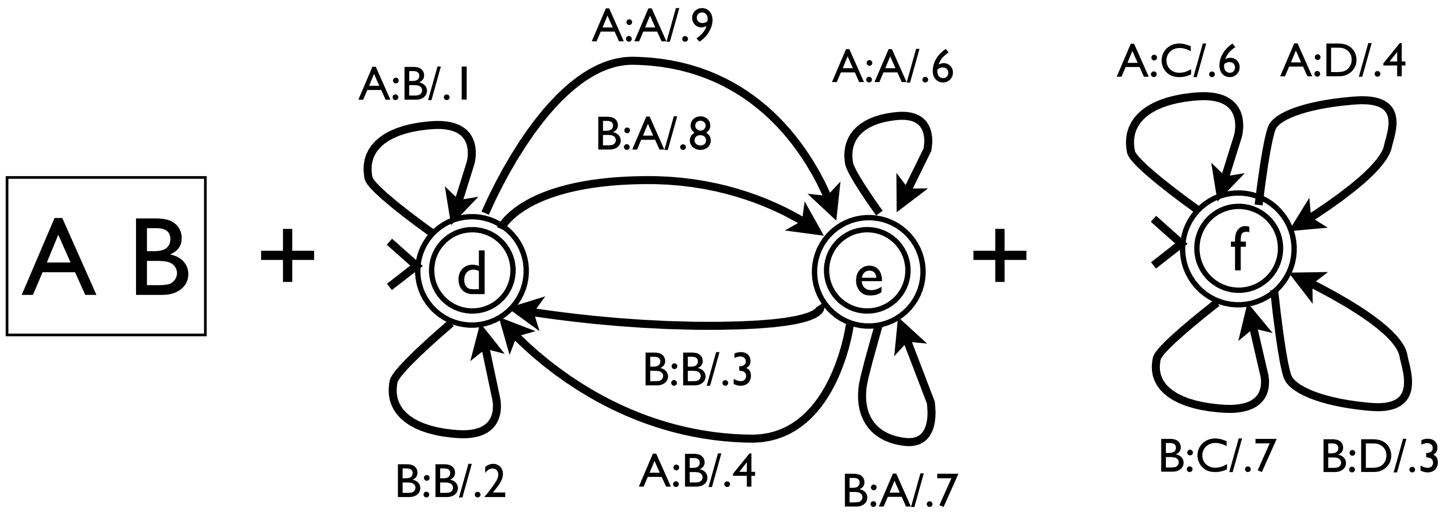
Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade



(Pereira & Riley, 1997)

Inference through string *cascades*

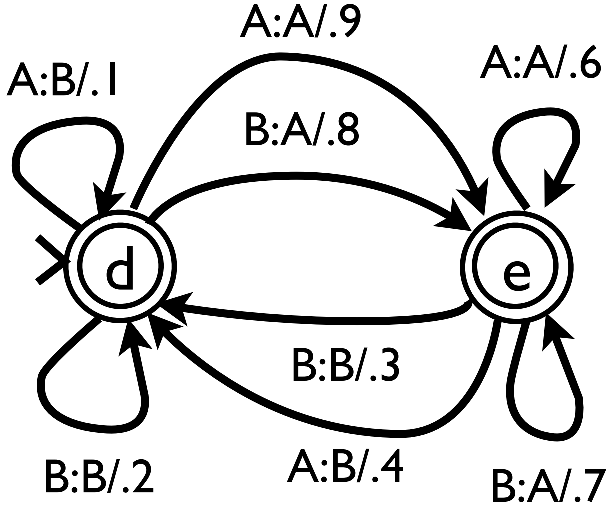
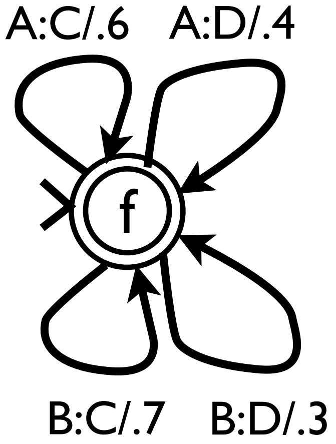
Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade



(Pereira & Riley, 1997)

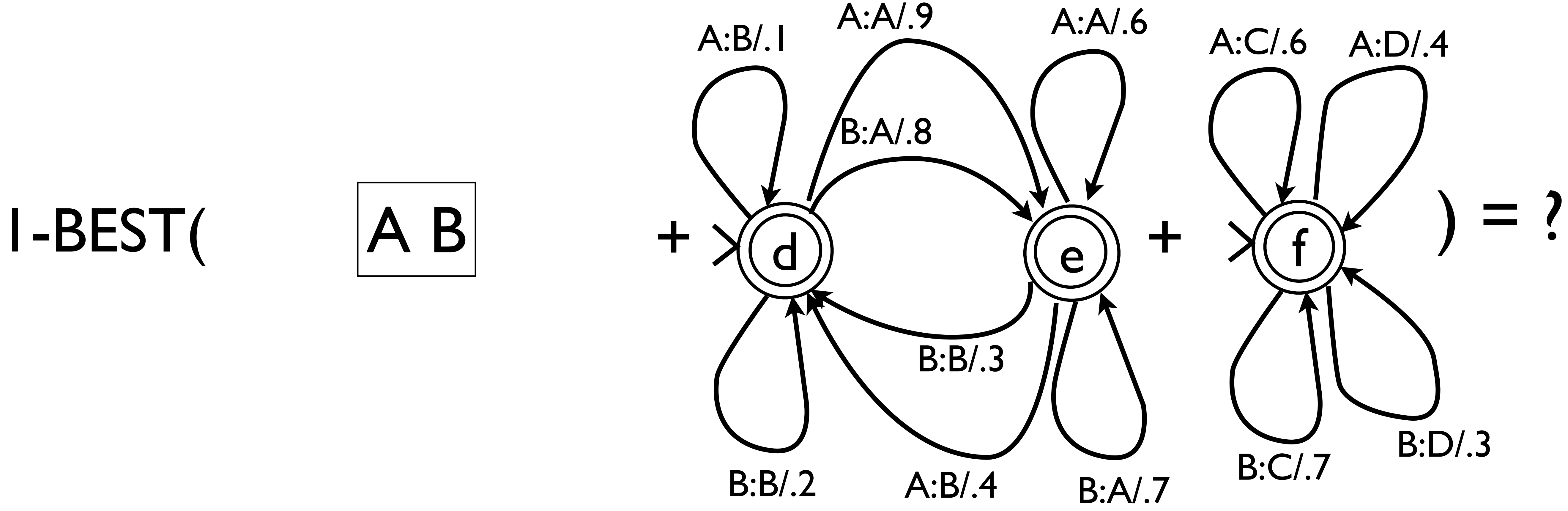
Inference through string *cascades*

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

1-BEST(A B +  + ) = ?

(Pereira & Riley, 1997)

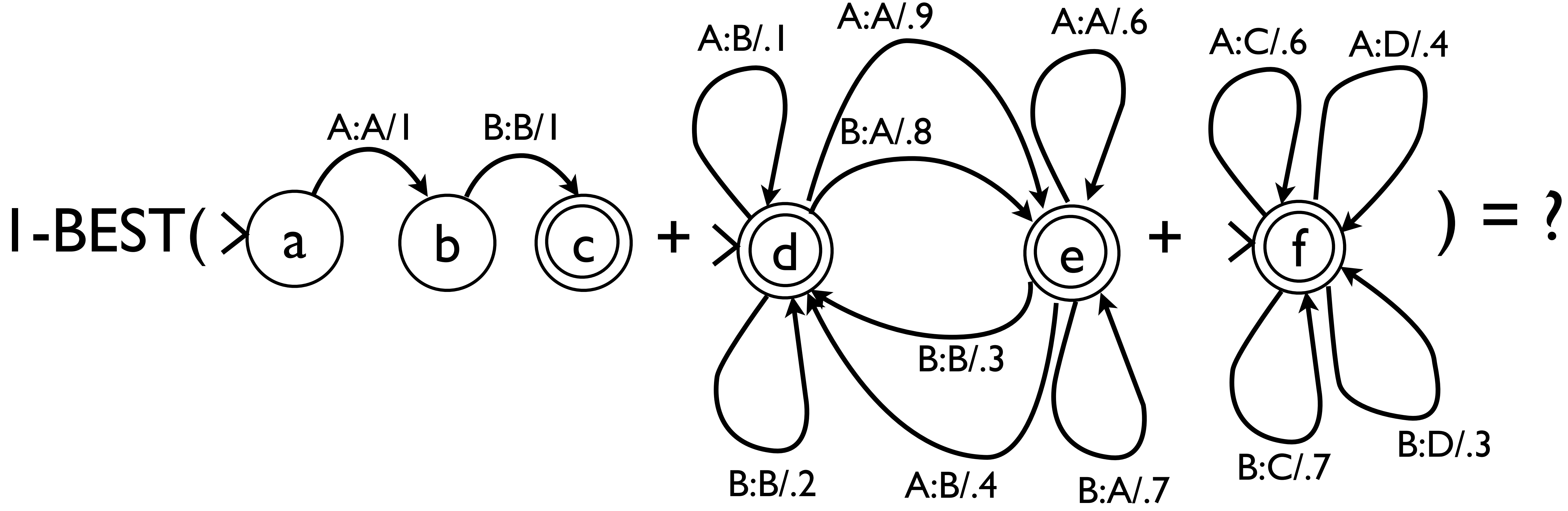
Pipeline approach



Embed the string

(Pereira & Riley, 1997)

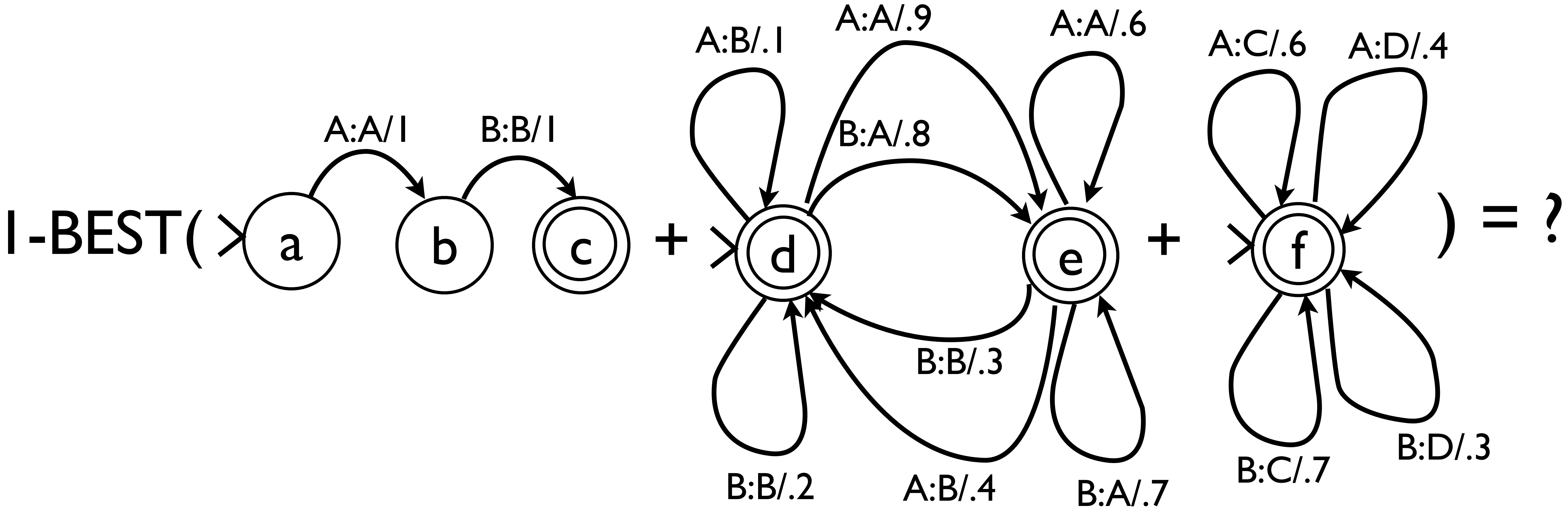
Pipeline approach



Embed the string

(Pereira & Riley, 1997)

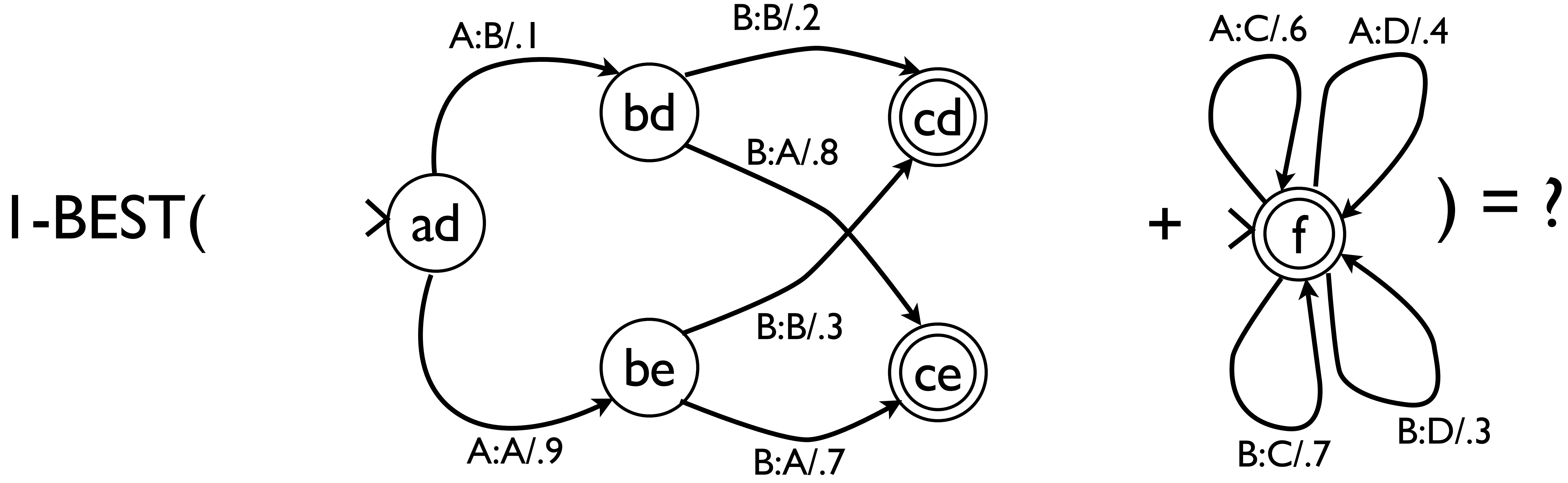
Pipeline approach



Compose the cascade

(Pereira & Riley, 1997)

Pipeline approach

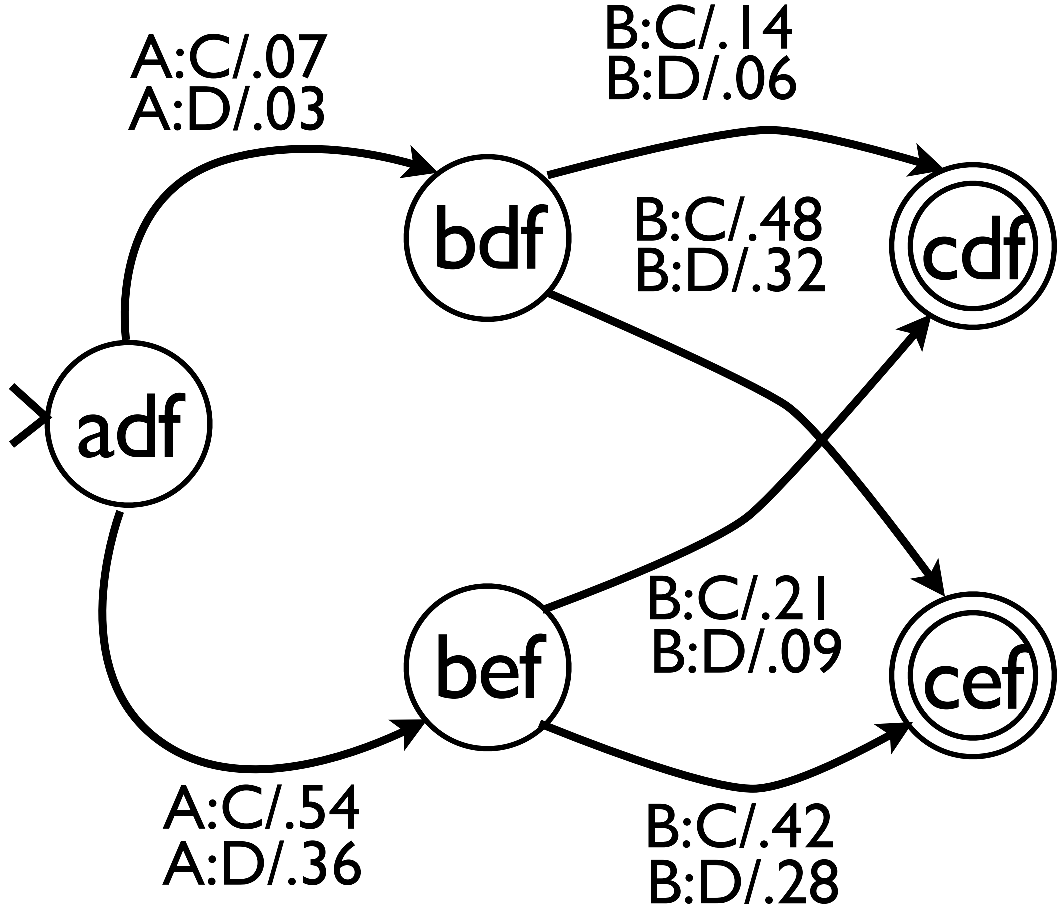


Compose the cascade

(Pereira & Riley, 1997)

Pipeline approach

1-BEST(

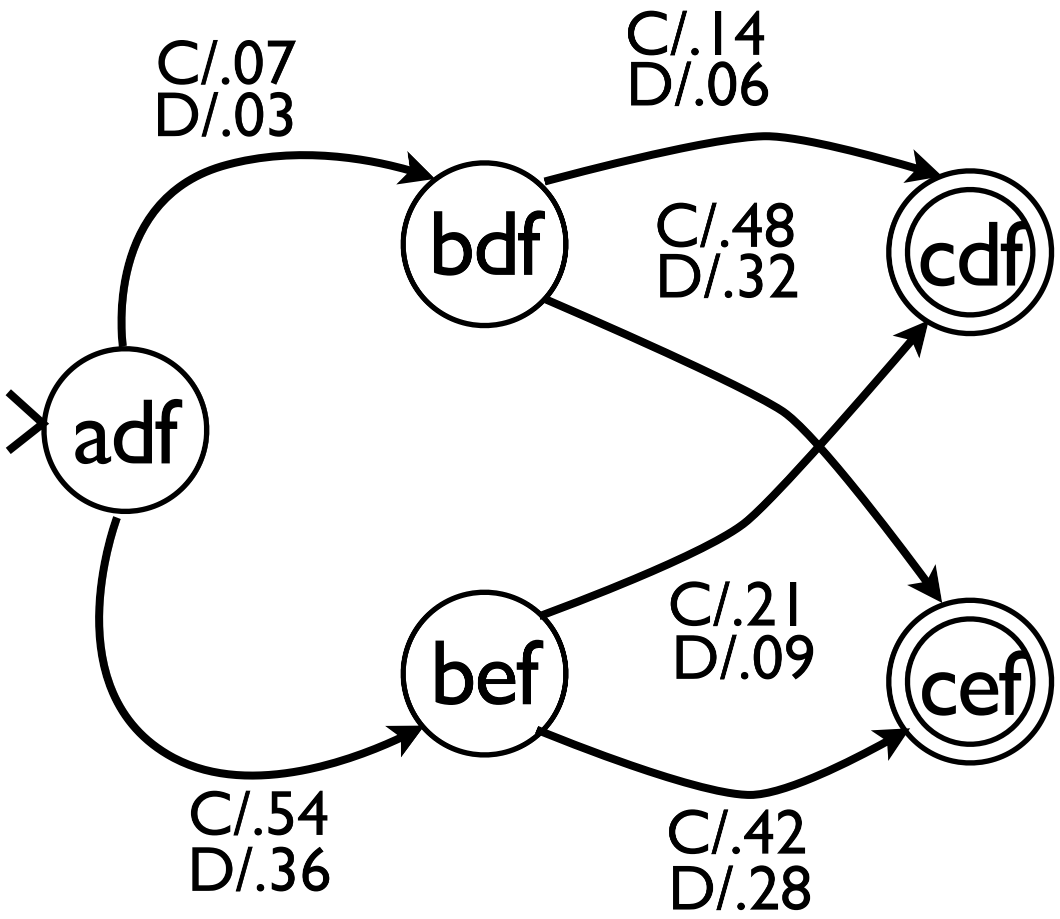


) = ?

Compose the cascade

(Pereira & Riley, 1997)

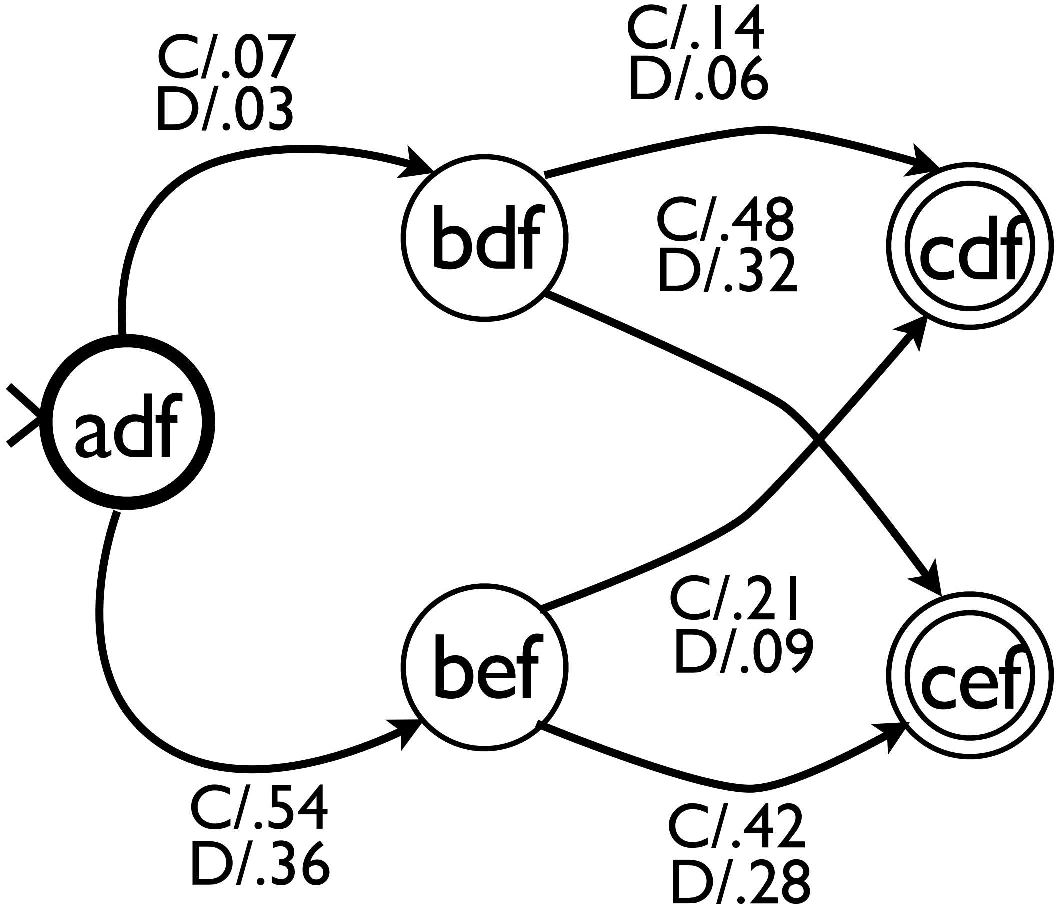
Pipeline approach

1-BEST() = ?

Project the range

(Pereira & Riley, 1997)

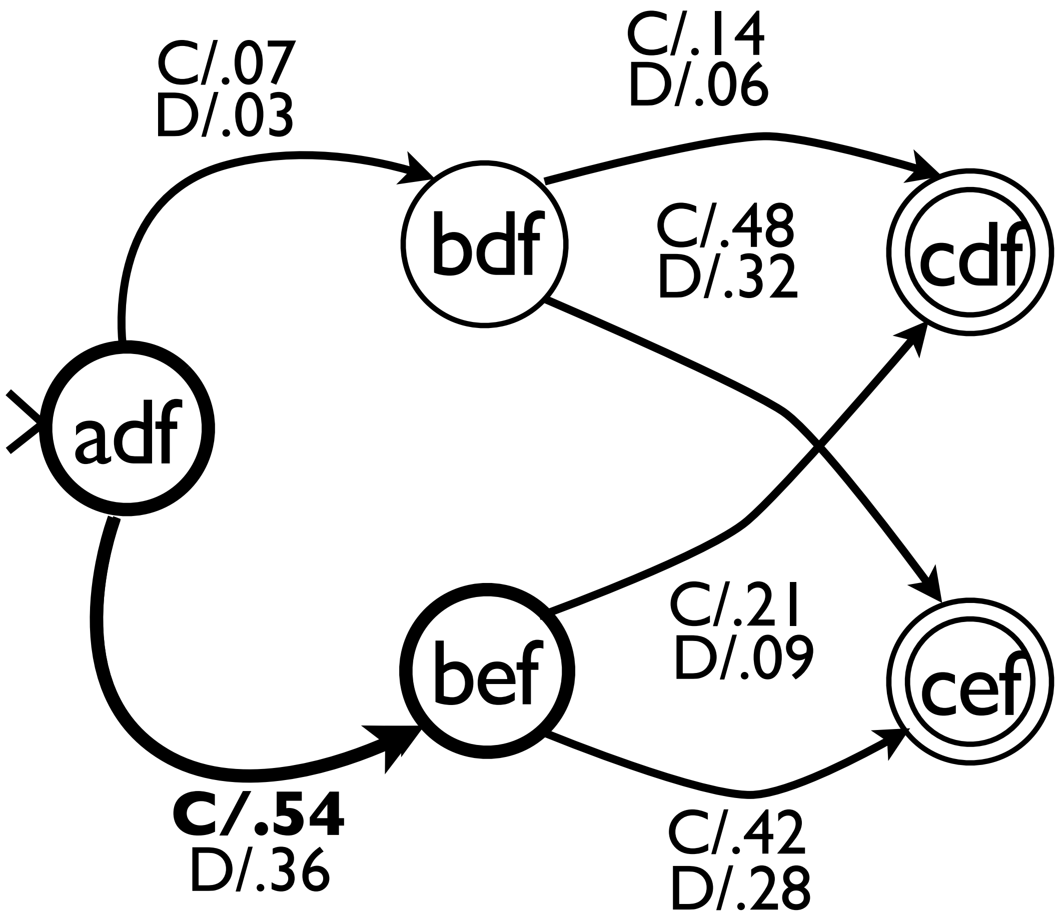
Pipeline approach

1-BEST() = ?

Find the 1-best path of the result

(Dijkstra, 1959)

Pipeline approach

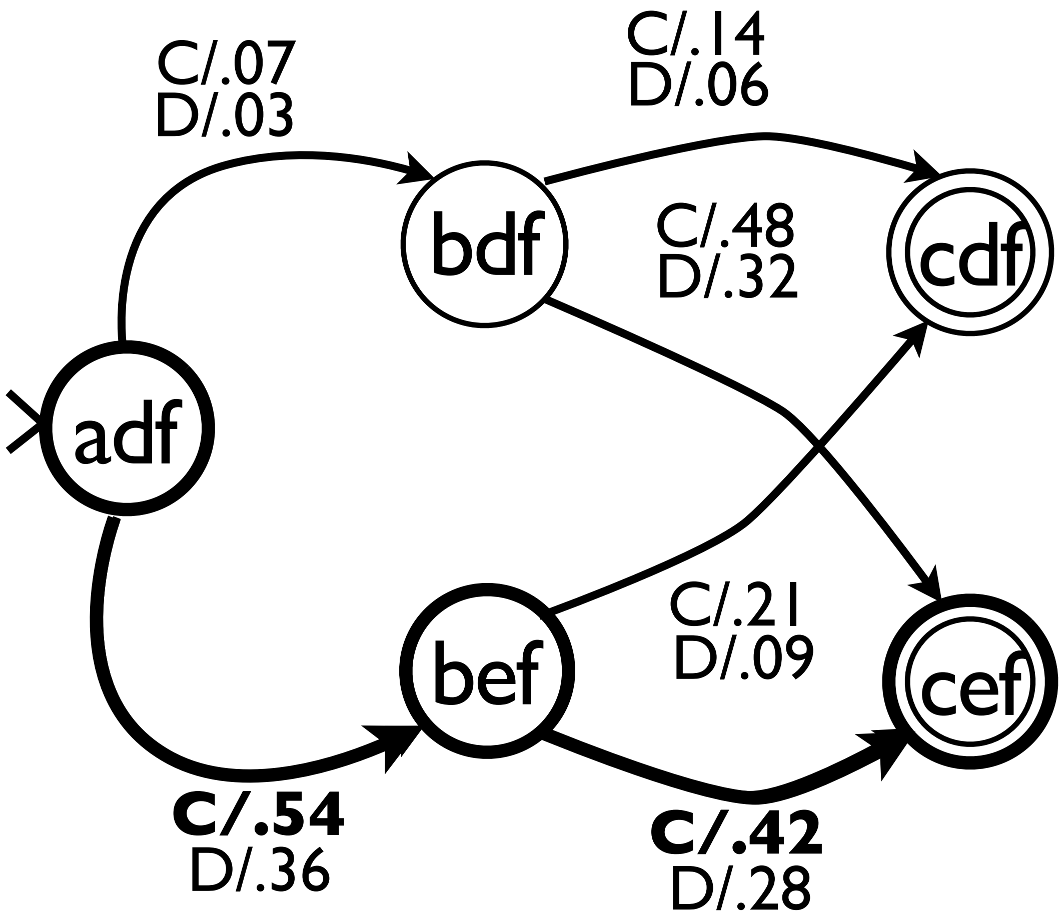
1-BEST() = ?

Find the 1-best path of the result

(Dijkstra, 1959)

Pipeline approach

1-BEST(



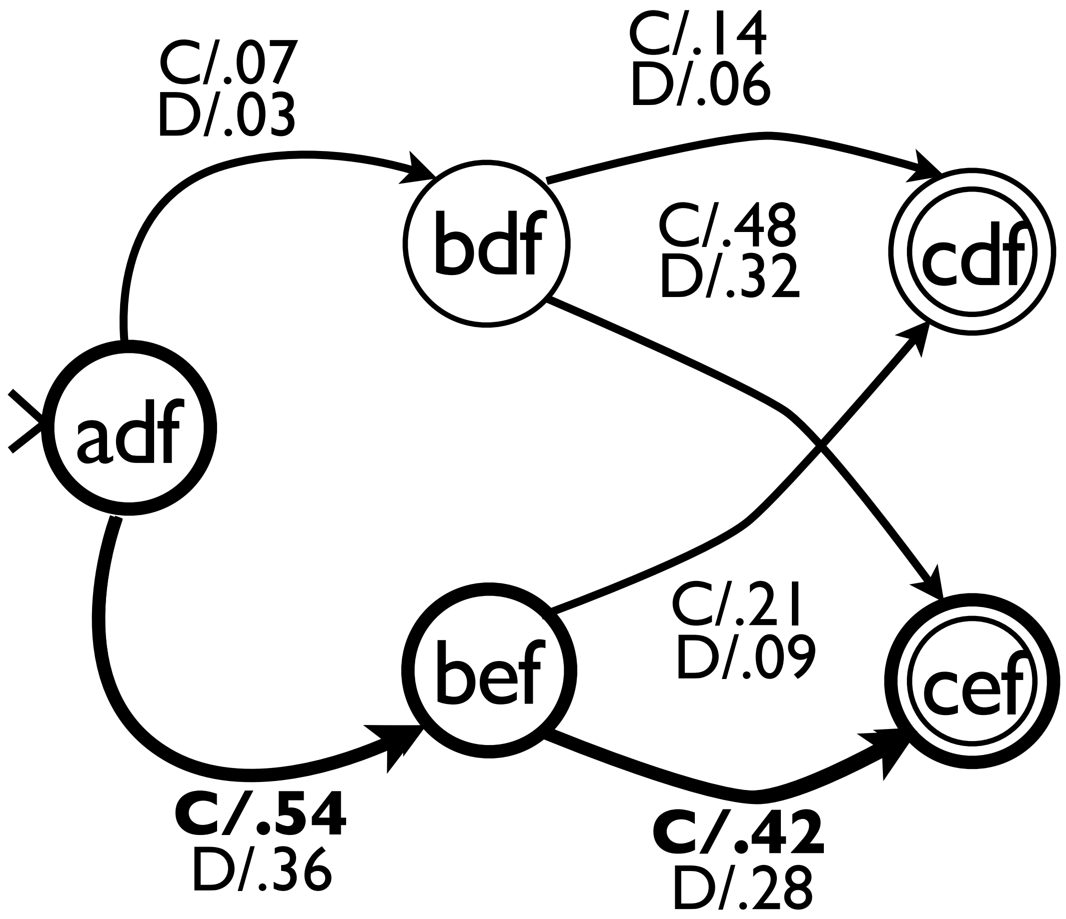
) = ?

Find the 1-best path of the result

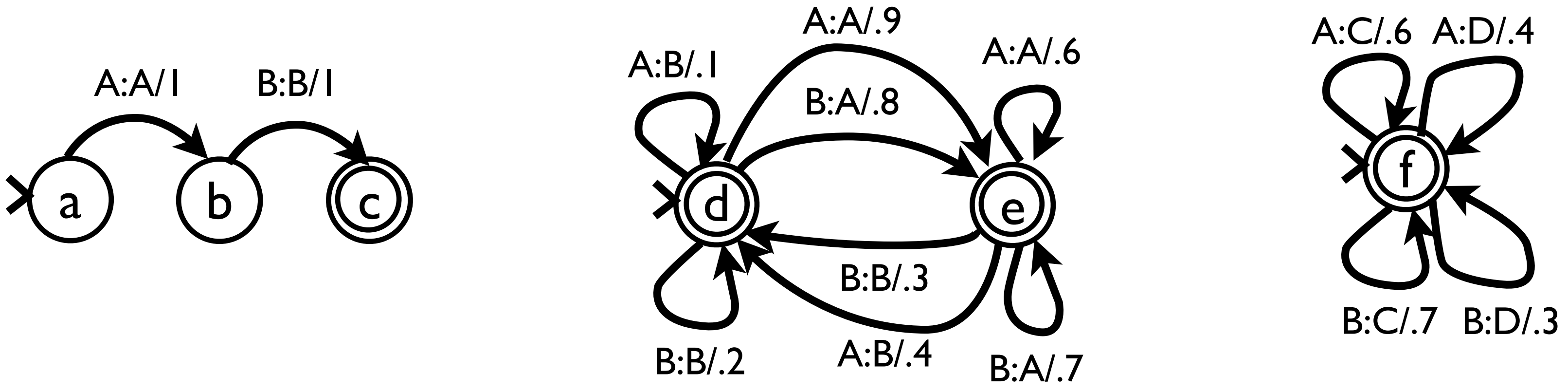
(Dijkstra, 1959)

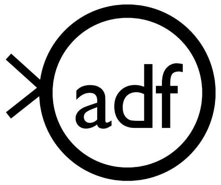
Problems with pipeline

- Extra work done to create unused arcs
- Building done without input of all cascade members



On-the-fly approach

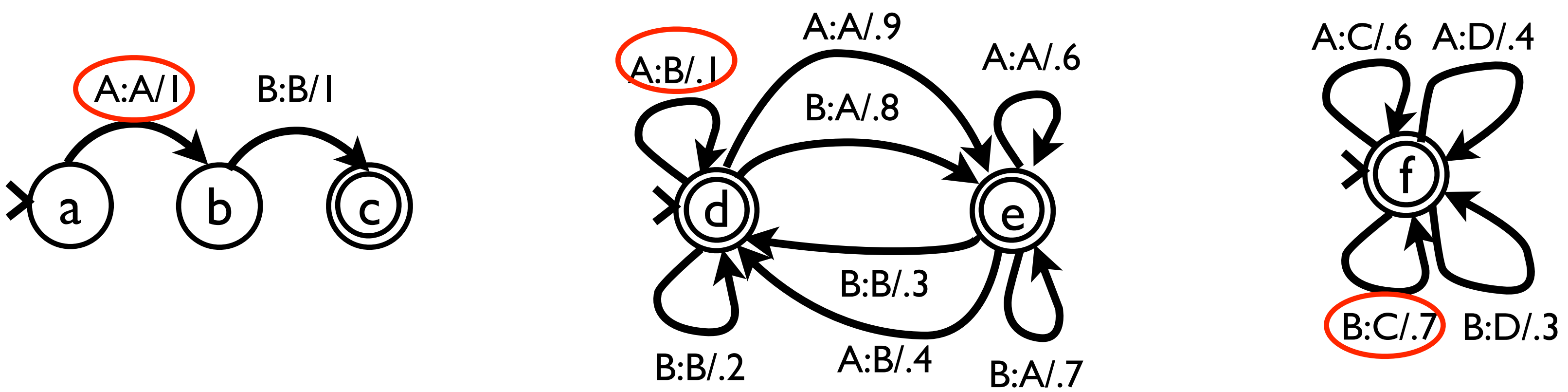


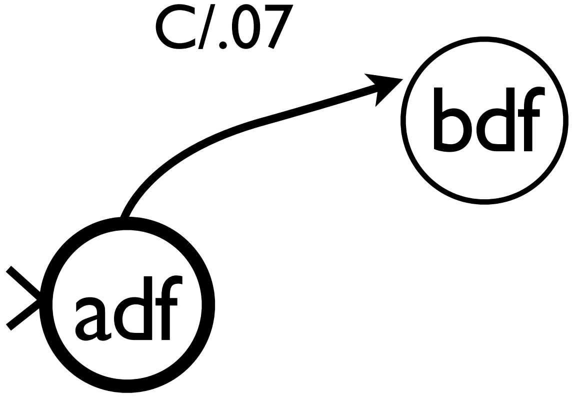
1-BEST() = ?

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach

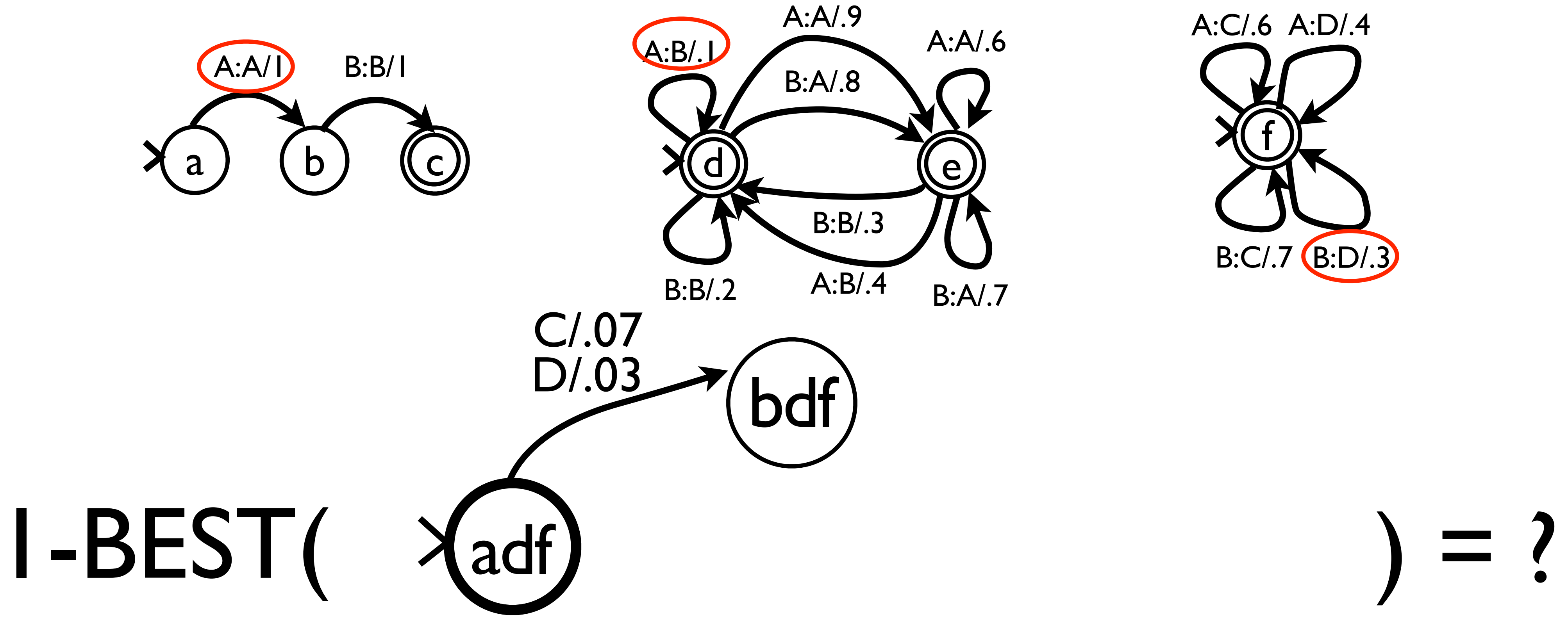


1-BEST() = ?

- Build arcs in result graph as needed
- All members of cascade "vote" simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

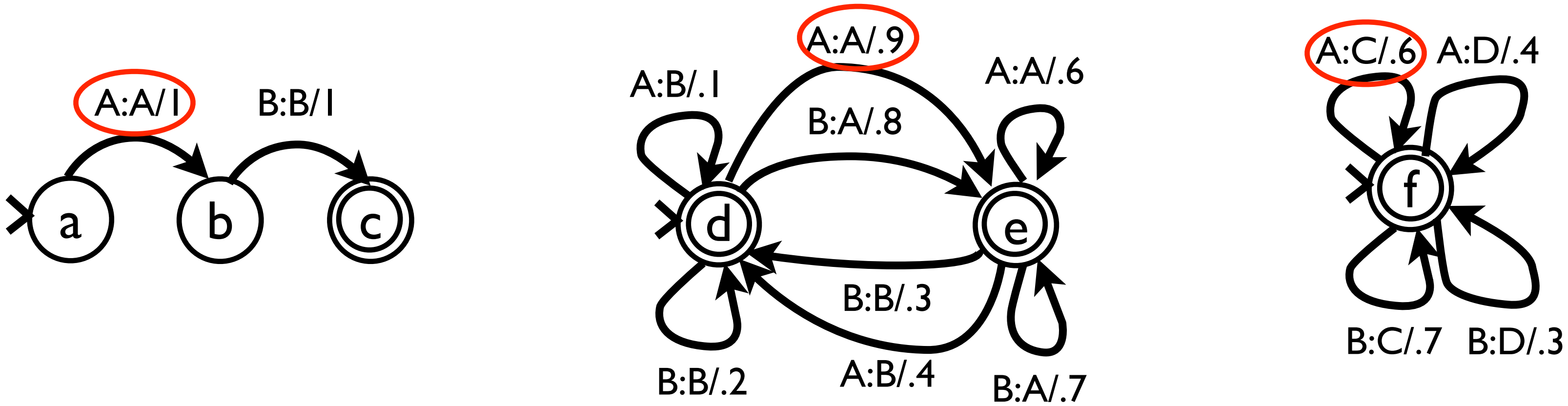
On-the-fly approach



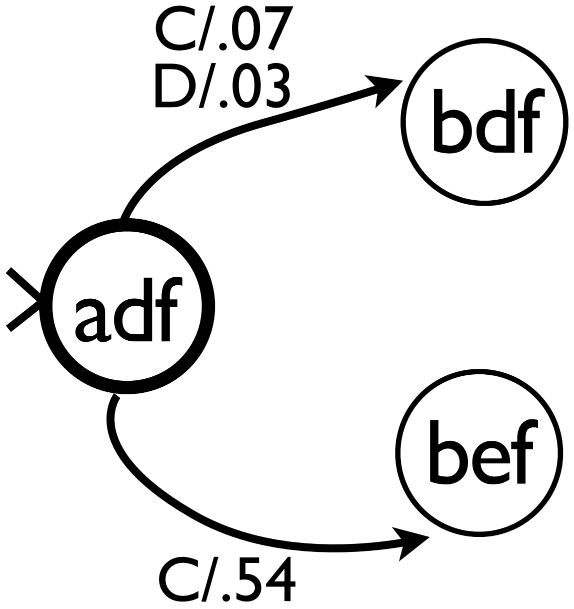
- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach



1-BEST(

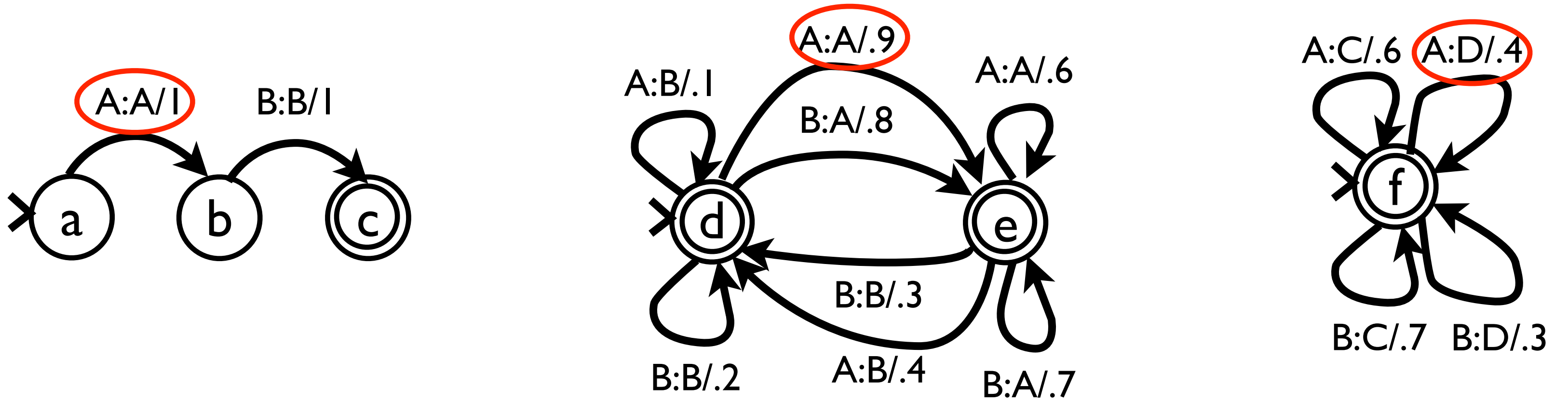


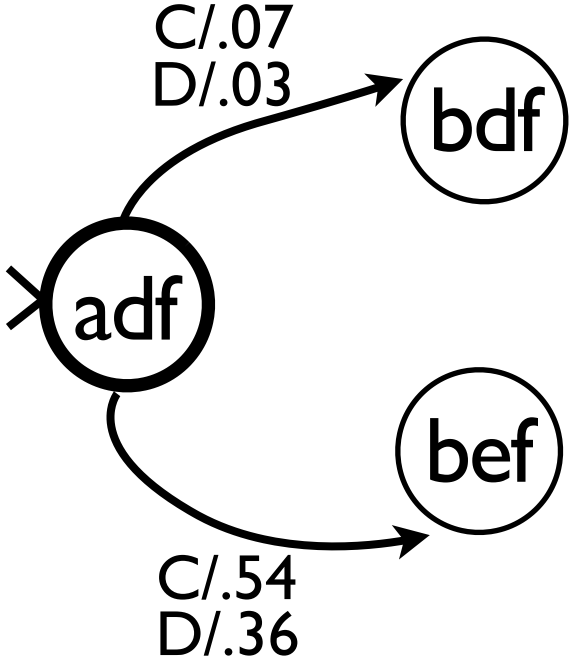
) = ?

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach

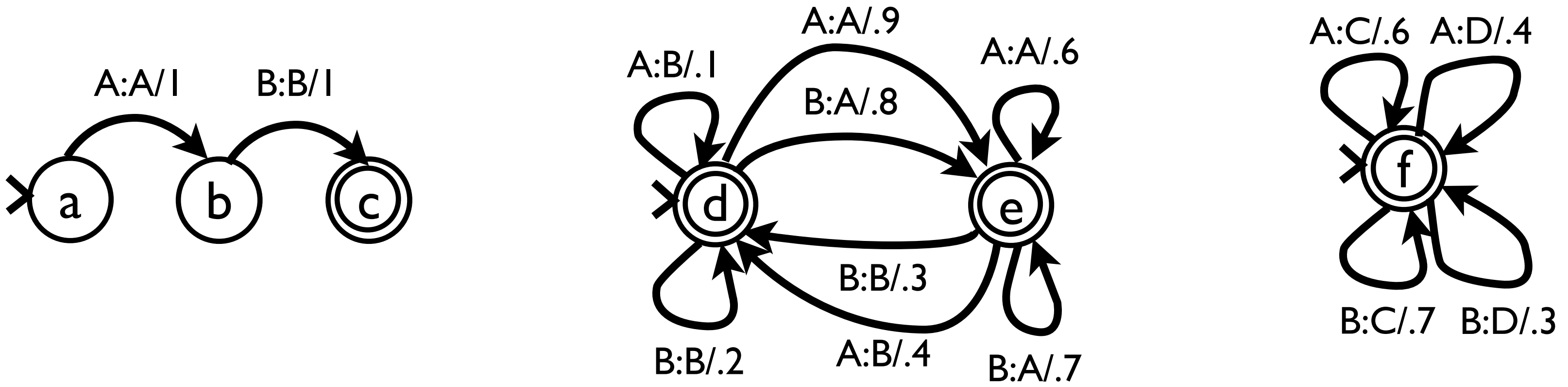


1-BEST() = ?

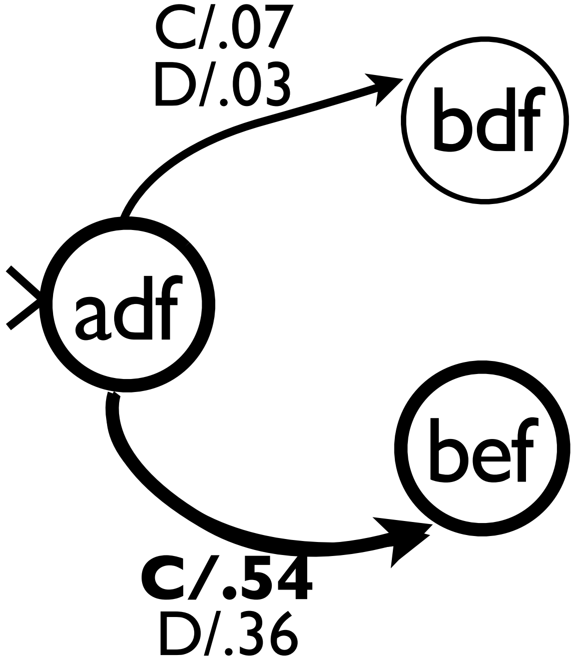
- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach



1-BEST(

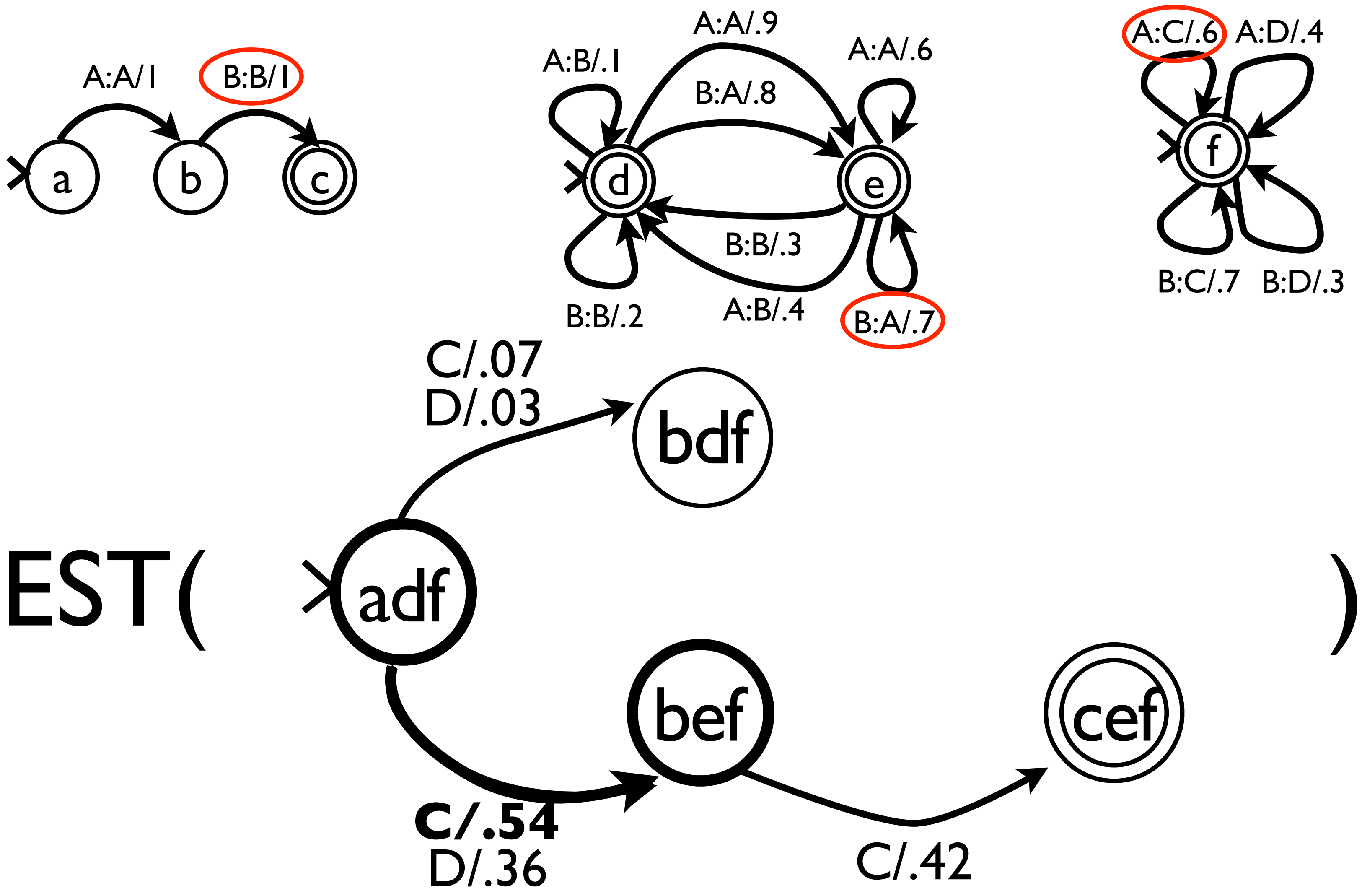


) = ?

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach

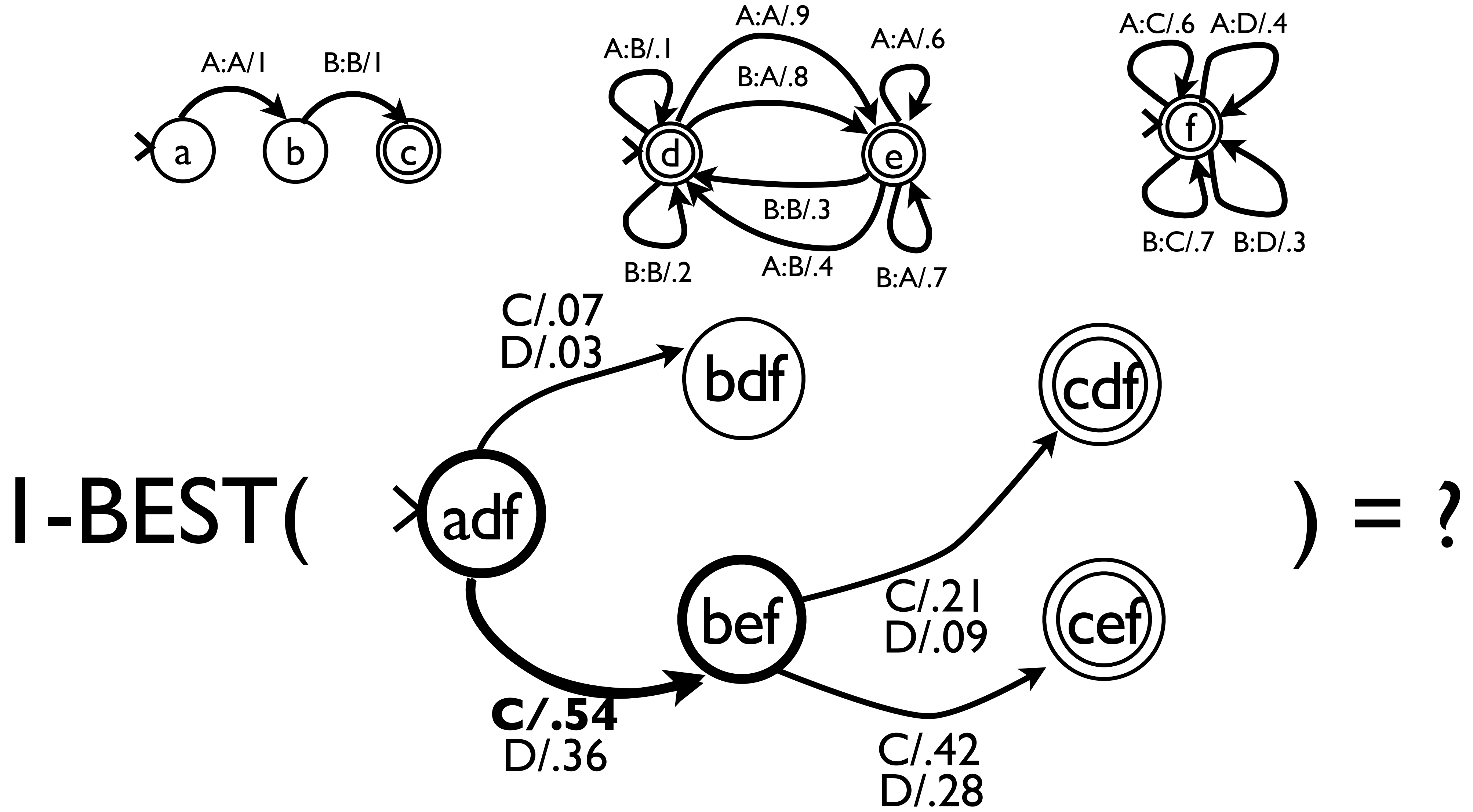


1-BEST( ) = ?

- Build arcs in result graph as needed
- All members of cascade "vote" simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

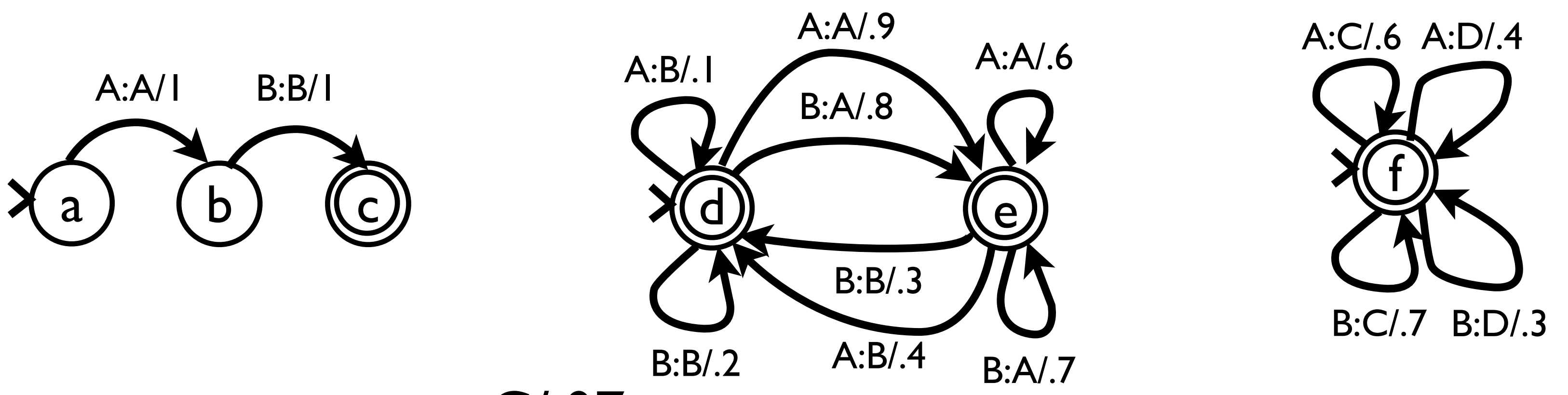
On-the-fly approach



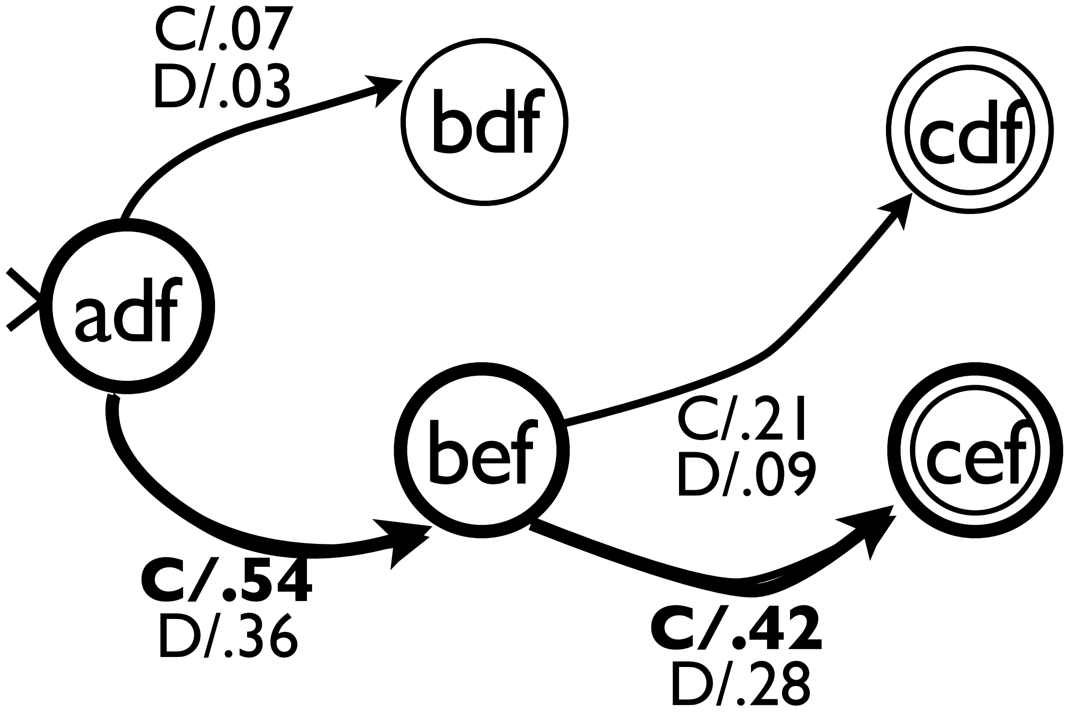
- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

On-the-fly approach



1-BEST(



) = ?

- Build arcs in result graph as needed
- All members of cascade "vote" simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

Inference through *tr*ee cascades?

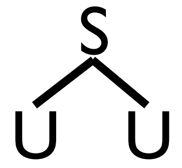
- In general, tree transducers are *not closed* under composition
- However, some classes are closed, and by adding additional steps to the process, we can conduct inference
- We provide pipeline and on-the-fly algorithms for applicable classes of weighted tree transducers

Inference through *tree* cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade

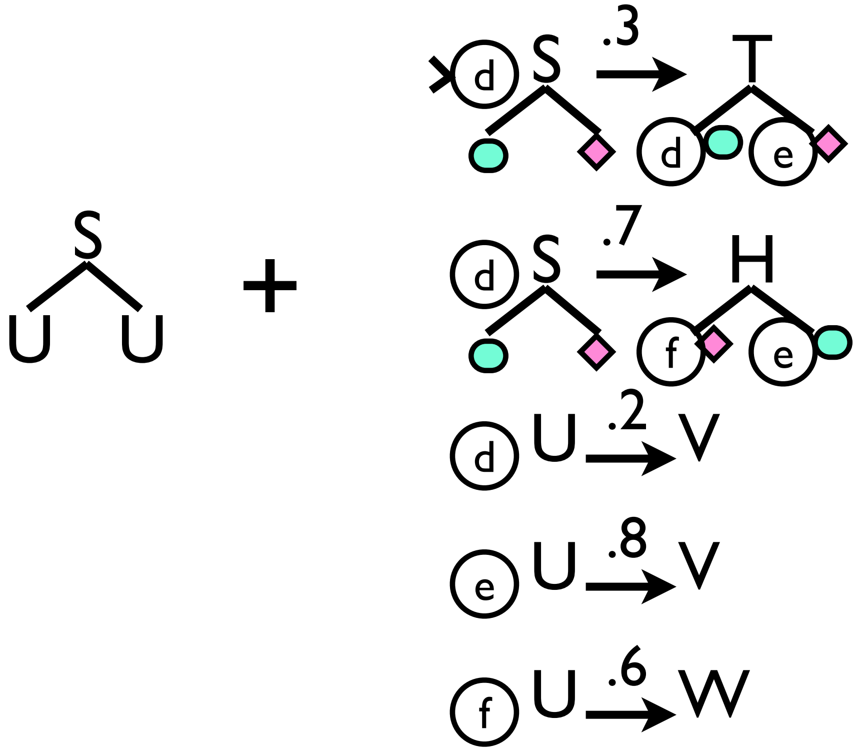
Inference through *tree* cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade



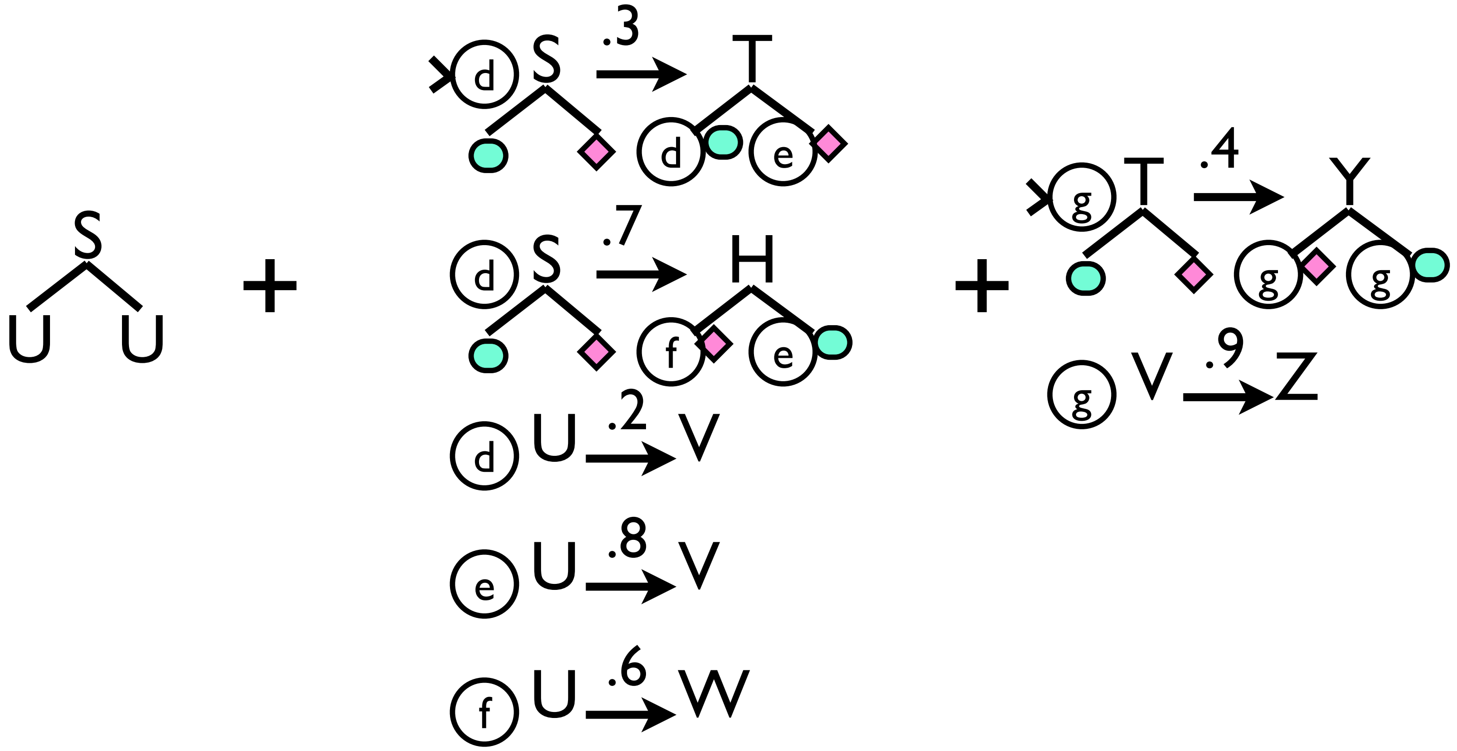
Inference through *tree* cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade



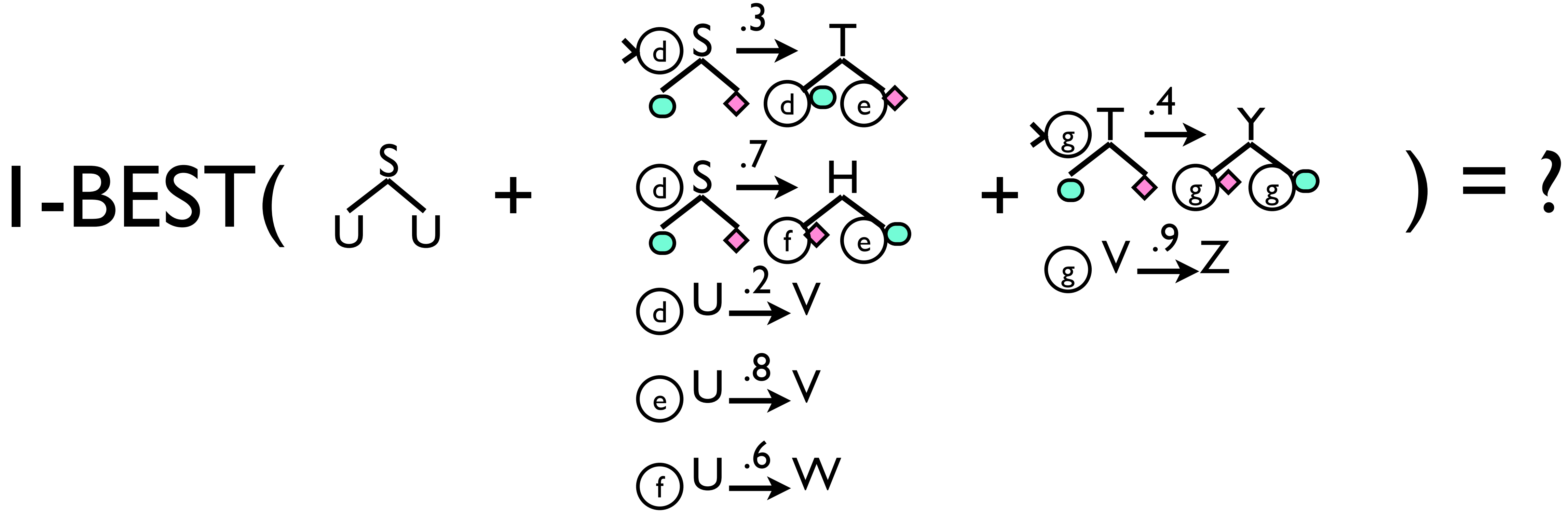
Inference through *tree* cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade

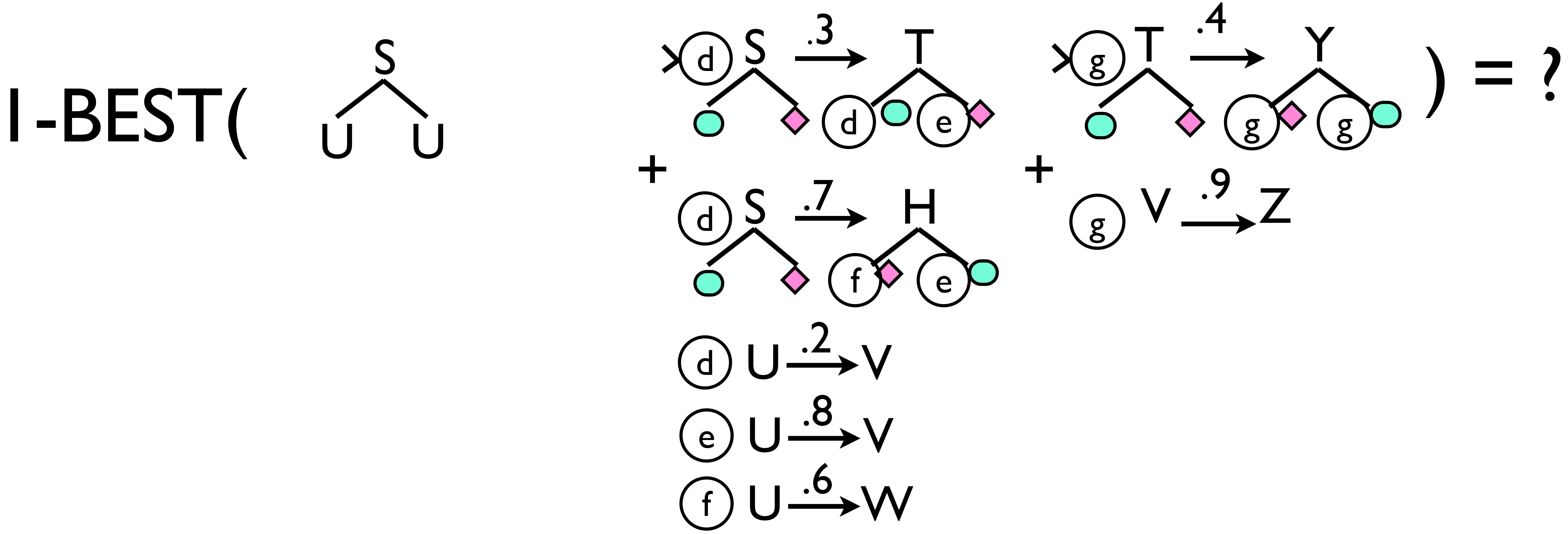


Inference through *tree* cascades

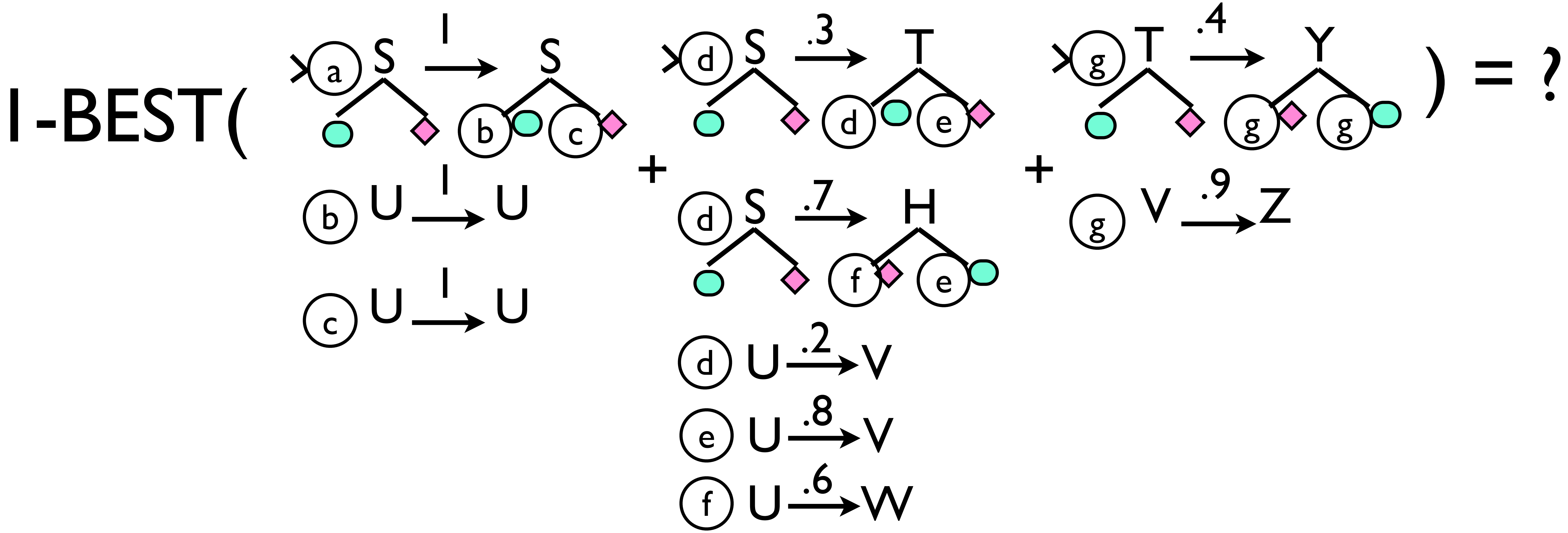
Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade



Pipeline approach

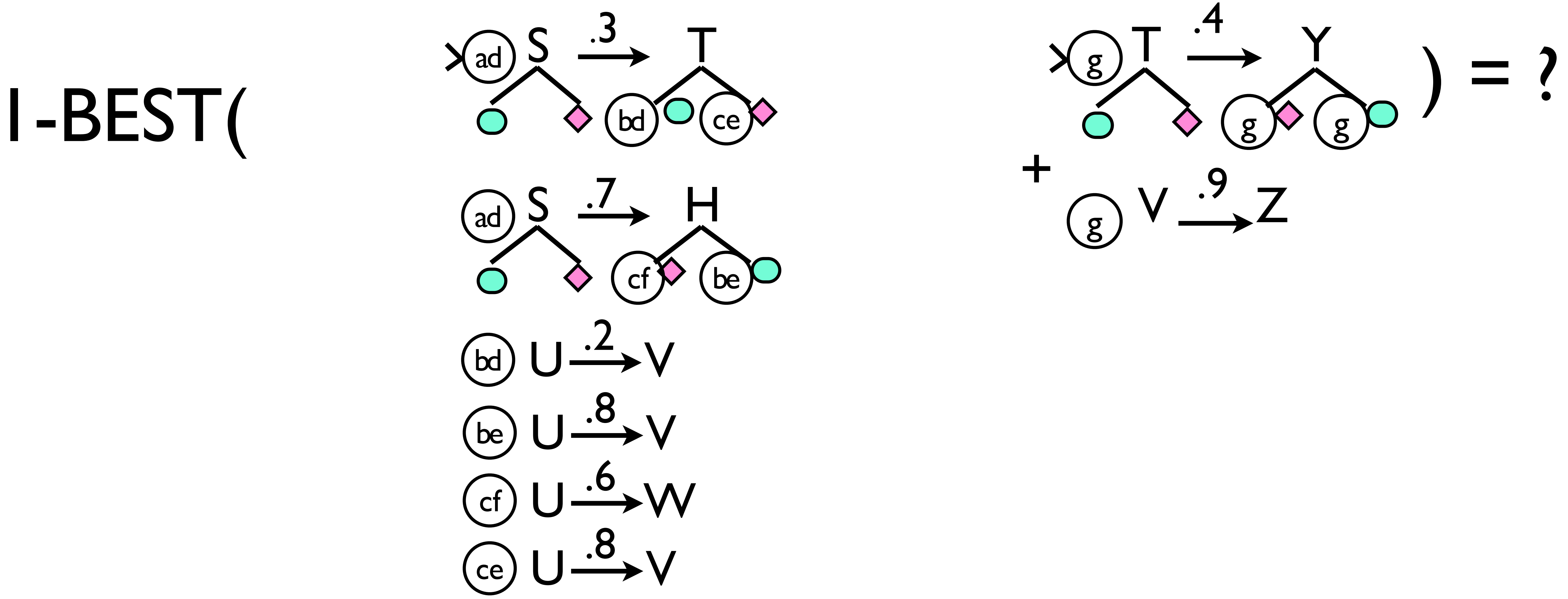


Pipeline approach



Embed the tree

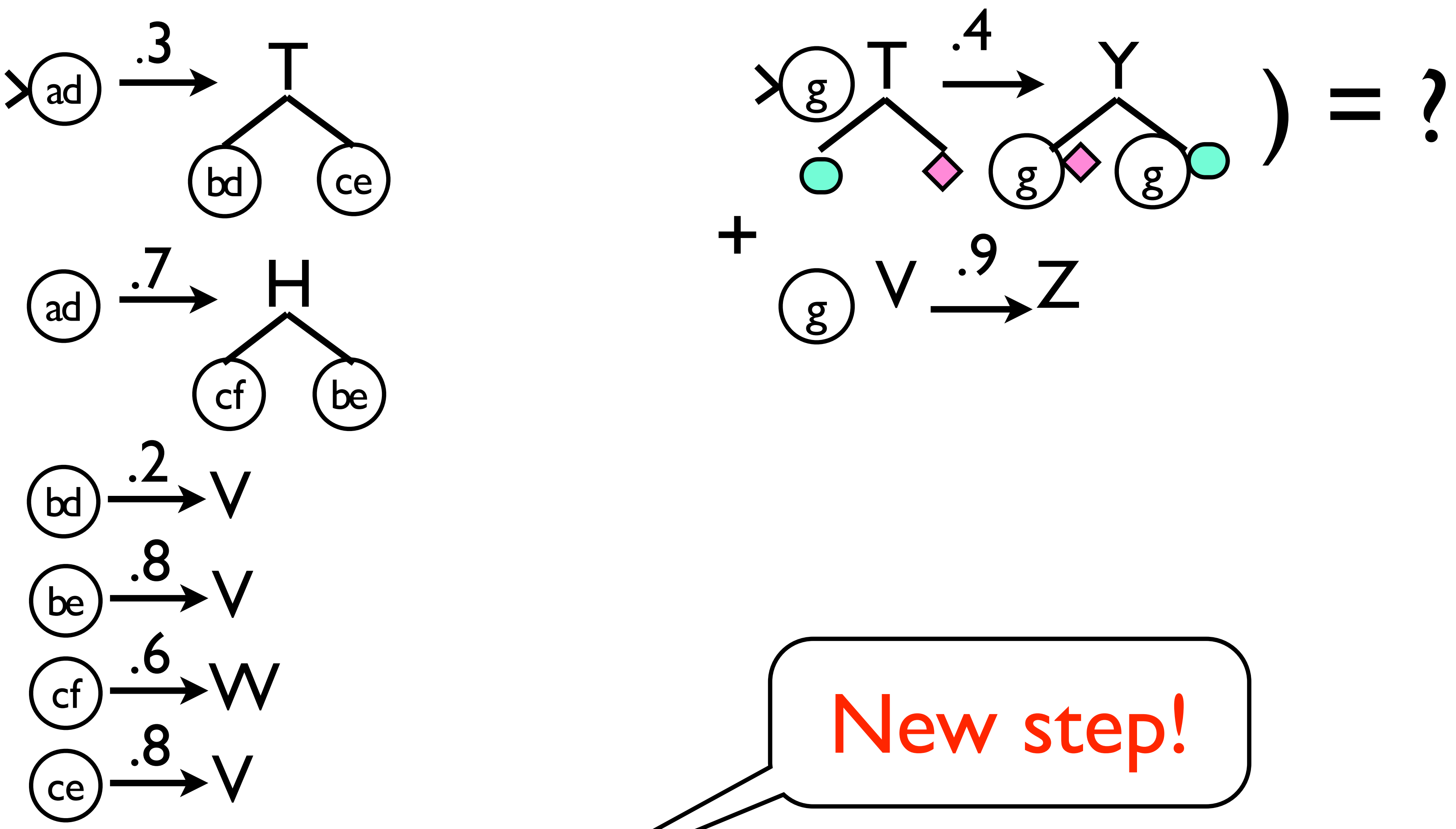
Pipeline approach



Compose adjacent transducers

Pipeline approach

1-BEST(

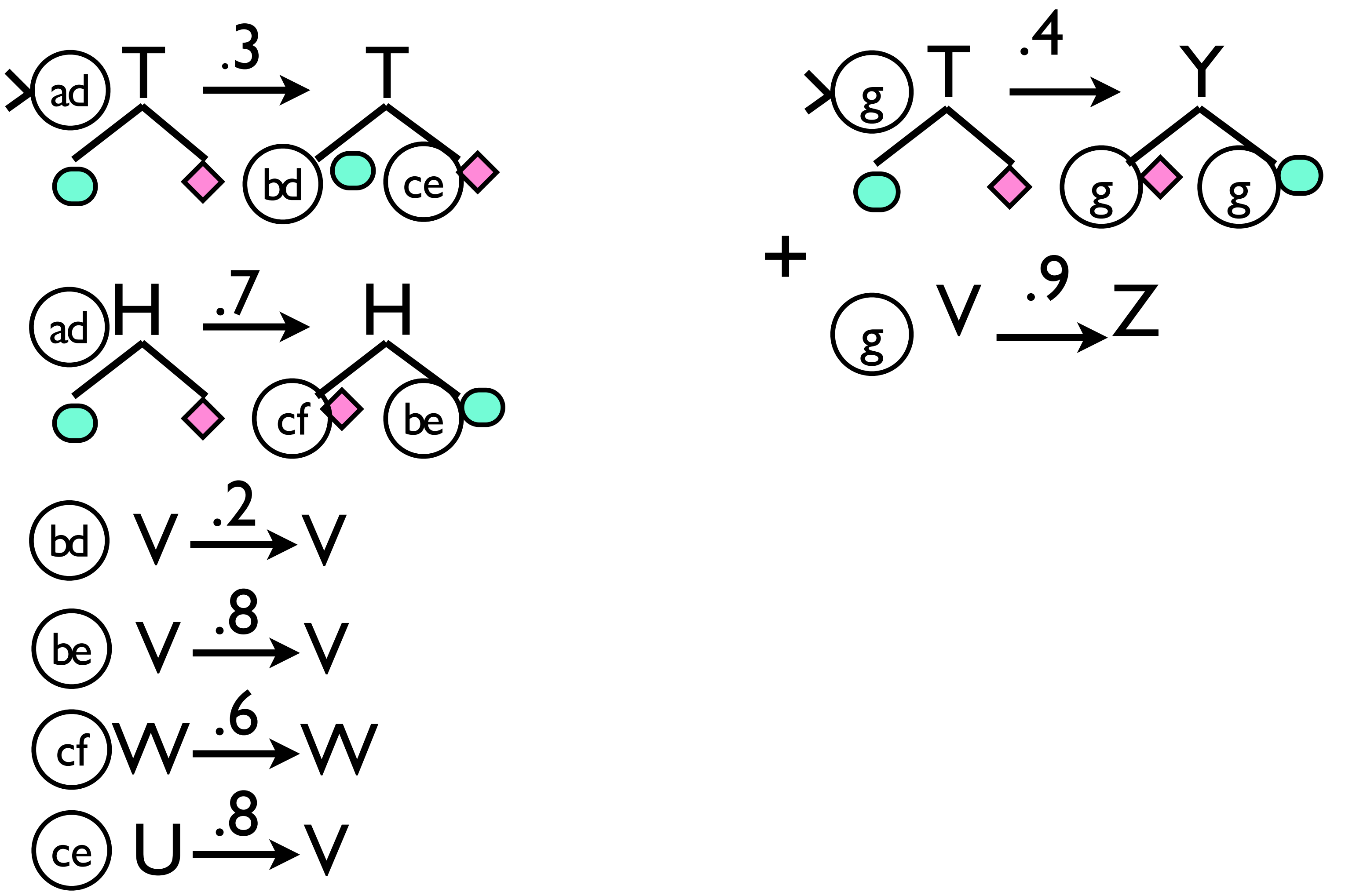


New step!

Project the range

Pipeline approach

1-BEST(

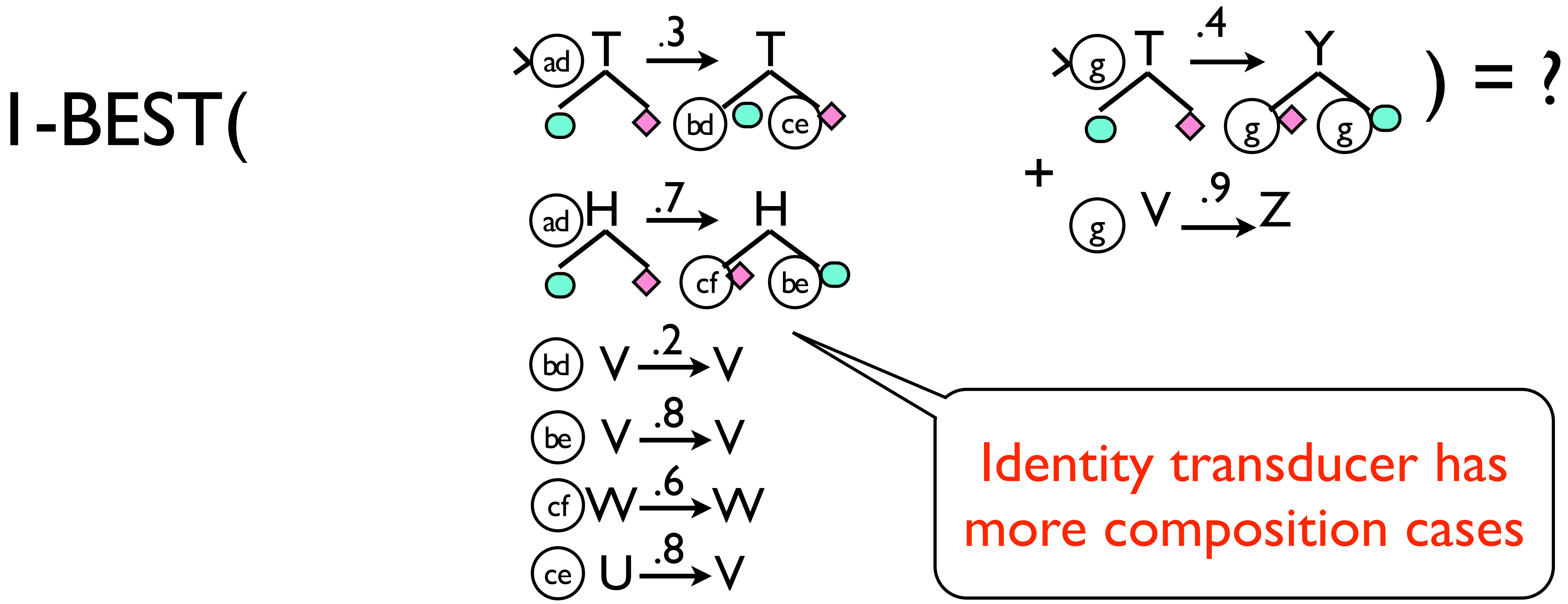


) = ?

New step!

Embed the grammar

Pipeline approach

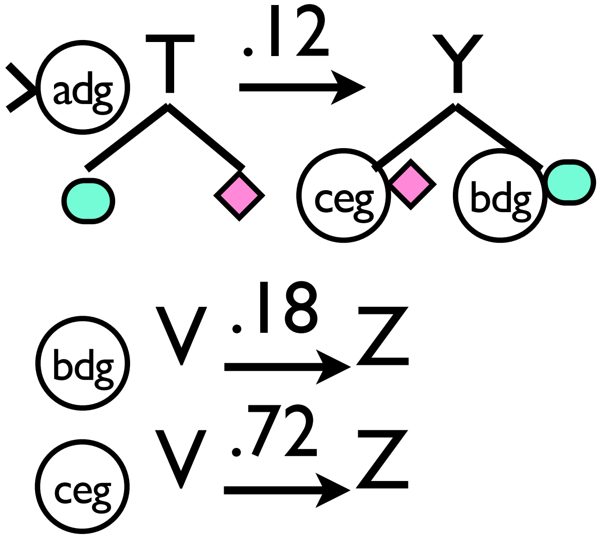


Identity transducer has more composition cases

Embed the grammar

Pipeline approach

1-BEST(

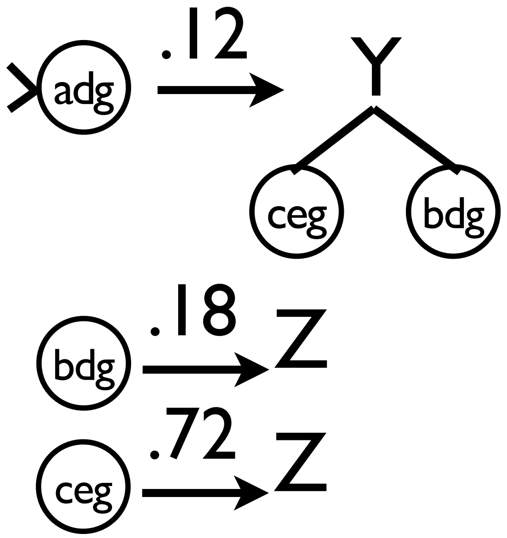


) = ?

Compose adjacent transducers

Pipeline approach

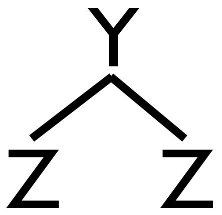
1-BEST(



) = ?

Project the range

Pipeline approach

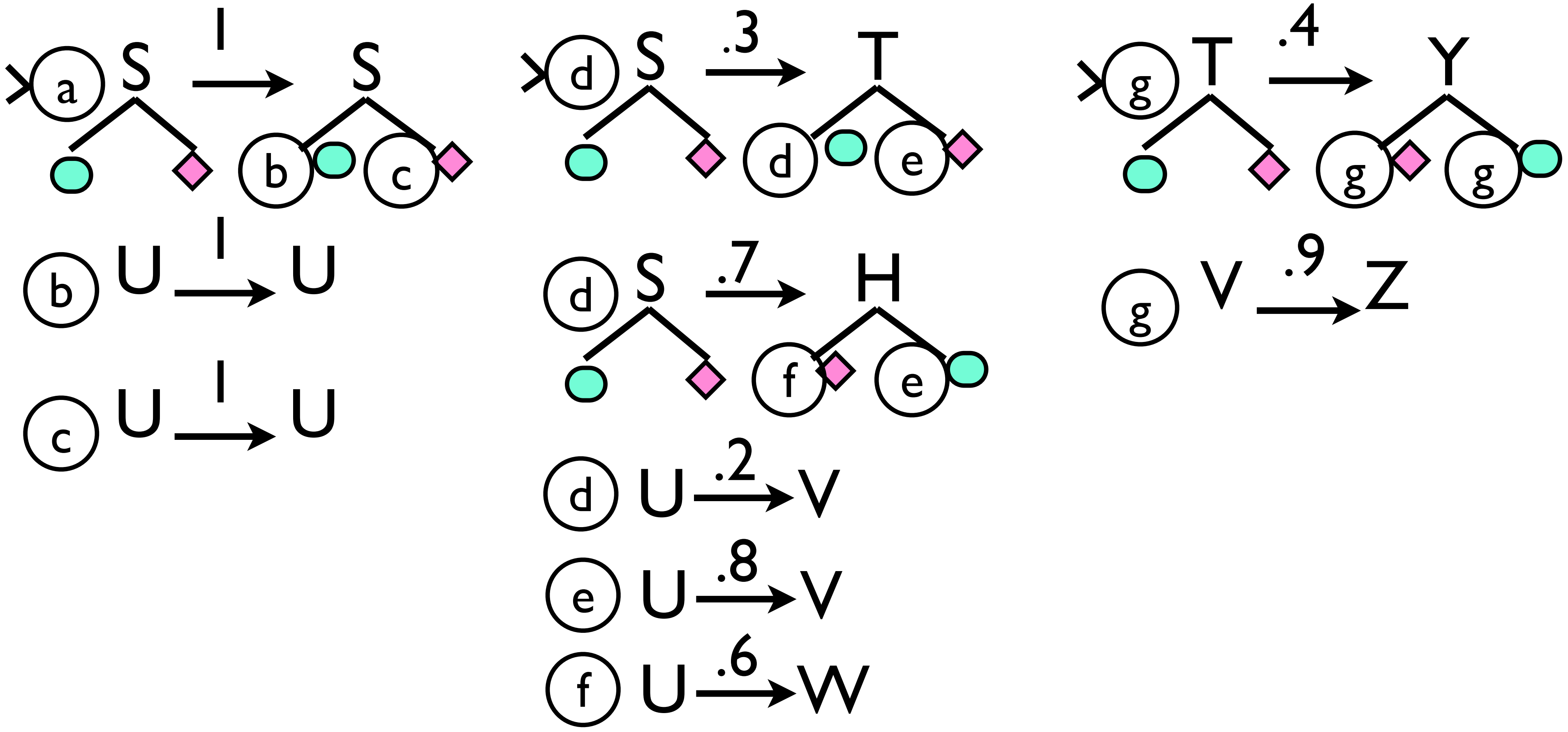


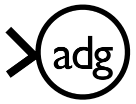
.016

Find 1-best path of the result

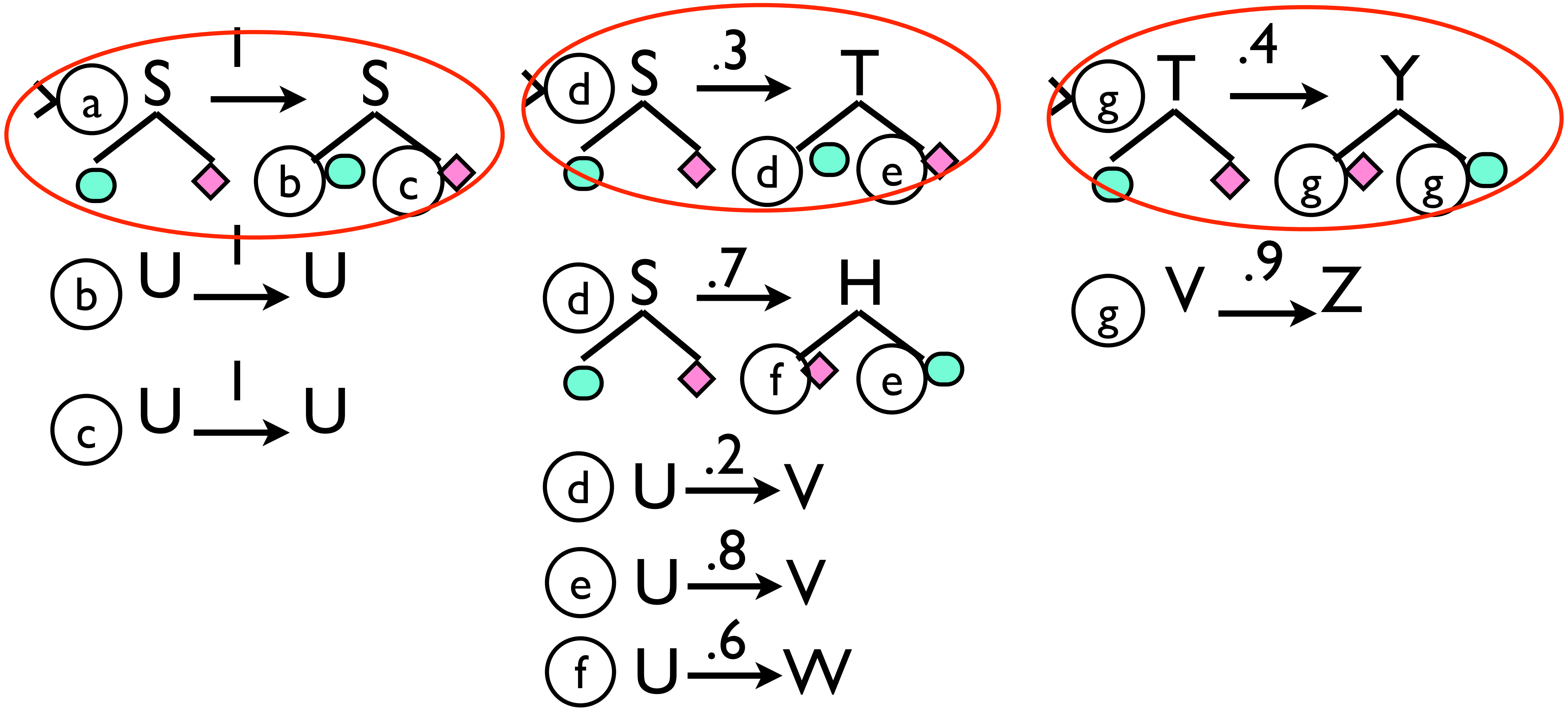
(Knuth ’77)

On-the-fly approach

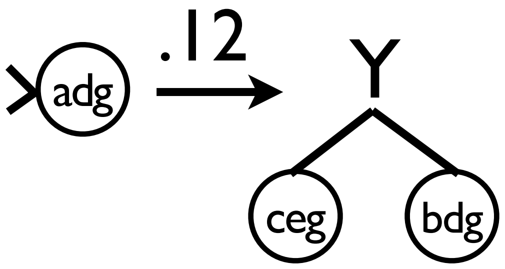


1-BEST() = ?

On-the-fly approach

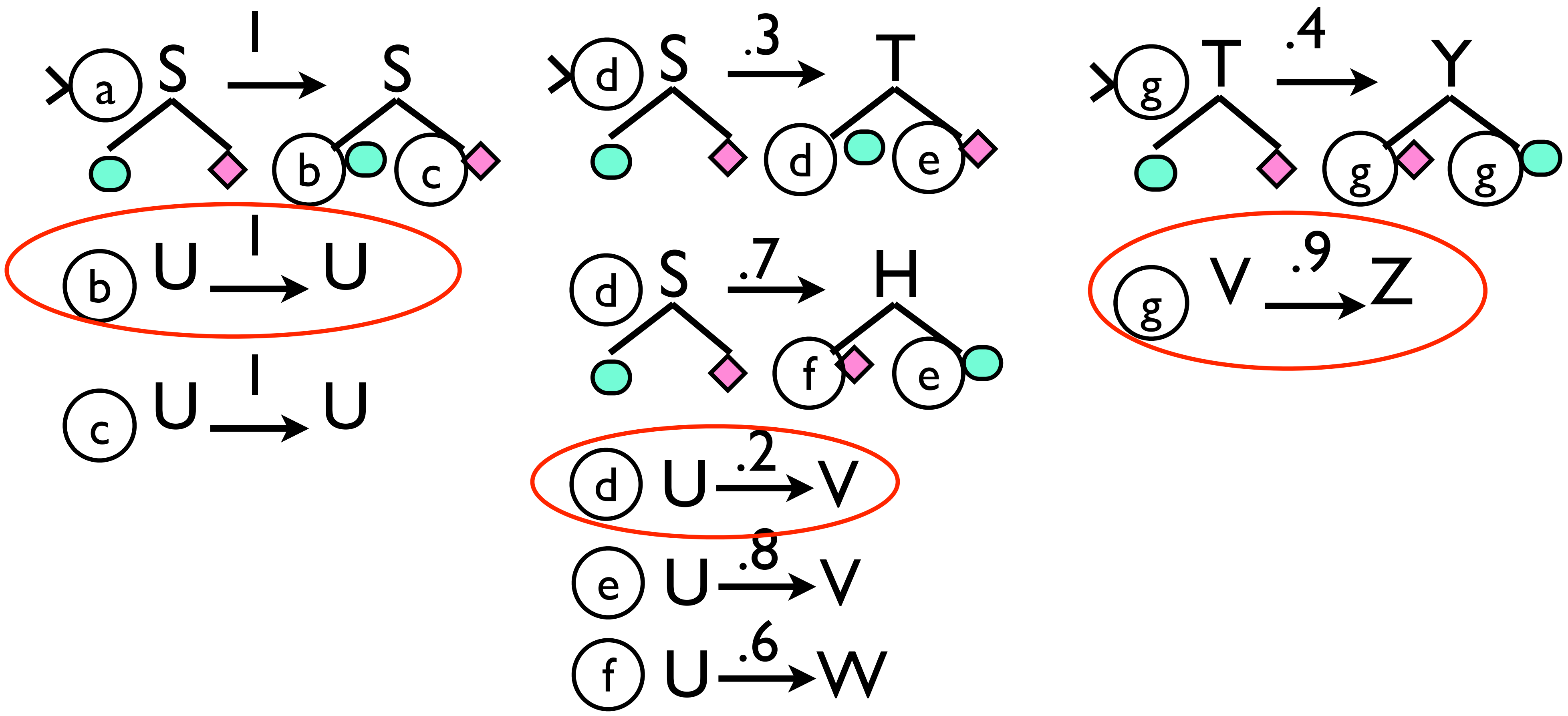


1-BEST(

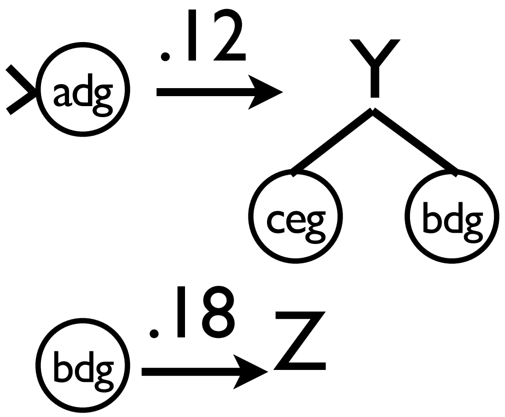


) = ?

On-the-fly approach

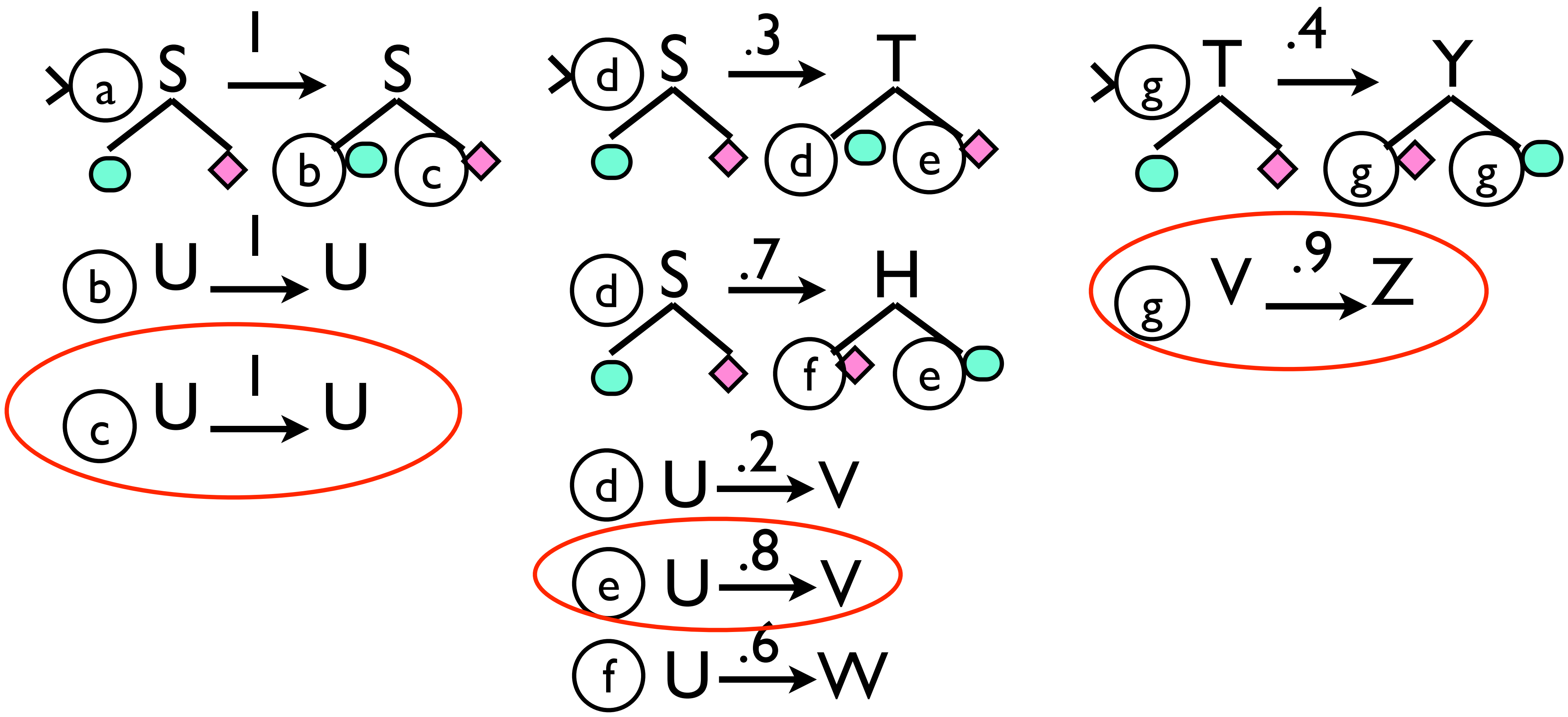


1-BEST(

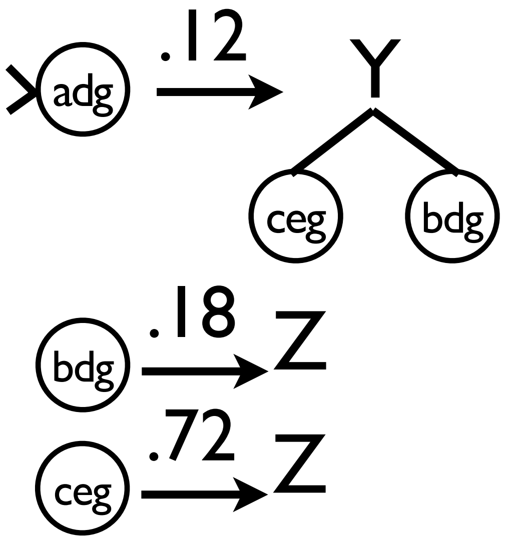


) = ?

On-the-fly approach

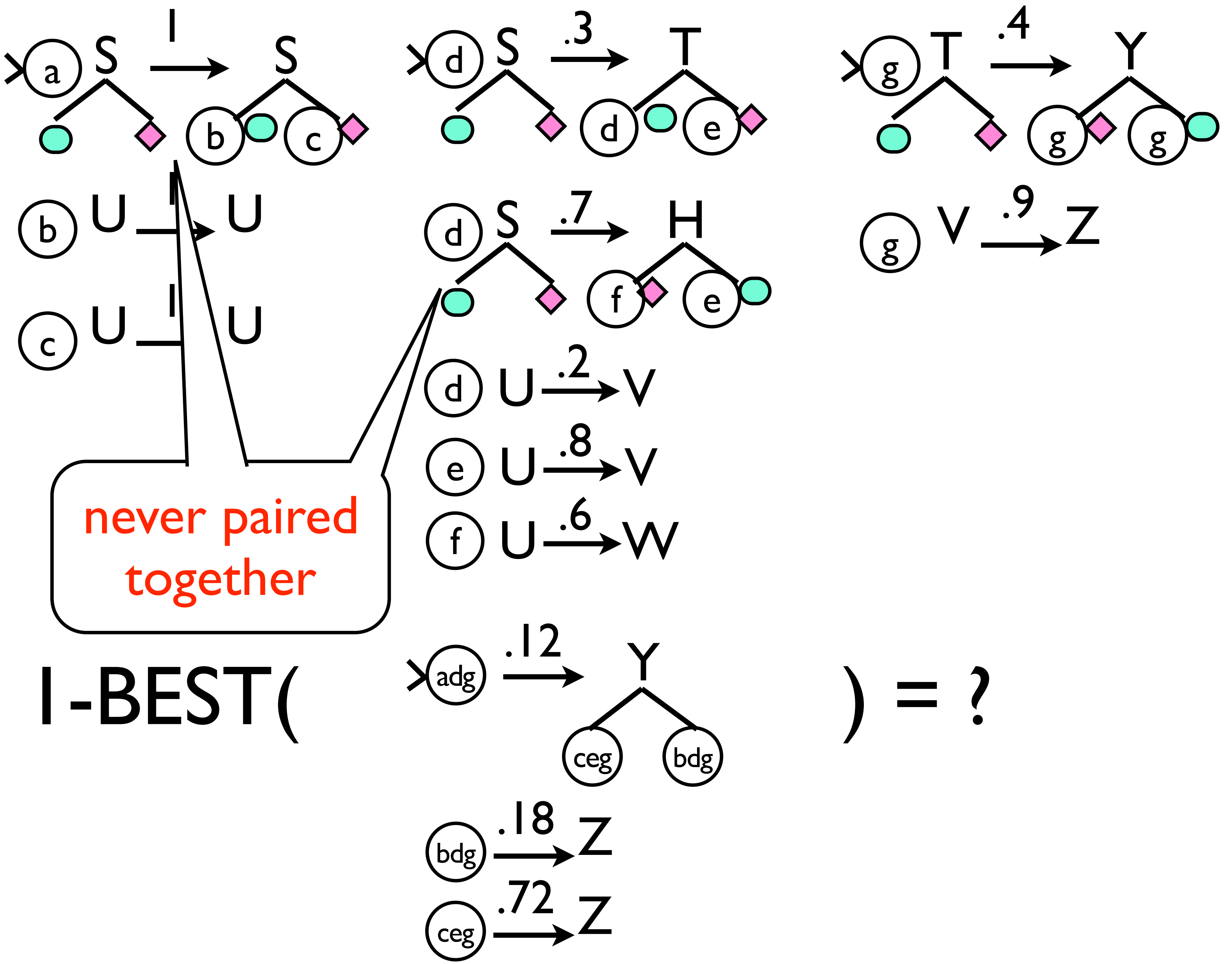


1-BEST(



) = ?

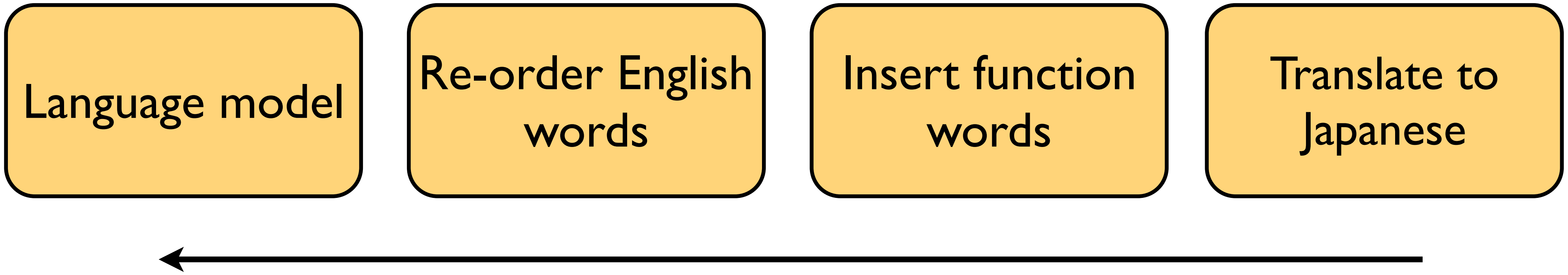
On-the-fly approach



never paired together

1-BEST( ) = ?

On-the-fly vs. pipeline


Language model

Re-order English words

Insert function words

Translate to Japanese

- We recovered 1-best English tree through this cascade
- We calculated time to complete for several language models and both pipeline and on-the-fly methods
- On-the-fly was much faster and in some cases the only method that worked in the memory allotted

(Yamada & Knight, 2001)

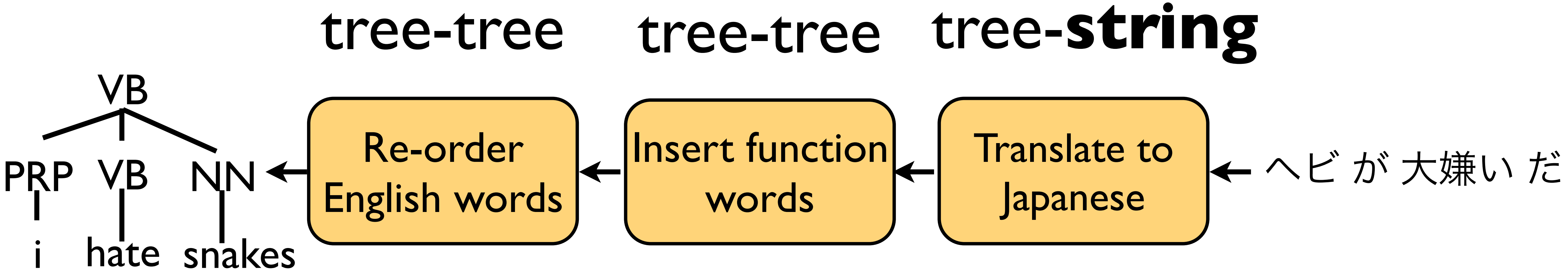
On-the-fly vs. pipeline

language model	method	time/sentence
weak	pipeline	28s
	on-the-fly	17s
strong & large	pipeline	>60s*
	on-the-fly	24s
strong & small	pipeline	2.5s
	on-the-fly	.06s

* Ran out of memory before completing

Extension for tree-string transducers

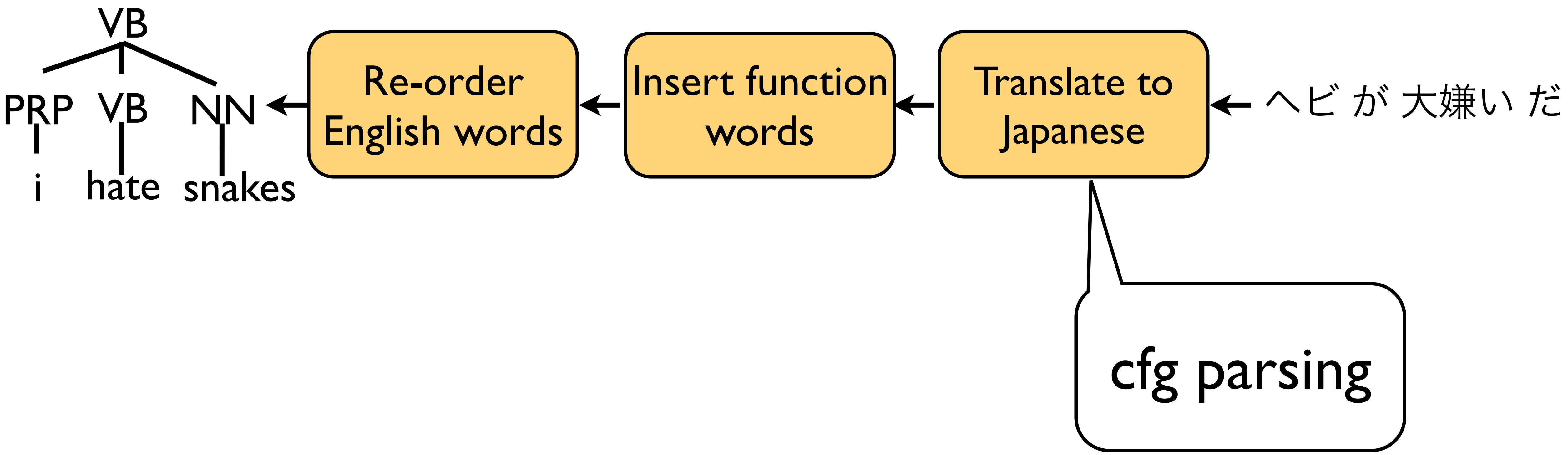
What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?



Extension for tree-string transducers

What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?

tree-tree tree-tree tree-**string**



Re-order English words

Insert function words

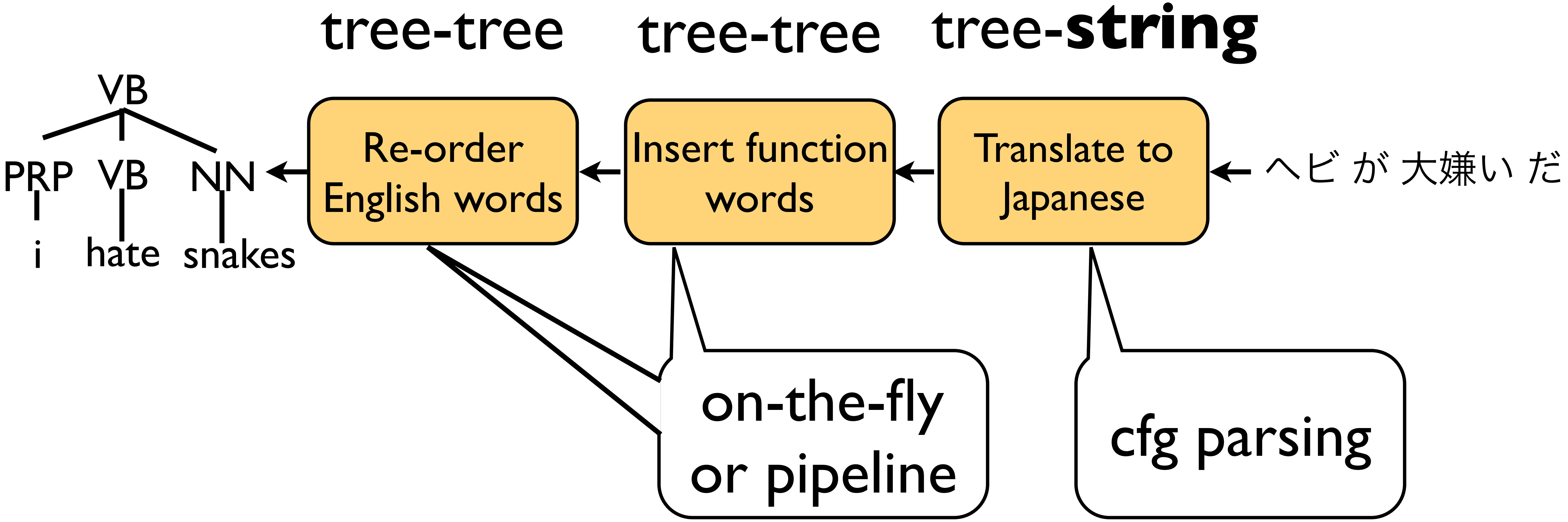
Translate to Japanese

ヘビ が 大嫌い だ

cfg parsing

Extension for tree-string transducers

What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?



A weighted tree automata and transducer toolkit

(May & Knight, CIAA '06)

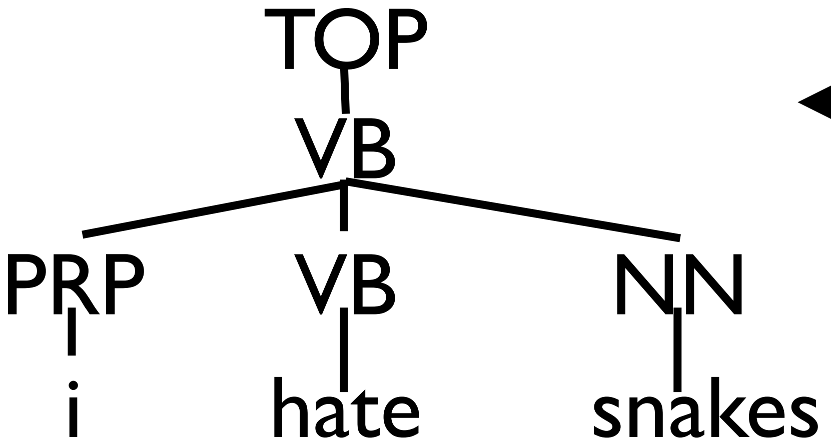
- Operations for inference, manipulation, and training of tree transducers and automata

- Very easy to experiment quickly, without coding

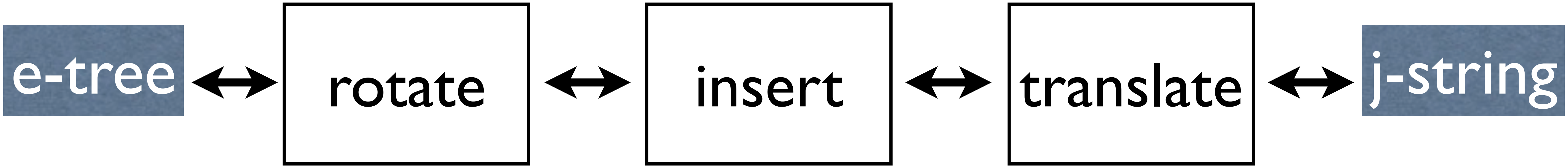
- http://www.isi.edu/licensed-sw/tiburon

Tiburon example 1: syntax MT cascade

Simplified English trees to Japanese strings



↔ ヘビ が 大嫌い だ

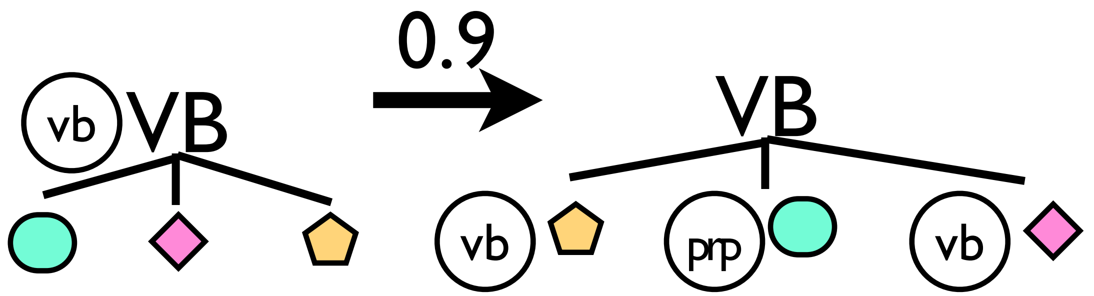


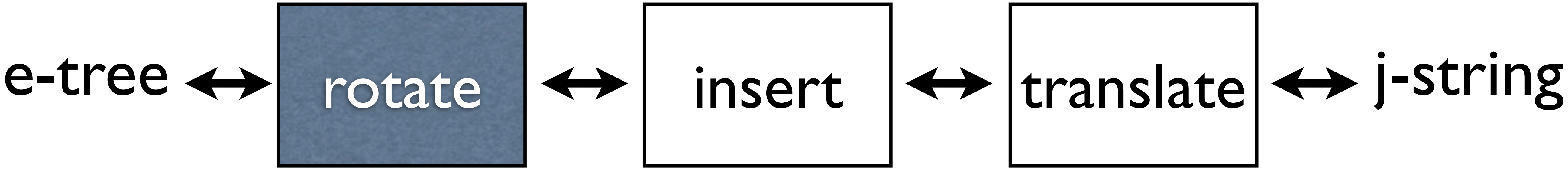
e-tree ↔ rotate ↔ insert ↔ translate ↔ j-string

(Yamada & Knight, 2001)

Tiburon example 1: syntax MT cascade

1) Rotate children

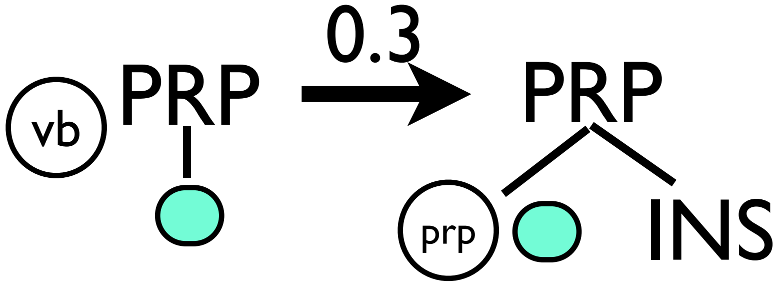


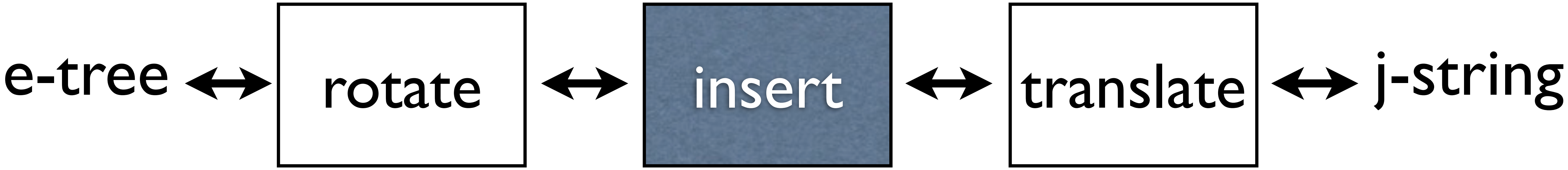


(Yamada & Knight, 2001)

Tiburon example 1: syntax MT cascade

2) Insert function words

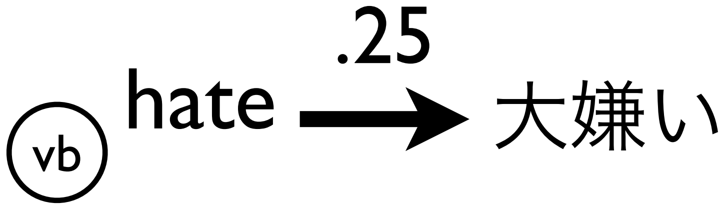


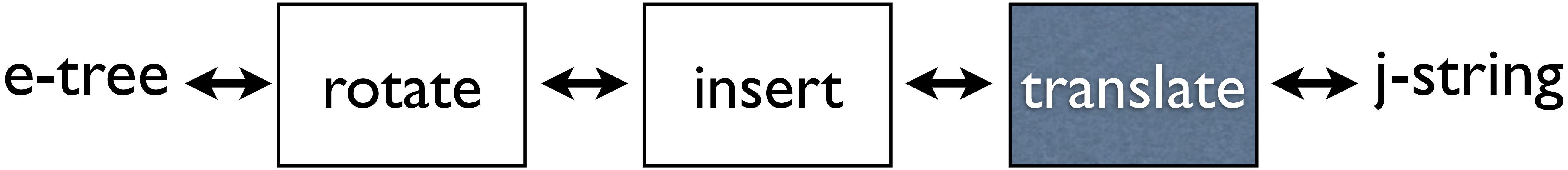


(Yamada & Knight, 2001)

Tiburon example 1: syntax MT cascade

3) Translate leaves




e-tree ↔ rotate ↔ insert ↔ translate ↔ j-string

(Yamada & Knight, 2001)

Tiburon example 1: syntax MT cascade

- Task: Decode candidate sentence, get top 5 answers
- Algorithms used: inference through cascade, k-best, determinization

Candidate: 彼ら は 偽善 が 大嫌い だ

Correct answer:

TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))))

Tiburon example 1: syntax MT cascade

Let’s try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

Tiburon example 1: syntax MT cascade

Let's try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

program

Tiburon example 1: syntax MT cascade

Let’s try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

5 best

Tiburon example 1: syntax MT cascade

Let’s try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

semiring

Tiburon example 1: syntax MT cascade

Let’s try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

character set

Tiburon example 1: syntax MT cascade

Let’s try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

cascade

Tiburon example 1: syntax MT cascade

Let's try it!

```
% tiburon  -k 5  -m tropical  -e euc-jp rot ins trans ej.1.f
```

input

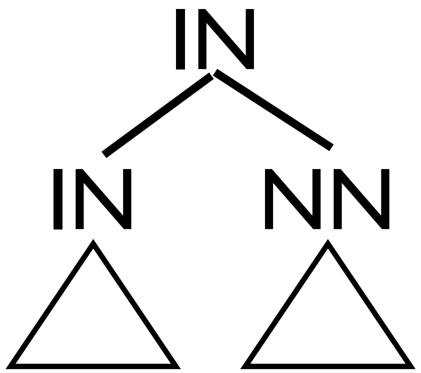
Tiburon example 1: syntax MT cascade

First try is not so good!

```
% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f
TOP(VB(PRP("him") VB("abominate") IN(IN("above") NN(JJ("abhorrent") NN("fanatic"))))) # 18.368
TOP(VB(PRP("them") VB("abominate") IN(IN("above") NN(JJ("abhorrent") NN("fanatic"))))) # 18.368
TOP(VB(PRP("him") VB("abominate") IN(IN("above") NN(JJ("abhorrent") NN("hypocrisy"))))) # 18.368
TOP(VB(PRP("them") VB("abominate") IN(IN("above") NN(JJ("abhorrent") NN("hypocrisy"))))) # 18.368
TOP(VB(PRP("him") VB("abominate") IN(IN("above") NN(JJ("abhorrent") NN("clouds"))))) # 18.368
```

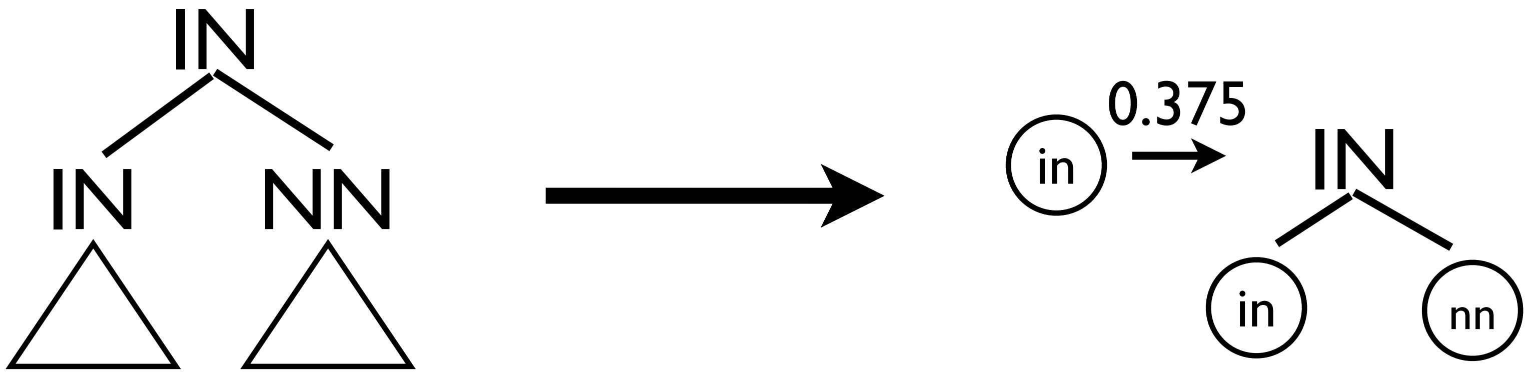
Tiburon example 1: syntax MT cascade

Add in a simple PCFG-based language model



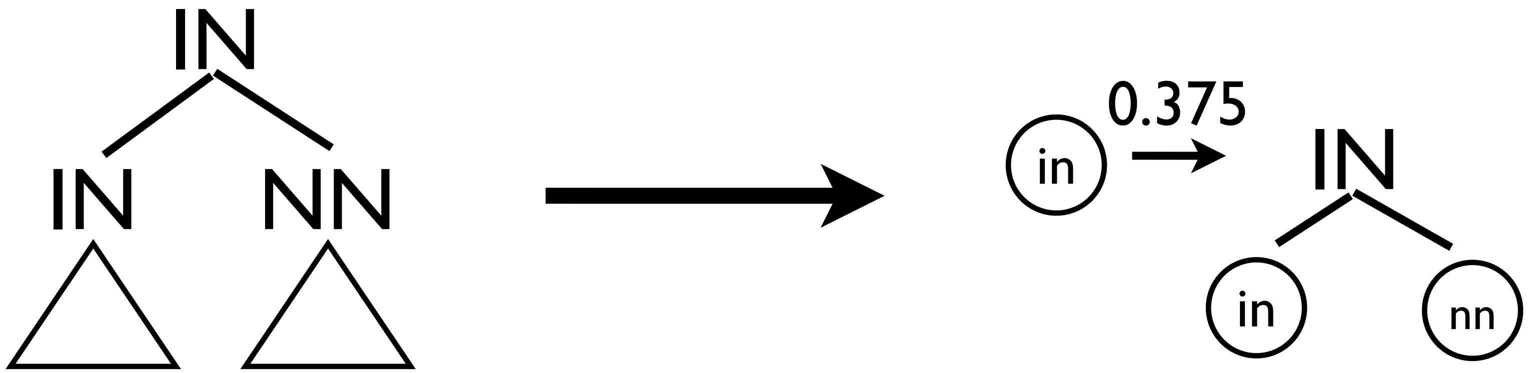
Tiburon example 1: syntax MT cascade

Add in a simple PCFG-based language model



Tiburon example 1: syntax MT cascade

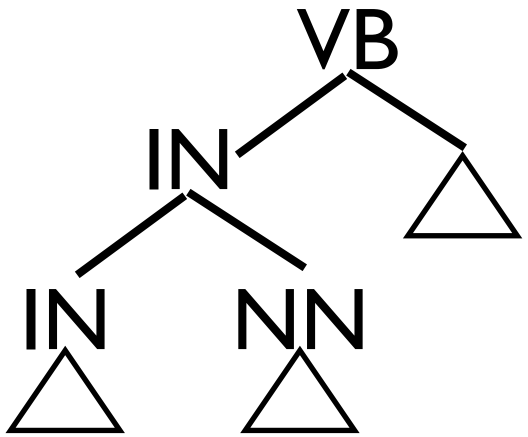
Add in a simple PCFG-based language model



```
% tiburon -k 5 -m tropical -e euc-jp pcfg-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))) # 33.024
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))) # 33.718
TOP(VB(PRP("him") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))) # 33.718
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("him"))))) # 33.718
TOP(VB(PRP("them") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))) # 33.718
```

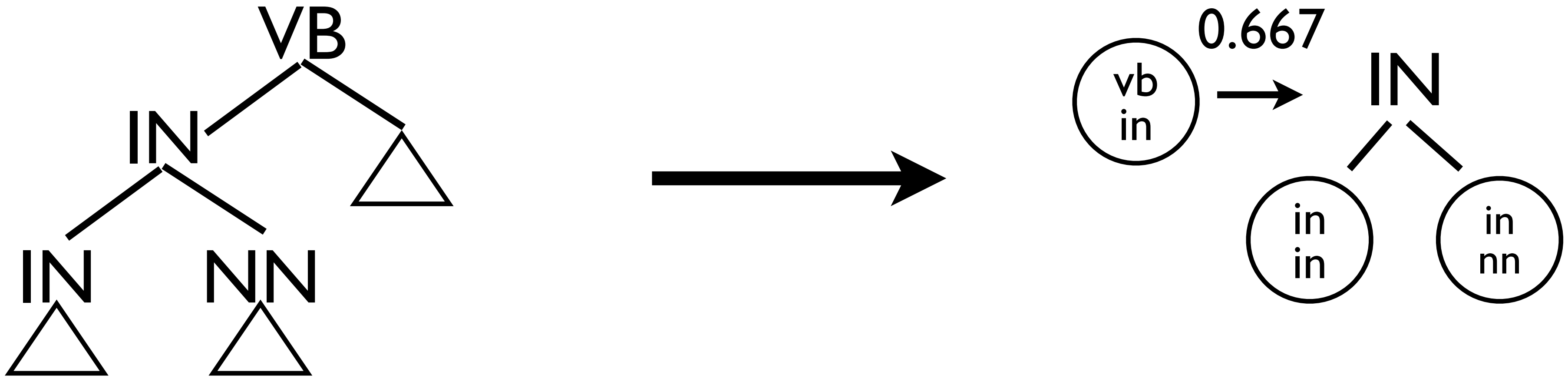
Tiburon example 1: syntax MT cascade

Try a grandparent language model



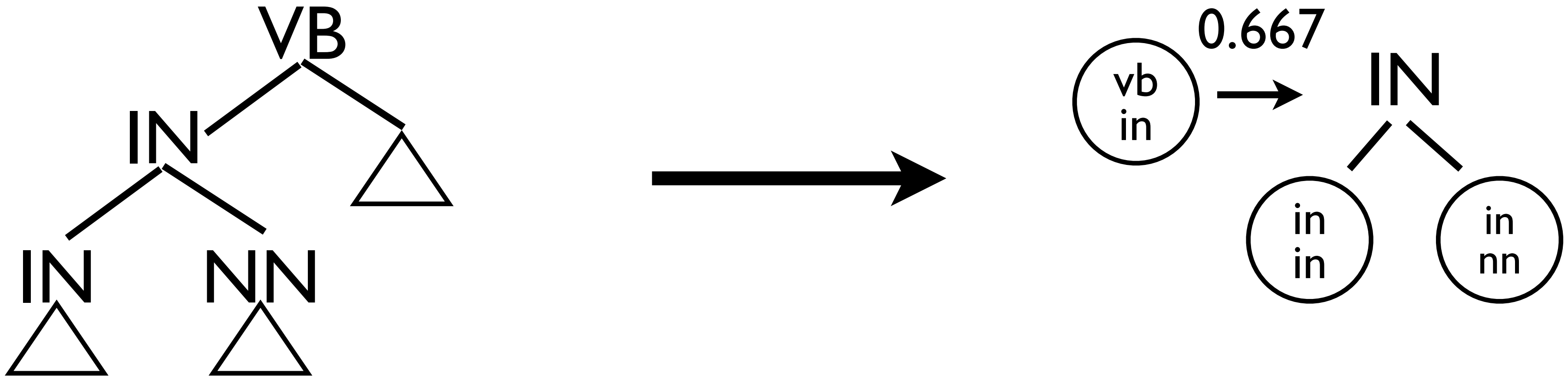
Tiburon example 1: syntax MT cascade

Try a grandparent language model



Tiburon example 1: syntax MT cascade

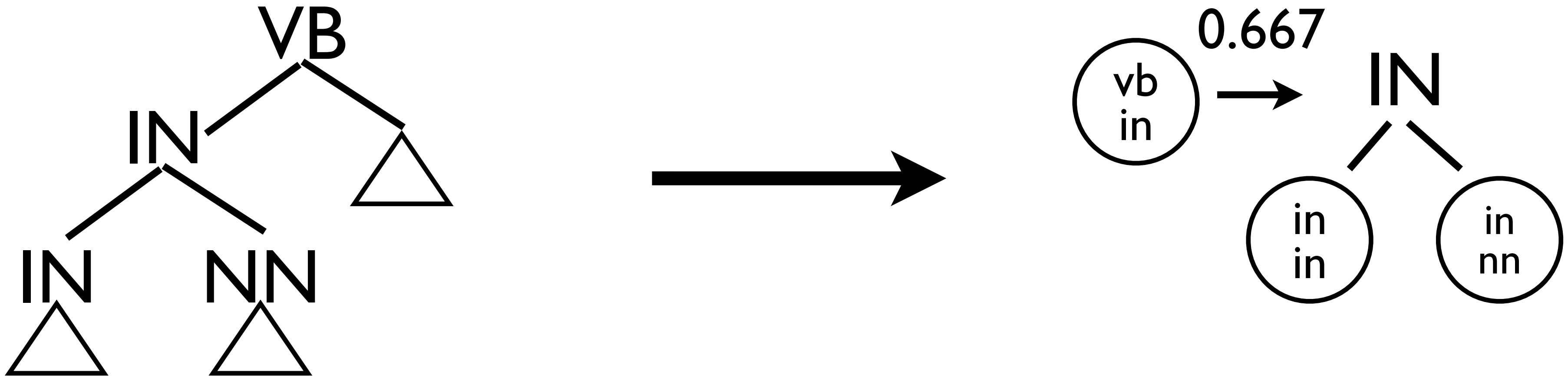
Try a grandparent language model



```
% tiburon -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.603
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.297
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.033
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.071
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.726
```

Tiburon example 1: syntax MT cascade

Try a grandparent language model

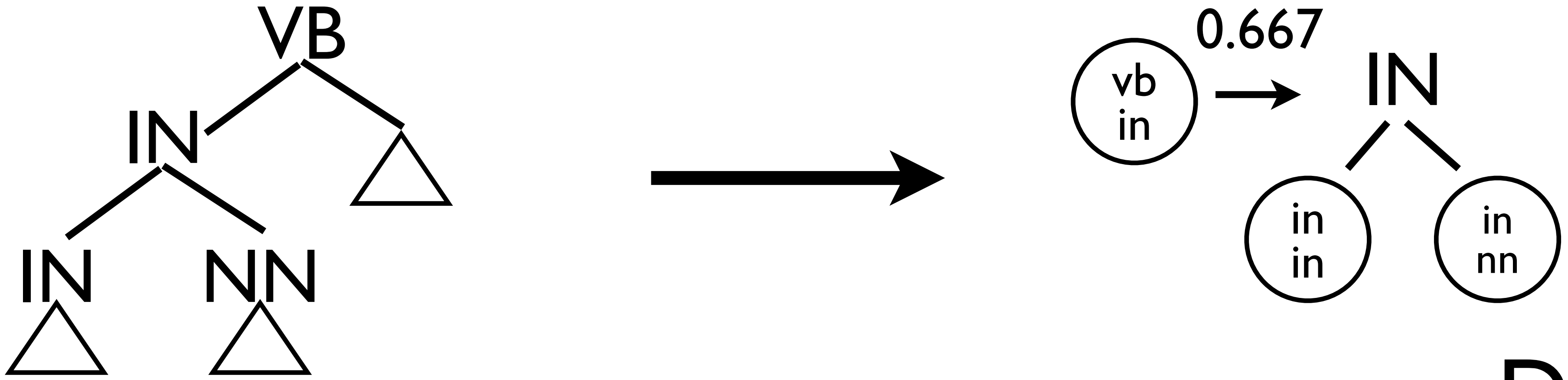


```
% tiburon -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.603
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.297
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.033
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.071
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.726
```

Correct sentence is 5th

Tiburon example 1: syntax MT cascade

Try a grandparent language model



Duplicates

```
% tiburon -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.603
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.297
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.033
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.071
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.726
```

Correct sentence is 5th

Tiburon example 1: syntax MT cascade

- Combine duplicate derivations in entire search space using *weighted determinization*

Tiburon example 1: syntax MT cascade

- Combine duplicate derivations in entire search space using *weighted determinization*

```
% tiburon -d 5 -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.329
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.023
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.759
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.452
TOP(VB(NN(DT("a") NN("clouds")) VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP
("them"))))) # 31.250
```

Tiburon example 1: syntax MT cascade

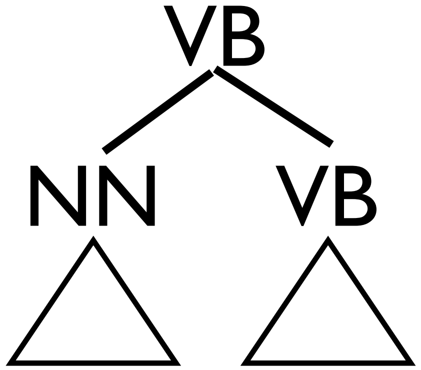
- Combine duplicate derivations in entire search space using *weighted determinization*

Now we’re 4th

```
% tiburon -d 5 -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.329
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.023
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.759
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.452
TOP(VB(NN(DT("a") NN("clouds")) VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP
("them"))))) # 31.250
```

Tiburon example 2: training a syntax LM

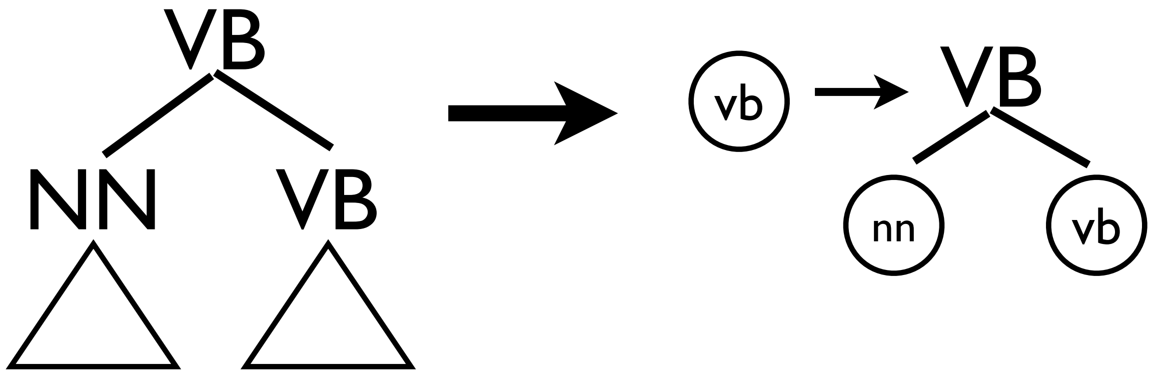
- The LMs we used before had no hidden states
- Let's introduce hidden states and learn weights with EM



(Petrov & Klein, '07)

Tiburon example 2: training a syntax LM

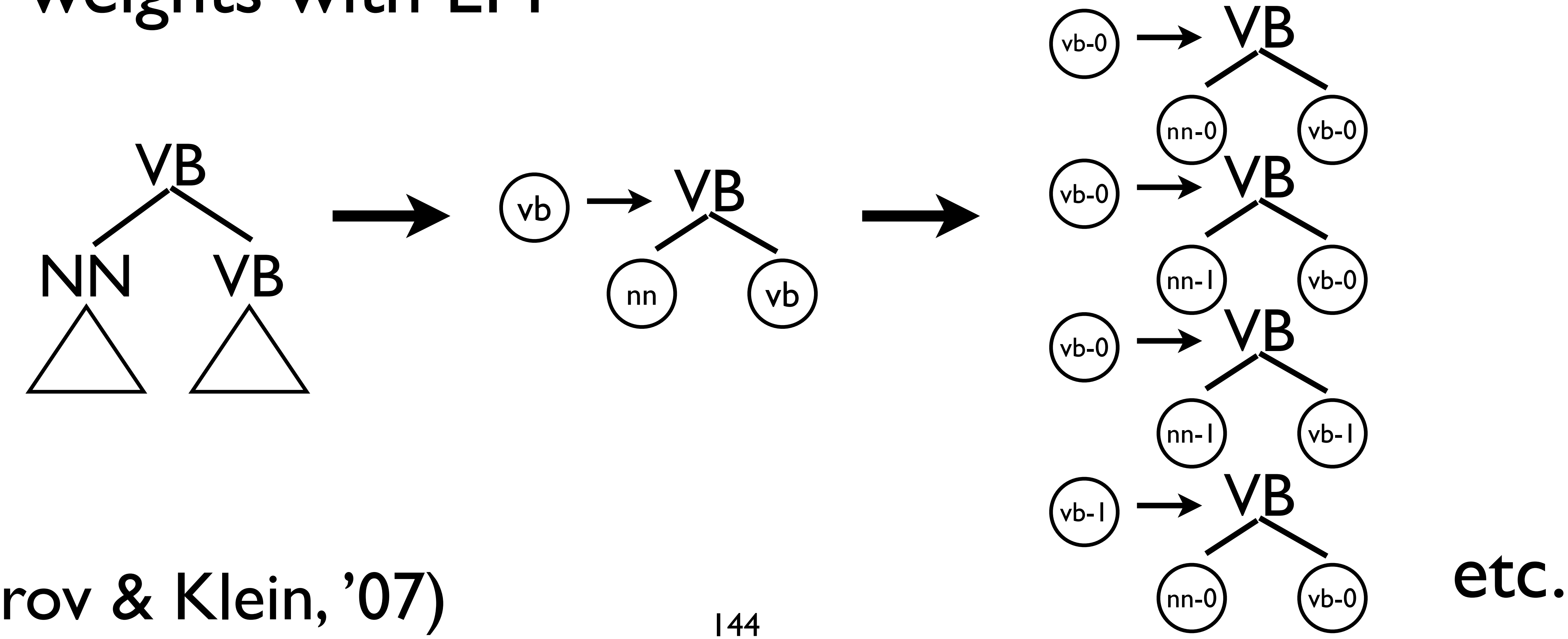
- The LMs we used before had no hidden states
- Let's introduce hidden states and learn weights with EM



(Petrov & Klein, '07)

Tiburon example 2: training a syntax LM

- The LMs we used before had no hidden states
- Let’s introduce hidden states and learn weights with EM



(Petrov & Klein, ’07)

Tiburon example 2: training a syntax LM

```
% tiburon  -t 50  --randomize  trees  rtg.4split > 4split-lm
```

Tiburon example 2: training a syntax LM

```
% tiburon  -t 50  --randomize  trees  rtg.4split > 4split-lm
```

50 iterations

Tiburon example 2: training a syntax LM

```
% tiburon  -t 50  --randomize  trees  rtg.4split > 4split-lm
```

random initial weights avoids saddles

Tiburon example 2: training a syntax LM

```
% tiburon  -t 50  --randomize   trees   rtg.4split  > 4split-lm
```

training data

Tiburon example 2: training a syntax LM

```
% tiburon  -t 50  --randomize  trees  rtg.4split > 4split-lm
```

4-way split

Tiburon example 2: training a syntax LM

```
% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Cross entropy with normalized initial weights is 1.868; corpus prob is e^-269.025
Cross entropy after 1 iterations is 1.190; corpus prob is e^-171.383
Cross entropy after 2 iterations is 1.138; corpus prob is e^-163.866
Cross entropy after 3 iterations is 1.036; corpus prob is e^-149.229
...
Cross entropy after 47 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 48 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 49 iterations is 0.581; corpus prob is e^-83.665
```

Tiburon example 2: training a syntax LM

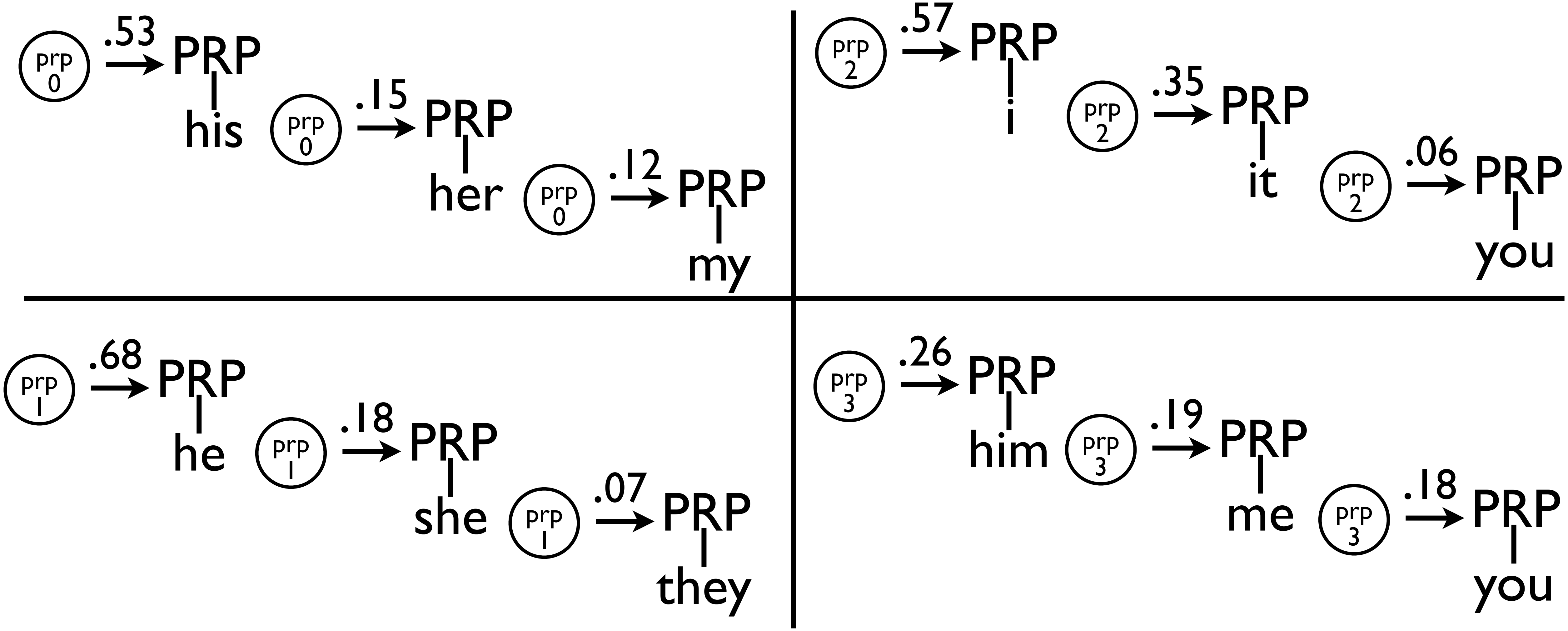
```
% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Cross entropy with normalized initial weights is 1.868; corpus prob is e^-269.025
Cross entropy after 1 iterations is 1.190; corpus prob is e^-171.383
Cross entropy after 2 iterations is 1.138; corpus prob is e^-163.866
Cross entropy after 3 iterations is 1.036; corpus prob is e^-149.229
...
Cross entropy after 47 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 48 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 49 iterations is 0.581; corpus prob is e^-83.665
```

Compare with GP-PCFG

```
% tiburon -t 3 --randomize trees rtg.gp.pcfg > lm
Cross entropy with normalized initial weights is 0.827; corpus prob is e^-119.022
Cross entropy after 1 iterations is 0.732; corpus prob is e^-105.448
Cross entropy after 2 iterations is 0.732; corpus prob is e^-105.448
```

Tiburon example 2: training a syntax LM

We can subjectively see state specialization



prp 0 .53 → PRP his

prp 0 .15 → PRP her

prp 0 .12 → PRP my

prp 2 .57 → PRP i

prp 2 .35 → PRP it

prp 2 .06 → PRP you

prp 1 .68 → PRP he

prp 1 .18 → PRP she

prp 1 .07 → PRP they

prp 3 .26 → PRP him

prp 3 .19 → PRP me

prp 3 .18 → PRP you

Tiburon example 2: training a syntax LM

Tied for first!

```
% tiburon -k 5 -m tropical -e euc-jp 4split-lm rot ins trans ej.1.f
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("fanatic") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("clouds") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("fanatic") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.717
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.717
```

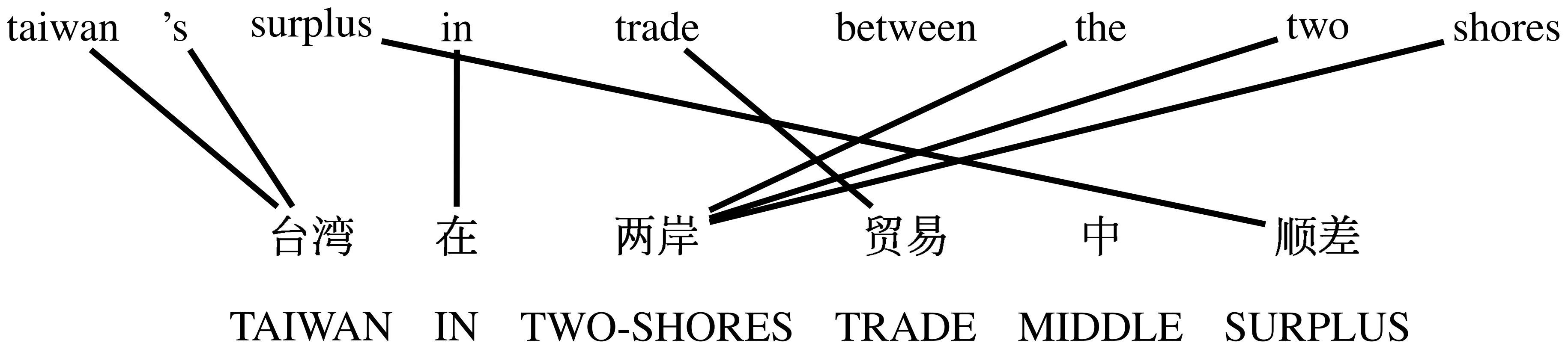
Using tree transducers to improve machine translation

(May & Knight, EMNLP '07)

- We will now shift focus to improving state-of-the-art syntax MT results
- At core, we're using the power of training tree transducers to achieve gains

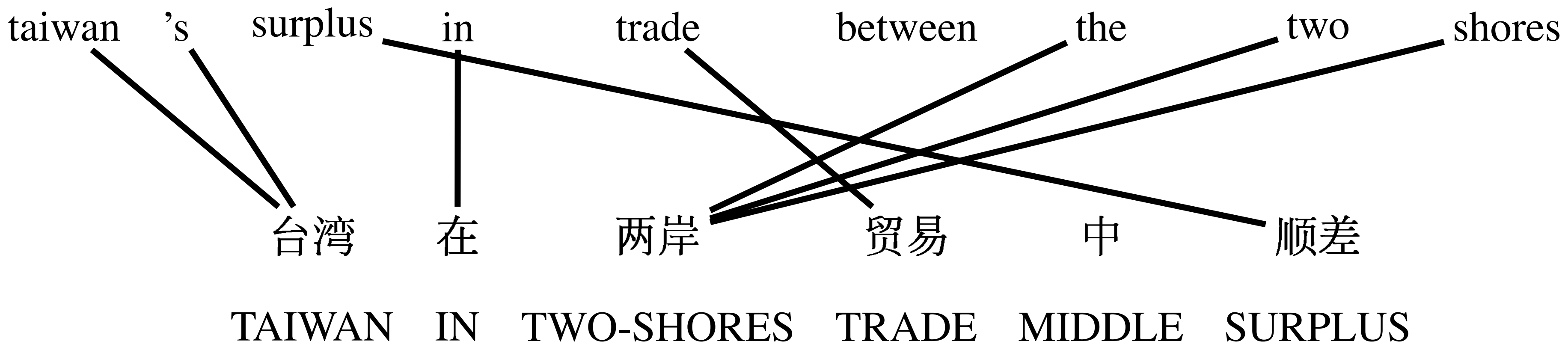
Extracting syntactic rules

1) Obtain alignments



Extracting syntactic rules

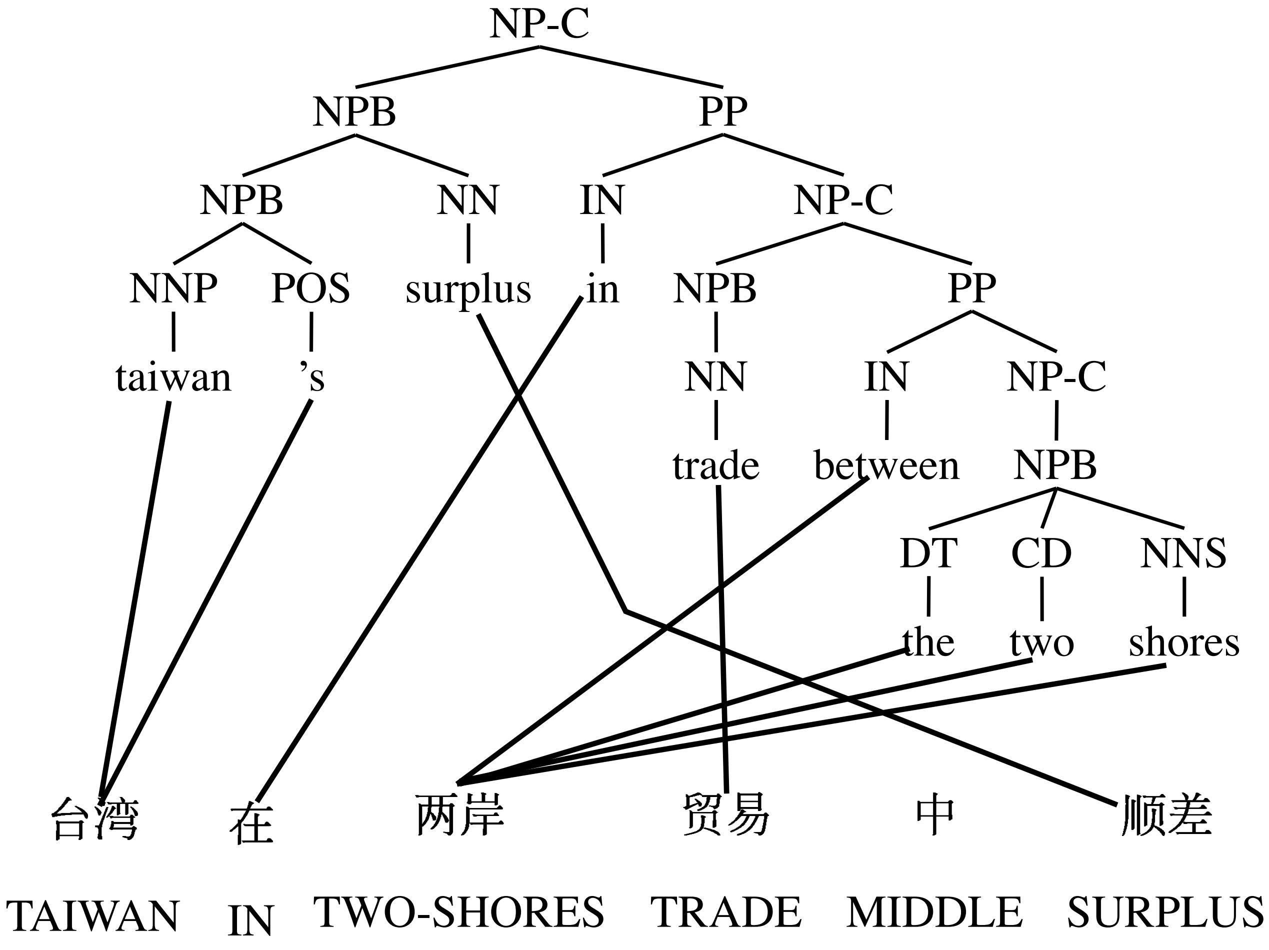
1) Obtain alignments



(Galley et al. ’04, ’06)

Extracting syntactic rules

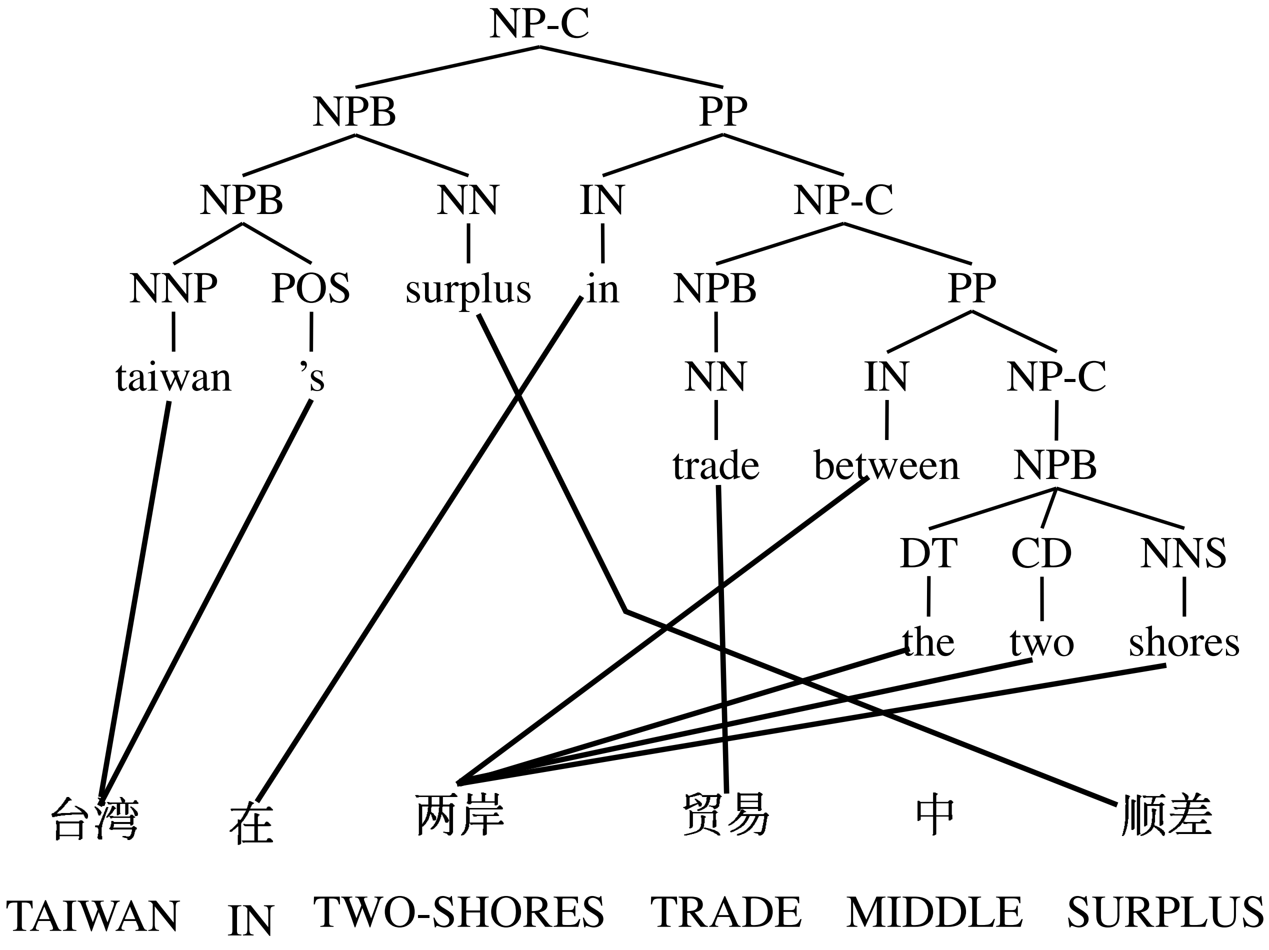
2) Add parse tree



(Galley et al. '04, '06)

Extracting syntactic rules

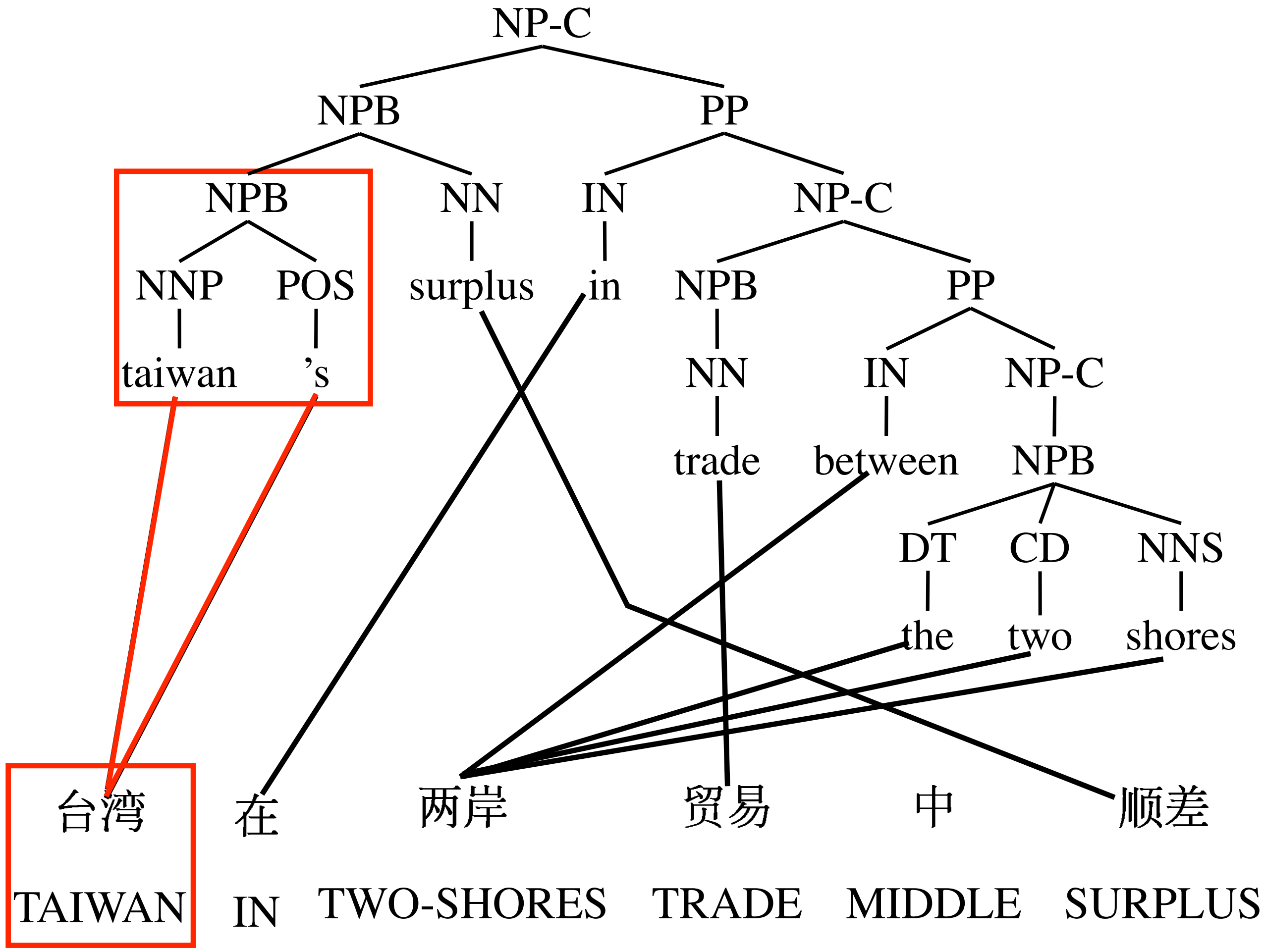
3) Extract rules



(Galley et al. '04, '06)

Extracting syntactic rules

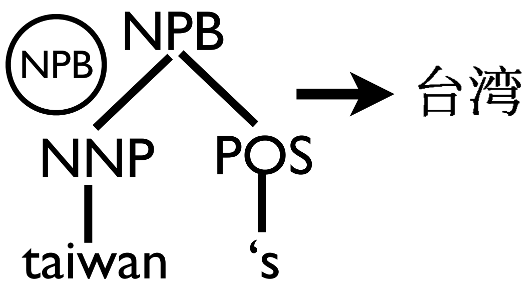
3) Extract rules

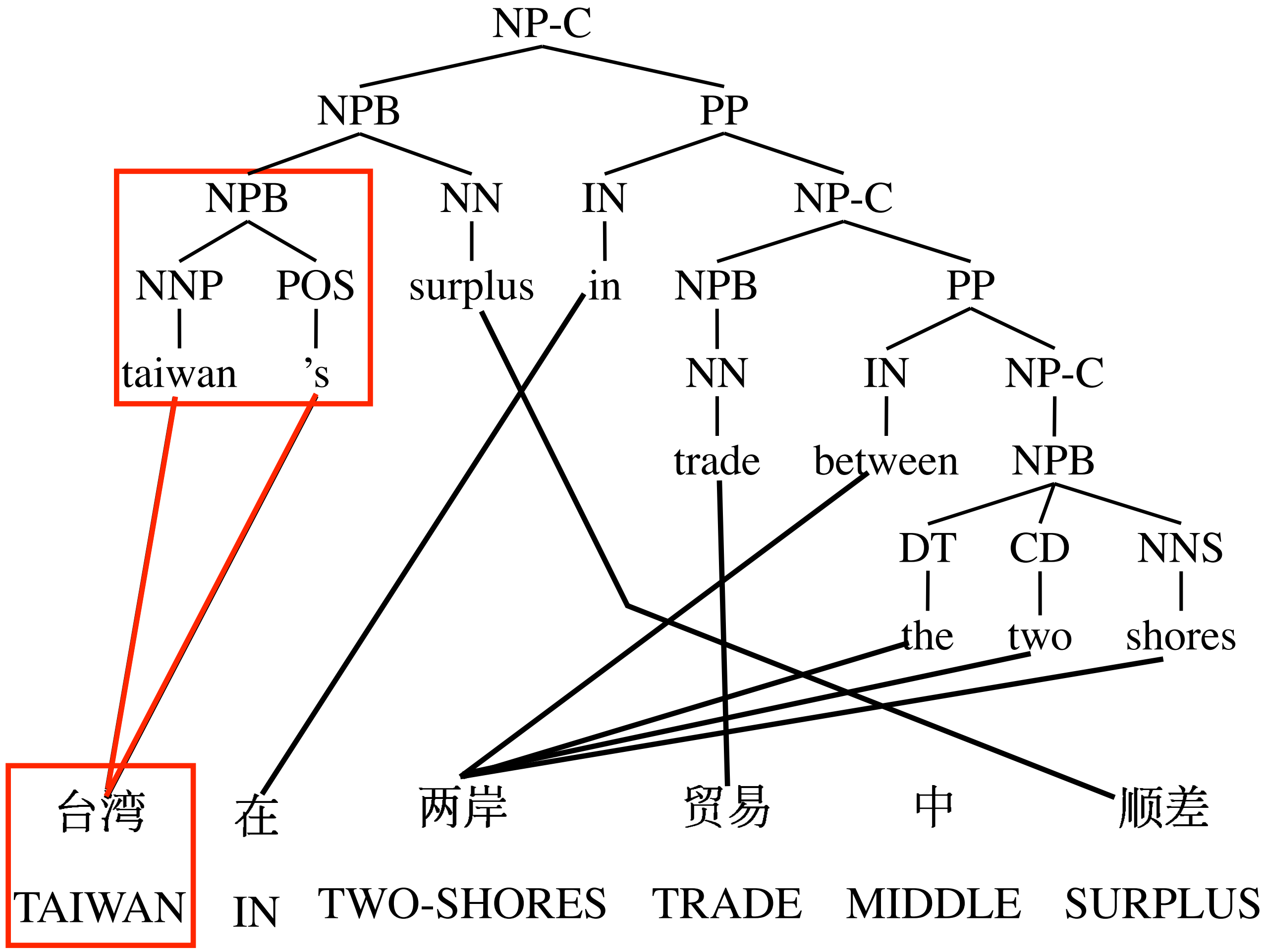


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

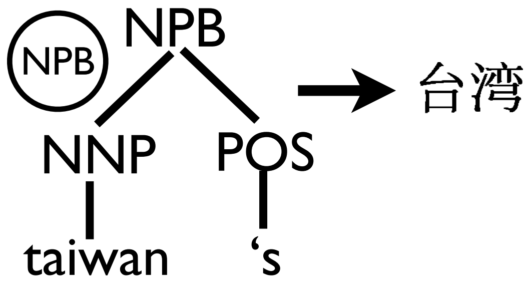


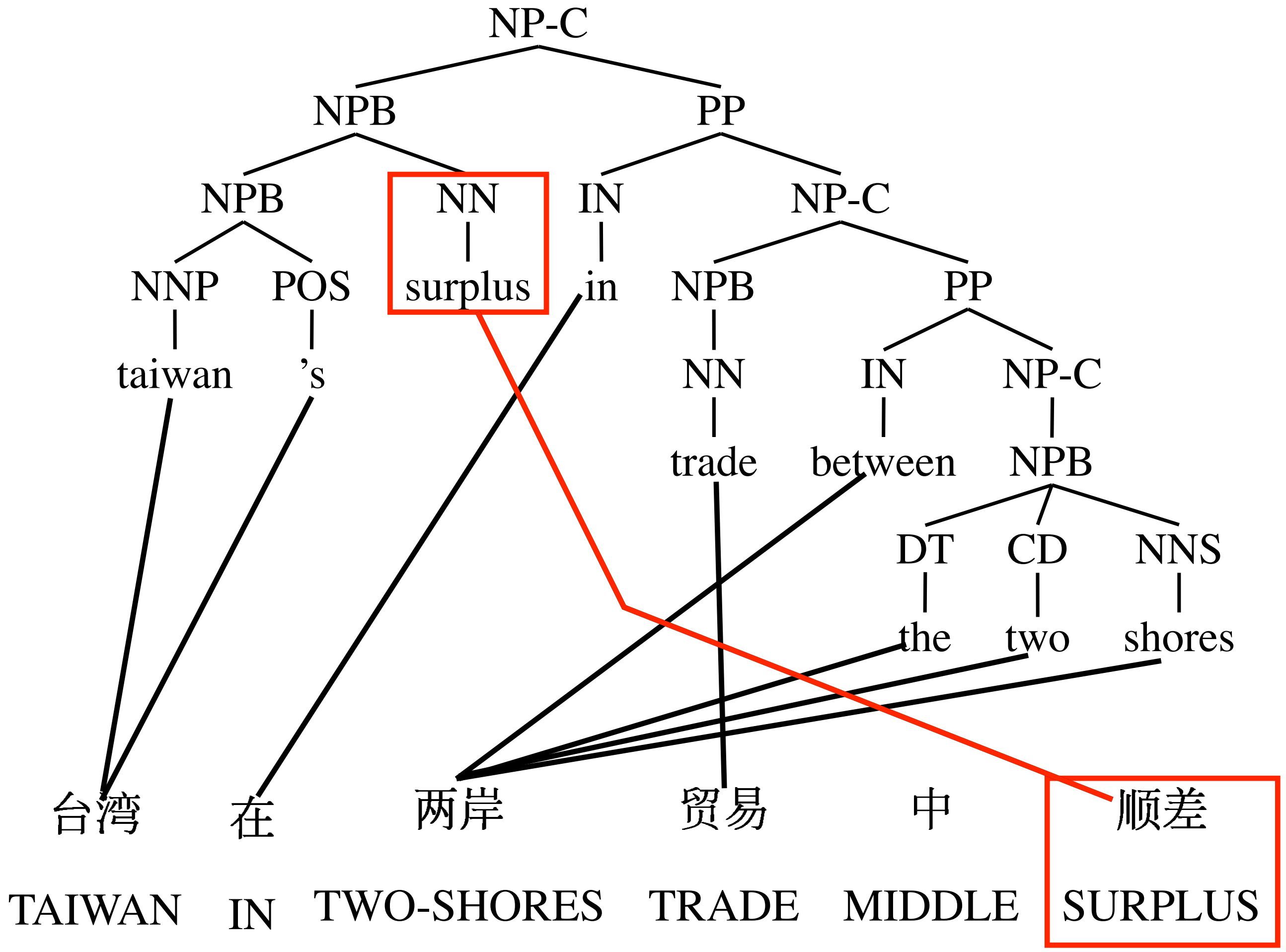


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

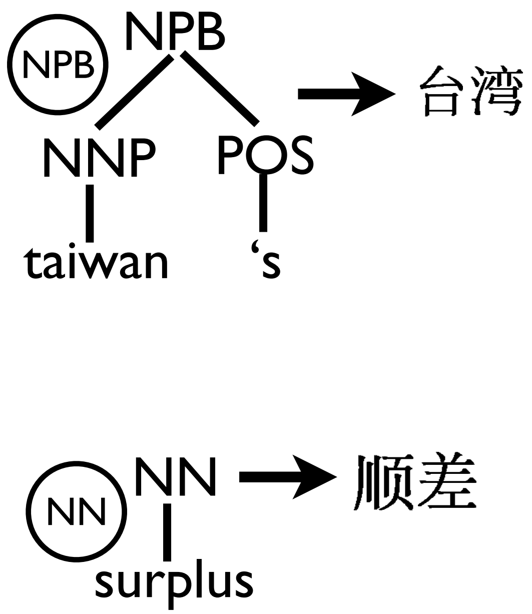


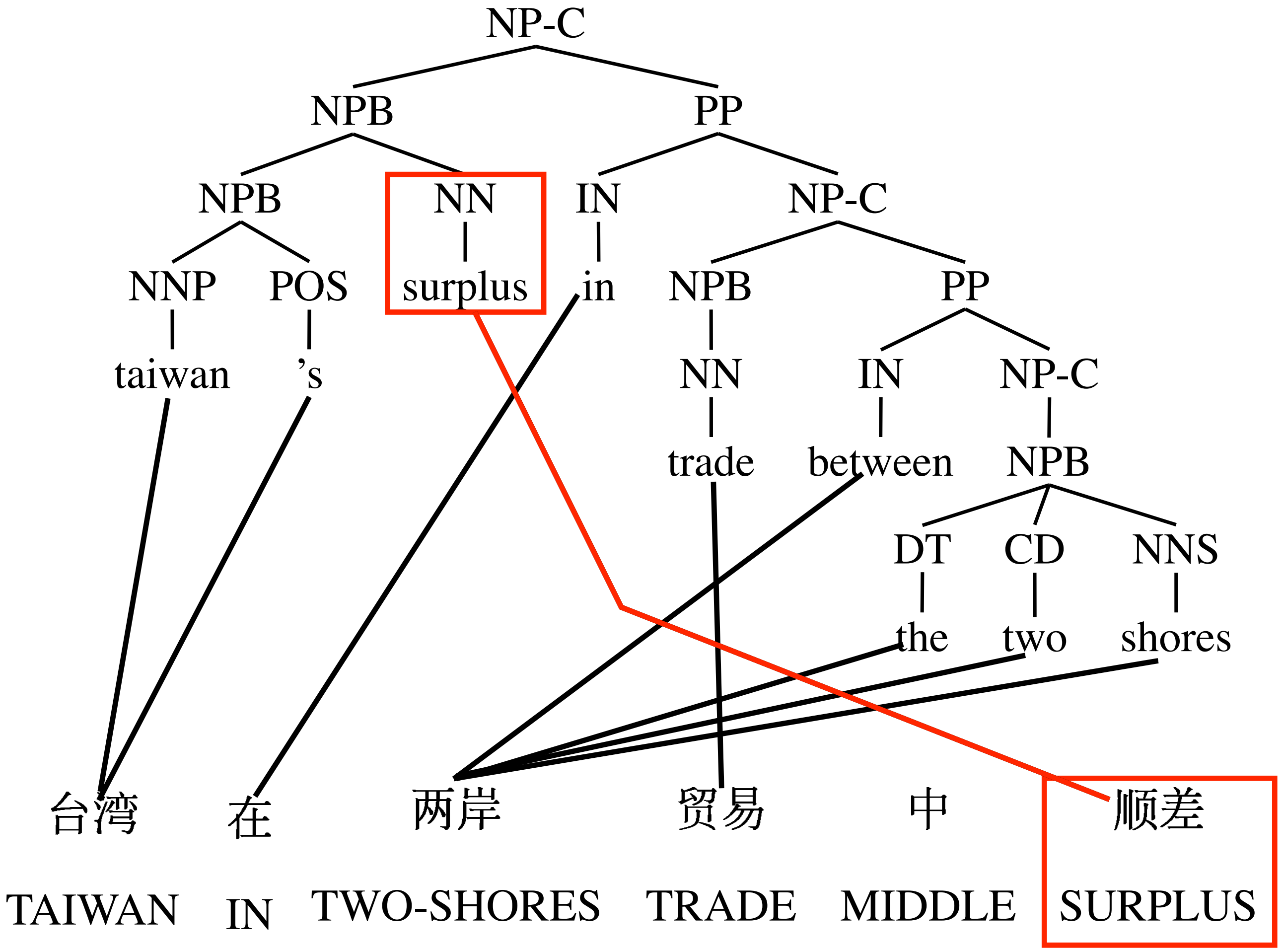


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

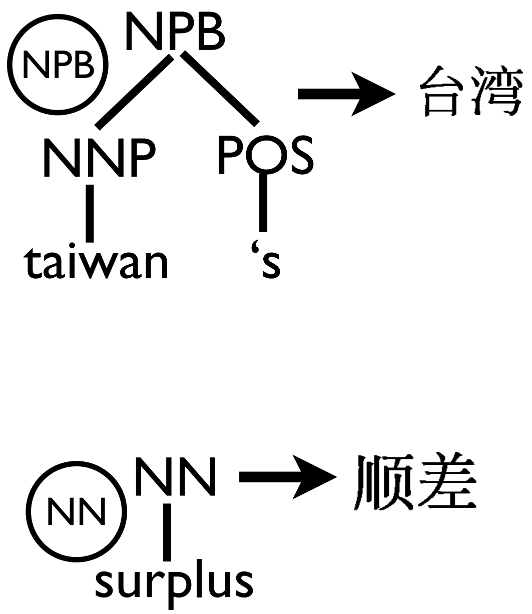


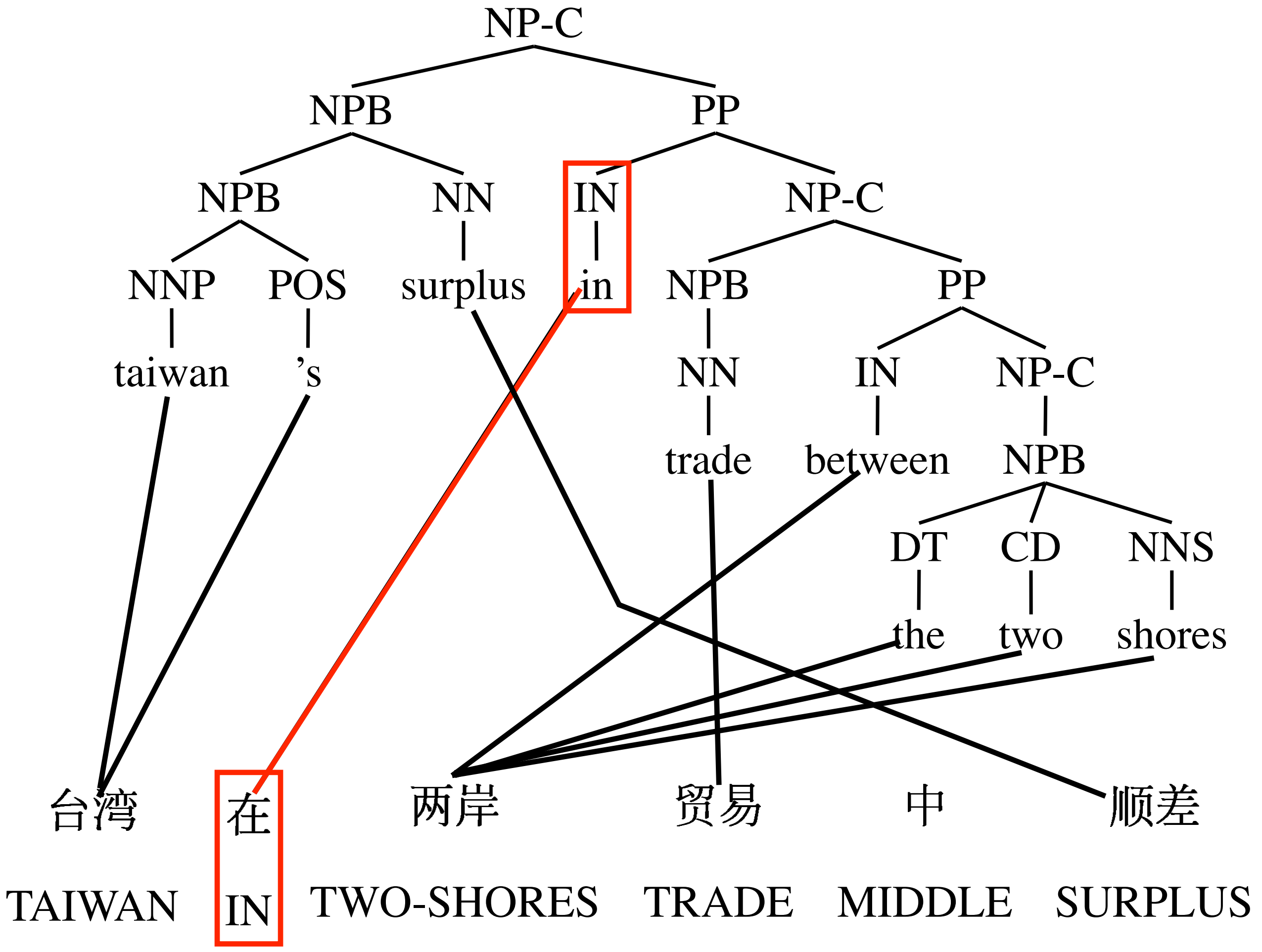


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

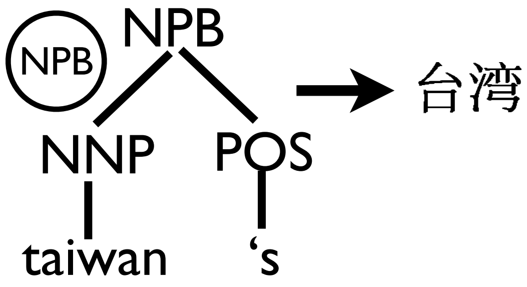


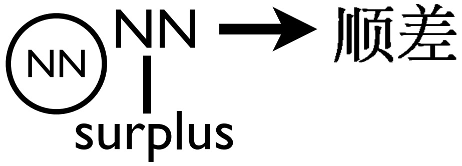


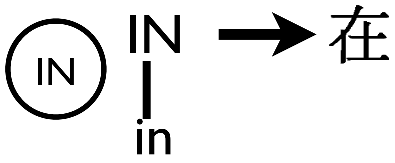
(Galley et al. '04, '06)

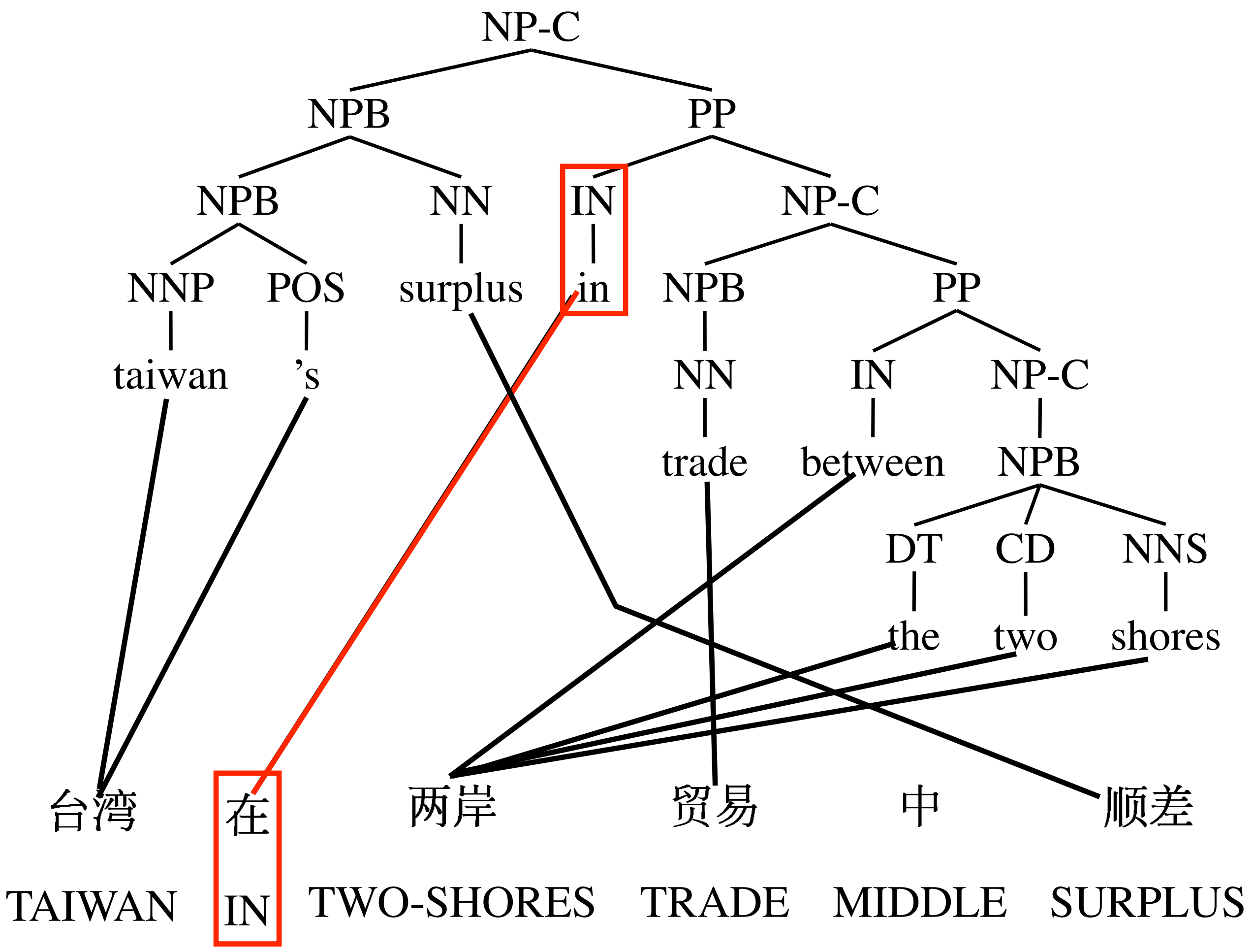
Extracting syntactic rules

3) Extract rules





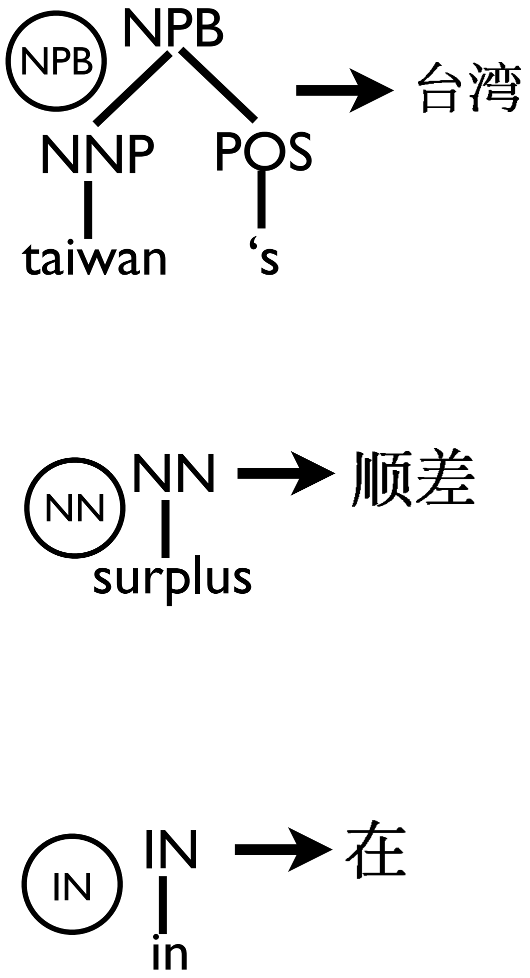


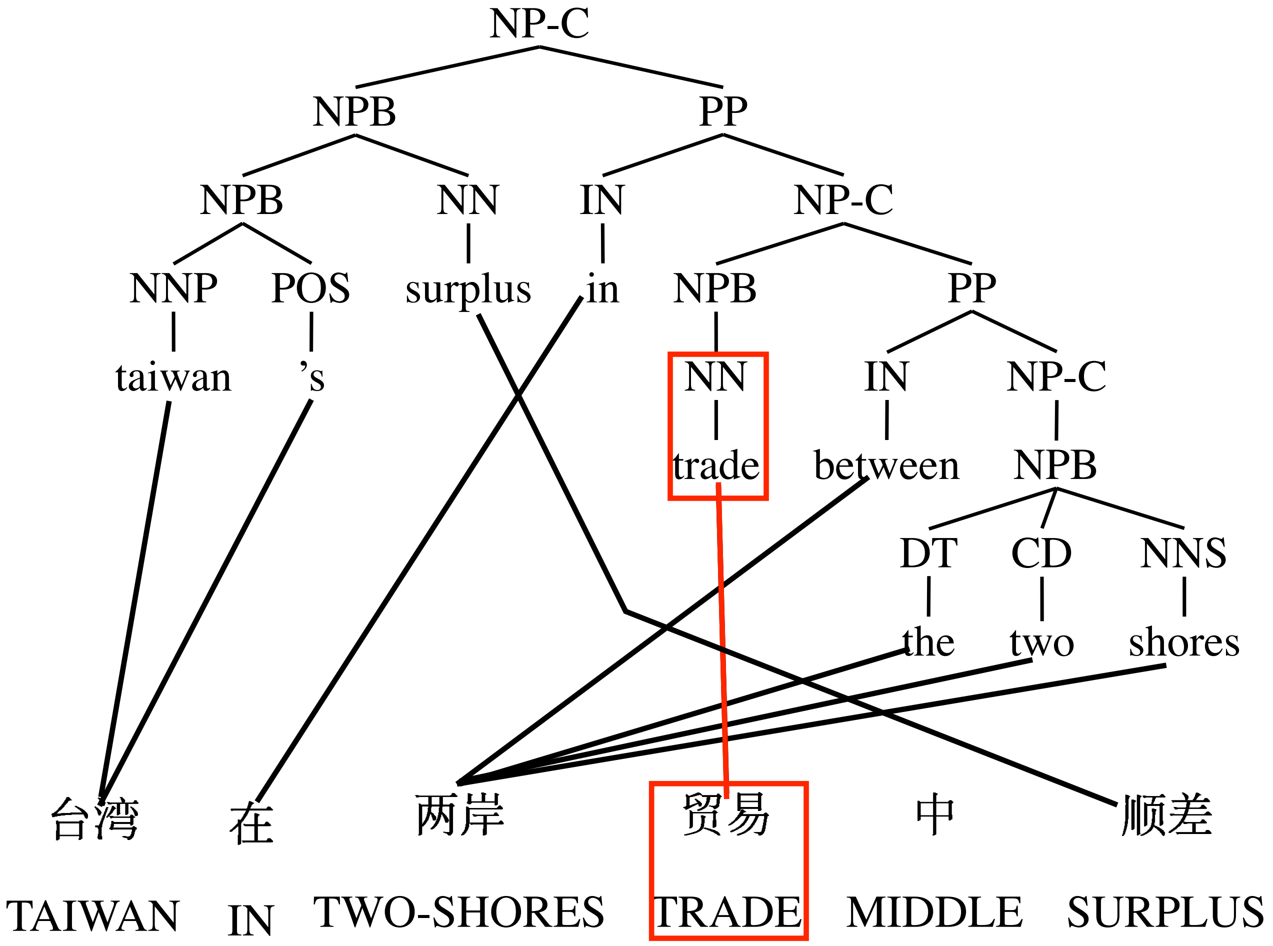


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

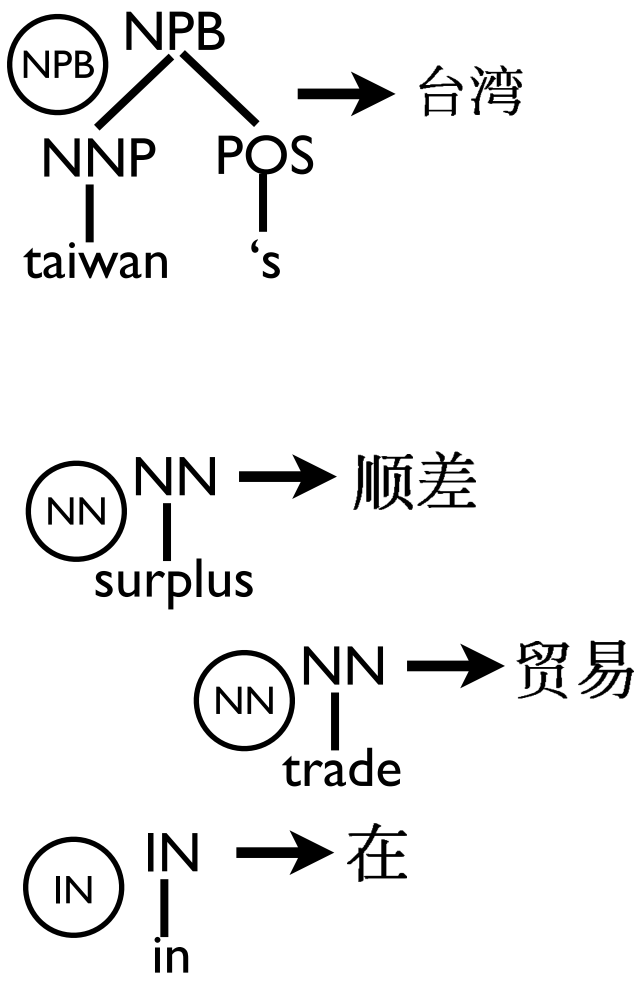


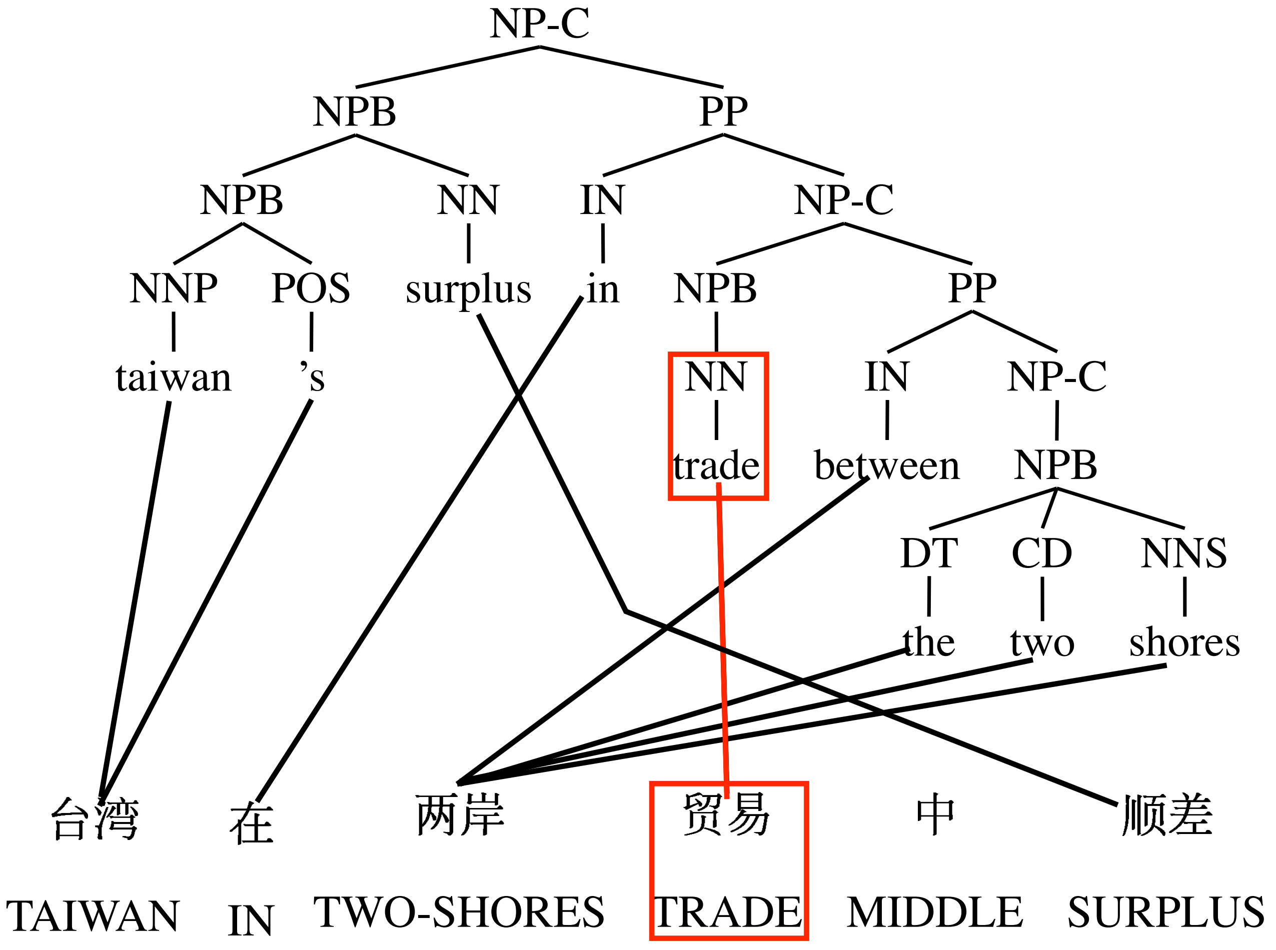


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

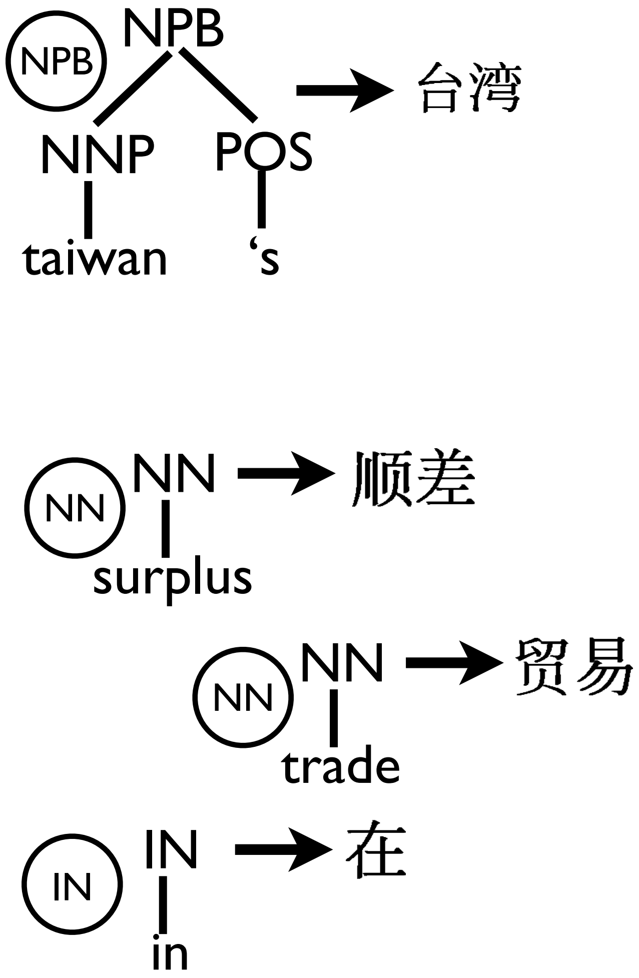


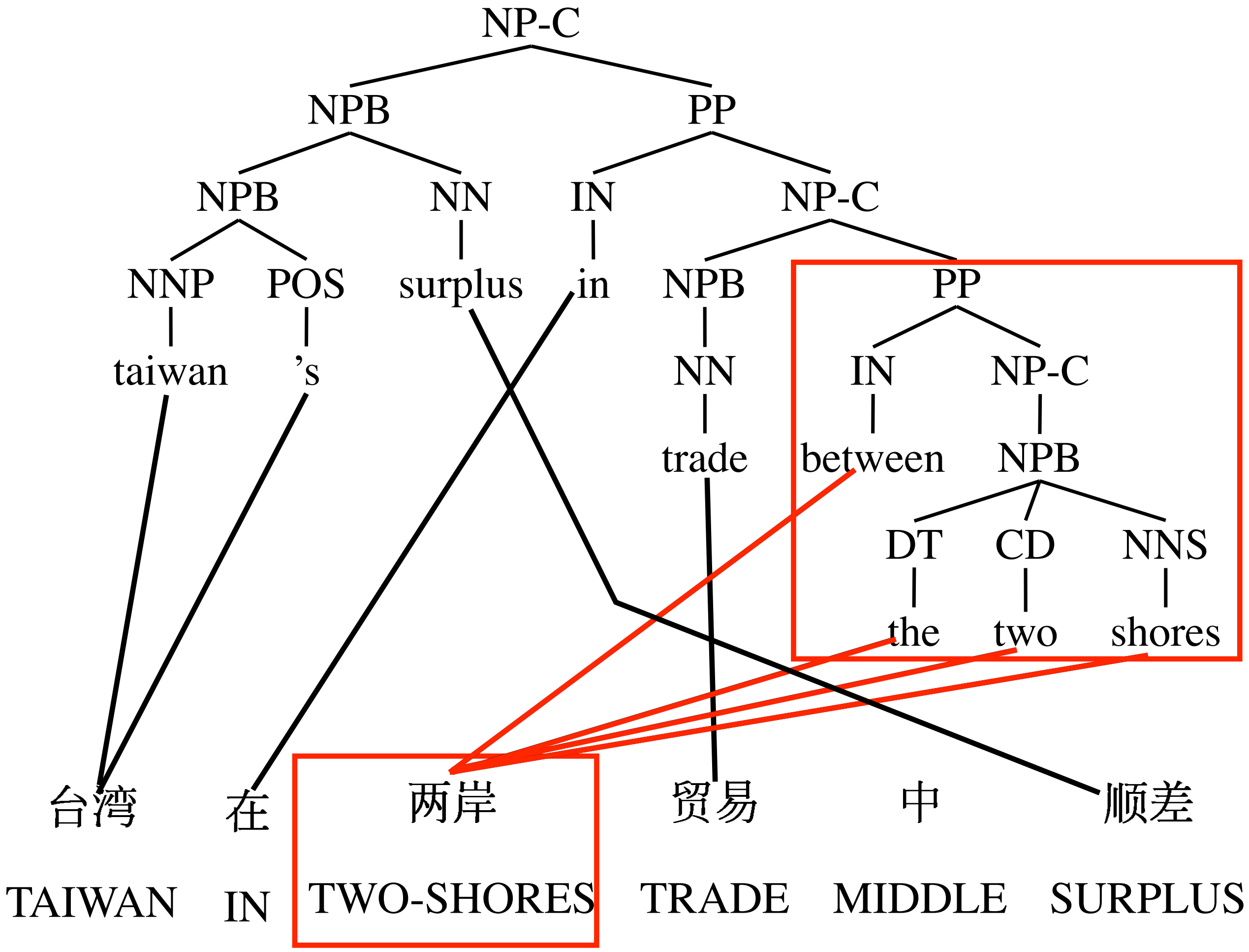


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

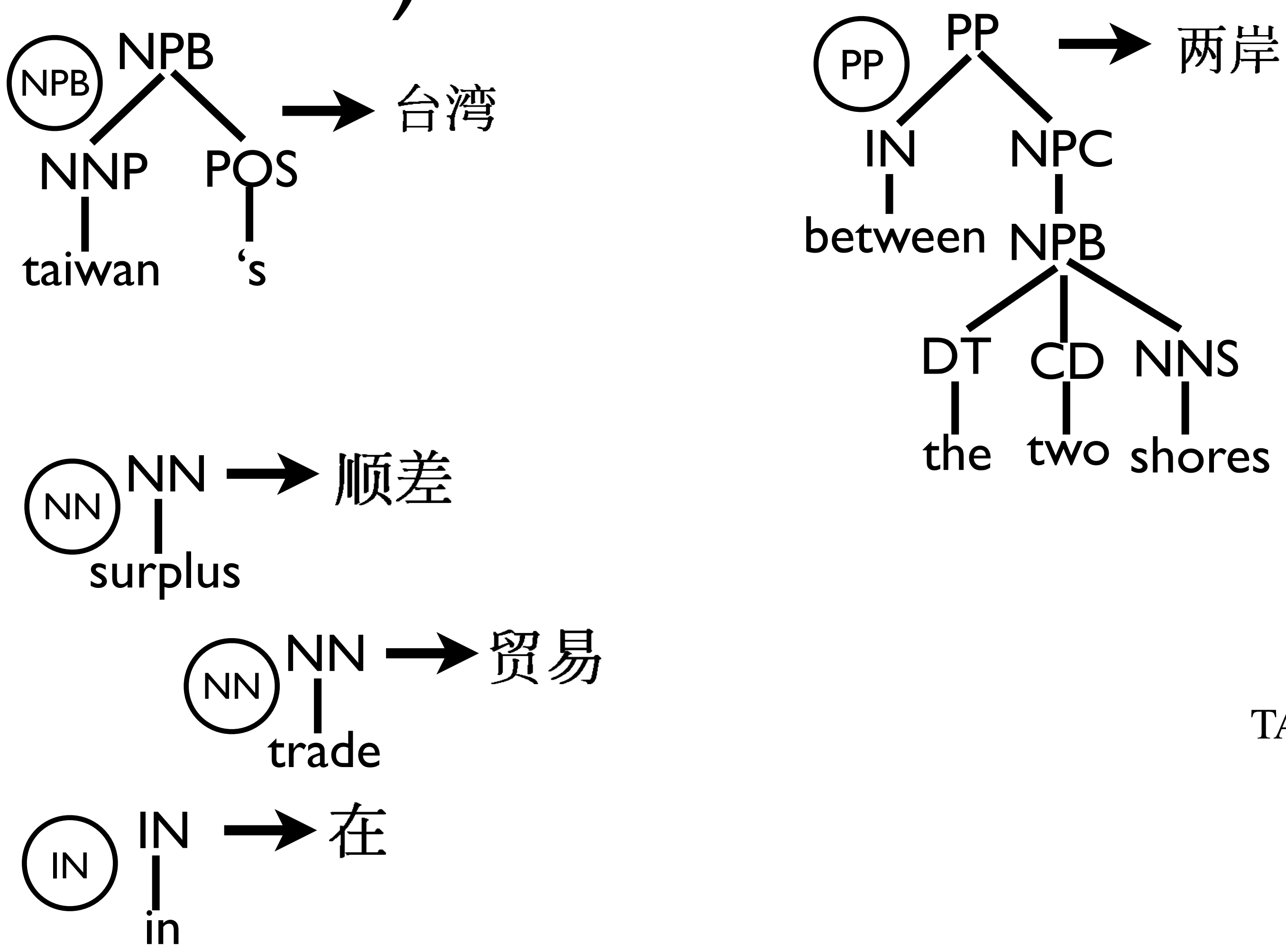


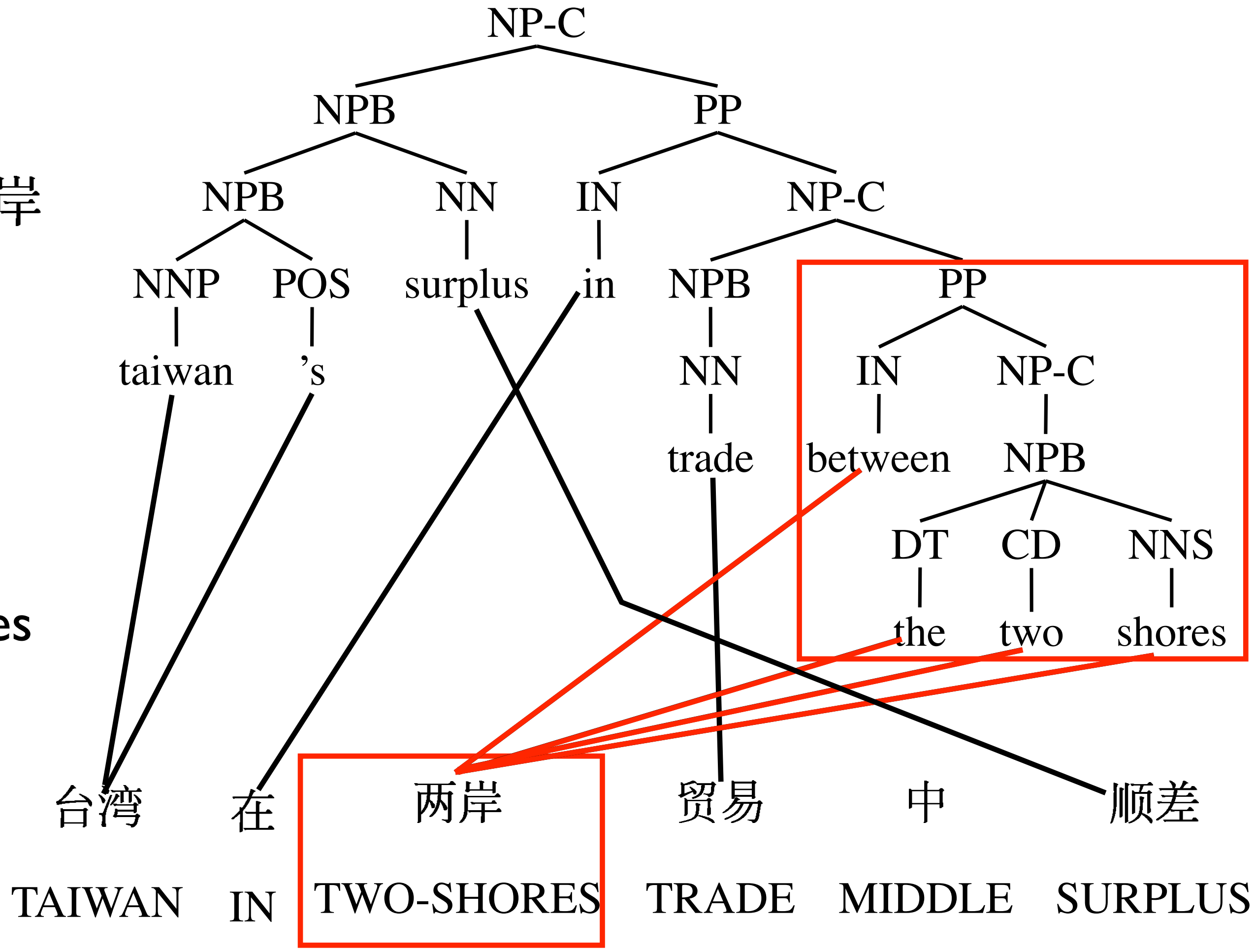


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

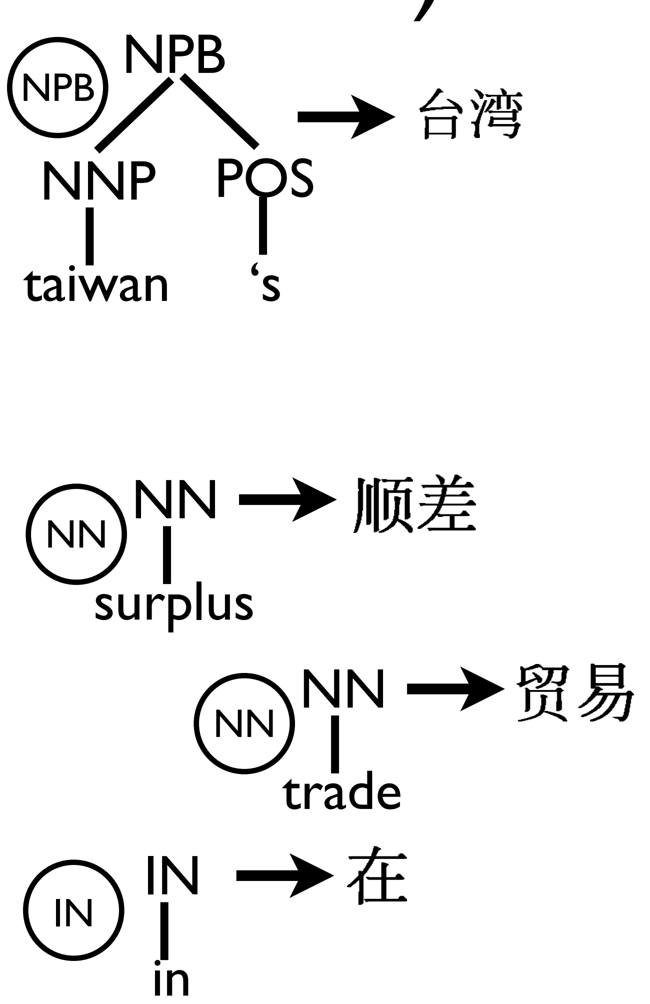


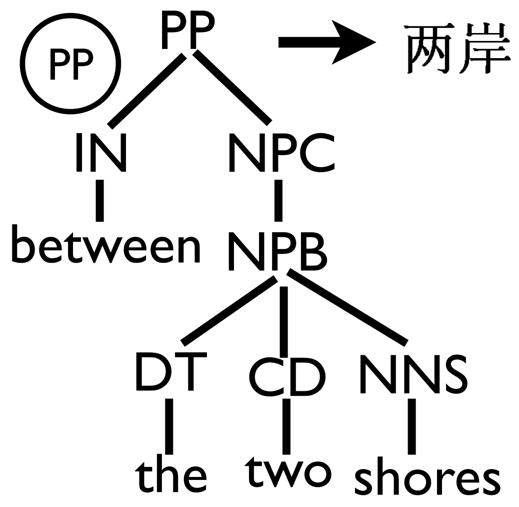


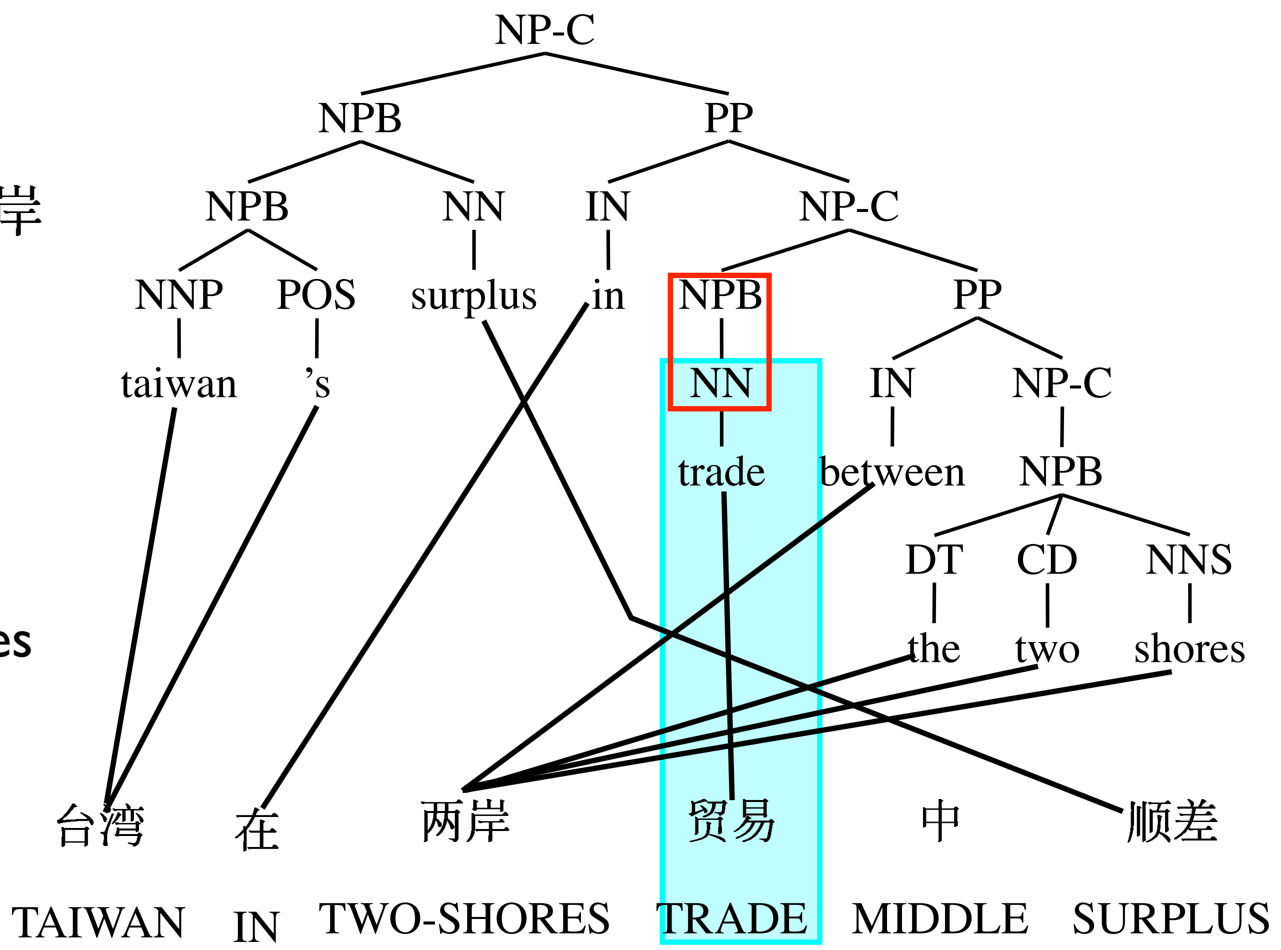
(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules



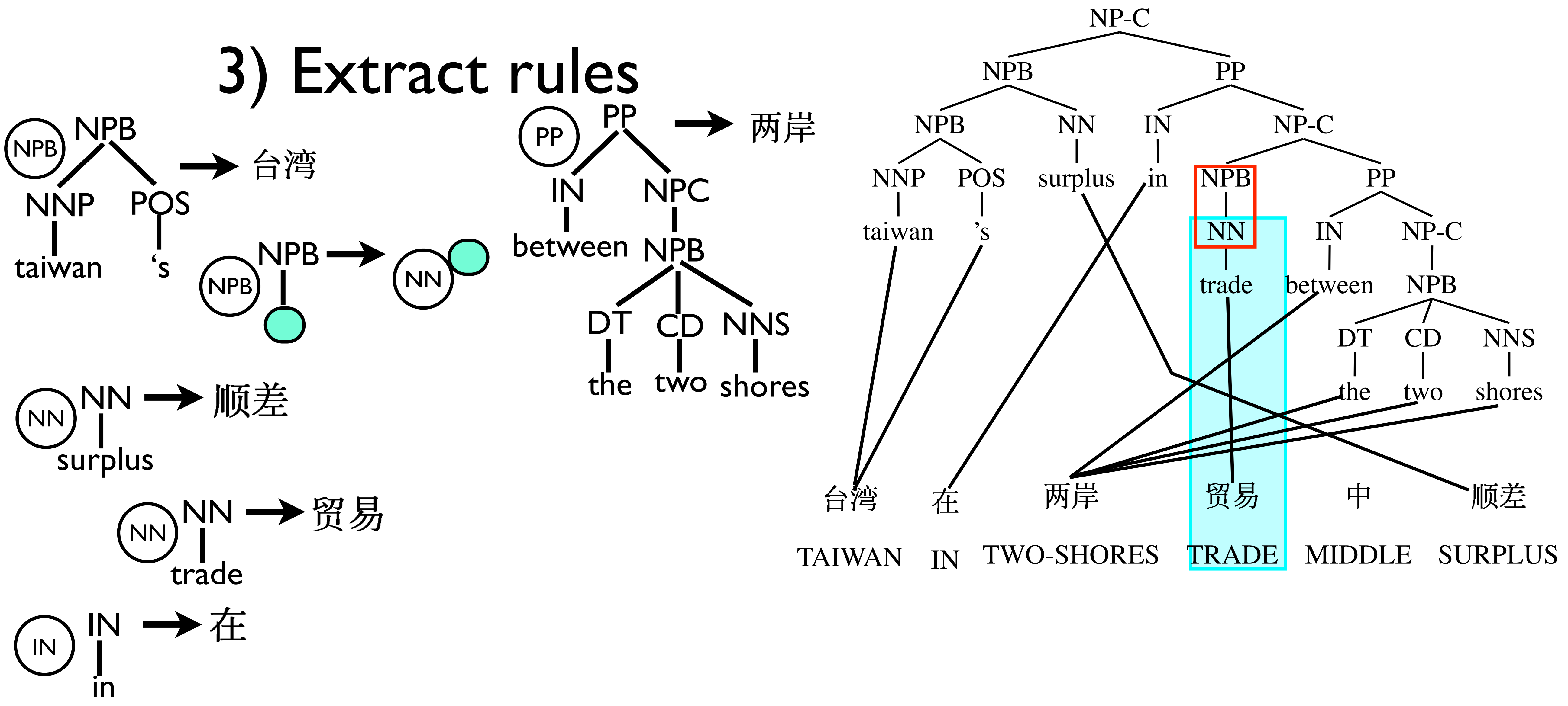




(Galley et al. '04, '06)

Extracting syntactic rules

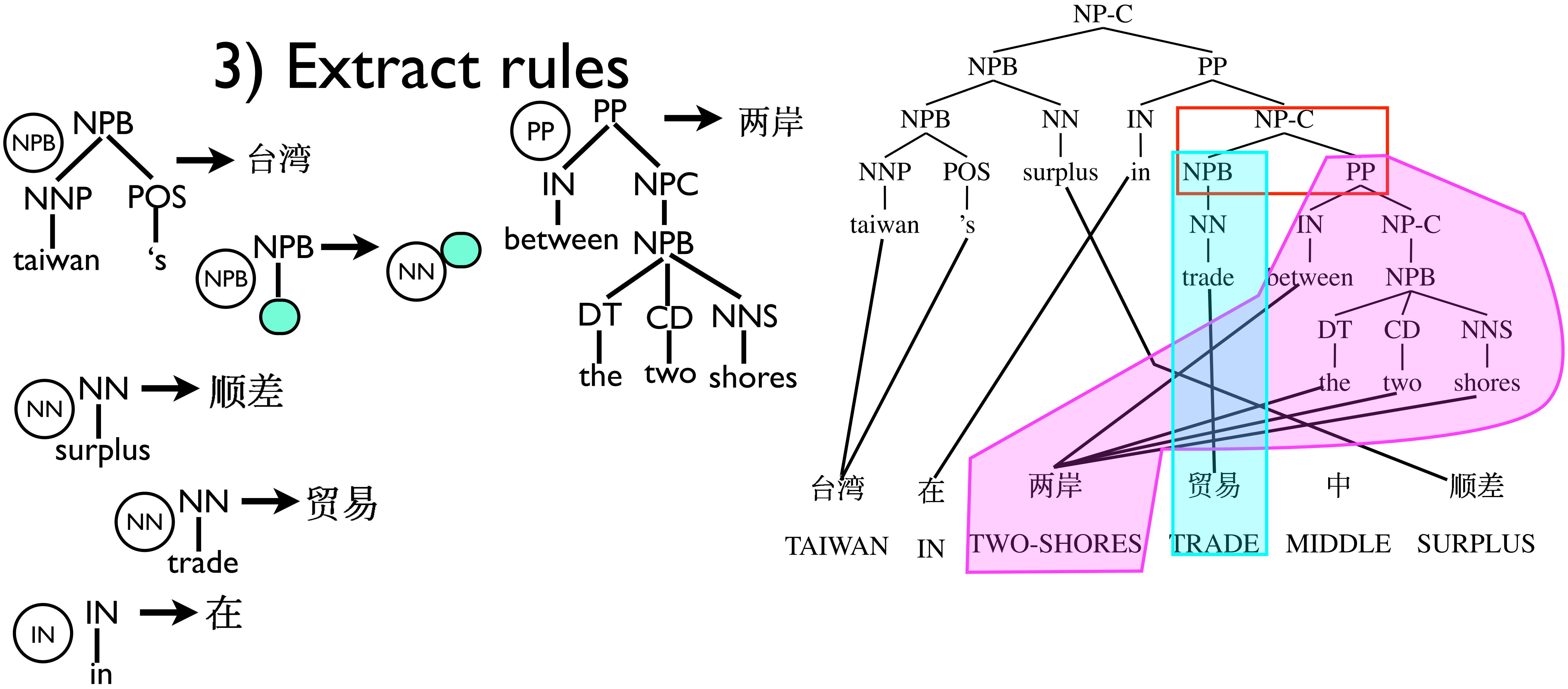
3) Extract rules



(Galley et al. '04, '06)

Extracting syntactic rules

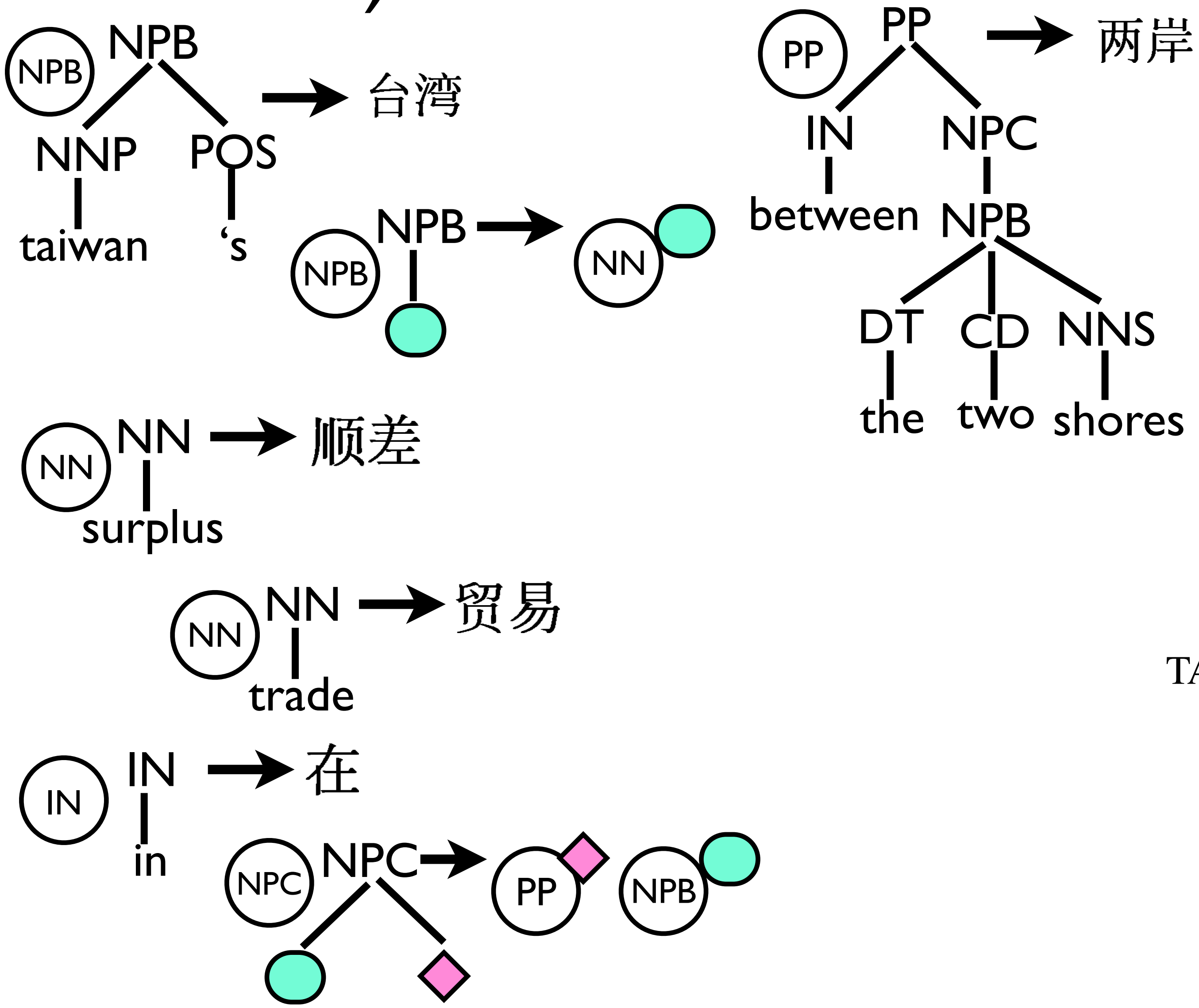
3) Extract rules

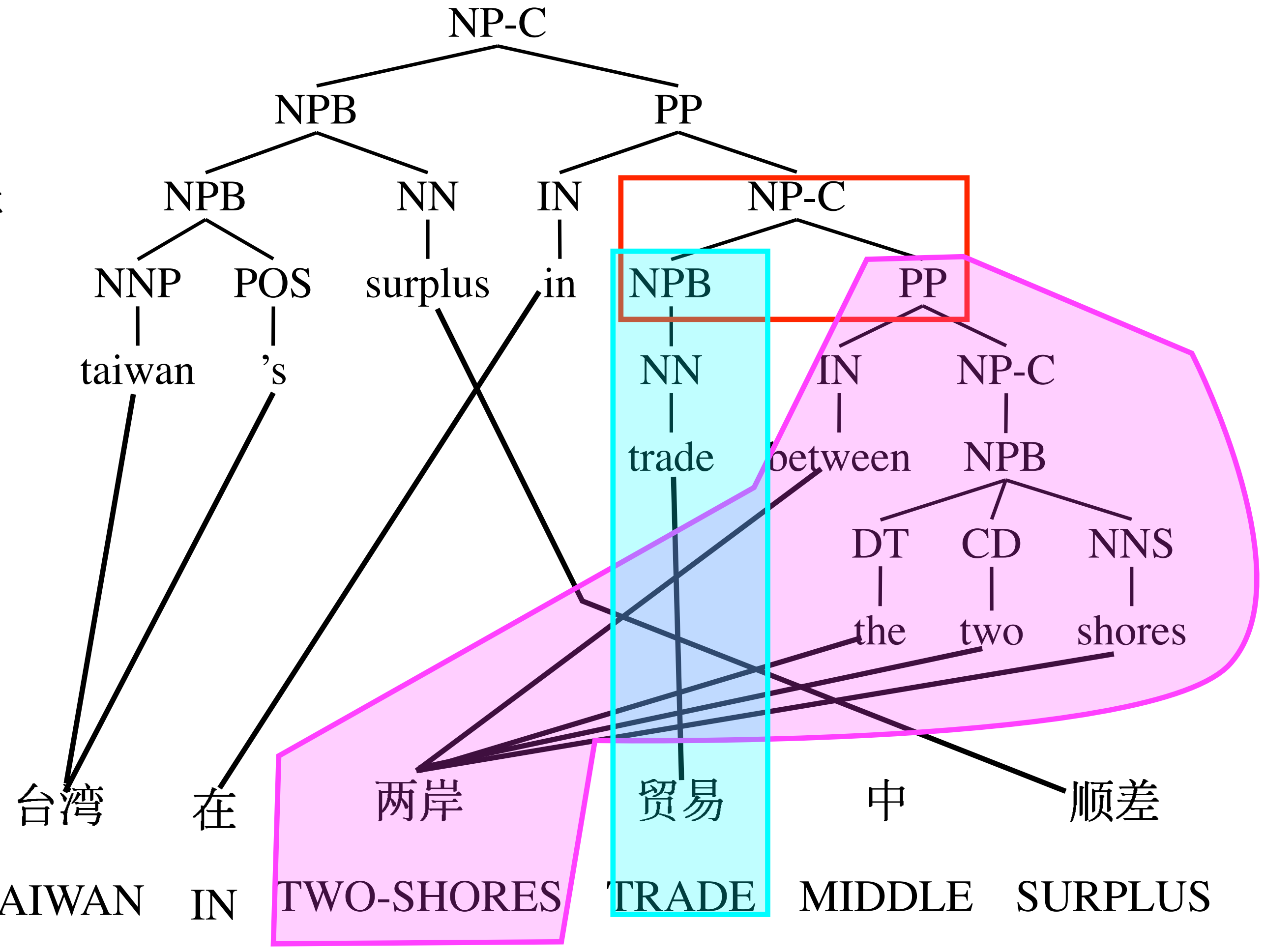


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

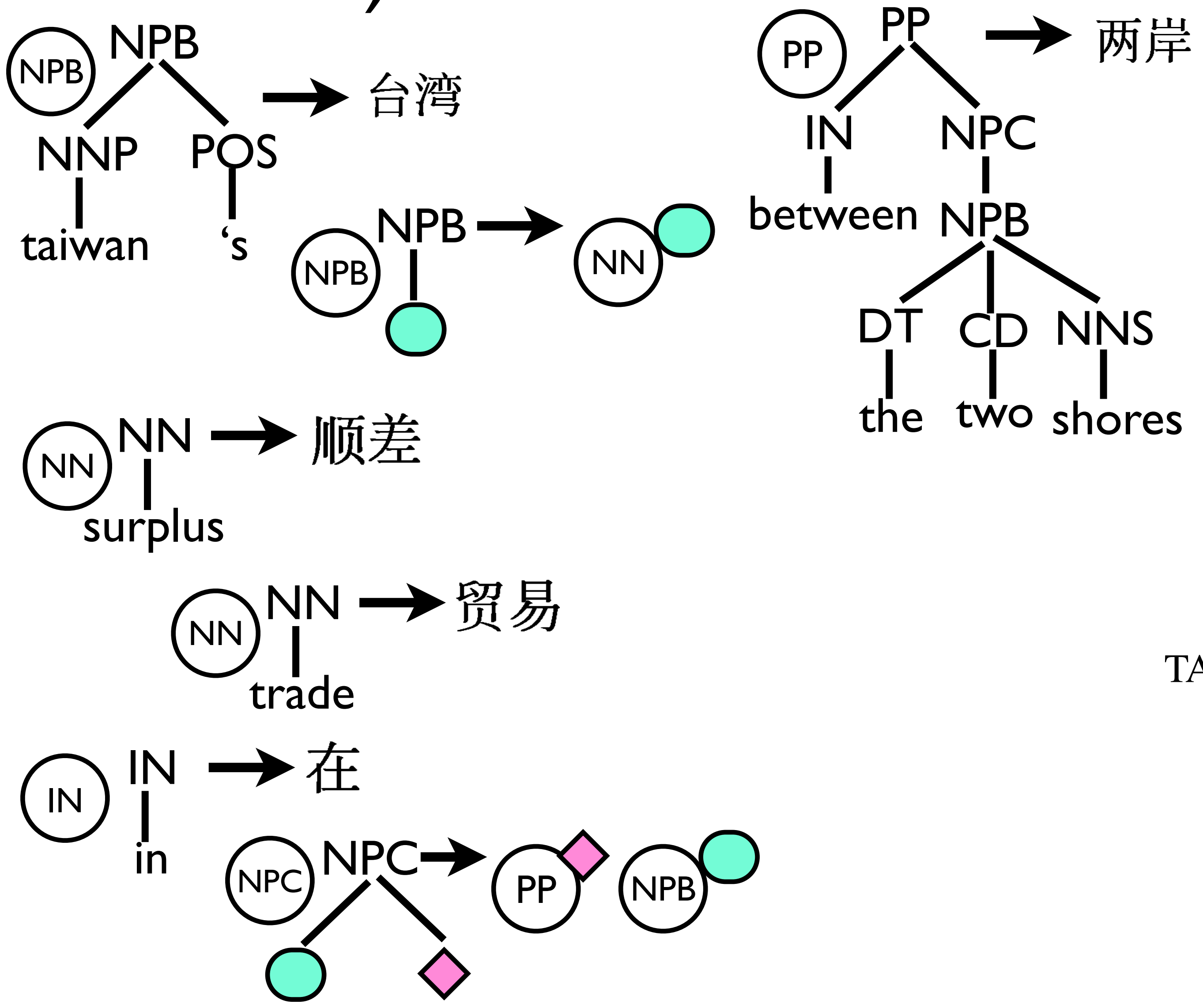


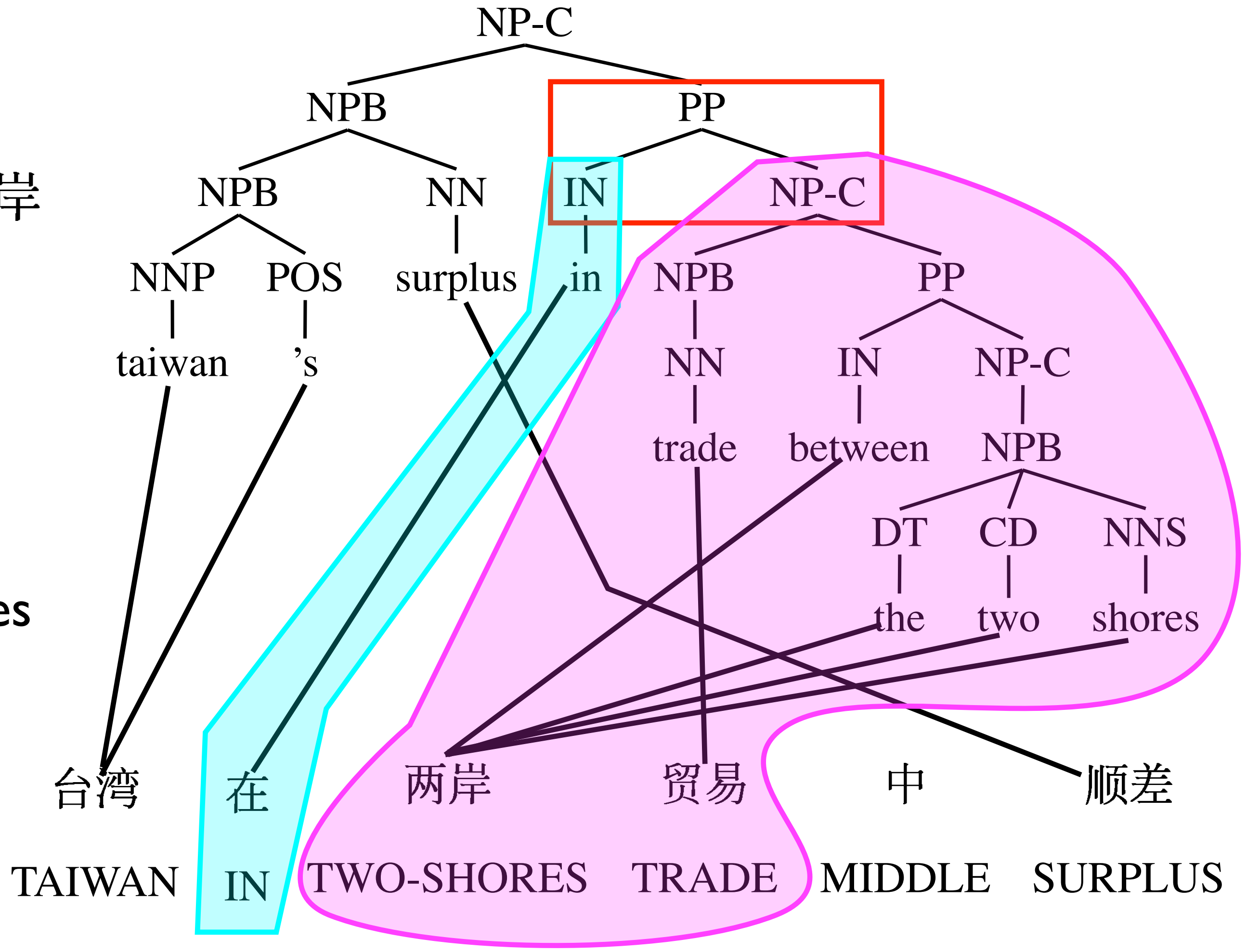


(Galley et al. '04, '06)

Extracting syntactic rules

3) Extract rules

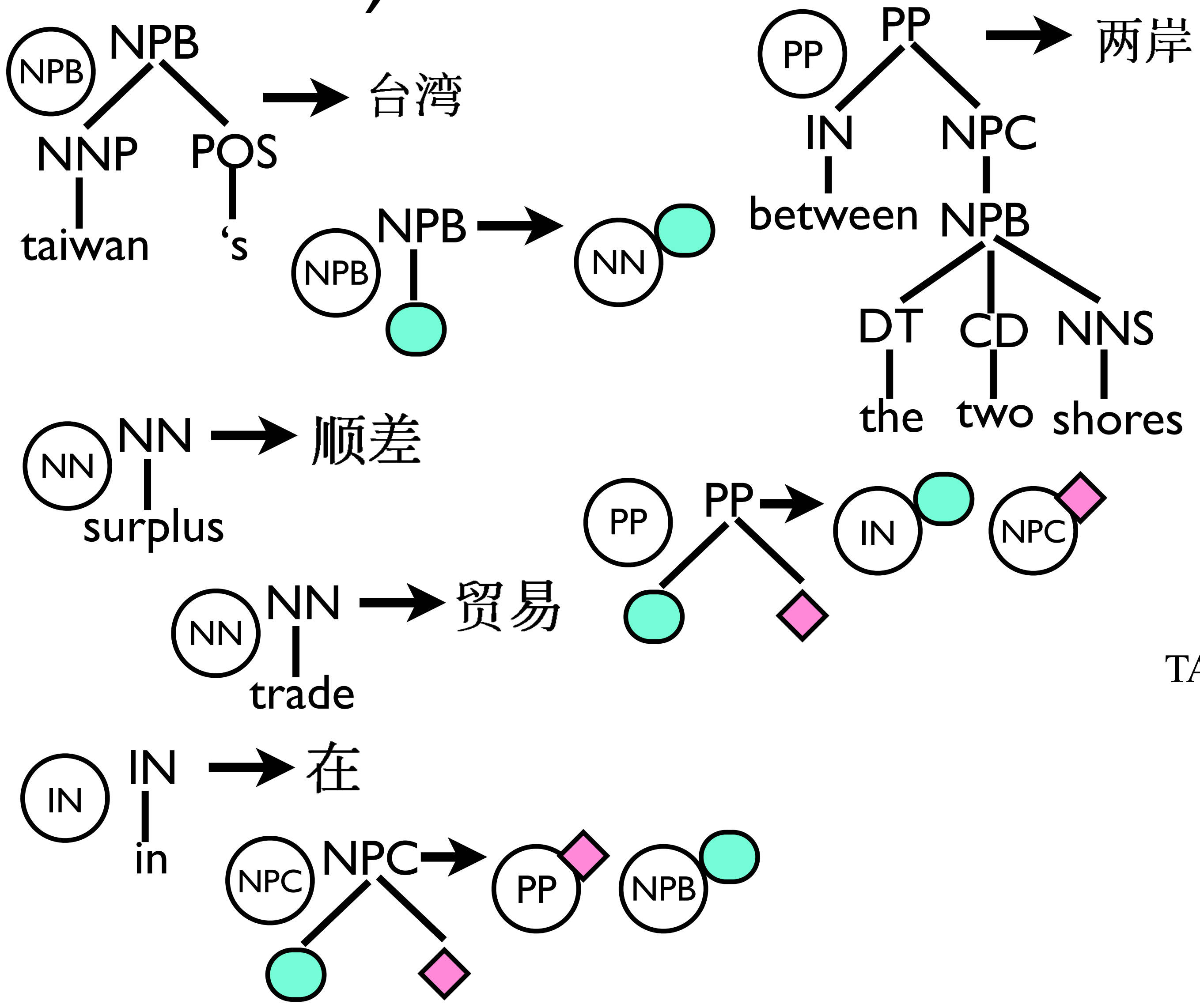


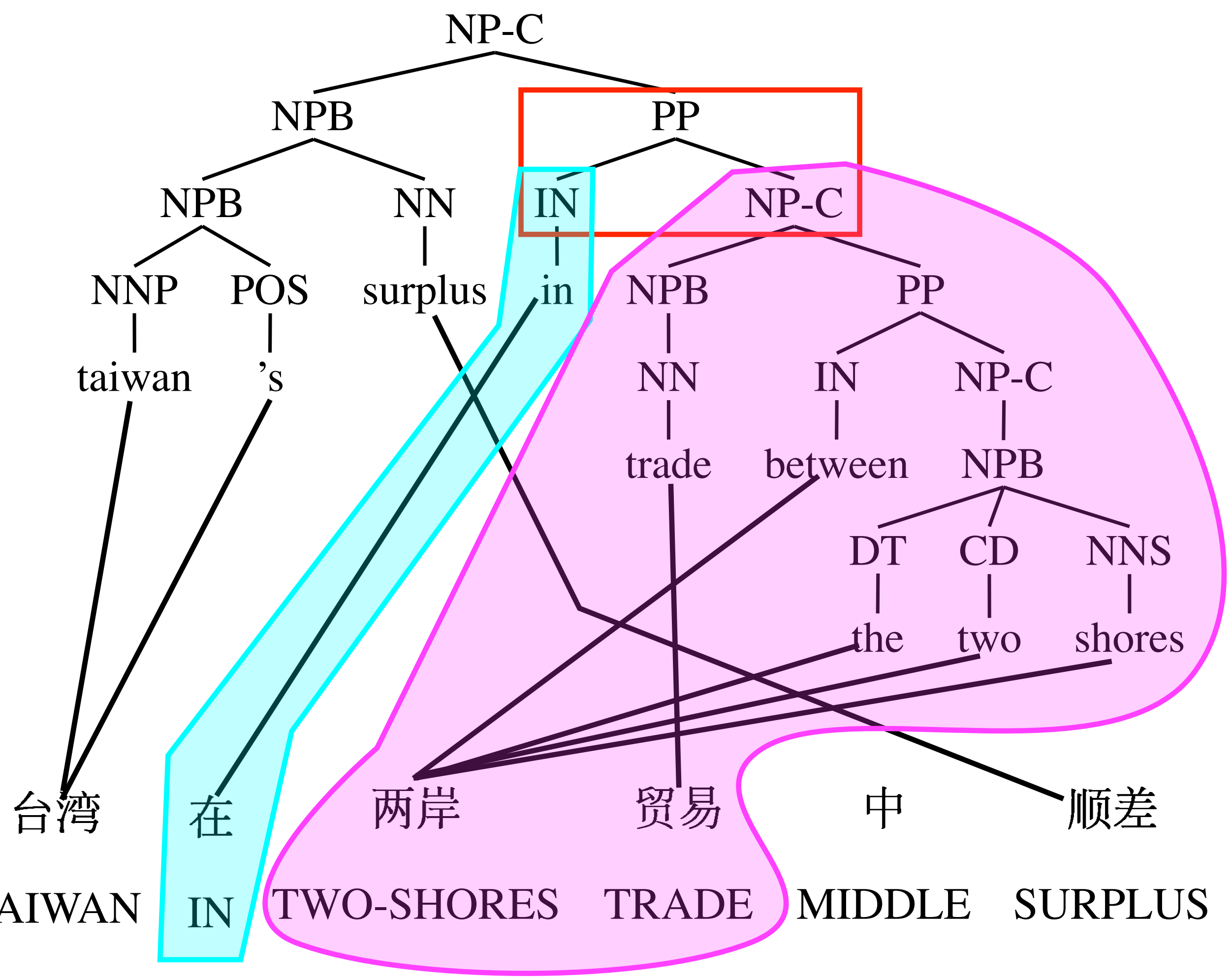


(Galley et al. ’04, ’06)

Extracting syntactic rules

3) Extract rules

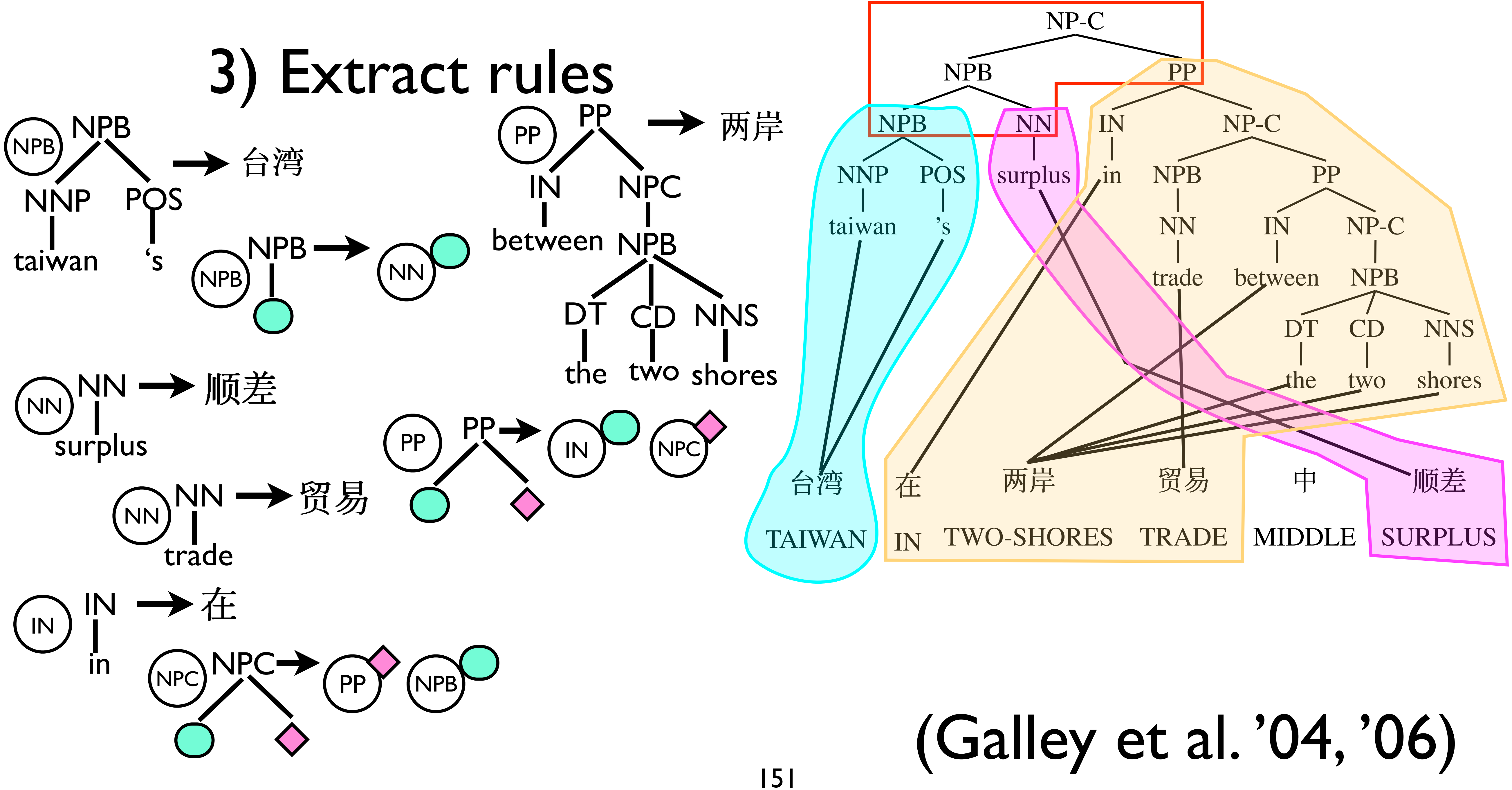




(Galley et al. '04, '06)

Extracting syntactic rules

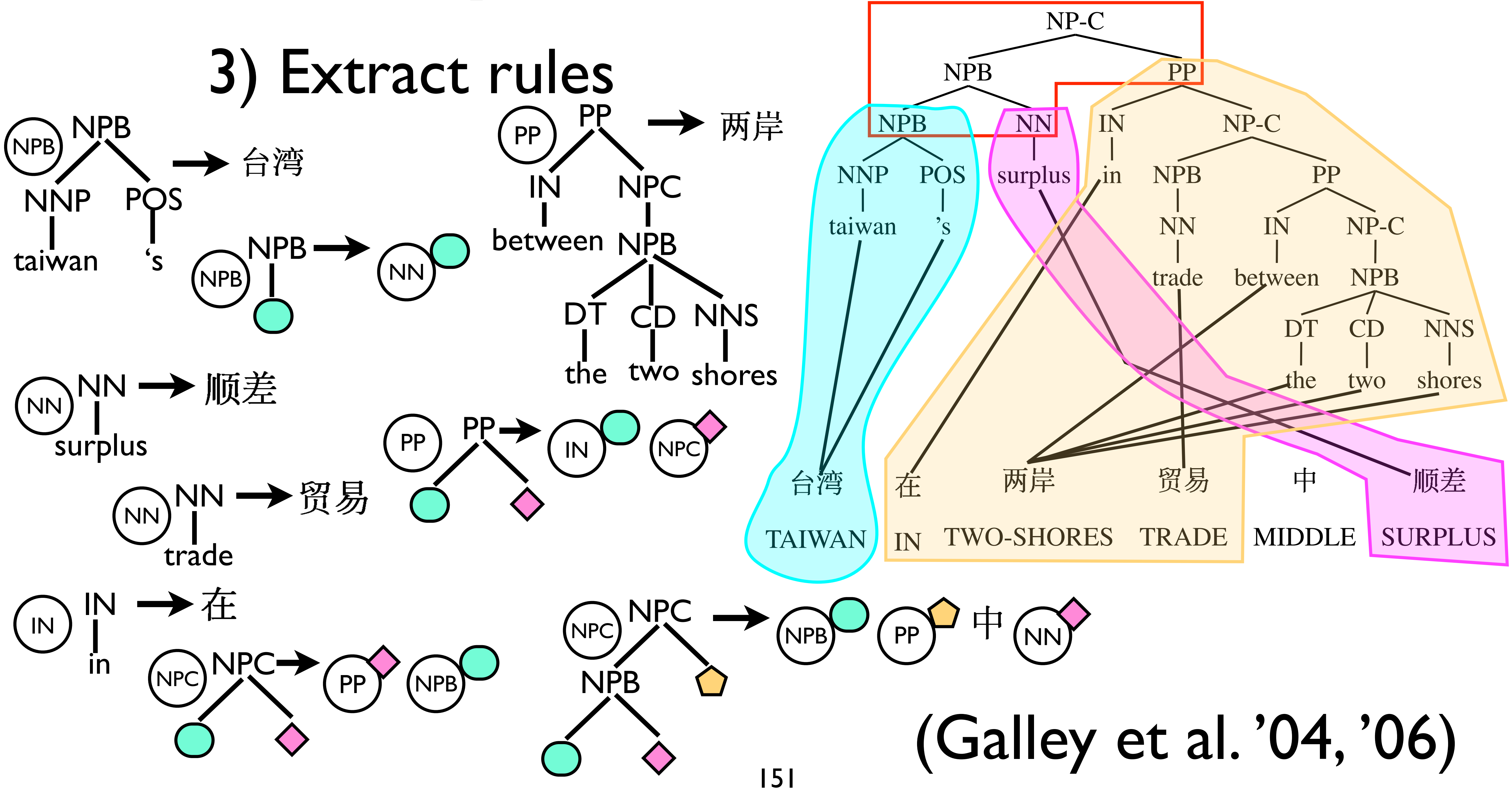
3) Extract rules



(Galley et al. '04, '06)

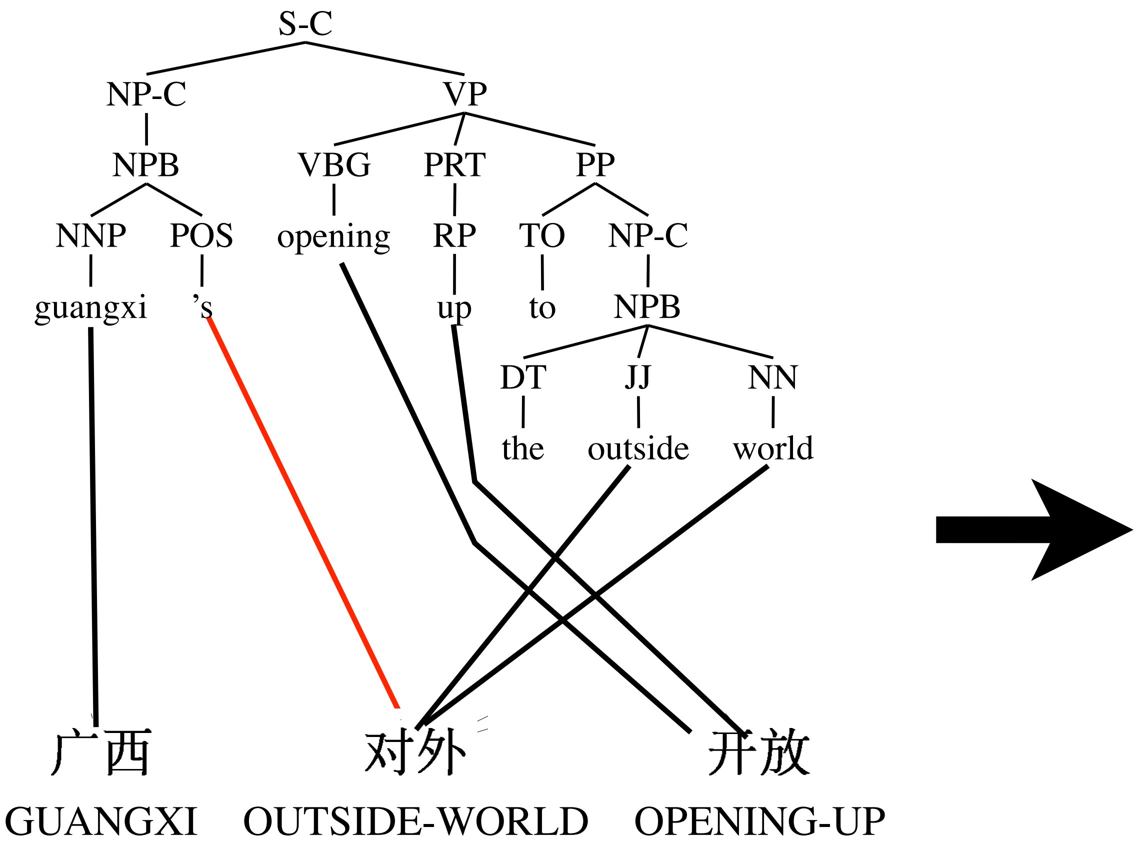
Extracting syntactic rules

3) Extract rules



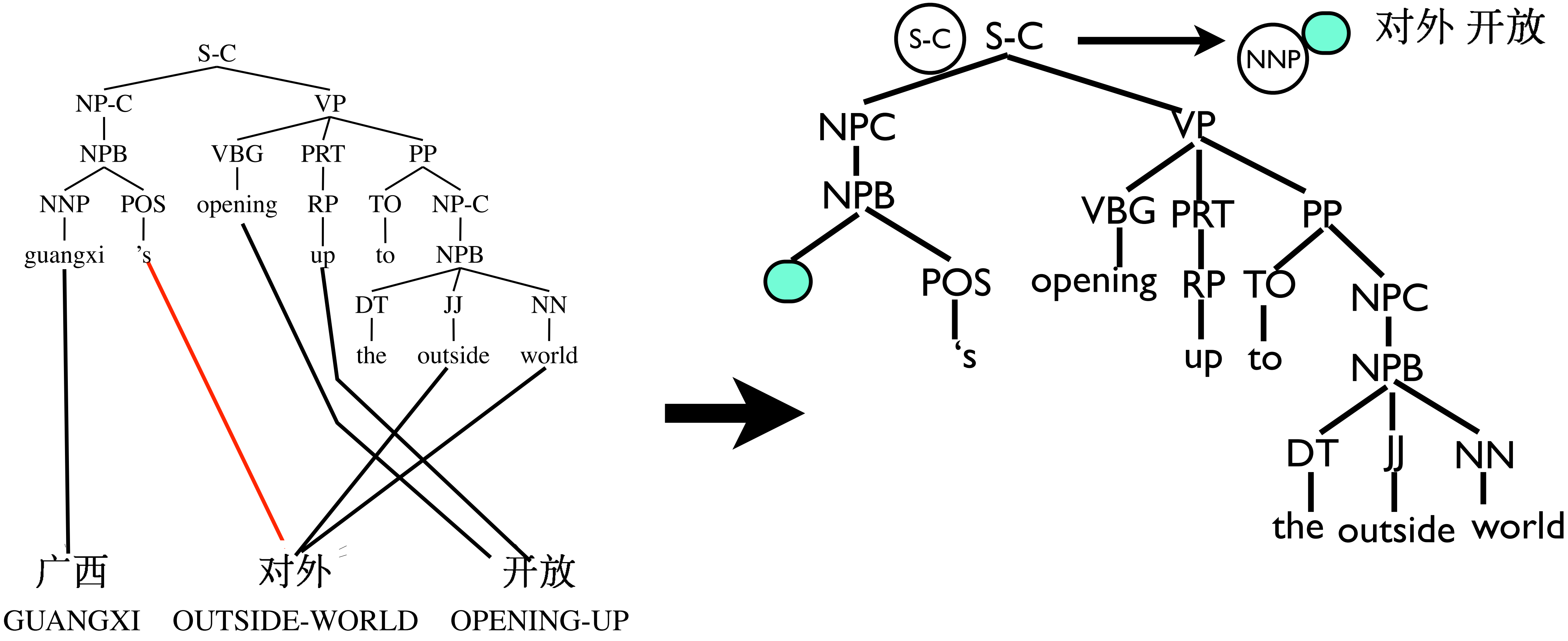
(Galley et al. '04, '06)

Bad alignments make bad rules



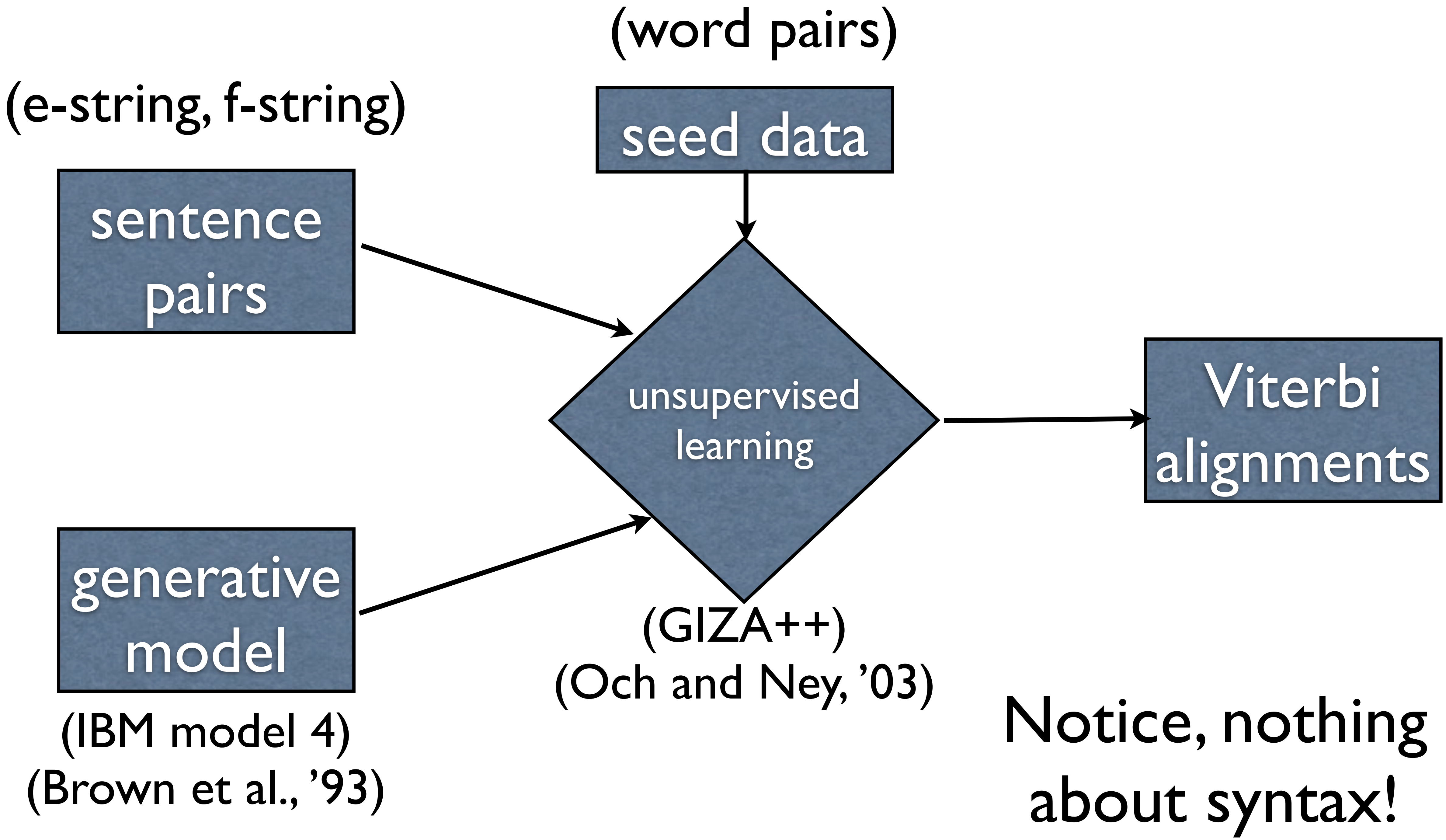
One bad link makes a totally unusable syntax rule!

Bad alignments make bad rules



One bad link makes a totally unusable syntax rule!

Where do the alignments come from?



(word pairs)

seed data

(e-string, f-string)

sentence pairs

unsupervised learning

Viterbi alignments

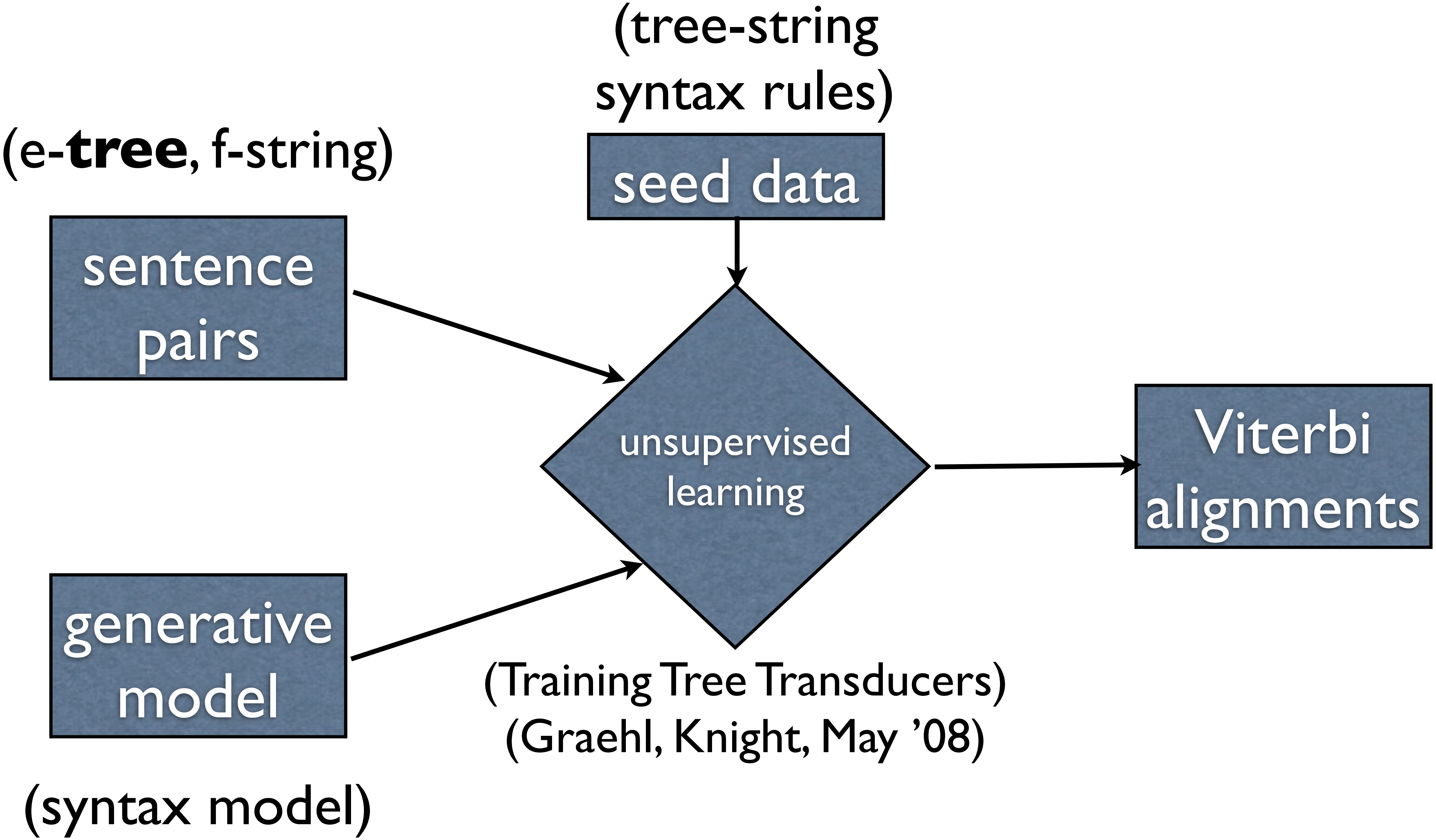
generative model

(IBM model 4)
(Brown et al., '93)

(GIZA++)
(Och and Ney, '03)

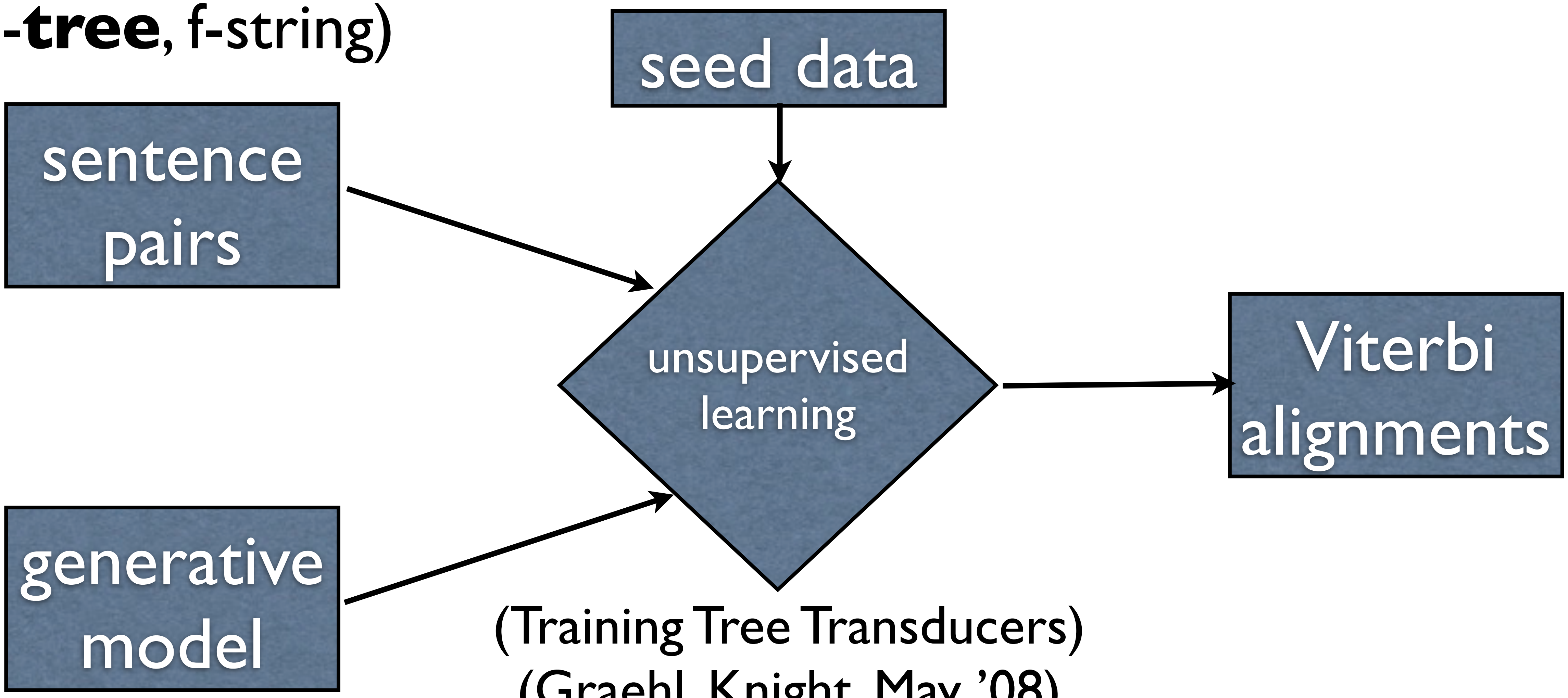
Notice, nothing about syntax!

Let’s add syntax!



(Training Tree Transducers)
(Graehl, Knight, May ’08)

Let's add syntax!



(tree-string syntax rules)

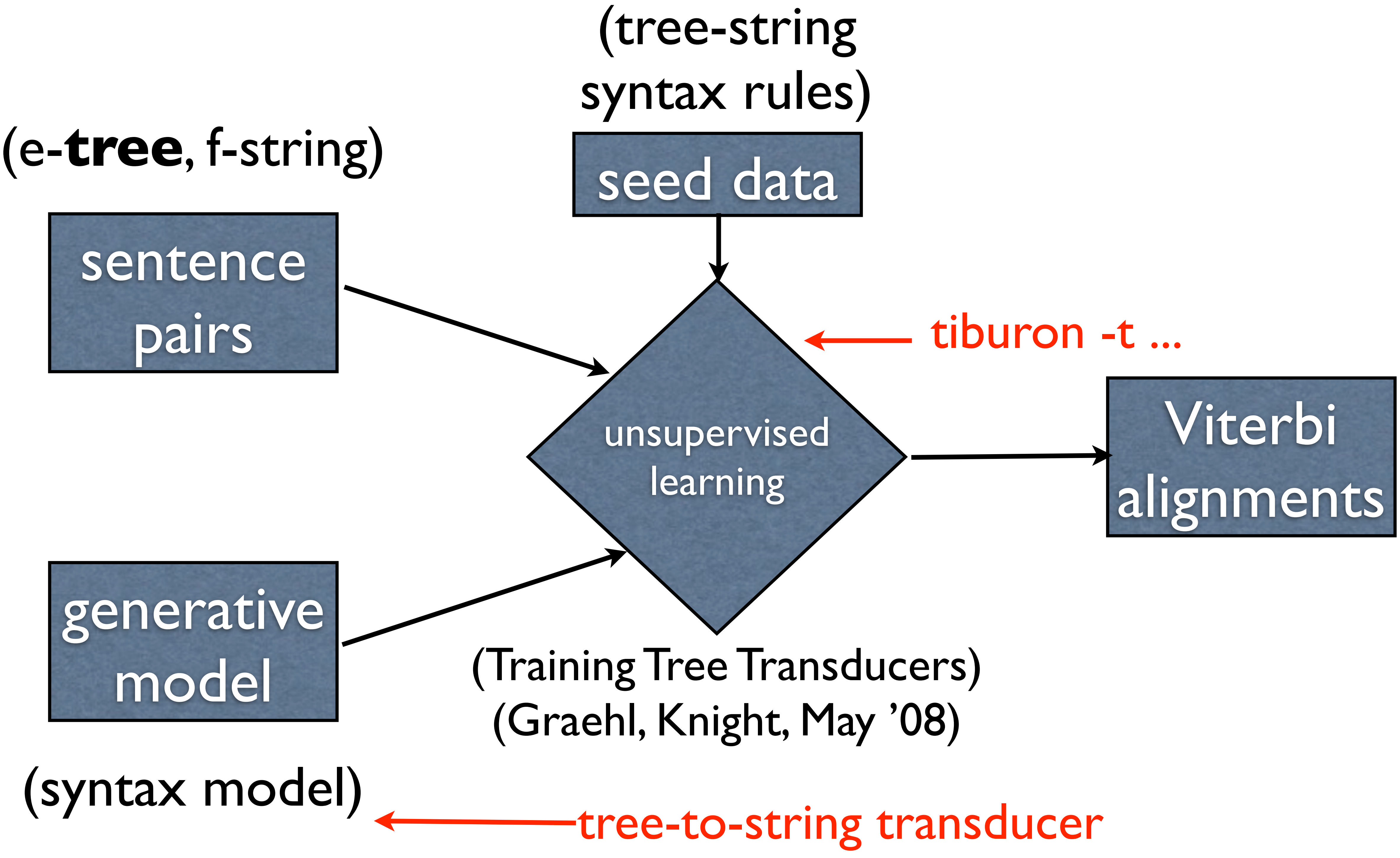
(e-**tree**, f-string)

(Training Tree Transducers)
(Graehl, Knight, May '08)

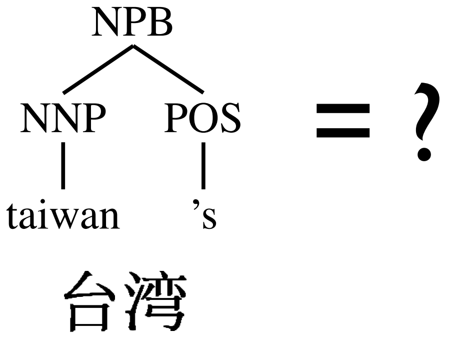
(syntax model)

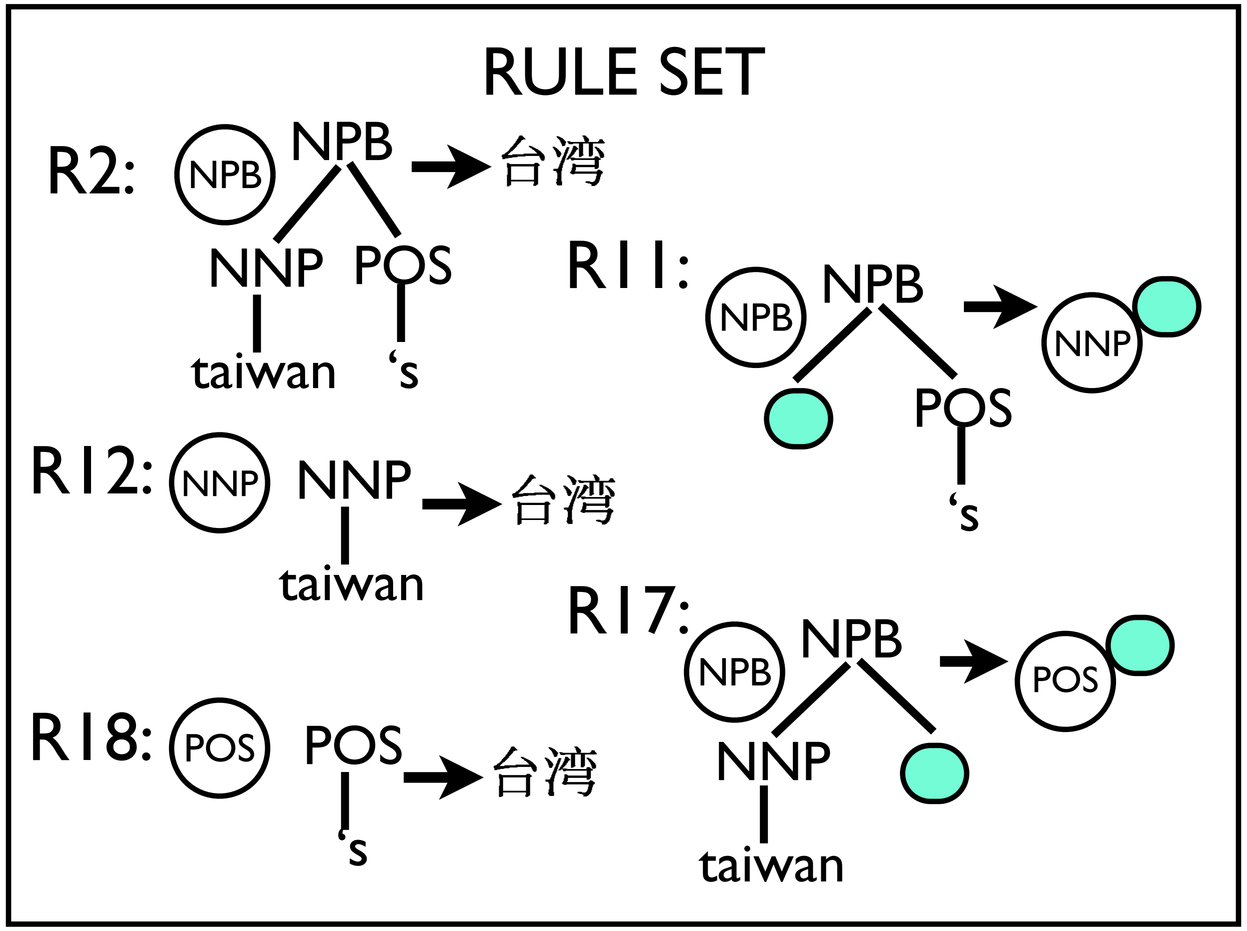
tree-to-string transducer

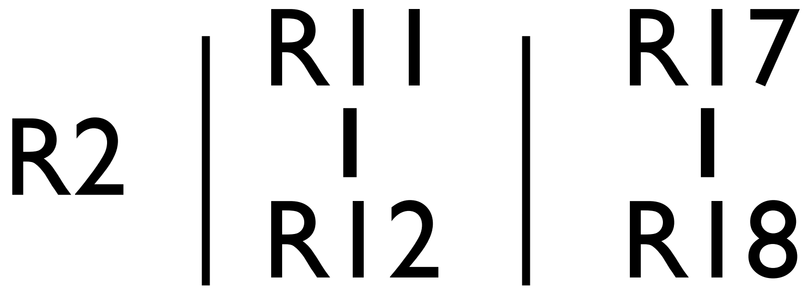
Let’s add syntax!



How we learn



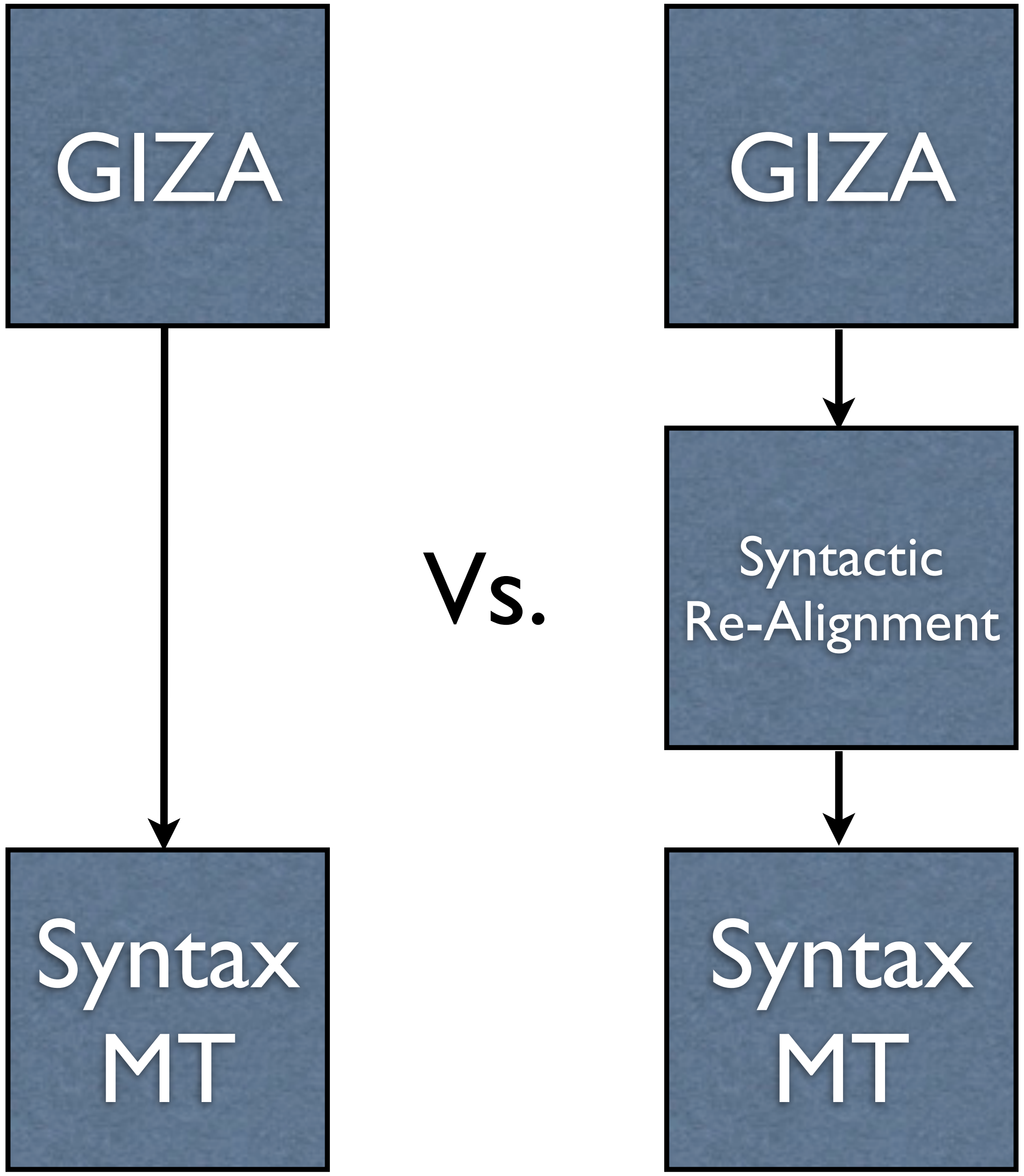




- For each training sentence, build *derivation forest* containing each possible tree of rules that satisfies the sentence pair
- EM iterations set highest probability to most useful rules
- Viterbi derivation has syntax-aware alignments and bad rules are not extracted

(Graehl, Knight, May, ’08)

Experiments



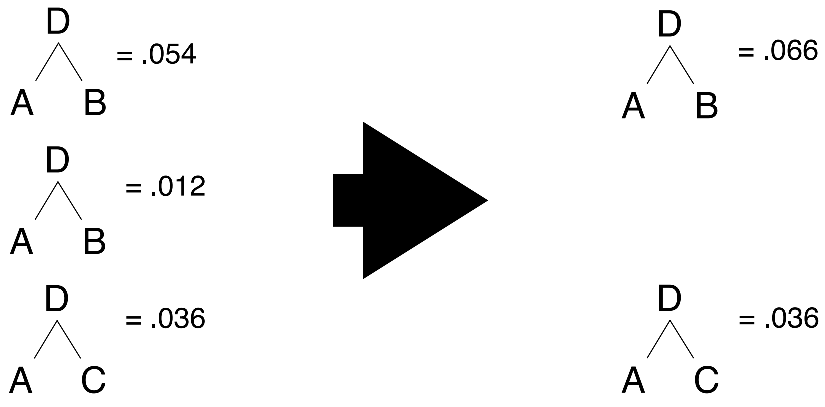
- Build a bootstrap alignment with GIZA
- Obtain new alignments with syntactic re-alignment
- Compare syntax MT system performance on rules extracted from each alignment

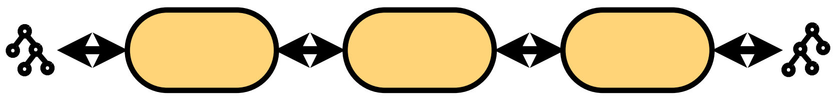
Results

source language	original alignments	type	MT system rules (millions)	NIST 2003 BLEU	Δ
Arabic	weak	baseline	2.3	47.3	+.6
		re-alignment	2.5	**47.9**	
	strong	baseline	3.2	49.6	+.4
		re-alignment	3.6	**50.0**	
Chinese	weak	baseline	19.1	37.8	+.9
		re-alignment	26.0	**38.7**	
	strong	baseline	23.4	38.9	+1.1
		re-alignment	33.4	**40.0**	

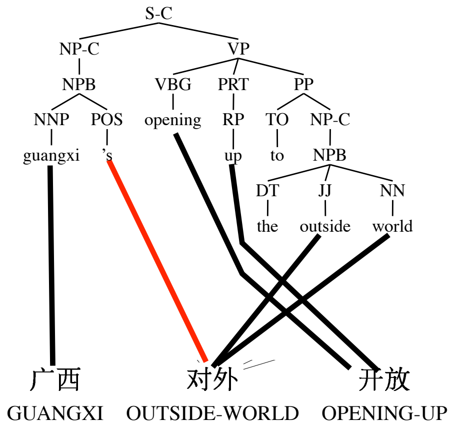
Conclusions and future work

- Algorithmic contributions
  - Determinization of weighted tree automata
  - Efficient inference through weighted tree transducer cascades





- Practical contributions
  - Weighted tree automata and transducer toolkit
  - Improvements in SMT using tree transducer EM



Future work

- More algorithms!
  - approximate linear k-best
  - on-the-fly tree-to-string inference
- More applications!
  - financial systems
  - gene sequencing
- More formalisms!
  - unranked automata
  - tree-adjoining grammars

Conclusions

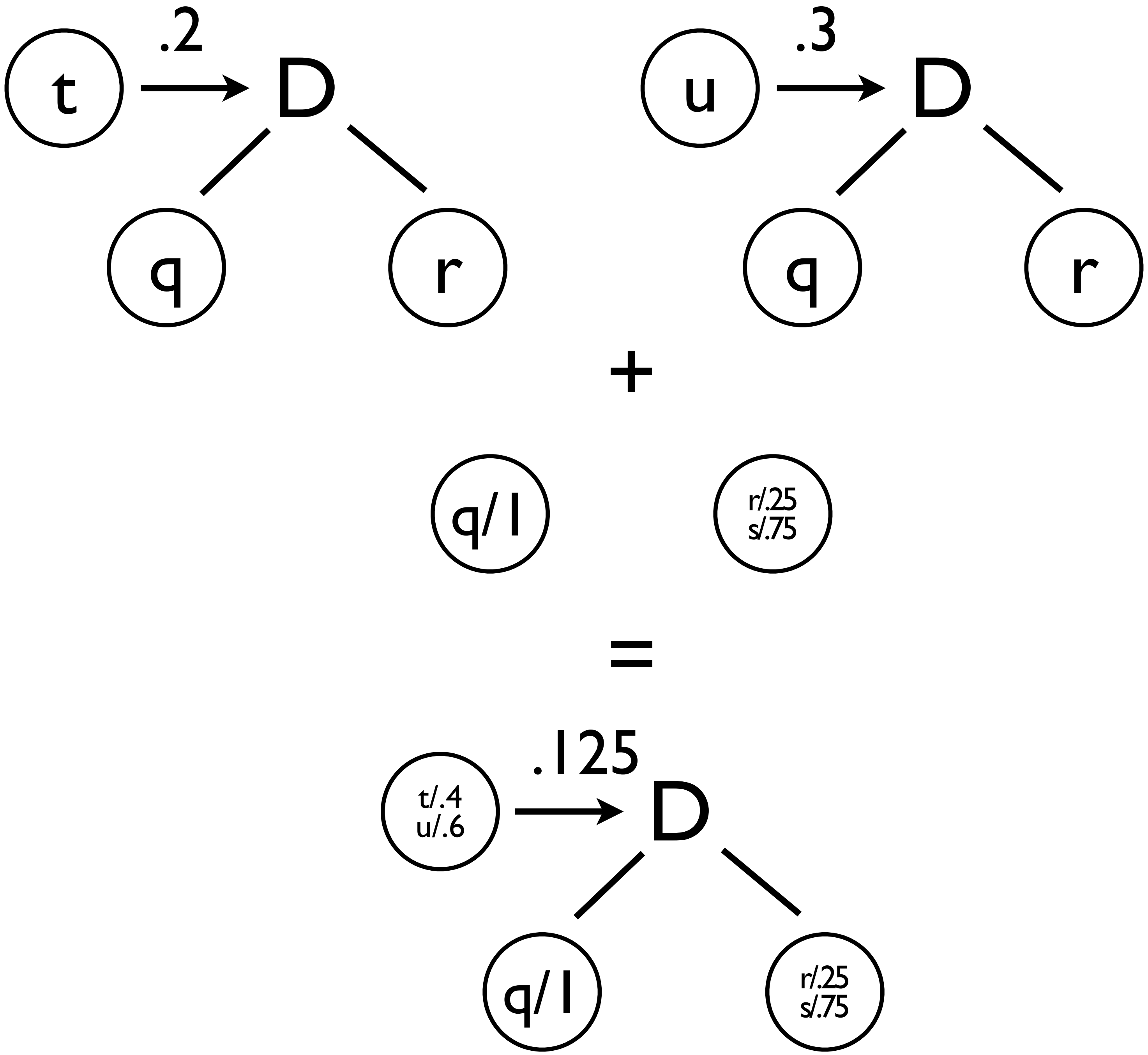
- Tiburon makes it easy to use tree transducers in NLP
- (known) Theses using Tiburon:
  - Alexander Radzievskiy -- Masters on parsing with semantic role labels
  - Joseph Tepperman -- PhD on pronunciation evaluation
  - Victoria Fossum -- PhD on machine translation and parsing
- July 2010: ATANLP in Uppsala!

Thanks!

Erika Barragan-Nunez, Rahul Bhagat, Marlynn Block, Matthias Büchse, Gully Burns, Marco Carbone, David Chiang, Hal Daumé III, Steve DeNeefe, John DeNero, Jason Eisner, Victoria Fossum, Alex Fraser, Jonathan Graehl, Erica Greene, Carmen Heger, Ulf Hermjakob, Johanna Högberg, Dirk Hovy, Ed Hovy, Liang Huang, David Kempe, Kevin Knight, Sven Koenig, Zornitsa Kozareva, Lorelei Laird, Kary Lau, Jerry Levine, Andreas Maletti, Daniel Marcu, Mitch Marcus, Howard May, Irena May, Rutu Mehta, Alma Nava, Adam Pauls, Fernando Pereira, Ben Plantan, Oana Postolache, Michael Pust, David Pynadath, Sujith Ravi, Deepak Ravichandran, Jason Riesa, Bill Rounds, Lee Rowland, Tom Russ, Shri Narayanan, Radu Soricut, Magnus Steinby, Shang-Hua Teng, Cătălin Tîrnăucă, Ashish Vaswani, Jens Vöckler, Heiko Vogler, David Foster Wallace, Wei Wang, Ralph Weischedel, Kenji Yamada

Backup Slides

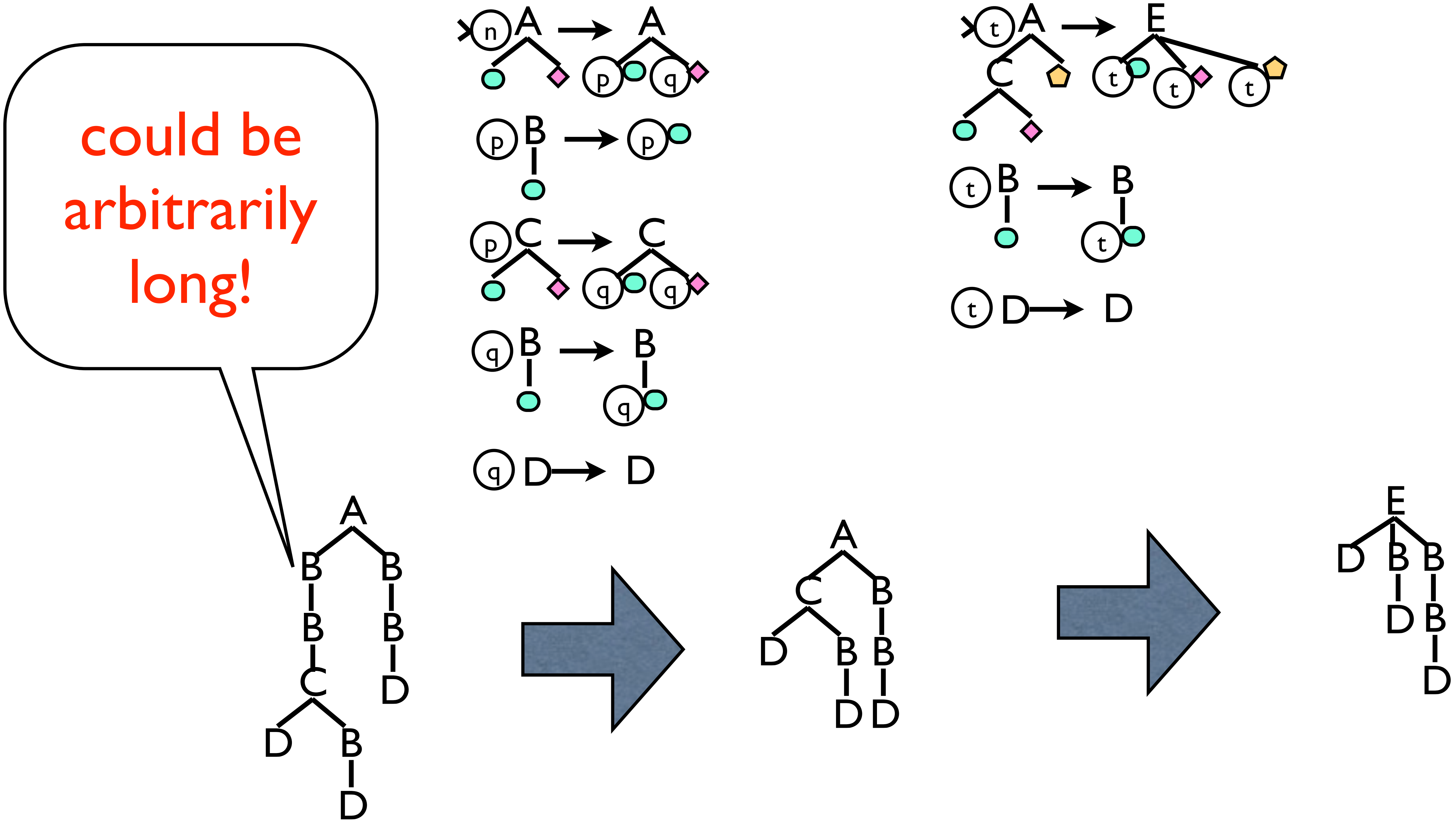
Non-deterministic and nonterminal?



MT Details

- Decoded 116 short Chinese sentences using the string-to-tree MT model based on (Galley et al. 2004)
  - No language model
  - No reranking
- Counted number of trees in each forest before and after determinization
- 86.3% trees in forest are duplicates on average
  - 1.4×10^{12} median per forest pre-determ
  - 2.0×10^{11} median per forest post-determ
- Determinization changes top tree 77.6% of the time
- Crunching matches determinization 50.6% of the time

xLNT not closed!



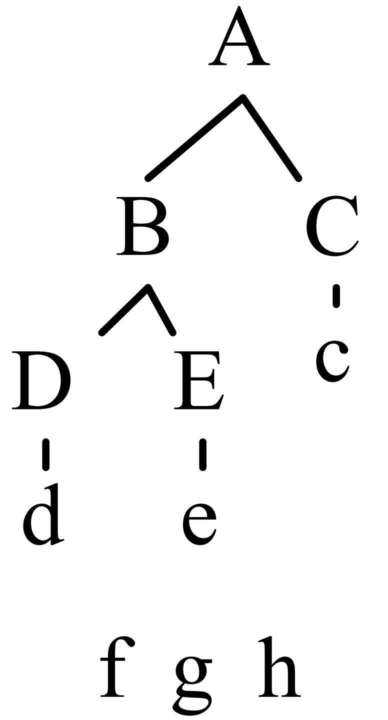
could be arbitrarily long!

(Maletti, Graehl, Hopkins, Knight, ’09)

Closure Under Composition and Recognizability Preservation

closed	forward recog	backward recog
wLNT	wxLNT	xT
		wxLT

Where do the rules come from?



=

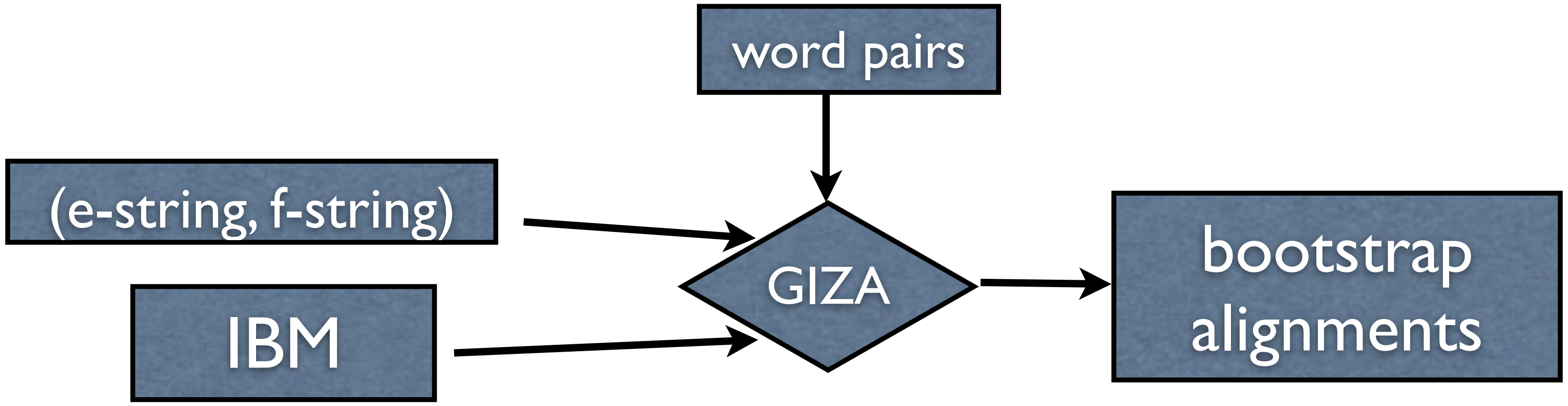
103 possible rules

- Ideally we would add all possible rules
- To avoid overflow, we bootstrap with a previous (syntax-free) alignment model
- This follows a rich history in MT (Och & Ney ’00, Fraser & Marcu ‘06)

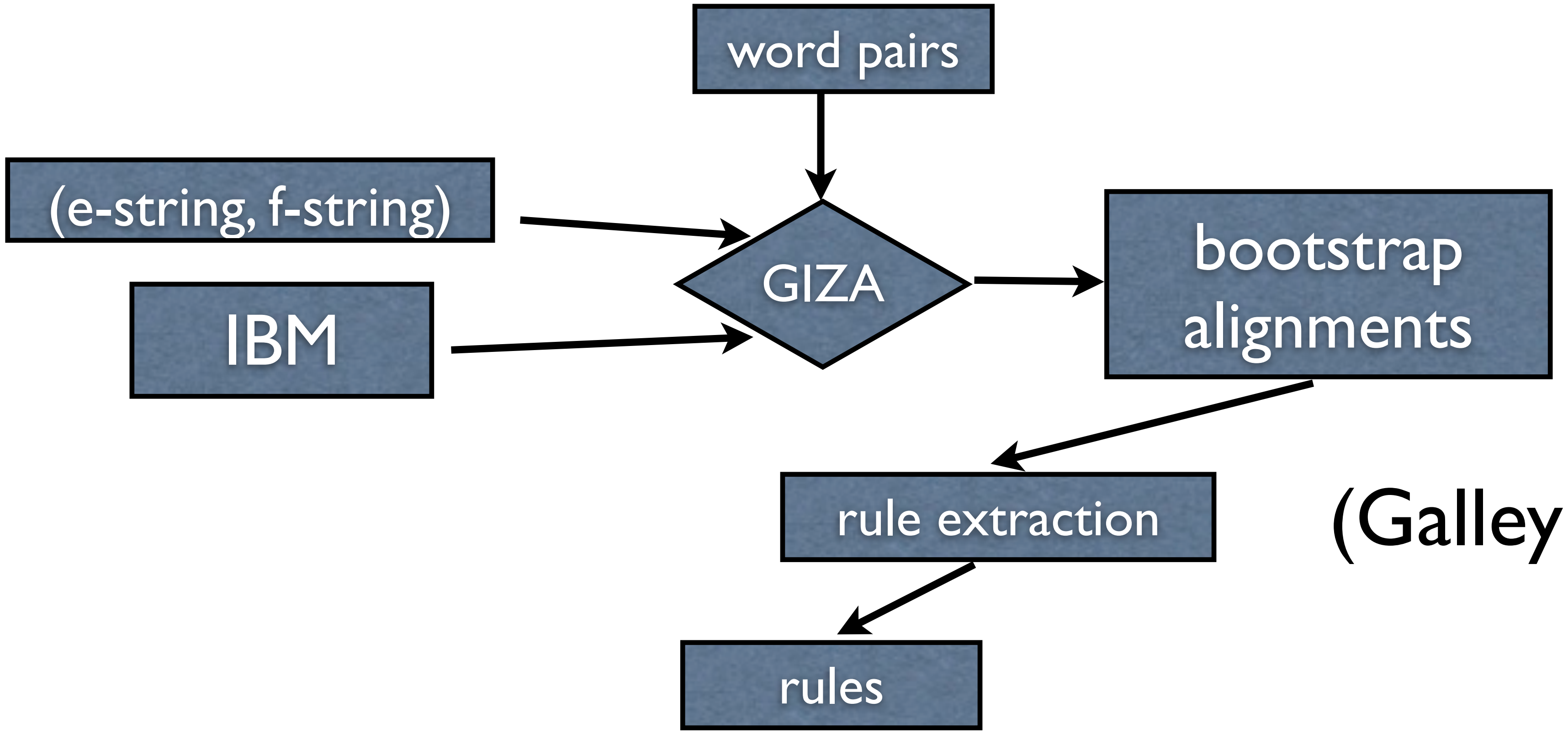
Other approaches to this problem

- Cherry and Lin ‘06: Discriminatively train ITG-based alignment model influenced by dependency graph
- DeNero and Klein ‘07: HMM model modified to incorporate syntax penalty into distortion
- Fossum et al. ‘08: Identify troublesome links and remove them

Where do the rules come from?

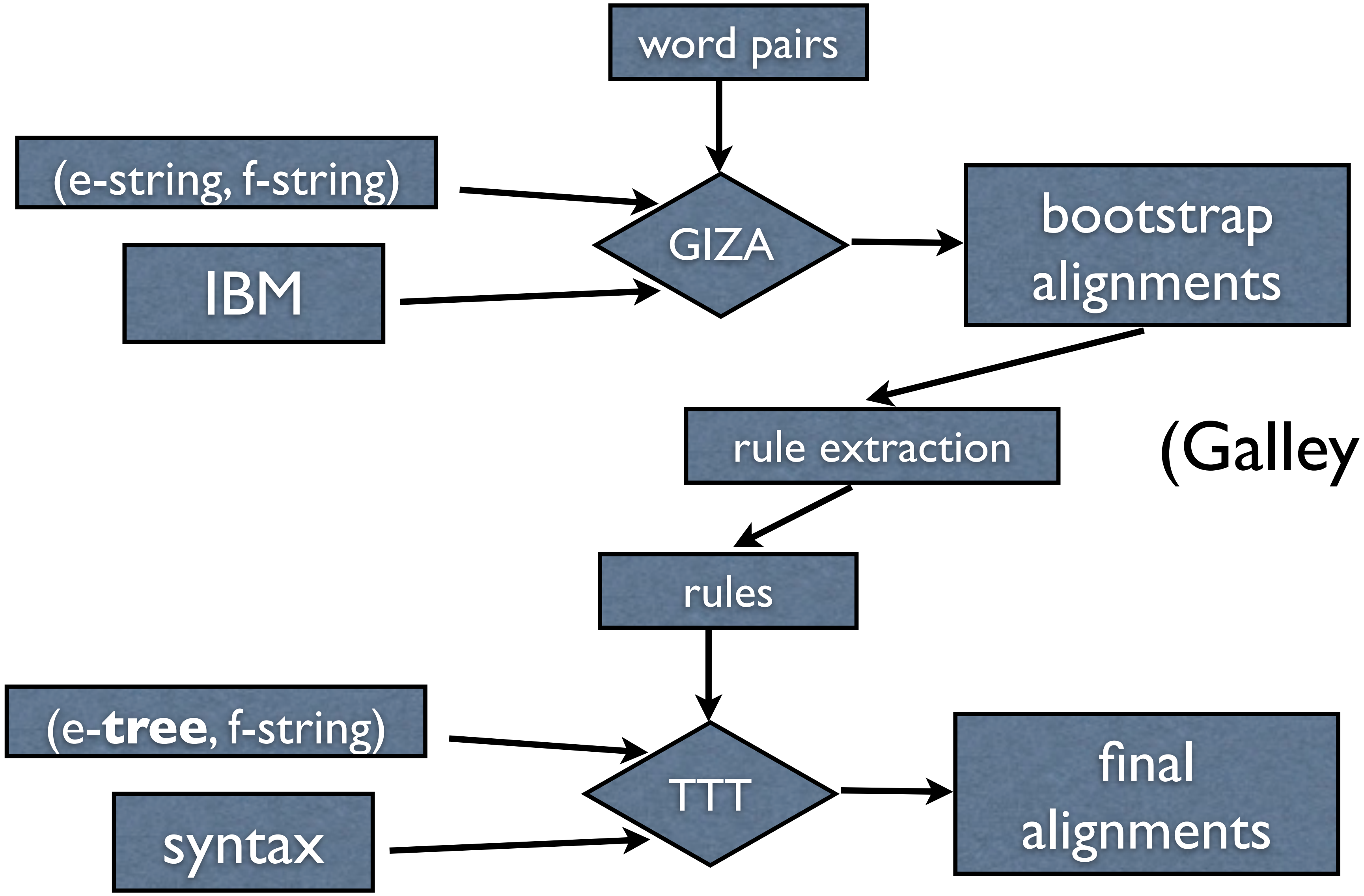


Where do the rules come from?



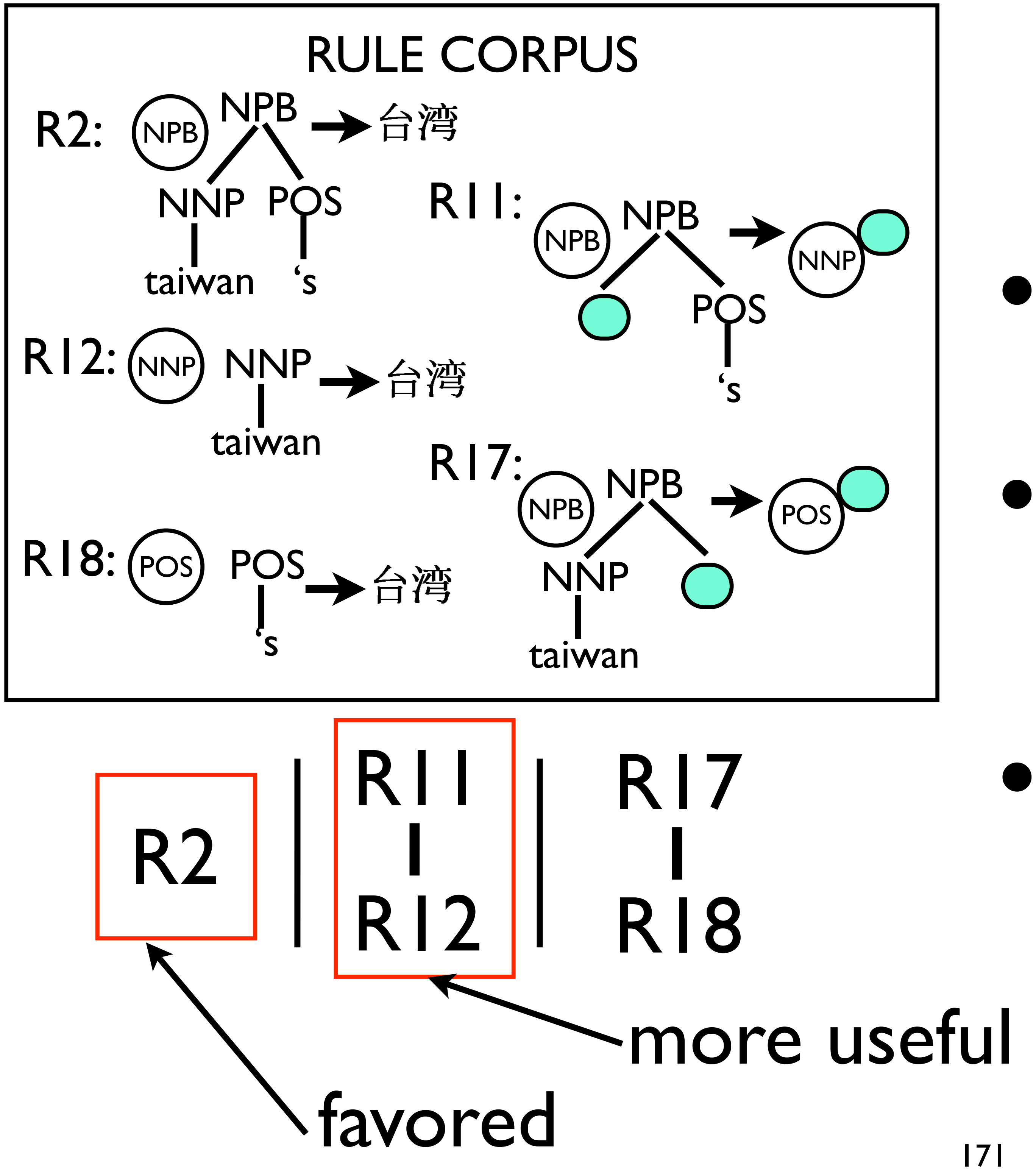
(Galley et al. '04)

Where do the rules come from?



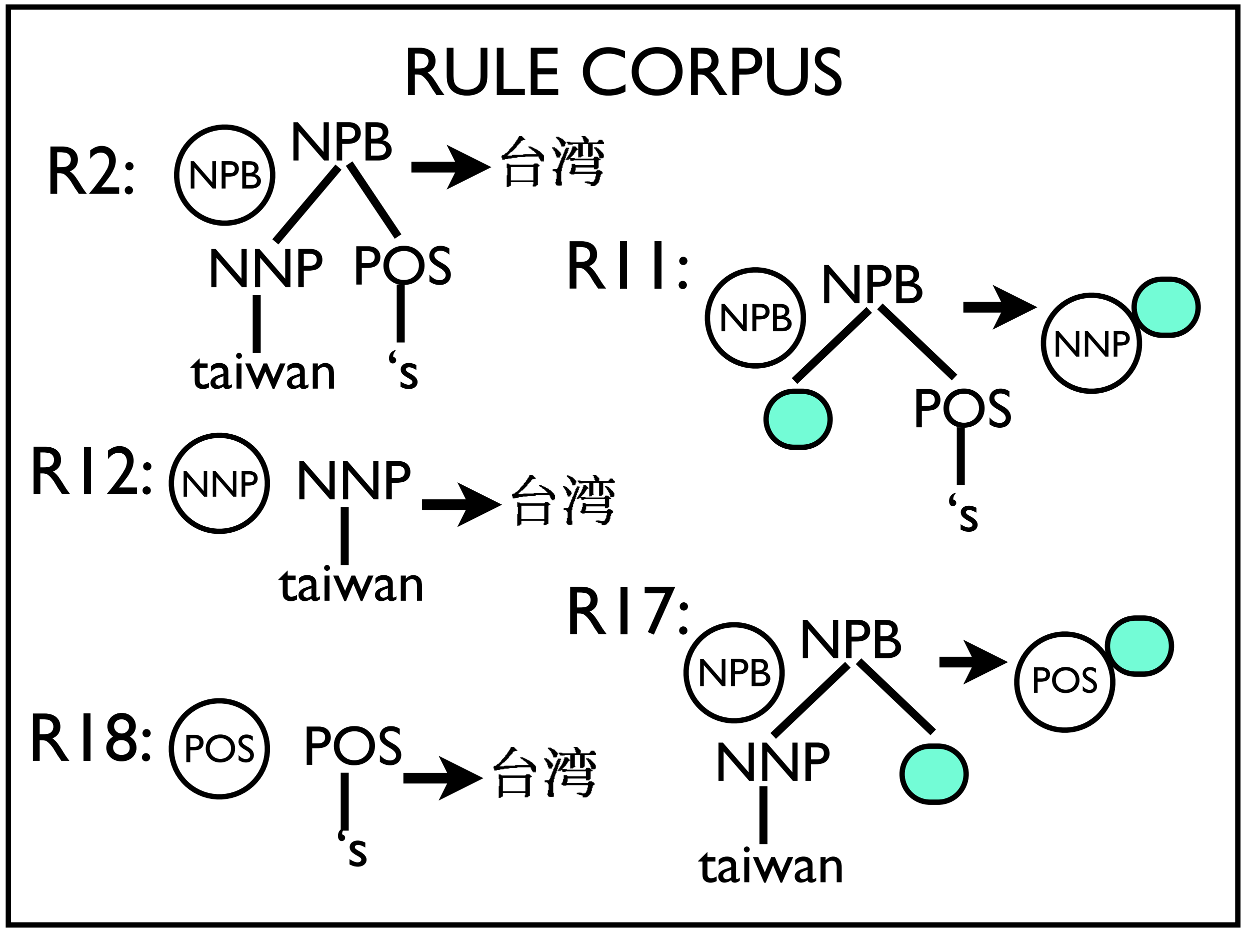
(Galley et al. '04)

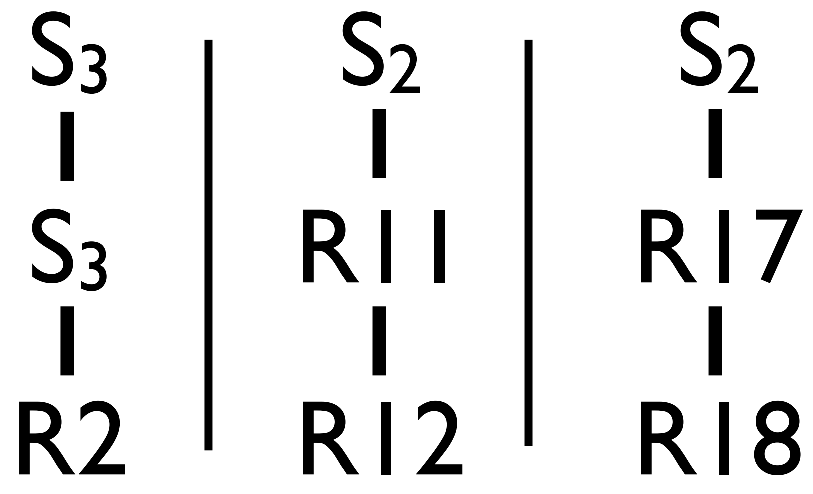
EM size bias



- EM attempts to learn derivations with highest probability.
- Shorter derivations have fewer chances to take a probability "hit" and are thus biased to be favored.
- This, then, tends to favor larger rules, generally the opposite of what we want.

Correcting size bias





- When using a rule with *n* non-leaf nodes, prepend *n-1* copies of a special *size rule* S_n
- Each competing derivation now has the same number of rules
- Size rules are built into the derivation forests and weights are learned by the same EM procedure

Complexity Analysis

k-best (H&C)	$O(P+ D_{max}k \log k)$	P = rtg rules D_{max} = max deriv

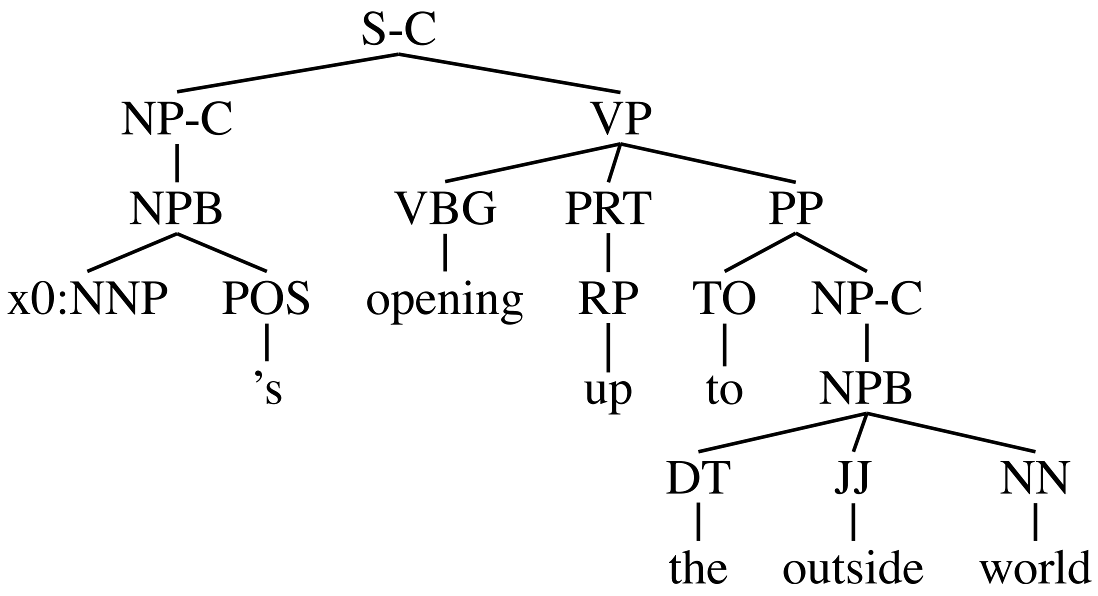
determinization	$O(Ak^{zL})$	A = alph size
		k = max rank
		z = max tree size
		L = lang size

rtg+xLNT	$O(RP^{l})$	R = trans rules
		P = rtg rules
		l = max trans lhs

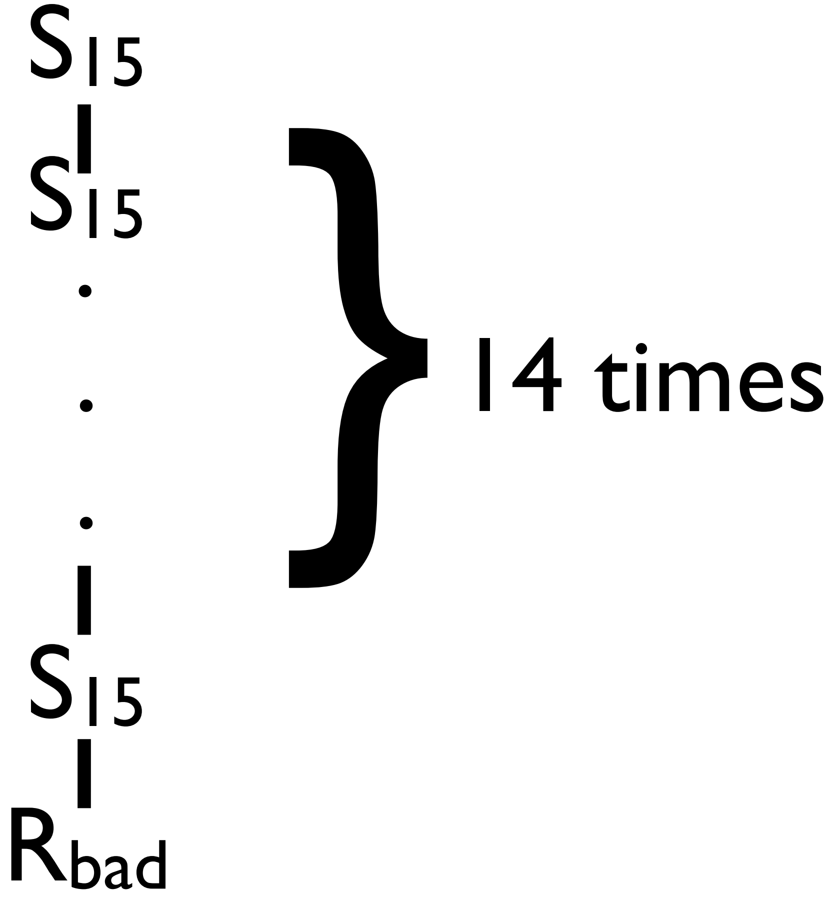
xT+LNT	$O(R_A R_B^{r})$	R_A = xT rules R_B = LNT rules r = max R_A rhs

Dramatic use of size rules

R_{bad}:



→ x0 对外 开放



Approximate Algorithms

- linear-time approximate *k*-best
- polynomial time determinization that fails to recognize some trees in the input
- weighted domain projection with relative ordering, but not exact weights, preserved
- mildly incorrect fast composition
- on-the-fly tree-to-string backward application

Engineering

- Battle-test Tiburon implementations and bring it up to production level
- Make greater use of system on biological sequencing and financial systems analysis -- leads to more interesting algorithmic questions, different types of transducers

Explore the limits of Tree Transducers

- Weighting scheme of Collins' parsing model[1] doesn't fit well
- Very large tree transducers needed in syntax MT[2]
- Can these models be simplified and still retain their power? Or should different formalisms be used?

[1]: Collins, 1997

[2]: DeNeefe and Knight, 2009