Towards Acceptors and Transducers for Semantics

kevin knight

columbia university, february 2012

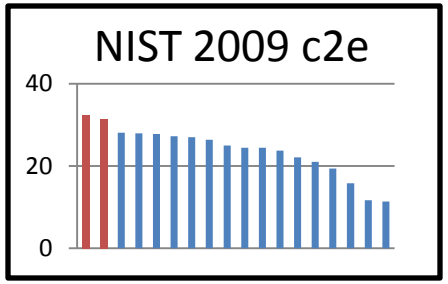
Machine Translation

Phrase-based MT

source string → target string

Syntax-based MT

source string → source tree → target tree → target string

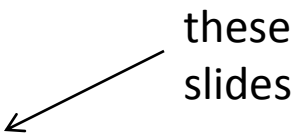


Meaning-based MT

source string → source tree → meaning representation → target tree → target string

Meaning-based MT

- What content goes into the meaning representation? 1
  - linguistics, annotation
- How are meaning representations probabilistically generated, transformed, scored? 2
  - automata theory, efficient algorithms
- How can a full MT system be built and tested? 3
  - understanding, generation, rule extraction, language modeling, features, training, engineering



Automata Frameworks

- How to represent and manipulate linguistic representations?
- Linguistics, NLP, and Automata Theory used to be together (1960s, 70s)
  - Context-free grammars were invented to model human language
  - Tree transducers were invented to model transformational grammar
- They drifted apart
- Renewed connections around MT (this century)
- Role: greatly simplify systems!

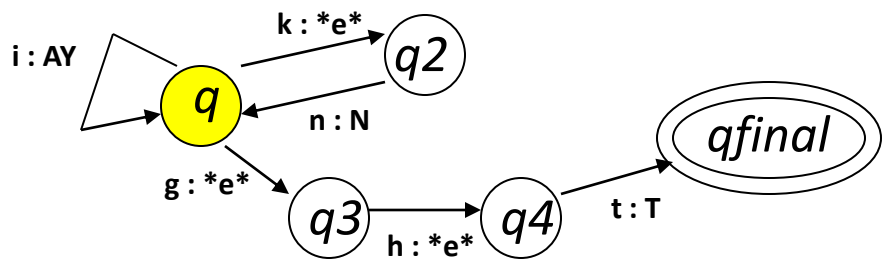
Finite-State Transducer (FST)

Original input:

k n i g h t

FST

q k → *q2* *e*
q i → *q* AY
q g → *q3* *e*
q2 n → *q* N
q3 h → *q4* *e*
q4 t → *qfinal* T



Transformation:

q k n i g h t

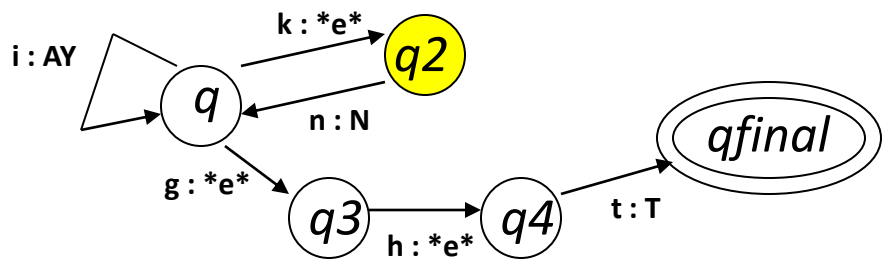
Finite-State Transducer (FST)

Original input:

k n i g h t

FST

q k → *q2* *e*
q i → *q* AY
q g → *q3* *e*
q2 n → *q* N
q3 h → *q4* *e*
q4 t → *qfinal* T



Transformation:

q2 n i g h t

Finite-State Transducer (FST)

Original input:

k n i g h t

FST

q k → *q2* *e*

q i → *q* AY

q g → *q3* *e*

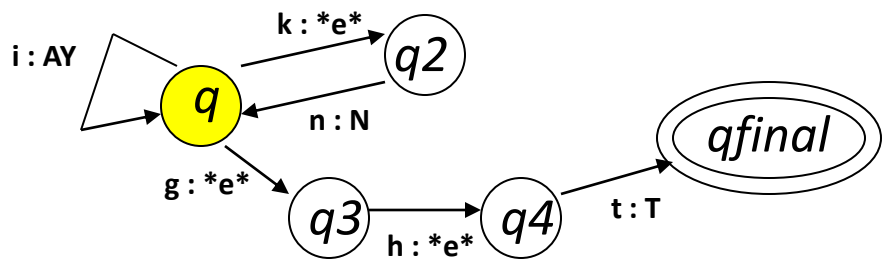
q2 n → *q* N

q3 h → *q4* *e*

q4 t → *qfinal* T

Transformation:

N *q* i g h t



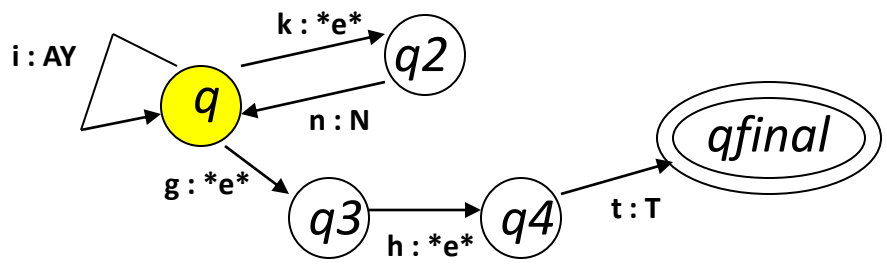
Finite-State Transducer (FST)

Original input:

k n i g h t

FST

q k → *q2* *e*
q i → *q* AY
q g → *q3* *e*
q2 n → *q* N
q3 h → *q4* *e*
q4 t → *qfinal* T



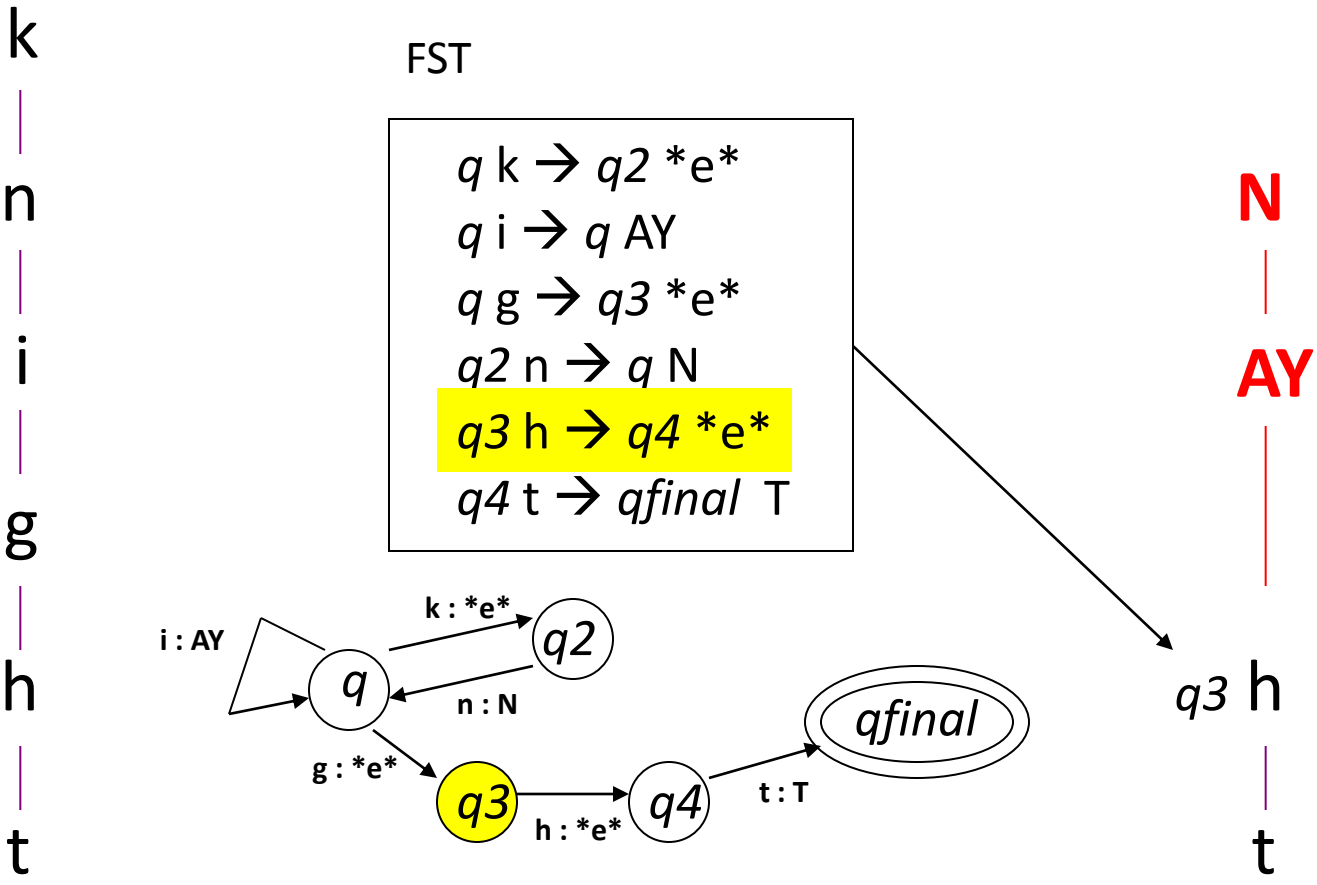
Transformation:

N AY *q* g h t

Finite-State Transducer (FST)

Original input:

k n i g h t



Transformation:

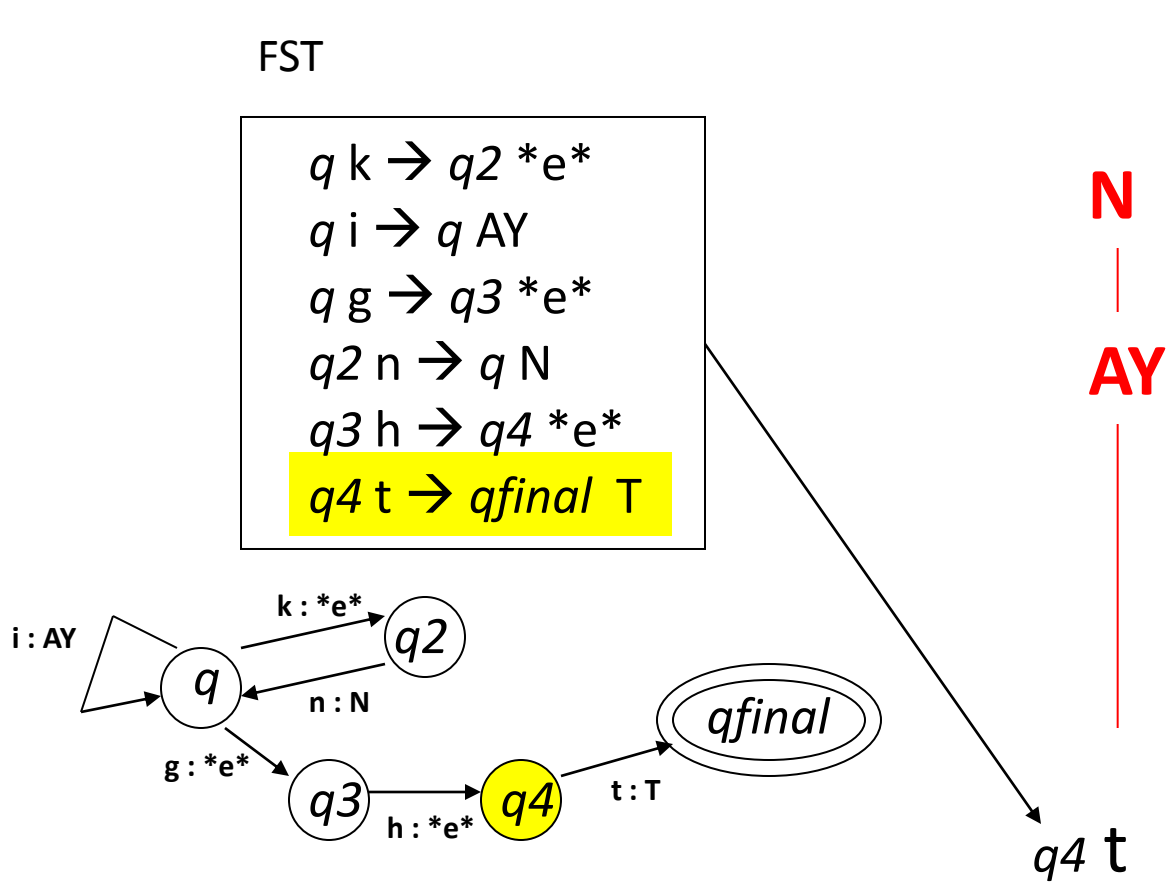
N AY q3 h t

Finite-State Transducer (FST)

Original input:

k
n
i
g
h
t

Transformation:

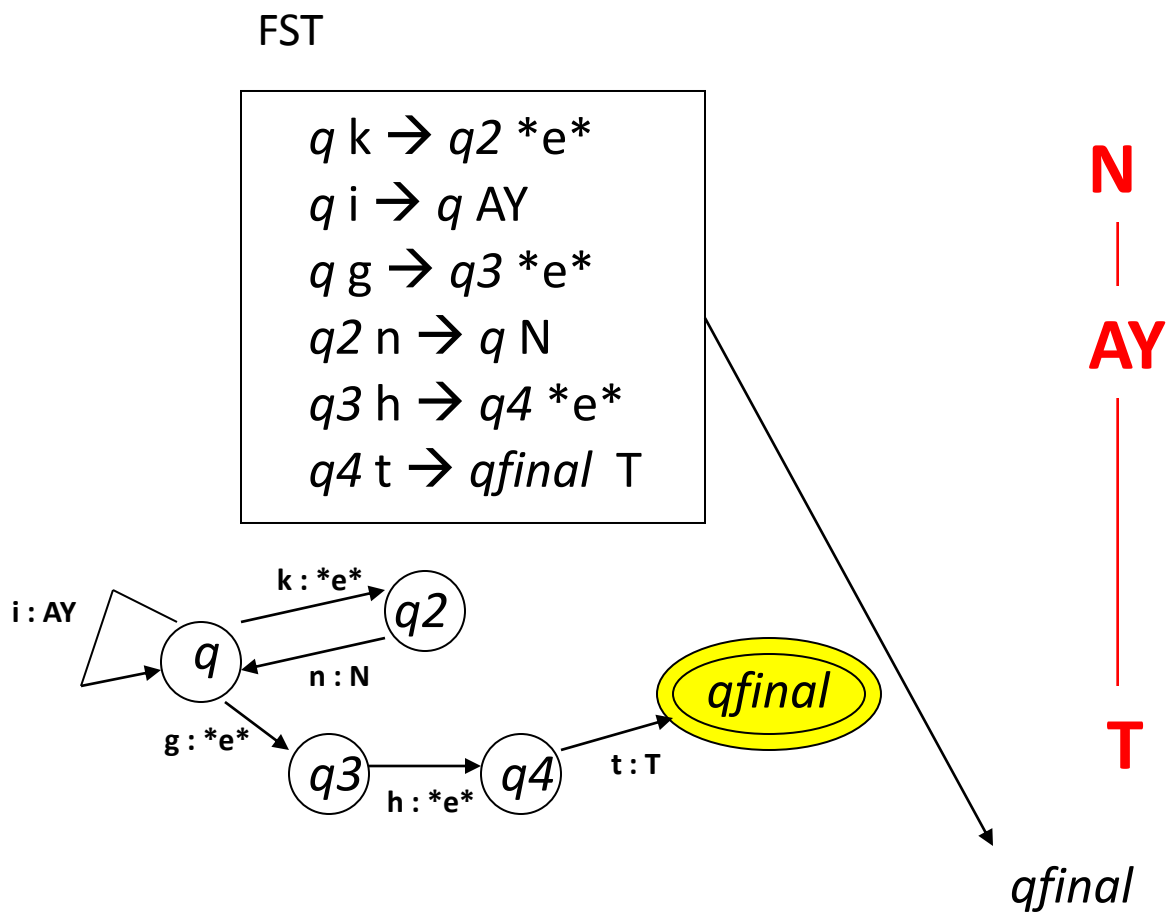


Finite-State Transducer (FST)

Original input:

k
n
i
g
h
t

Transformation:

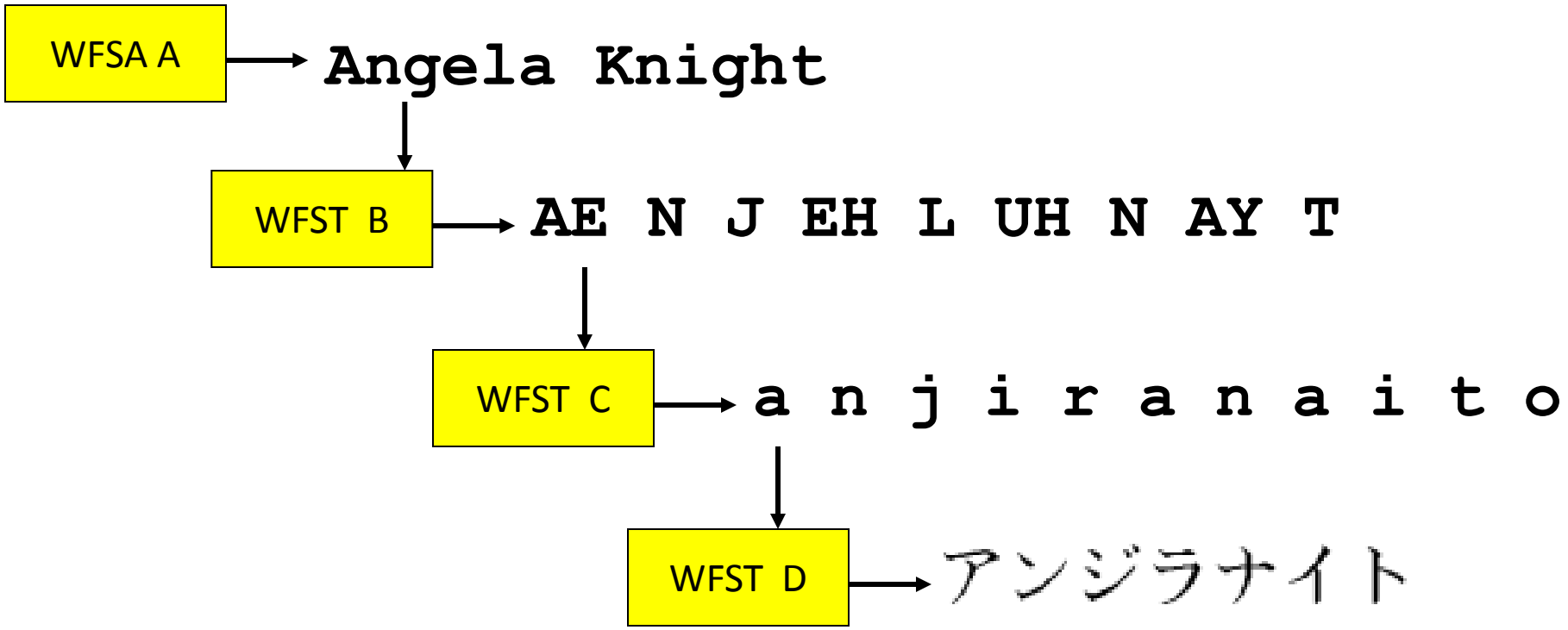


Transliteration

7 input symbols

13 output symbols

Transliteration



Angela Knight
Angela Nite
Andy Law Knight
Angela Nate
+ millions more

WFSA A

Ann Gere Uh
Anne Jill Ahh
Angy Rugh
Ann Zillah
+ millions more

WFST B

AE N J IH R UH N AY T
AH N J IH L UH N AY T OH

+ millions more

WFST C

a n j i r a n a i t o

WFST D

アンジラナイト

DECODE

$$\underset{E}{\operatorname{argmax}}\, P(E|K) =$$

$$\underset{E}{\operatorname{argmax}} \sum_{es} \sum_{js} P(E) \cdot P(es|E) \cdot P(js|es) \cdot P(K|js)$$

Finite-State String Transducers

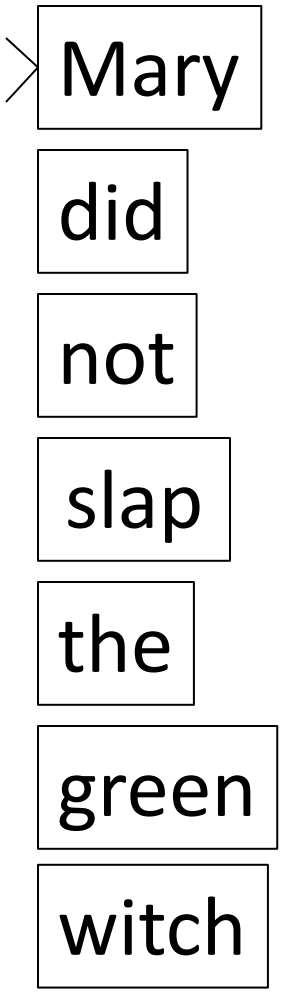
- non-deterministic
- invertible
- composable

etc

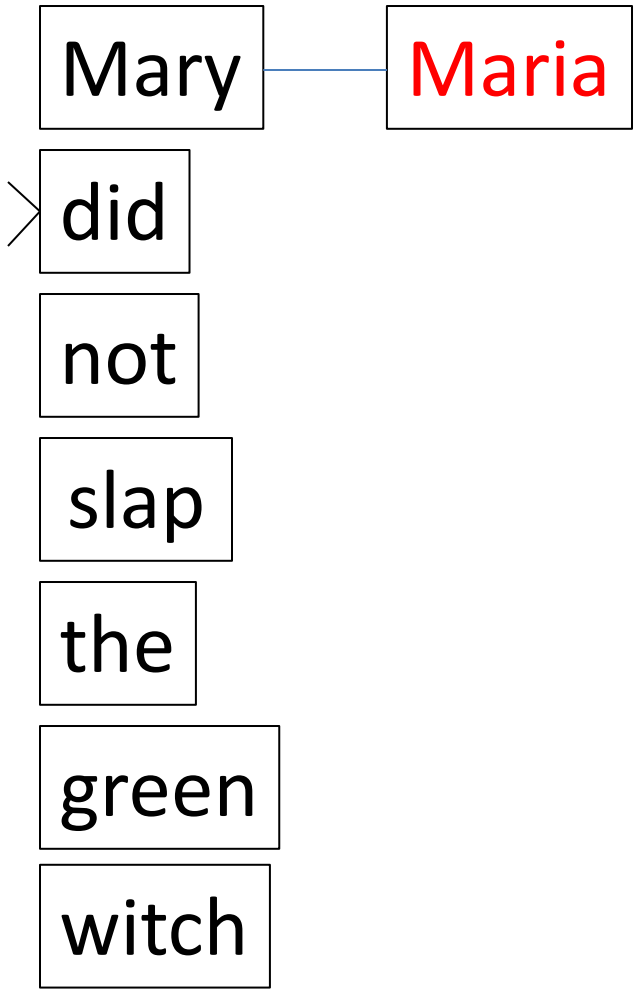
General-Purpose Algorithms for String Automata

N-best …	… paths through an WFSA (Viterbi, 1967; Eppstein, 1998)
EM training	Forward-backward EM (Baum & Welch, 1971; Eisner 2001)
Determinization …	… of weighted string acceptors (Mohri, 1997)
Intersection	WFSA intersection
Application	string → WFST → WFSA
Transducer composition	WFST composition (Pereira & Riley, 1996)
General-purpose toolkit	FSM (AT&T), Carmel (USC/ISI), OpenFST (Google), ...

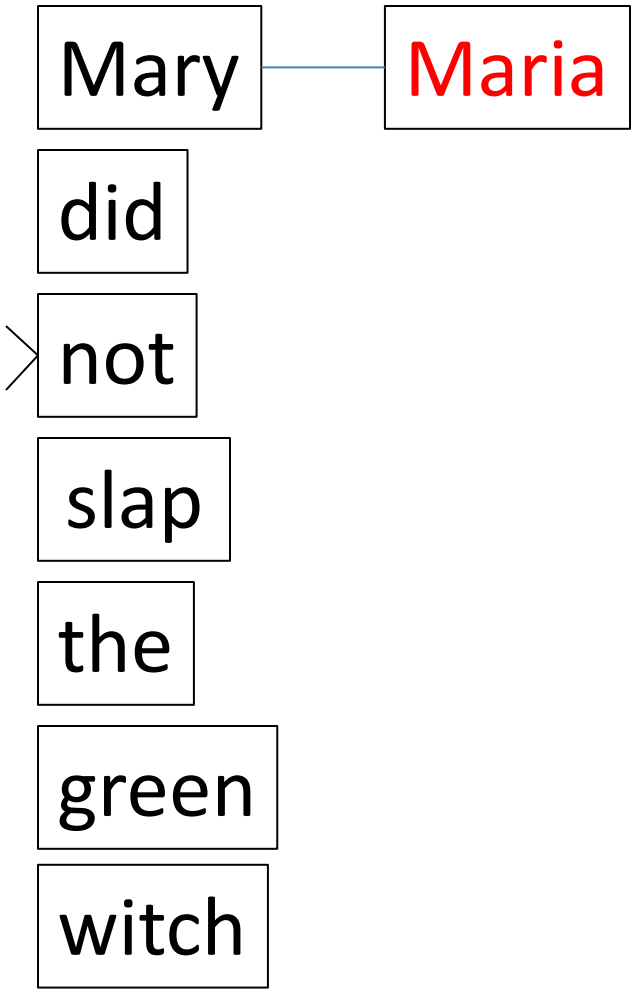
String Transduction for MT



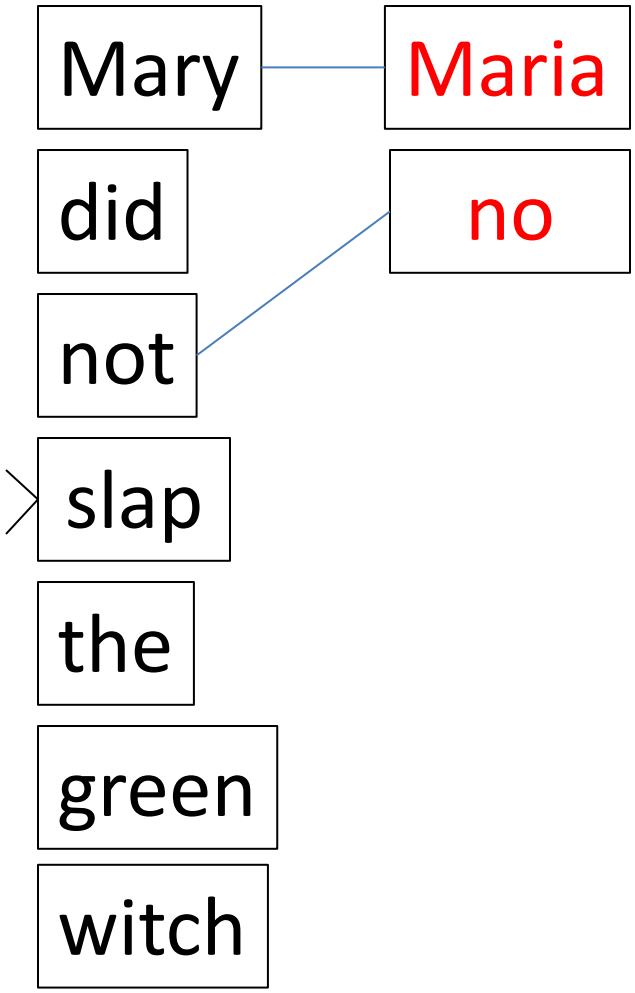
String Transduction for MT



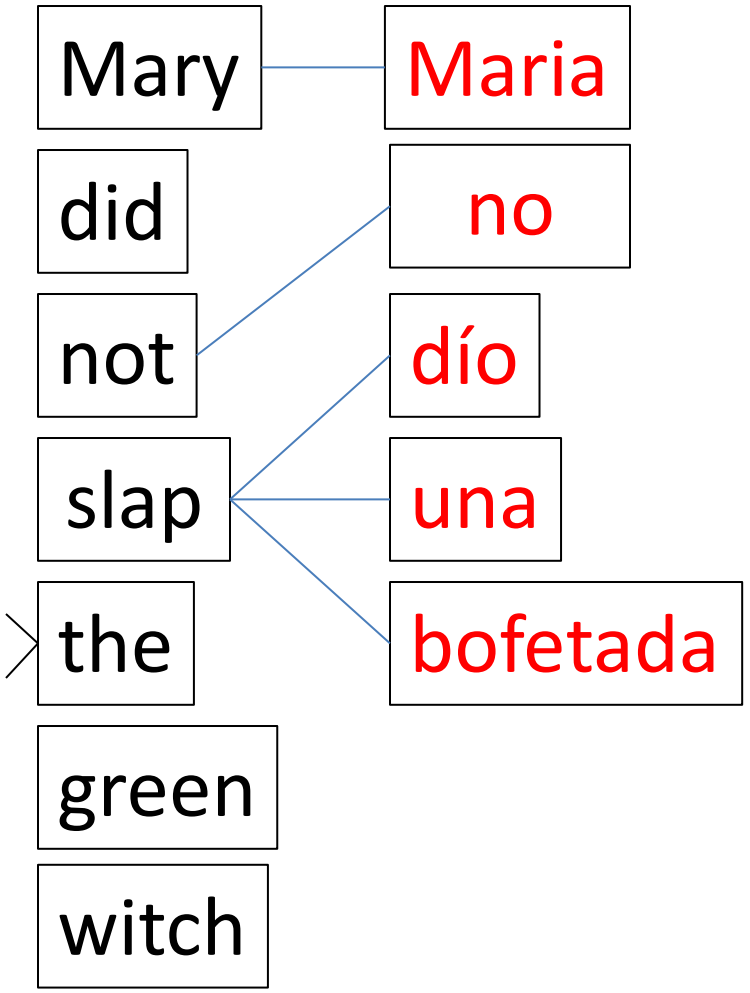
String Transduction for MT



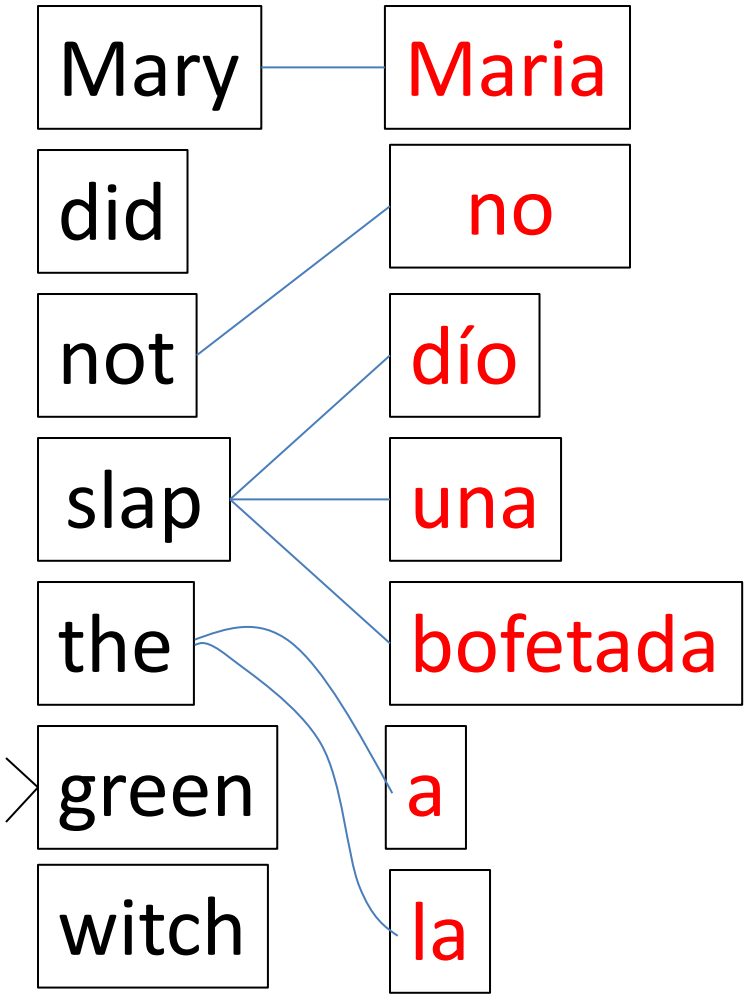
String Transduction for MT



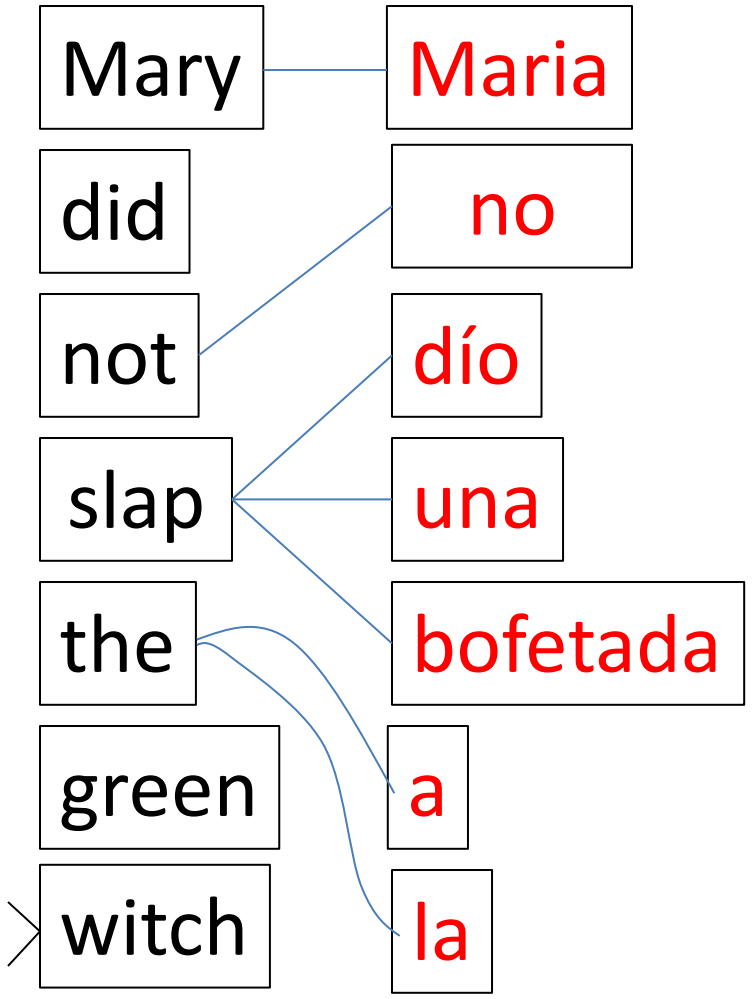
String Transduction for MT



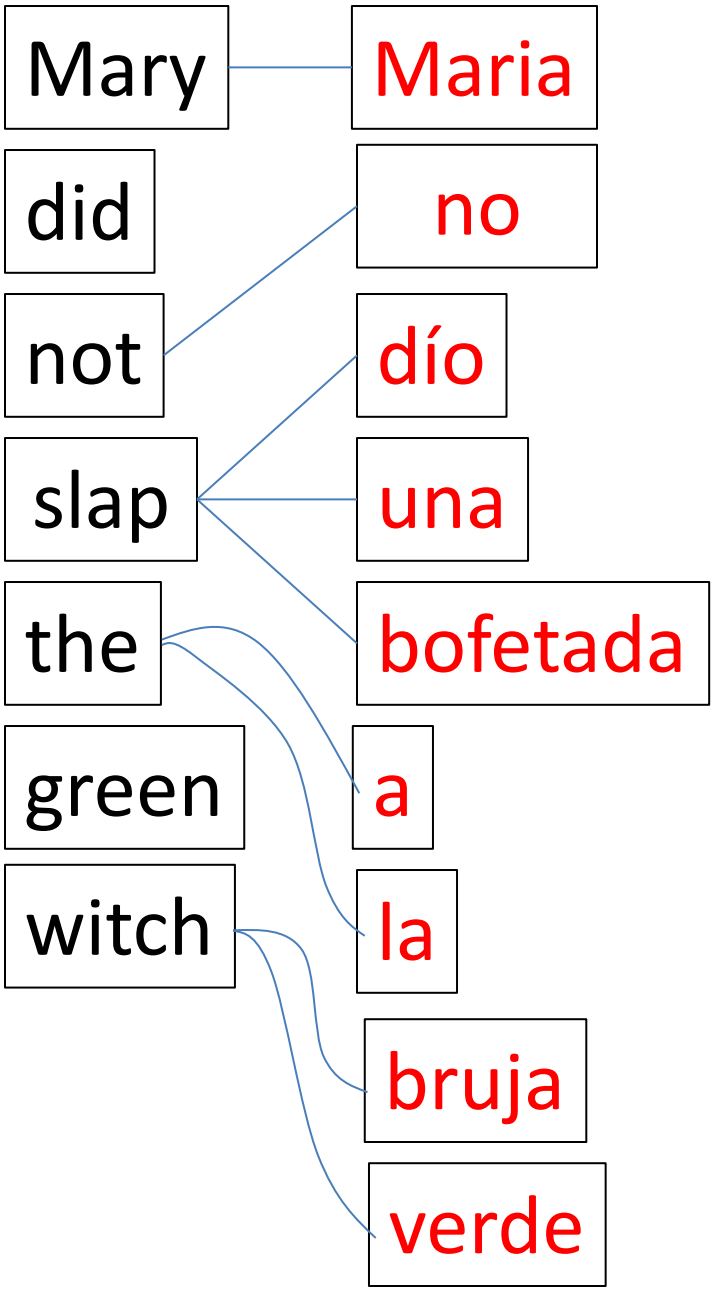
String Transduction for MT



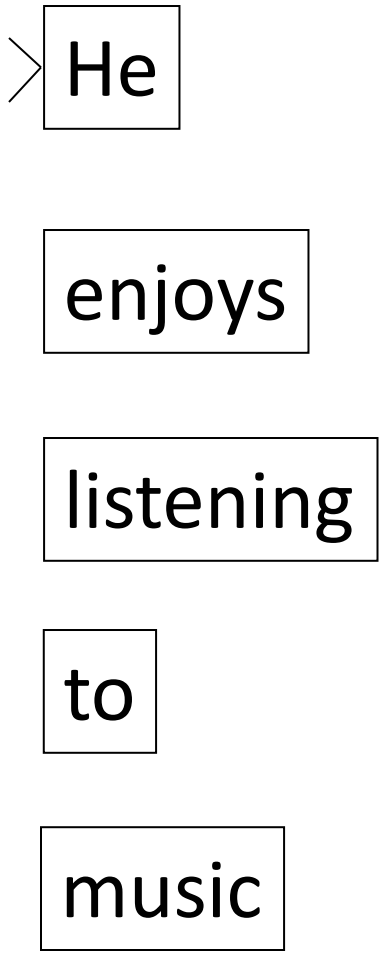
String Transduction for MT



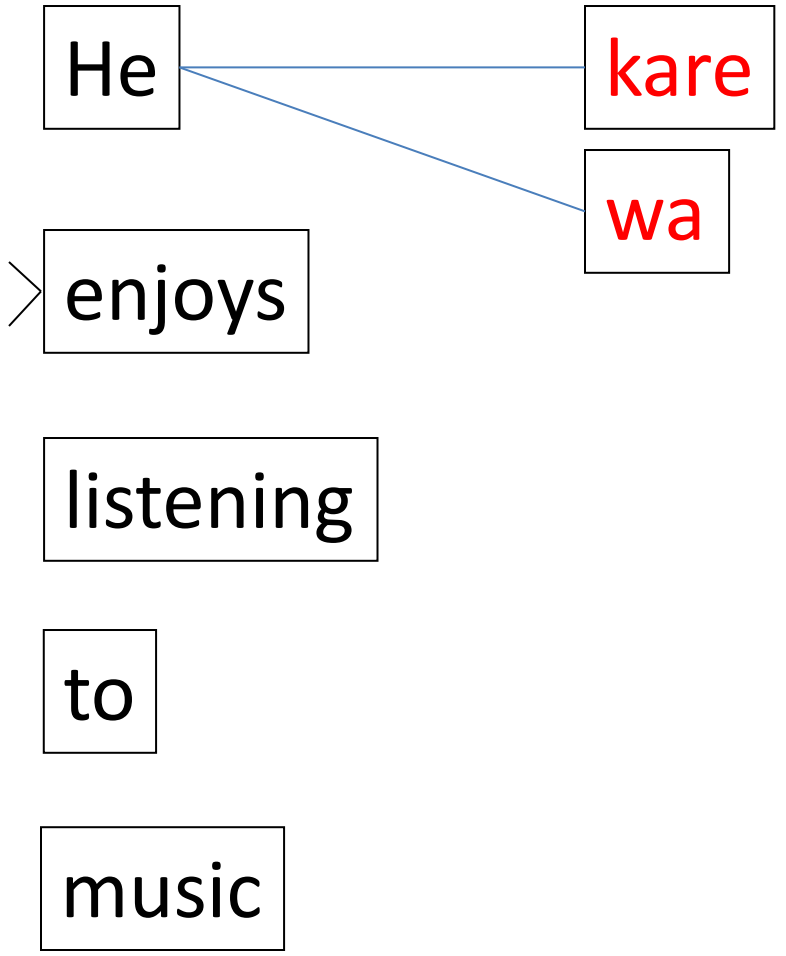
String Transduction for MT



String Transduction for MT

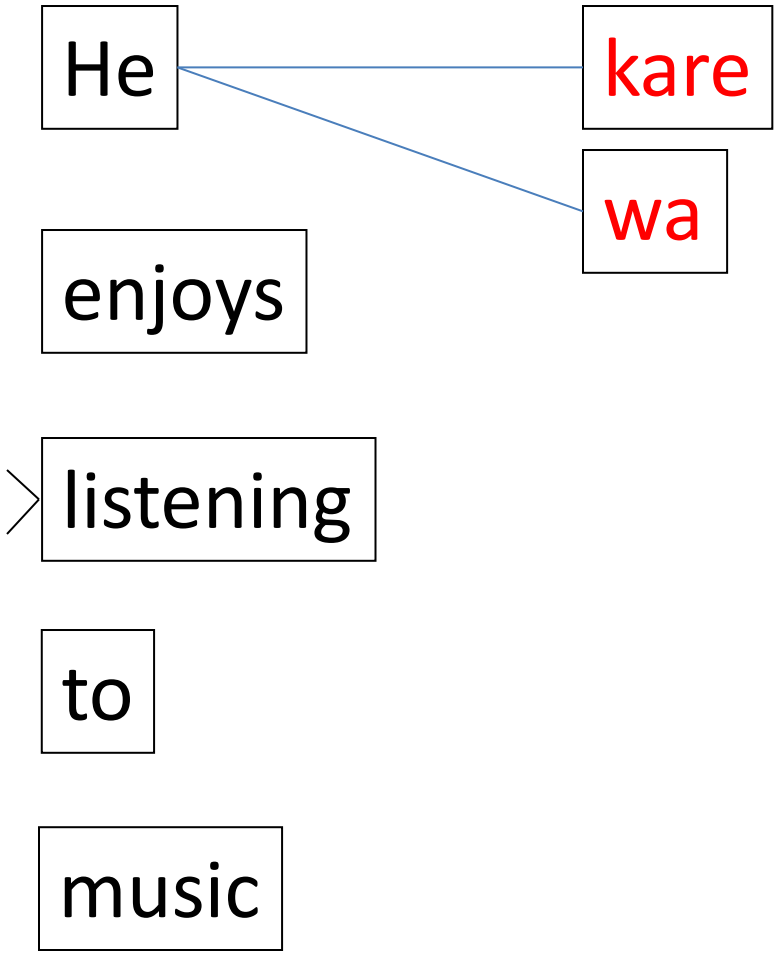


String Transduction for MT

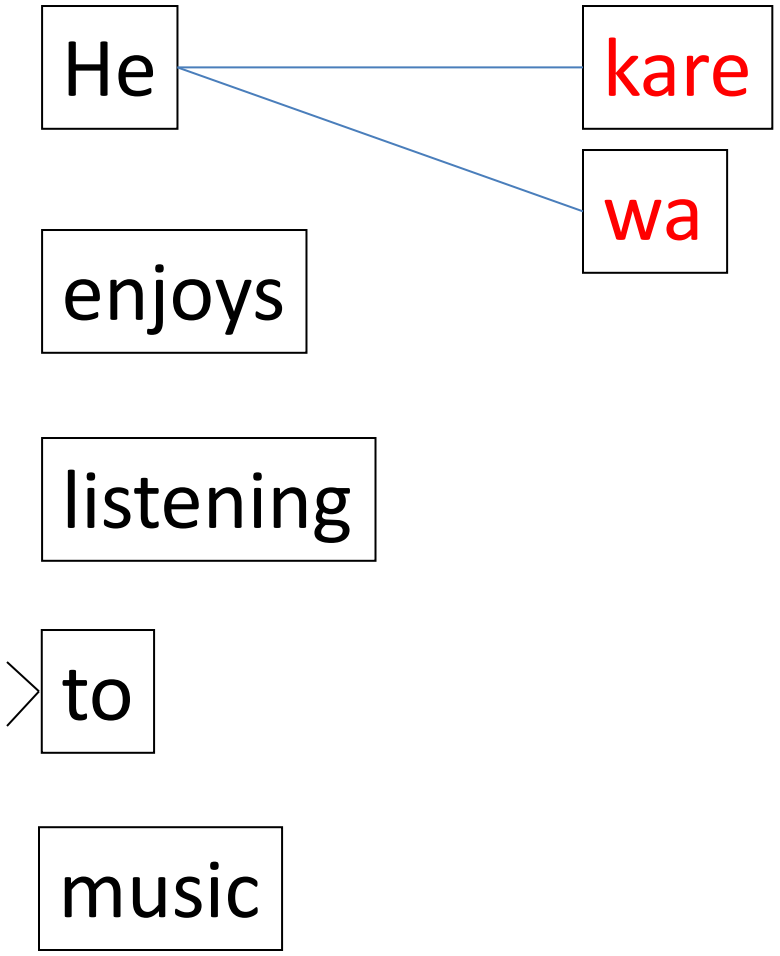


String Transduction for MT

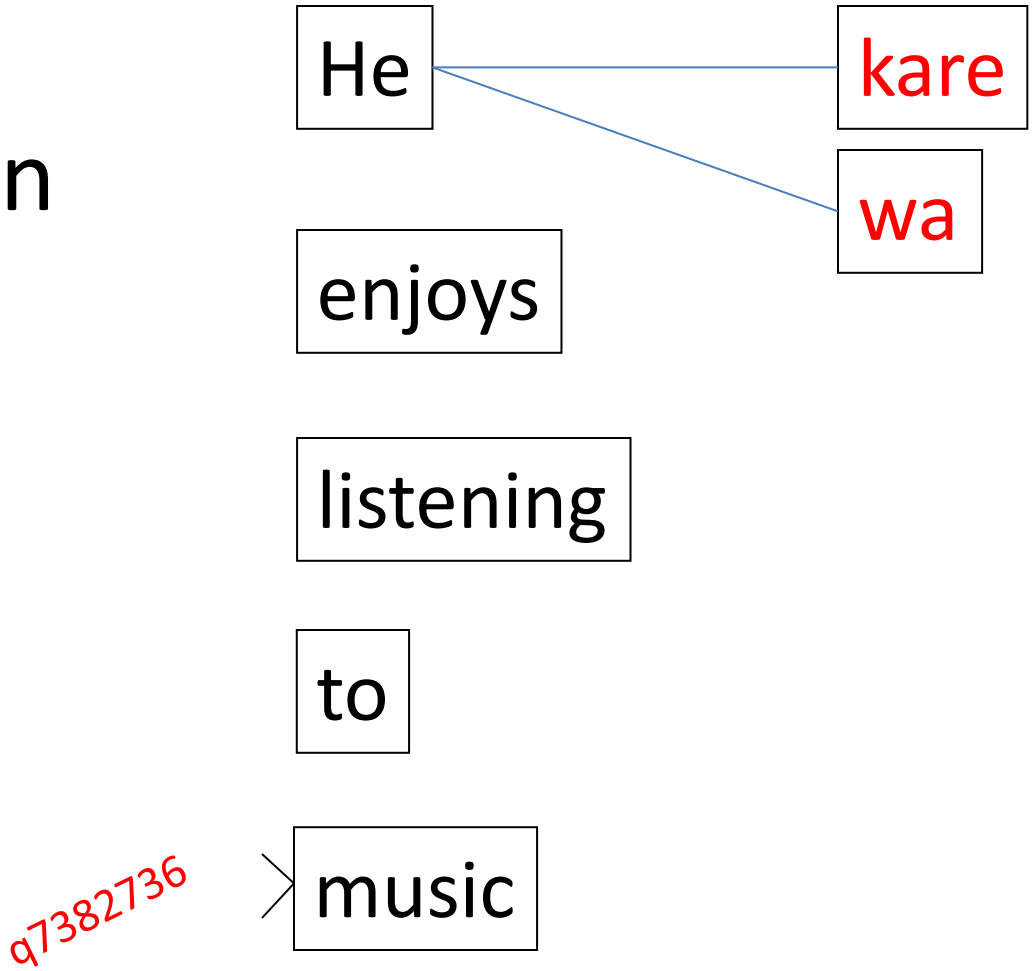
String Transduction for MT



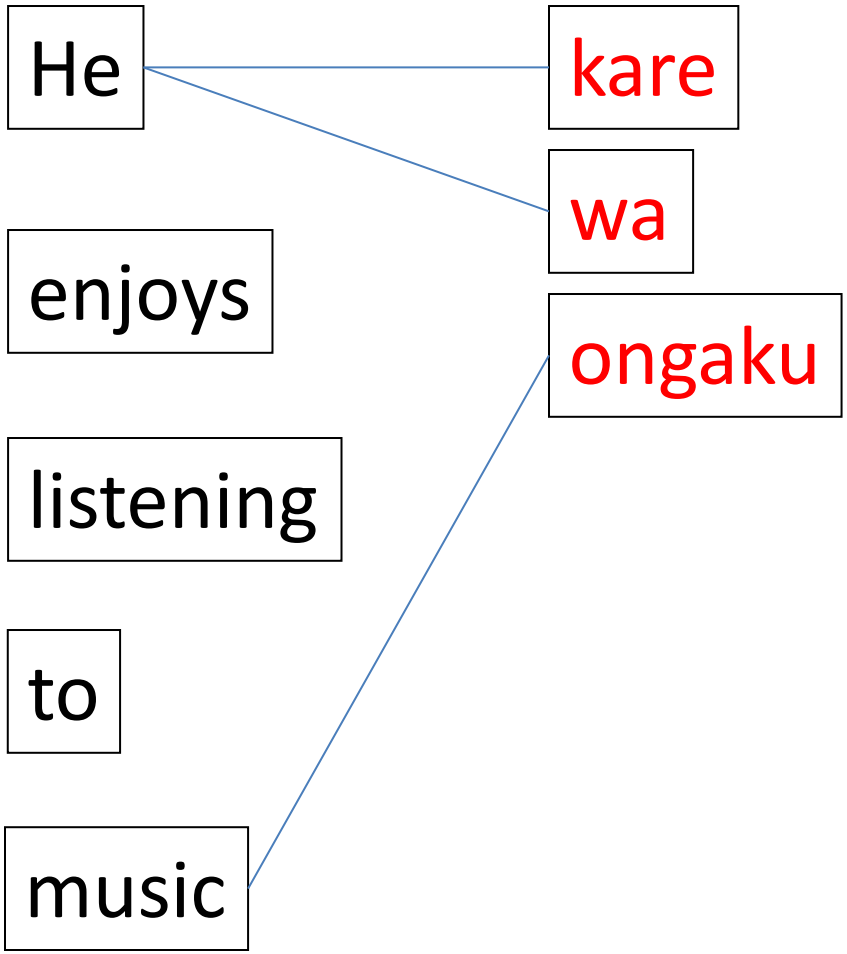
String Transduction for MT



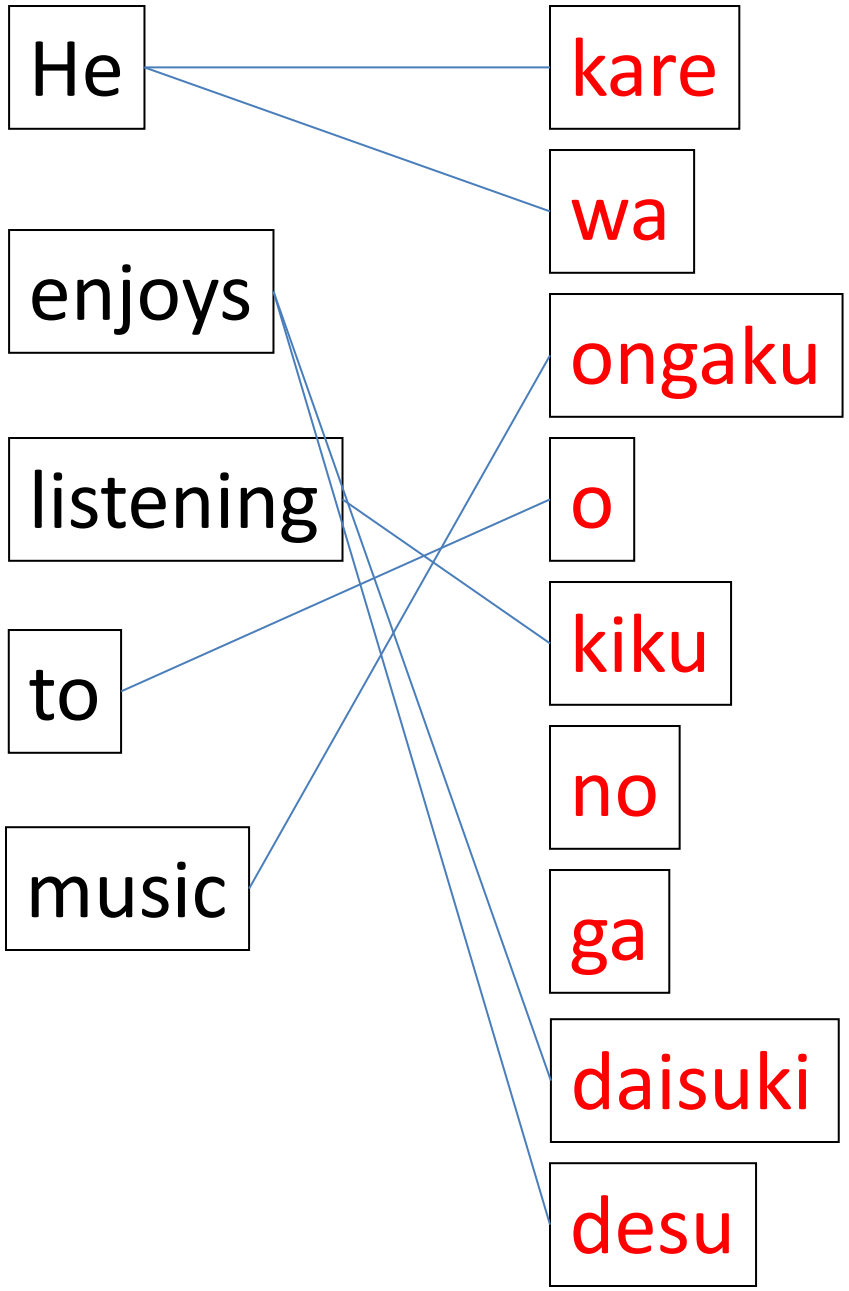
String Transduction for MT



String Transduction for MT



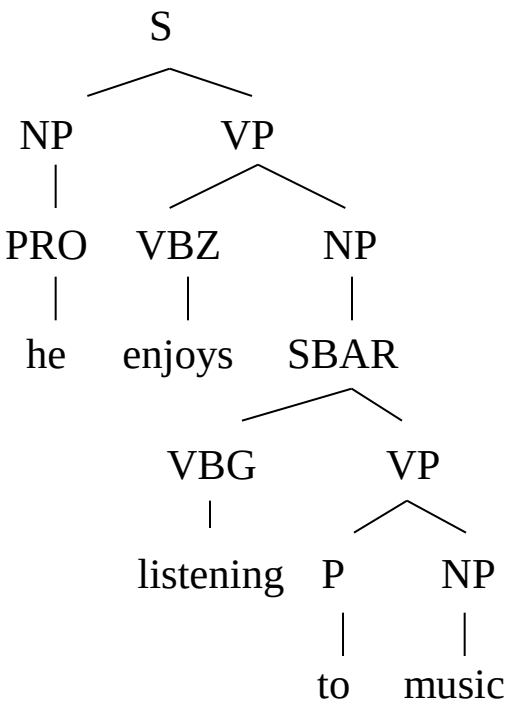
String Transduction for MT

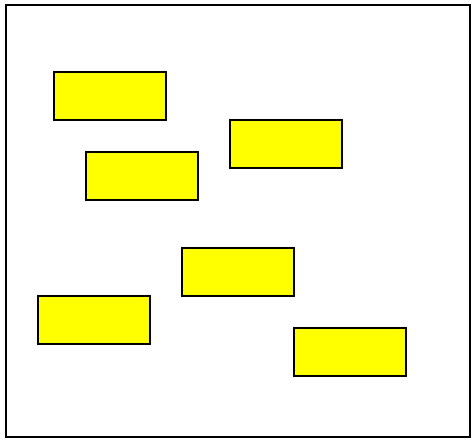


Top-Down Tree Transducer

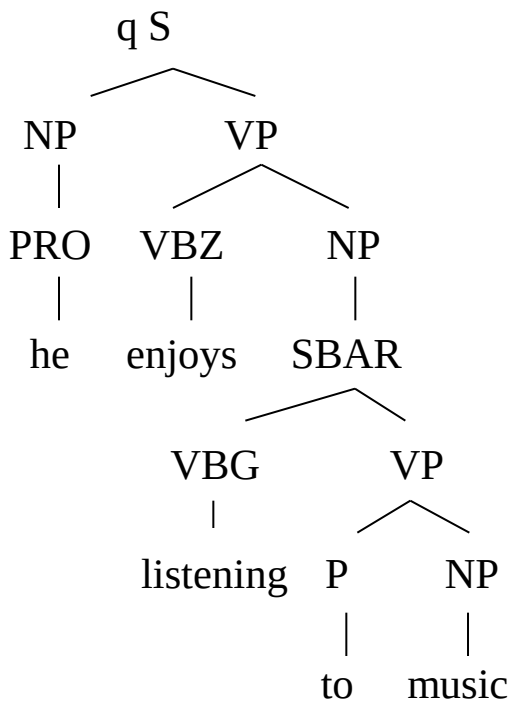
(W. Rounds 1970; J. Thatcher 1970)

Original input:



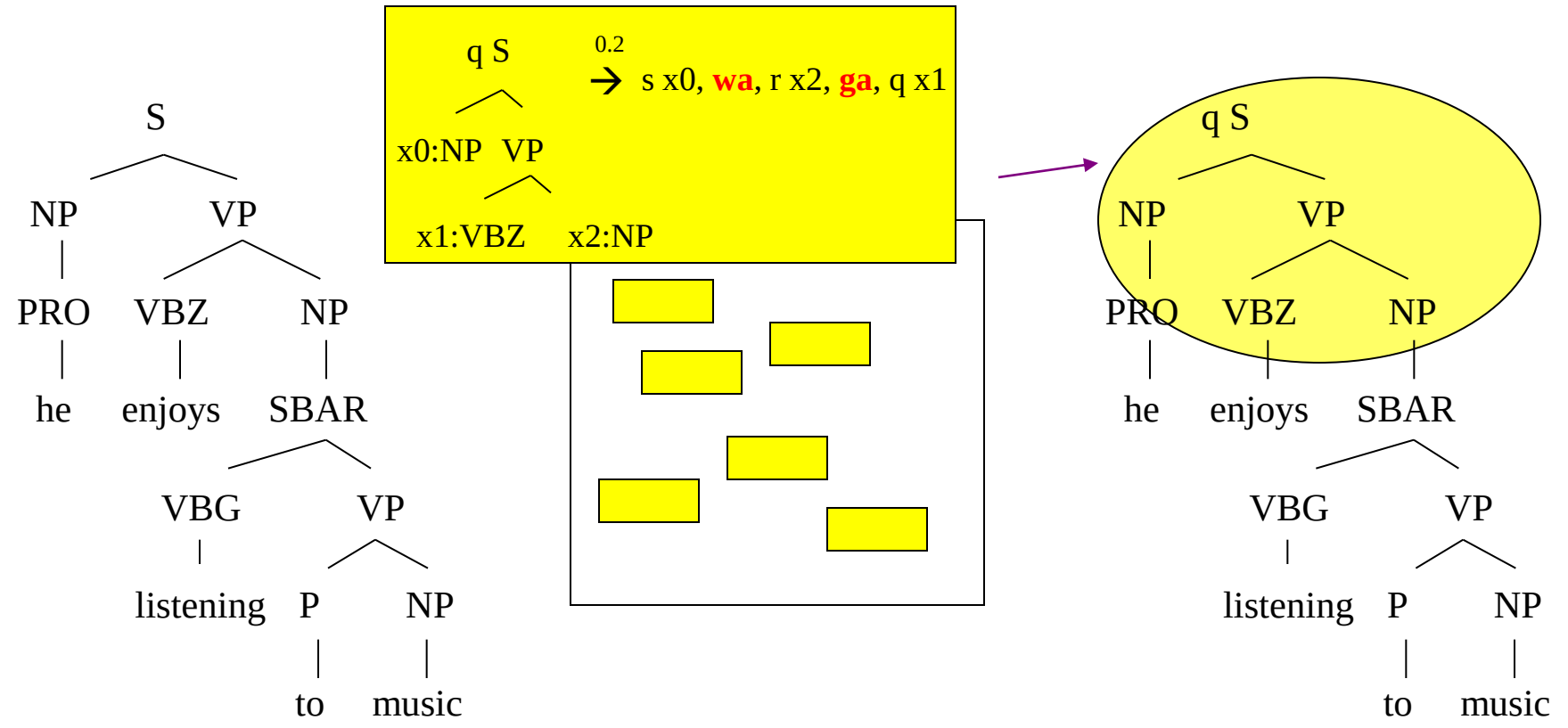


Transformation:



Top-Down Tree Transducer

(W. Rounds 1970; J. Thatcher 1970)

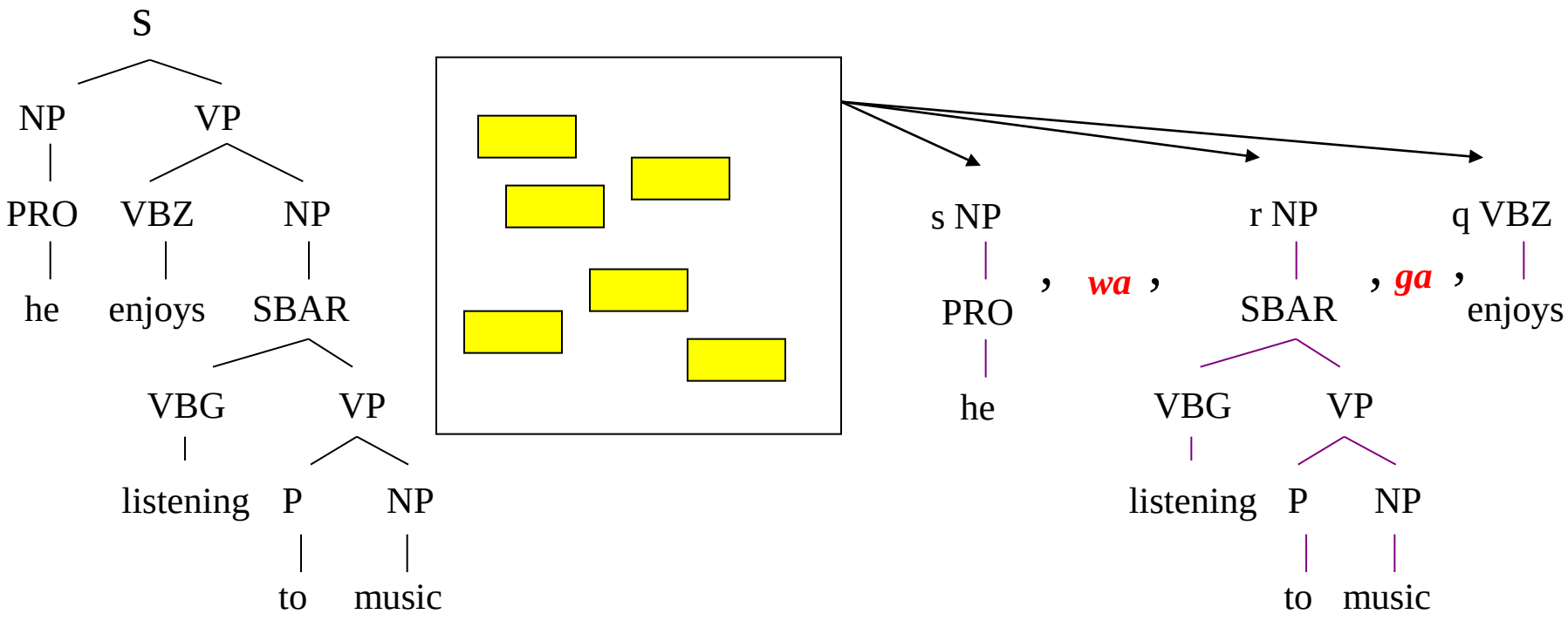


Top-Down Tree Transducer

(W. Rounds 1970; J. Thatcher 1970)

Original input:

Transformation:

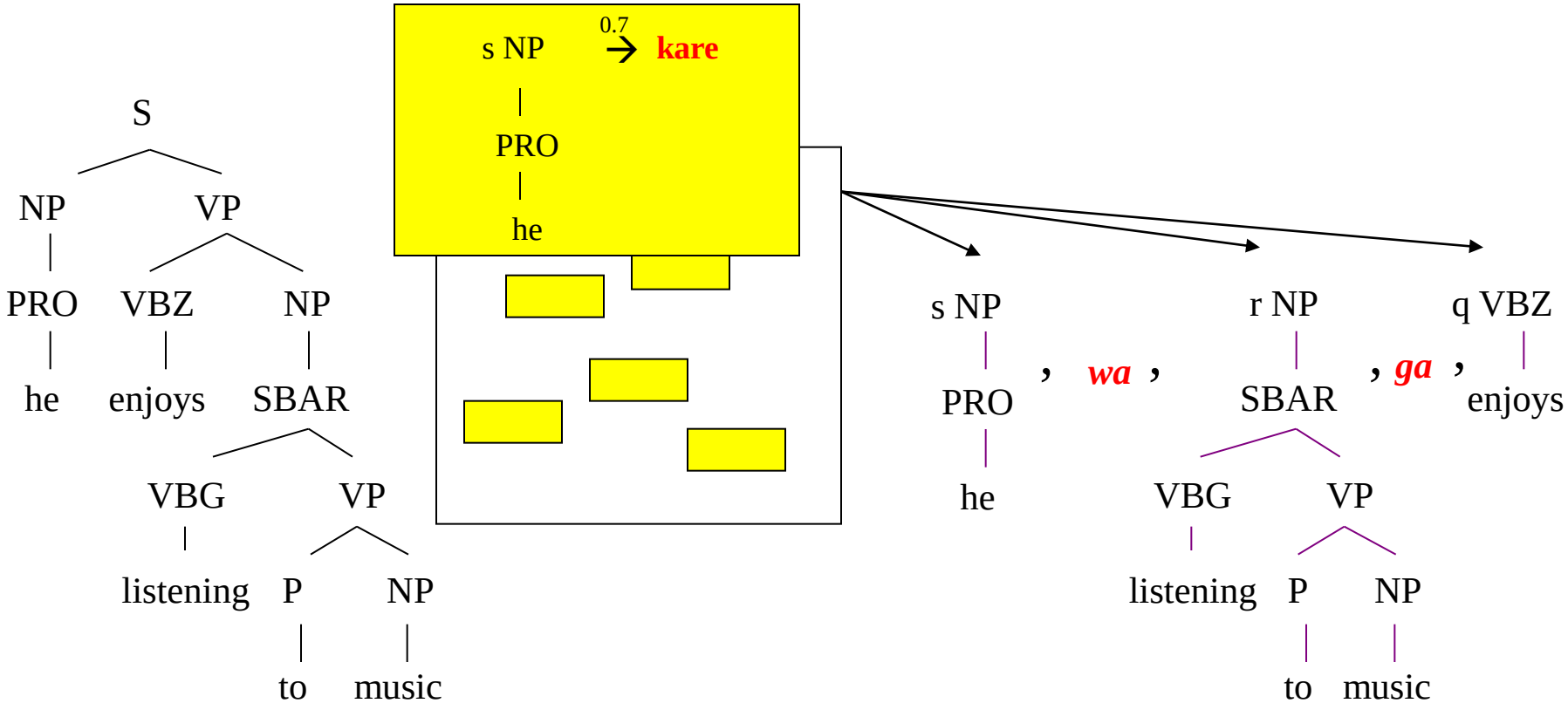


Top-Down Tree Transducer

(W. Rounds 1970; J. Thatcher 1970)

Original input:

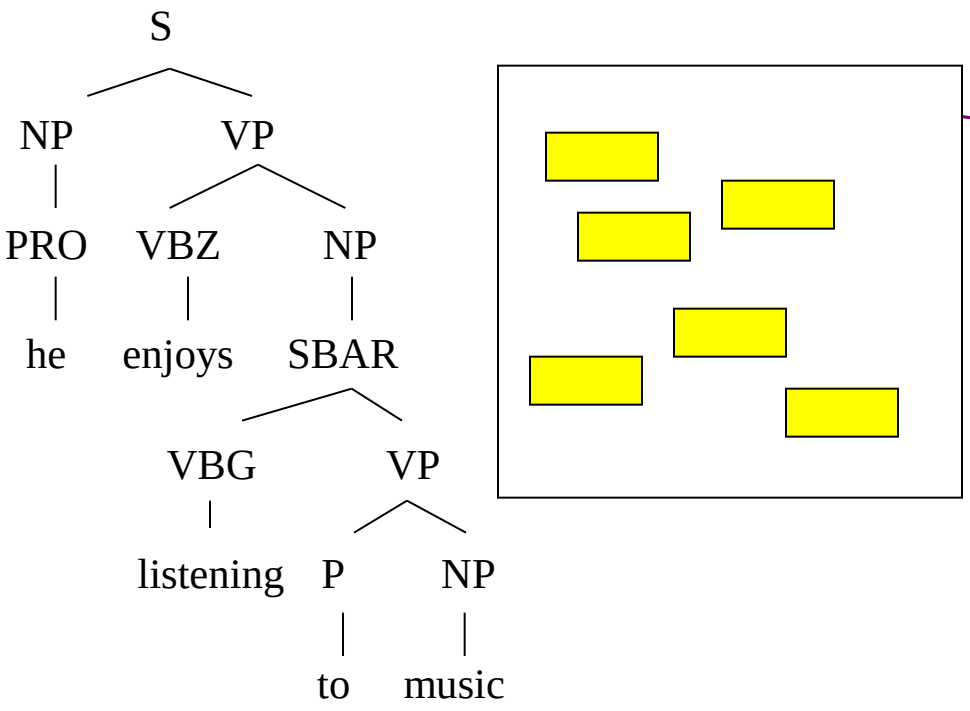
Transformation:



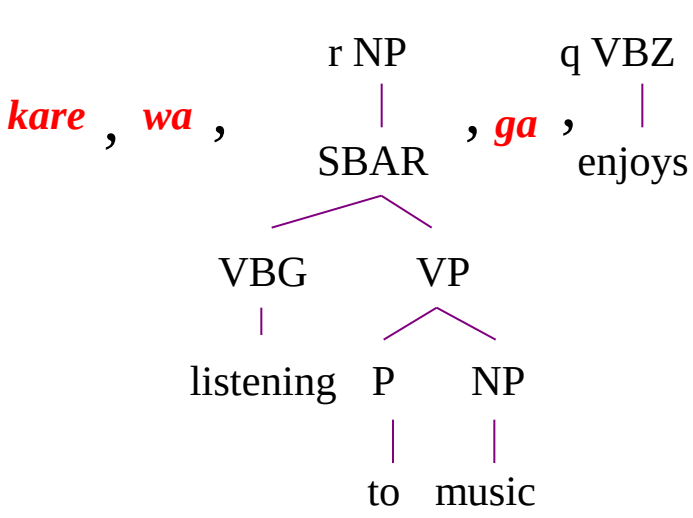
Top-Down Tree Transducer

(W. Rounds 1970; J. Thatcher 1970)

Original input:



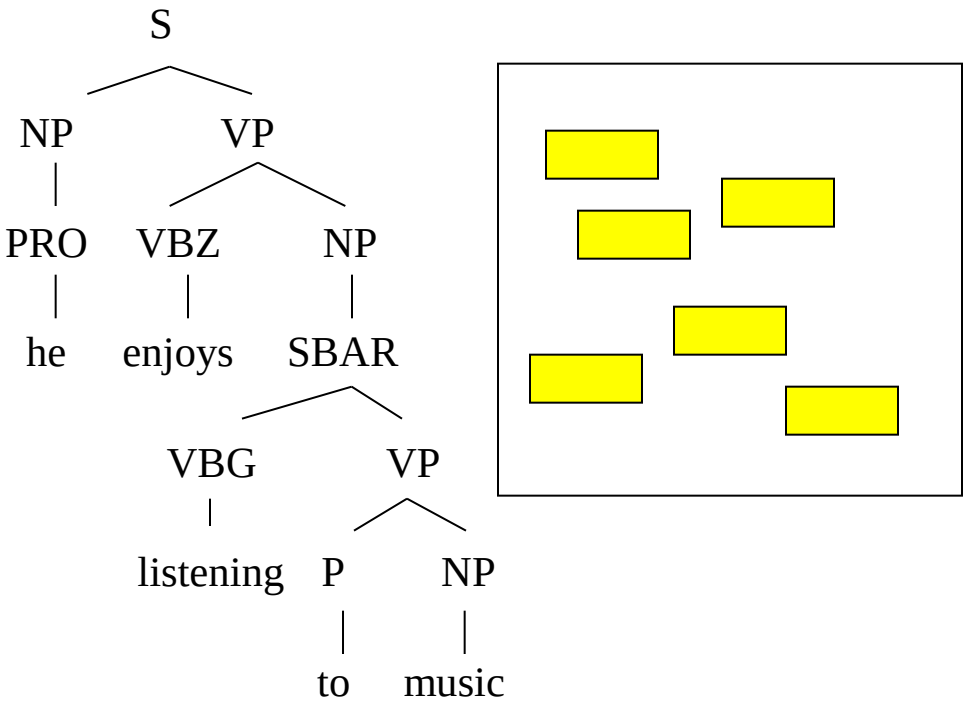
Transformation:



Top-Down Tree Transducer

(W. Rounds 1970; J. Thatcher 1970)

Original input:



Final output:

kare **,** ***wa*** **,** ***ongaku*** **,** ***o*** **,** ***kiku*** **,** ***no*** **,** ***ga*** **,** ***daisuki*** **,** ***desu***

Top-Down Tree Transducer

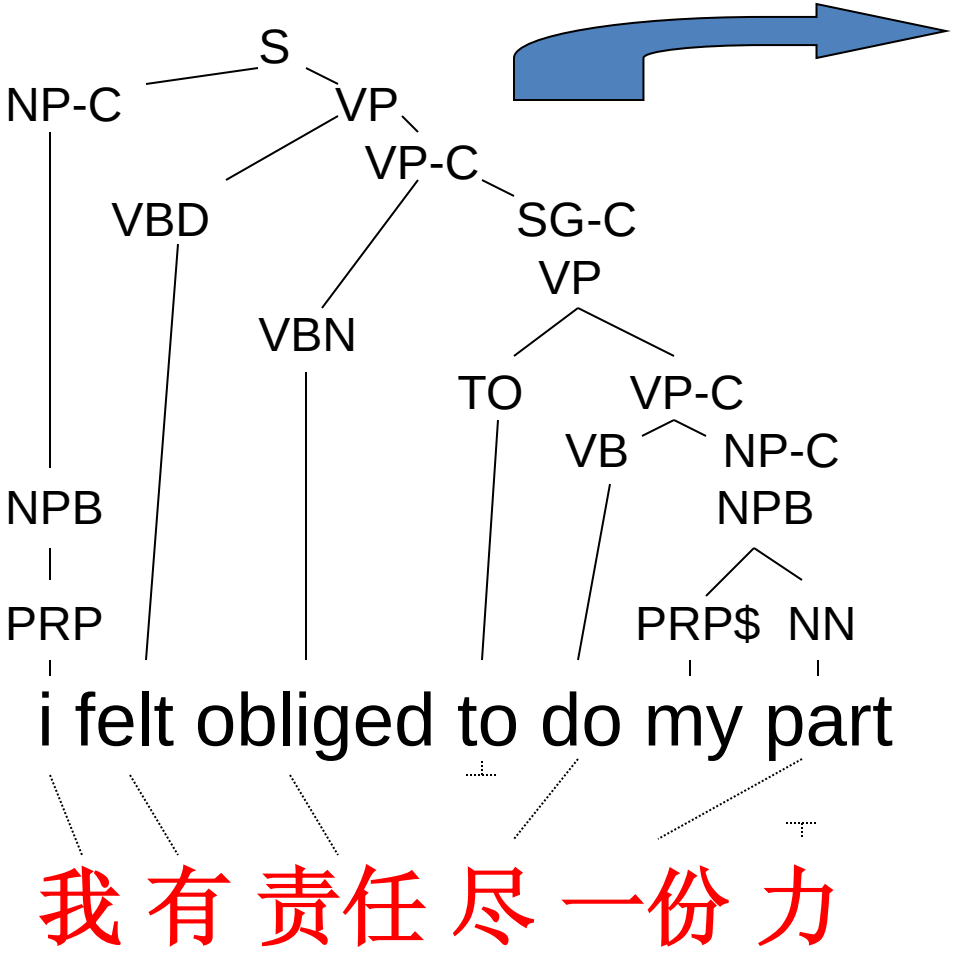
- Introduced by Rounds (1970) & Thatcher (1970)

  "Recent developments in the theory of automata have pointed to an extension of the domain of definition of automata from strings to trees … parts of **mathematical linguistics can be formalized easily** in a tree-automaton setting …"

  (Rounds 1970, "Mappings on Grammars and Trees", *Math. Systems Theory* 4(3))

- Large theory literature
  - e.g., Gécseg & Steinby (1984), Comon et al (1997)
- Re-connecting with NLP practice
  - e.g., Knight & Graehl (2005), Galley et al (2004, 2006), May & Knight (2006, 2010), Maletti et al (2009)

Tree Transducers Can be Extracted from Bilingual Data (Galley et al, 04)



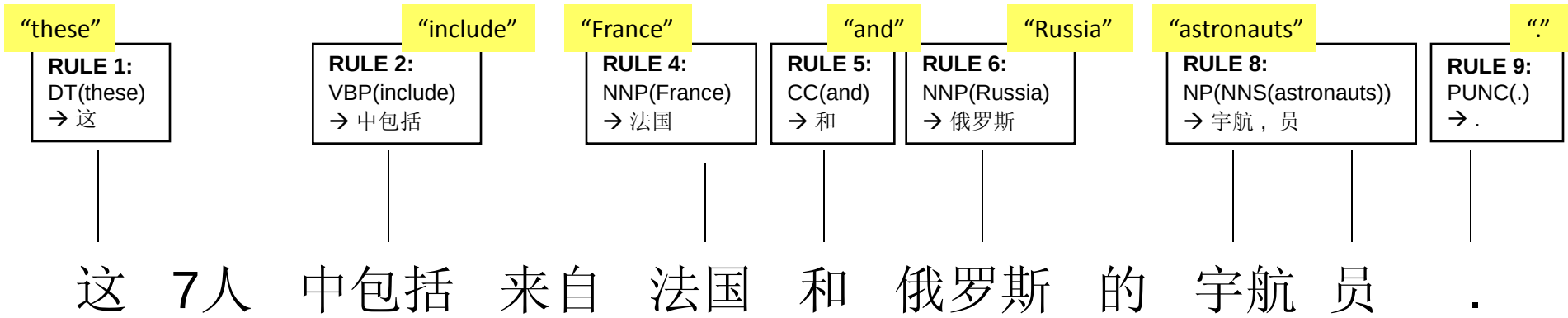
TREE TRANSDUCER RULES:

VBD(felt) → 有
VBN(obliged) → 责任
VB(do) → 尽
NN(part) → 一份
NN(part) → 一份 力
VP-C(x0:VBN x1:SG-C) → x0 x1
VP(TO(to) x0:VP-C) → x0
…
S(x0:NP-C x1:VP) → x0 x1

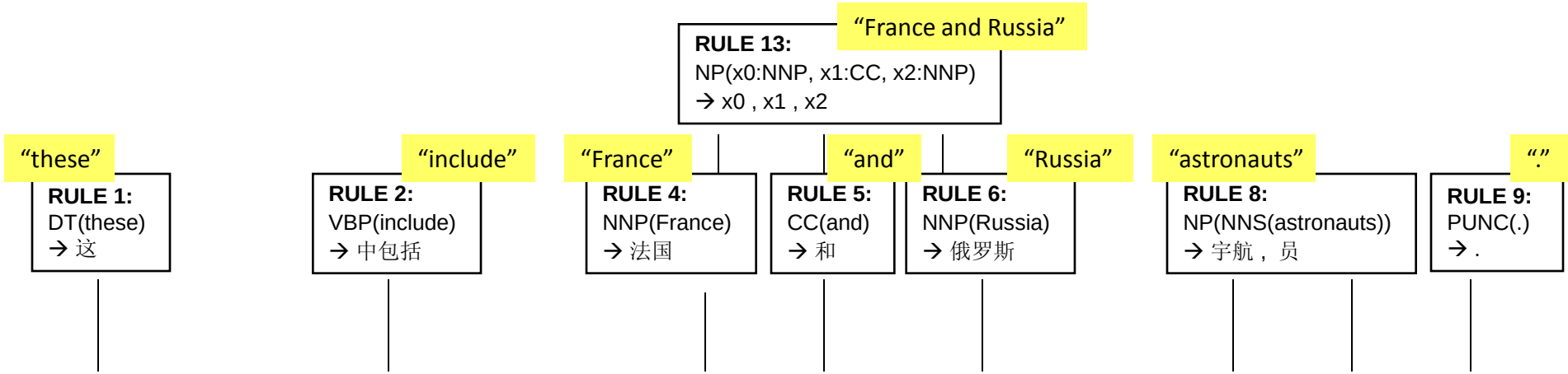
Syntax-Based Decoding

这 7人 中包括 来自 法国 和 俄罗斯 的 宇航 员 .

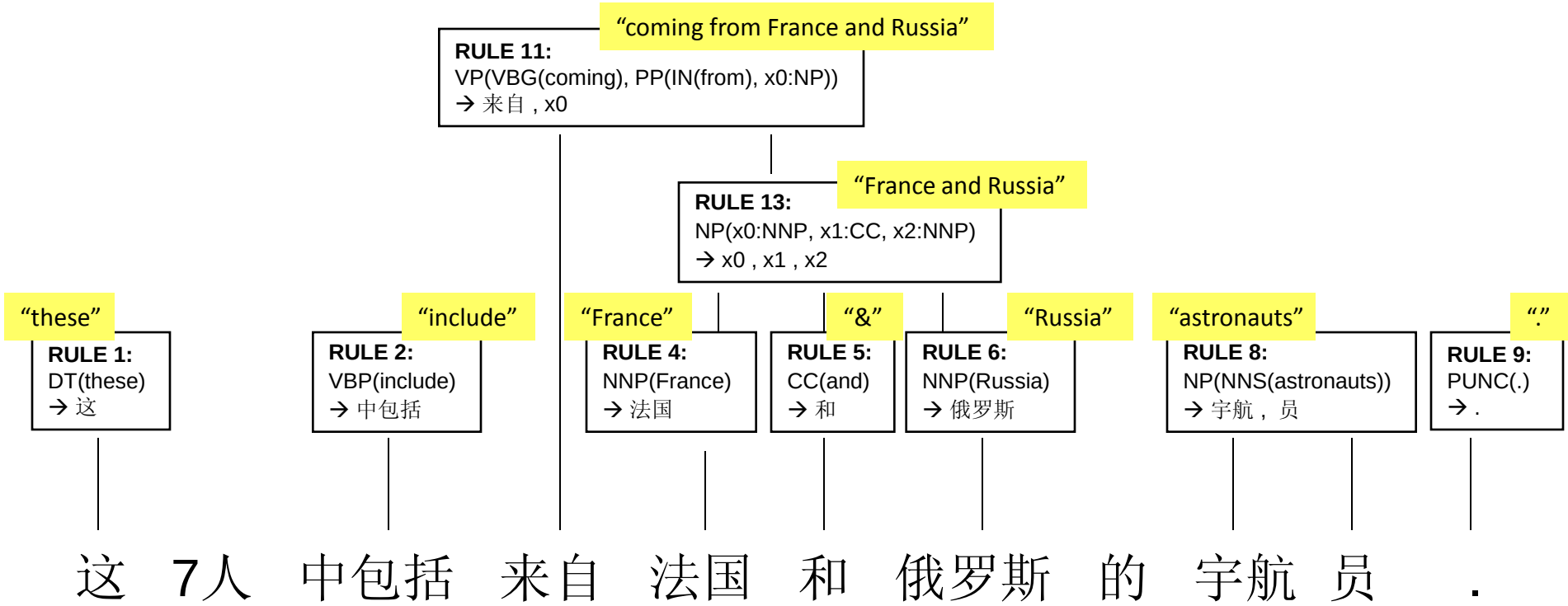
Syntax-Based Decoding



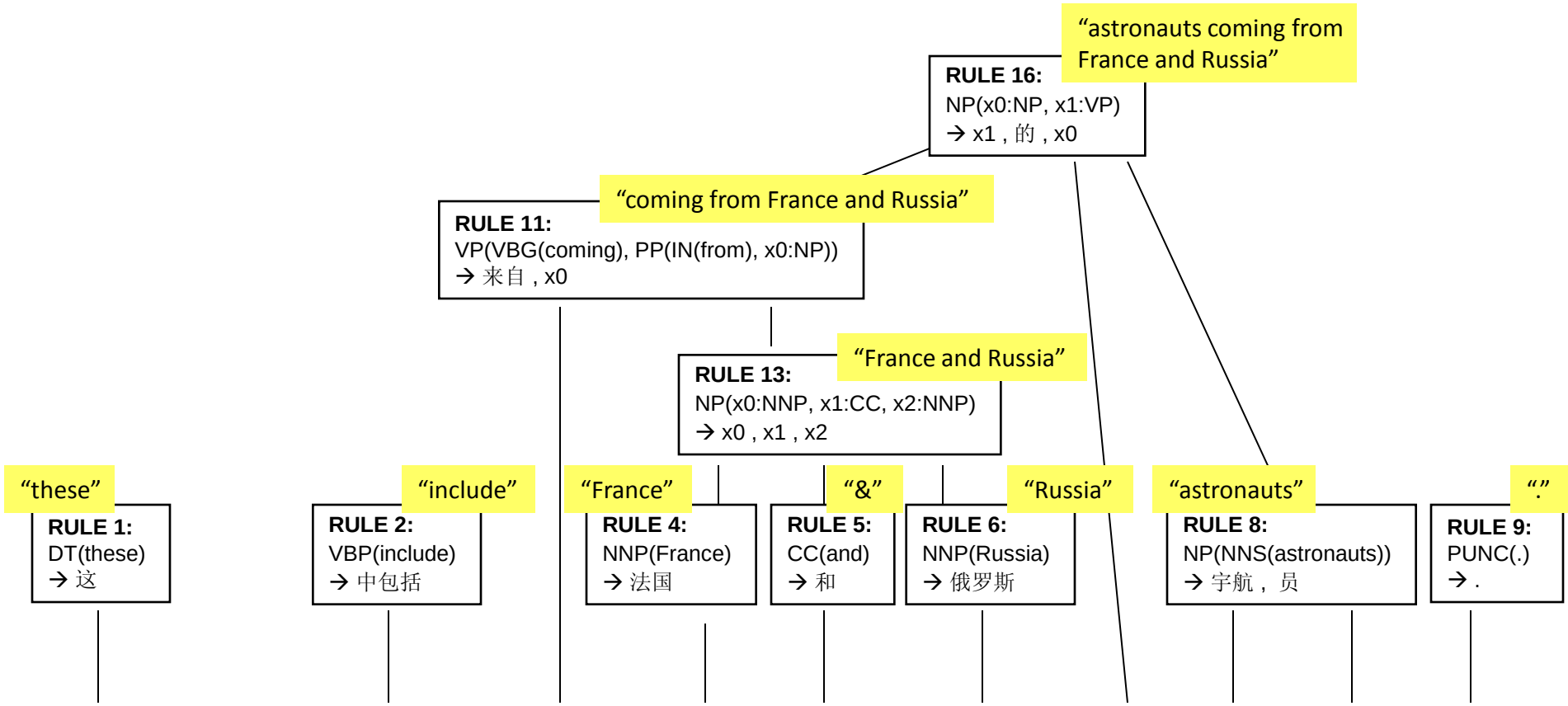
Syntax-Based Decoding

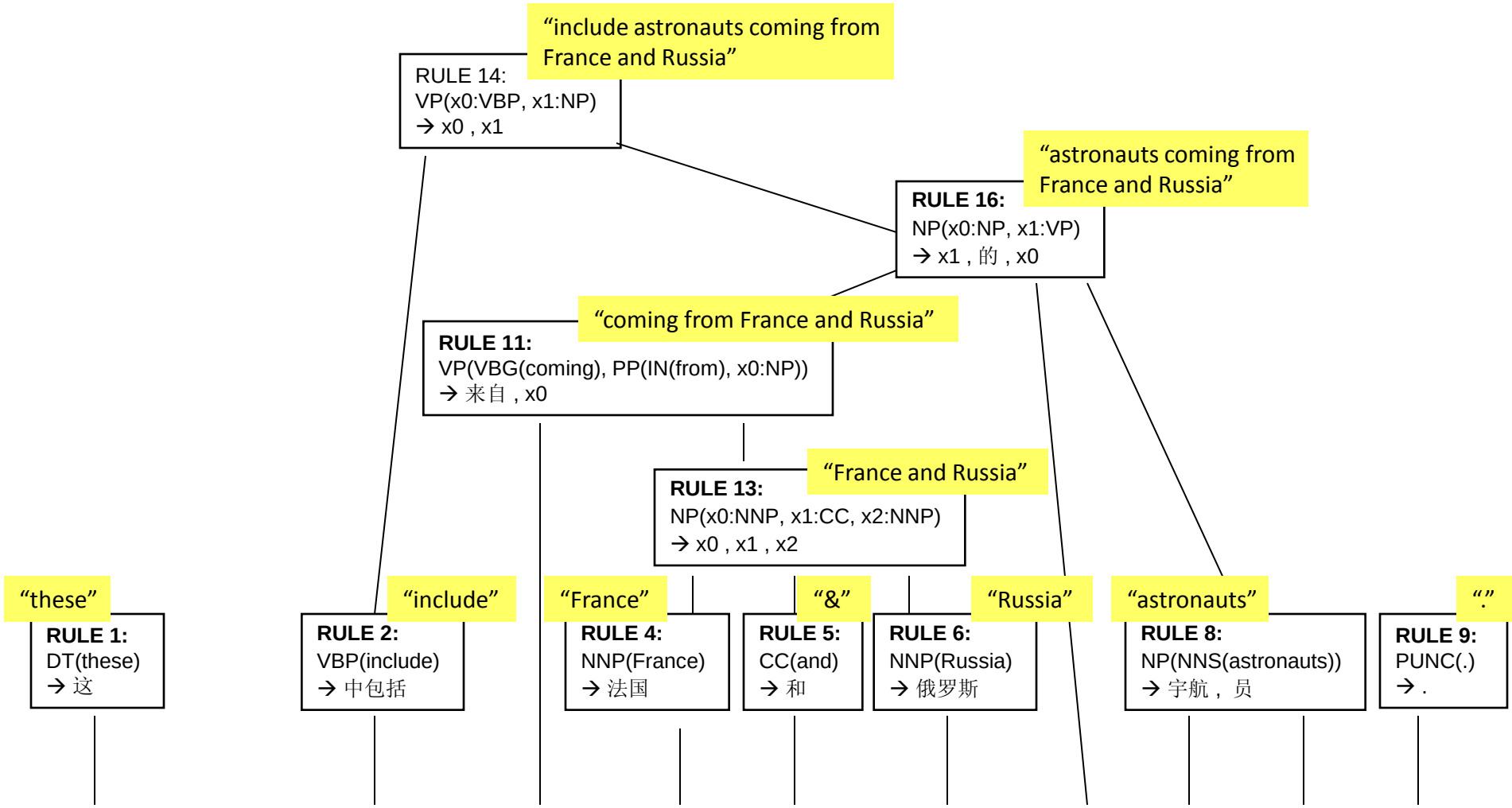


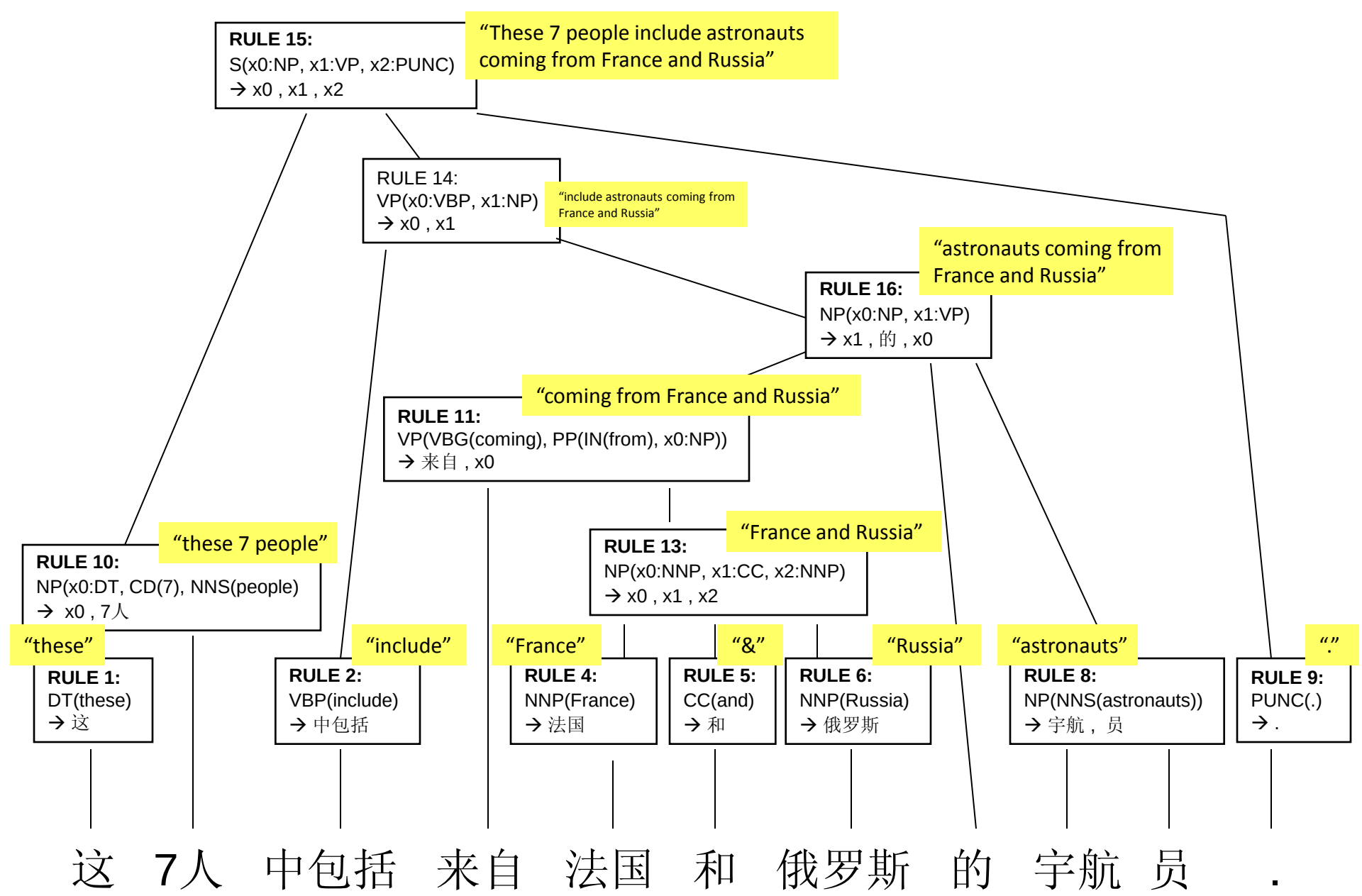
Syntax-Based Decoding



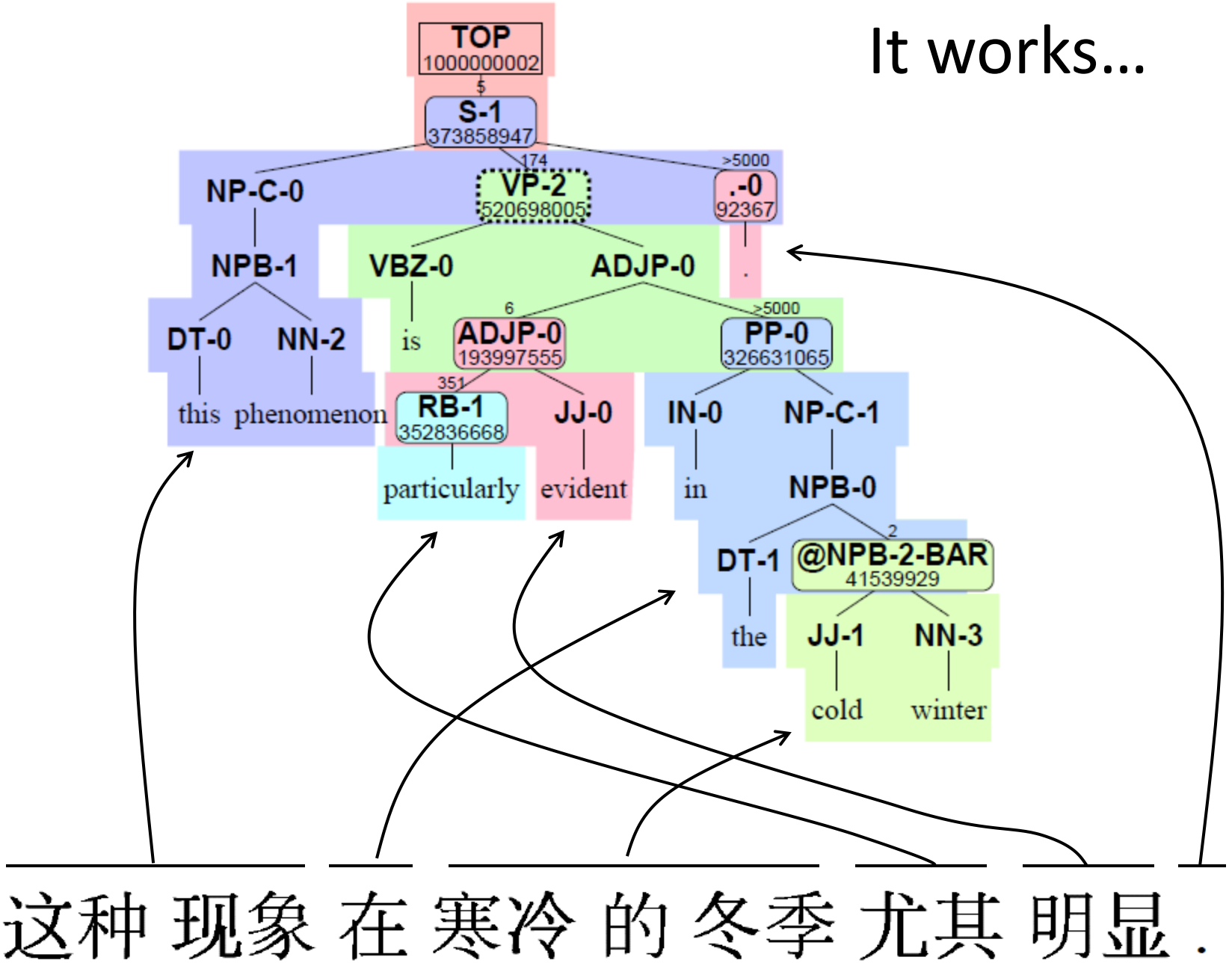
Syntax-Based Decoding



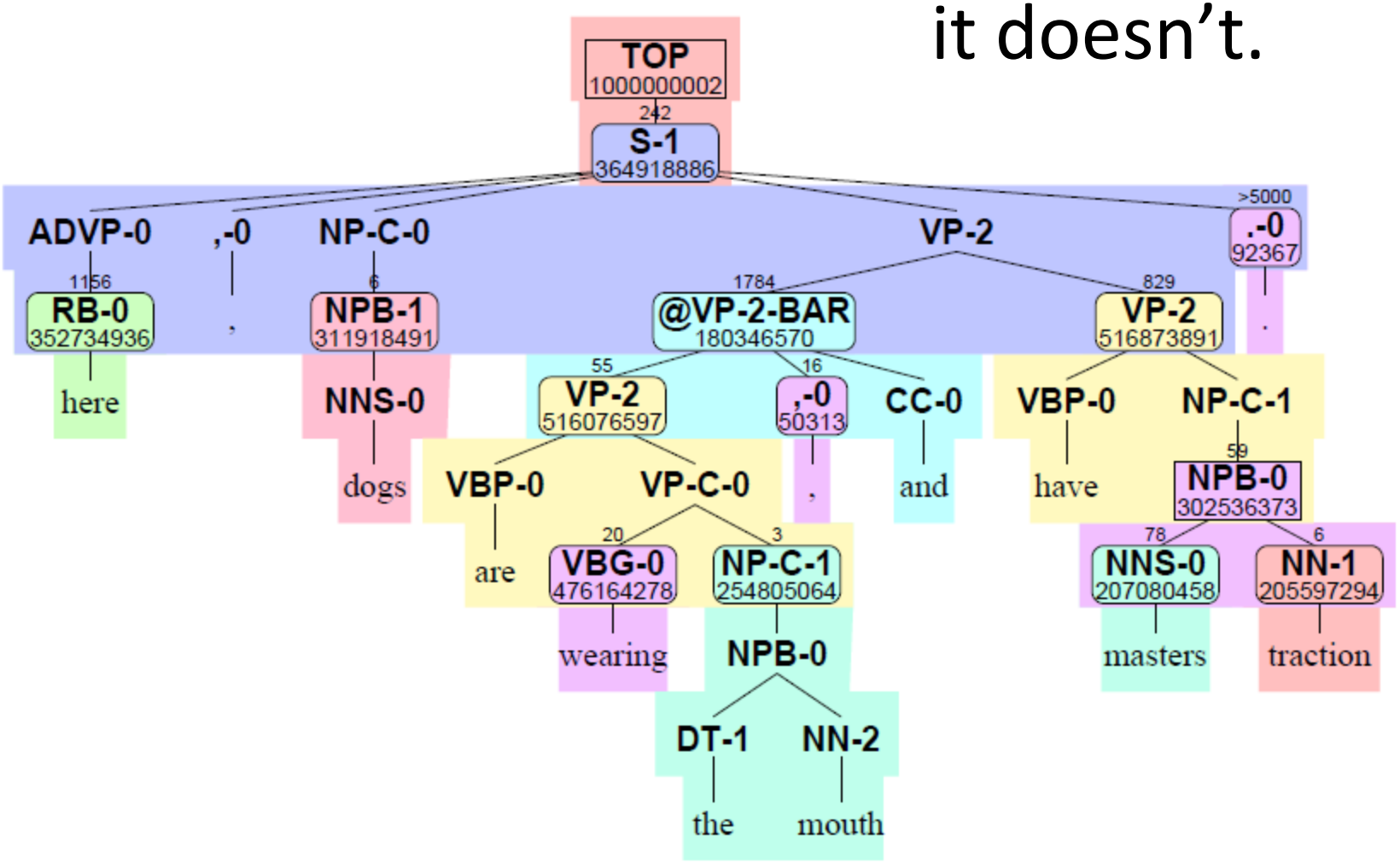

"include astronauts coming from France and Russia"
RULE 14:
VP(x0:VBP, x1:NP)
→ x0 , x1
"astronauts coming from France and Russia"
RULE 16:
NP(x0:NP, x1:VP)
→ x1 , 的 , x0
"coming from France and Russia"
RULE 11:
VP(VBG(coming), PP(IN(from), x0:NP))
→ 来自 , x0
"France and Russia"
RULE 13:
NP(x0:NNP, x1:CC, x2:NNP)
→ x0 , x1 , x2
"these"
RULE 1:
DT(these)
→ 这
"include"
RULE 2:
VBP(include)
→ 中包括
"France"
RULE 4:
NNP(France)
→ 法国
"&"
RULE 5:
CC(and)
→ 和
"Russia"
RULE 6:
NNP(Russia)
→ 俄罗斯
"astronauts"
RULE 8:
NP(NNS(astronauts))
→ 宇航 , 员
"."
RULE 9:
PUNC(.)
→ .
这 7人 中包括 来自 法国 和 俄罗斯 的 宇航 员 .


RULE 15:
S(x0:NP, x1:VP, x2:PUNC)
→ x0 , x1 , x2
“These 7 people include astronauts coming from France and Russia”
RULE 14:
VP(x0:VBP, x1:NP)
→ x0 , x1
“include astronauts coming from France and Russia”
RULE 16:
NP(x0:NP, x1:VP)
→ x1 , 的 , x0
“astronauts coming from France and Russia”
RULE 11:
VP(VBG(coming), PP(IN(from), x0:NP))
→ 来自 , x0
“coming from France and Russia”
RULE 13:
NP(x0:NNP, x1:CC, x2:NNP)
→ x0 , x1 , x2
“France and Russia”
RULE 10:
NP(x0:DT, CD(7), NNS(people)
→ x0 , 7人
“these 7 people”
“these”
RULE 1:
DT(these)
→ 这
“include”
RULE 2:
VBP(include)
→ 中包括
“France”
RULE 4:
NNP(France)
→ 法国
“&”
RULE 5:
CC(and)
→ 和
“Russia”
RULE 6:
NNP(Russia)
→ 俄罗斯
“astronauts”
RULE 8:
NP(NNS(astronauts))
→ 宇航 , 员
“.”
RULE 9:
PUNC(.)
→ .
这 7人 中包括 来自 法国 和 俄罗斯 的 宇航 员 .

It works…

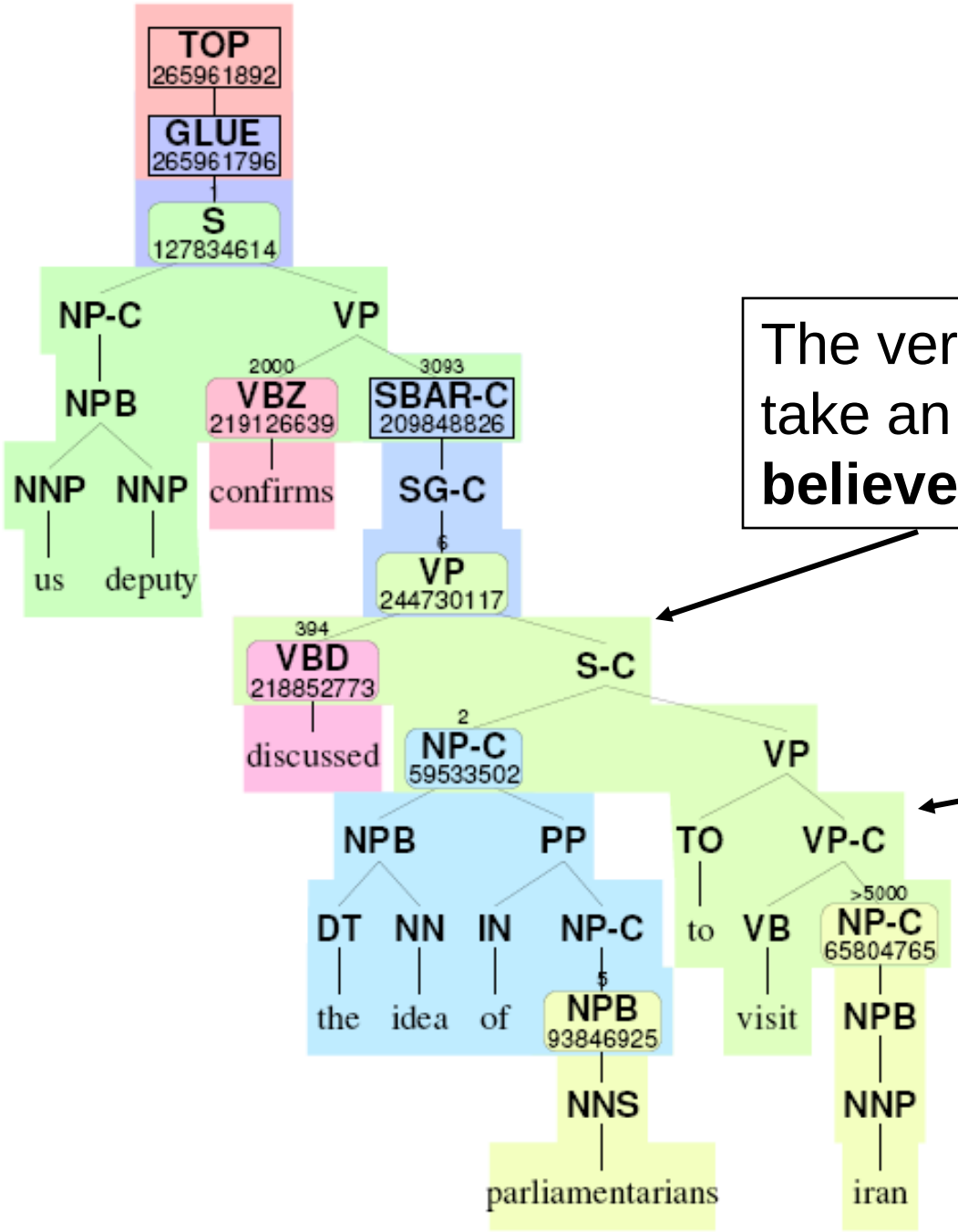

TOP
1000000002
5
S-1
373858947
NP-C-0
174
VP-2
520698005
>5000
.-0
92367
NPB-1
VBZ-0
ADJP-0
.
DT-0
NN-2
is
6
ADJP-0
193997555
>5000
PP-0
326631065
this
phenomenon
351
RB-1
352836668
JJ-0
IN-0
NP-C-1
particularly
evident
in
NPB-0
2
DT-1
@NPB-2-BAR
41539929
the
JJ-1
NN-3
cold
winter
这种 现象 在 寒冷 的 冬季 尤其 明显 .

…except when it doesn't.



在这里，狗都配戴嘴套，并有主人牵引。

…except when it doesn't.



The verb **discuss** doesn't take an S argument (like **believe** and **realize** do)

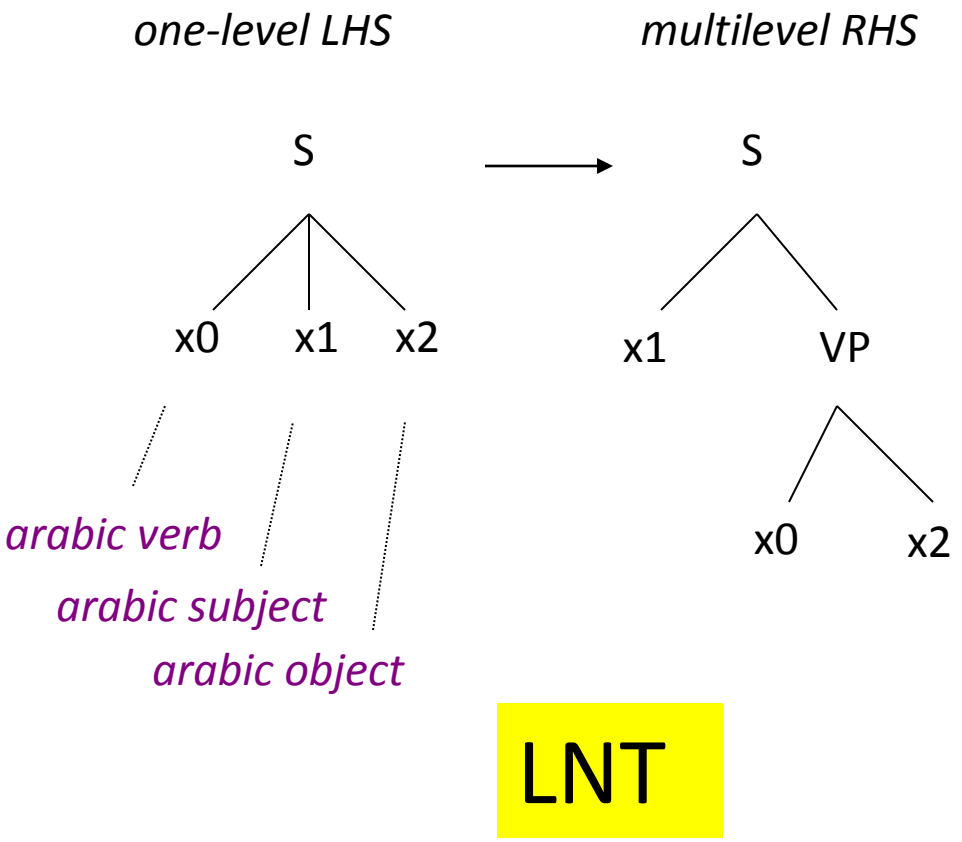
An **idea** can't **visit** a **place**

General-Purpose Algorithms for Tree Automata

	String Automata Algorithms	Tree Automata Algorithms
N-best …	… paths through an WFSA (Viterbi, 1967; Eppstein, 1998)	… trees in a weighted forest (Jiménez & Marzal, 2000; Huang & Chiang, 2005)
EM training	Forward-backward EM (Baum/Welch, 1971; Eisner 2003)	Tree transducer EM training (Graehl & Knight, 2004)
Determinization …	… of weighted string acceptors (Mohri, 1997)	… of weighted tree acceptors (Borchardt & Vogler, 2003; May & Knight, 2005)
Intersection	WFSA intersection	Tree acceptor intersection
Applying transducers	string → WFST → WFSA	tree → TT → weighted tree acceptor
Transducer composition	WFST composition (Pereira & Riley, 1996)	Many tree transducers not closed under composition (Maletti et al 09)
General-purpose tools	FSM, Carmel, OpenFST	Tiburon (May & Knight 10)

Tree Transducers

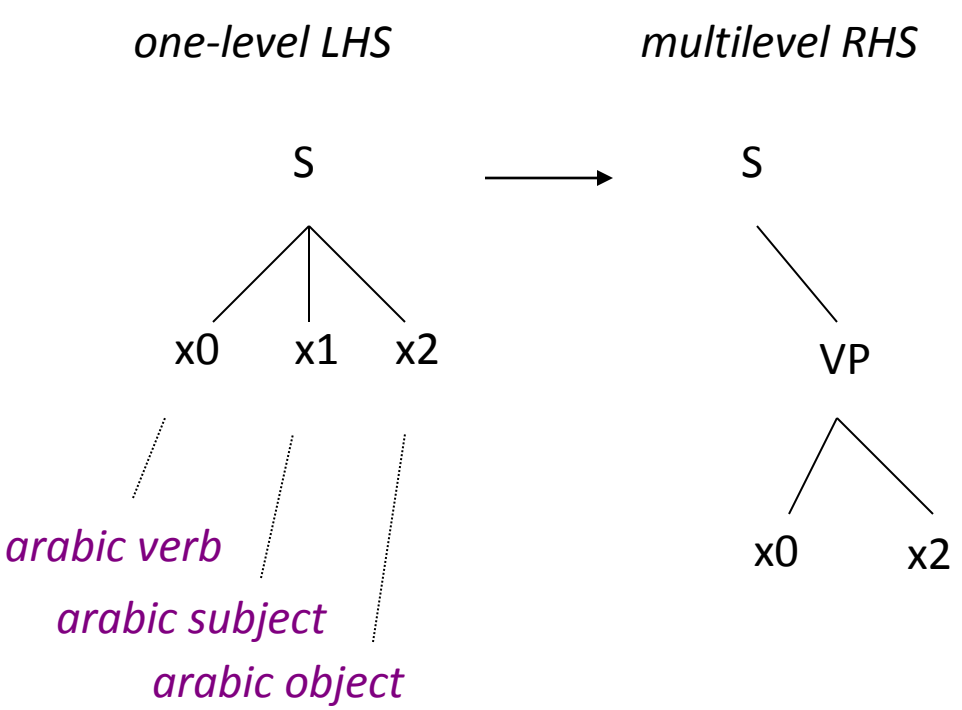
every rule has this form



T – top-down
L – linear (non-copying)
N – non-deleting

LNT

Tree Transducers

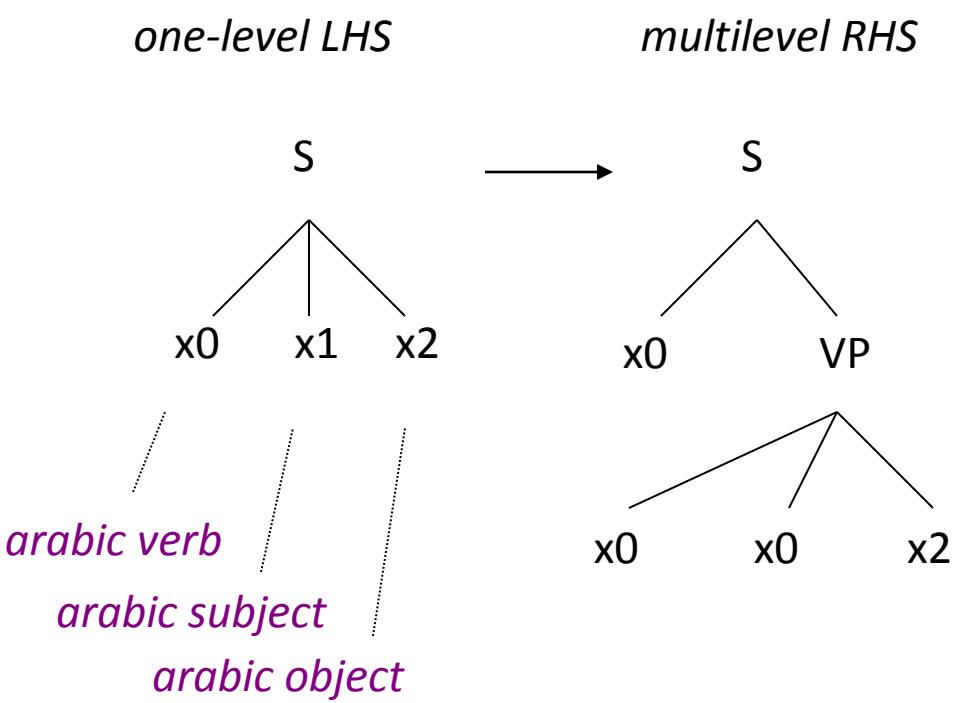


T – top-down
L – linear (non-copying)
N – non-deleting

LT

can delete subtrees

Tree Transducers

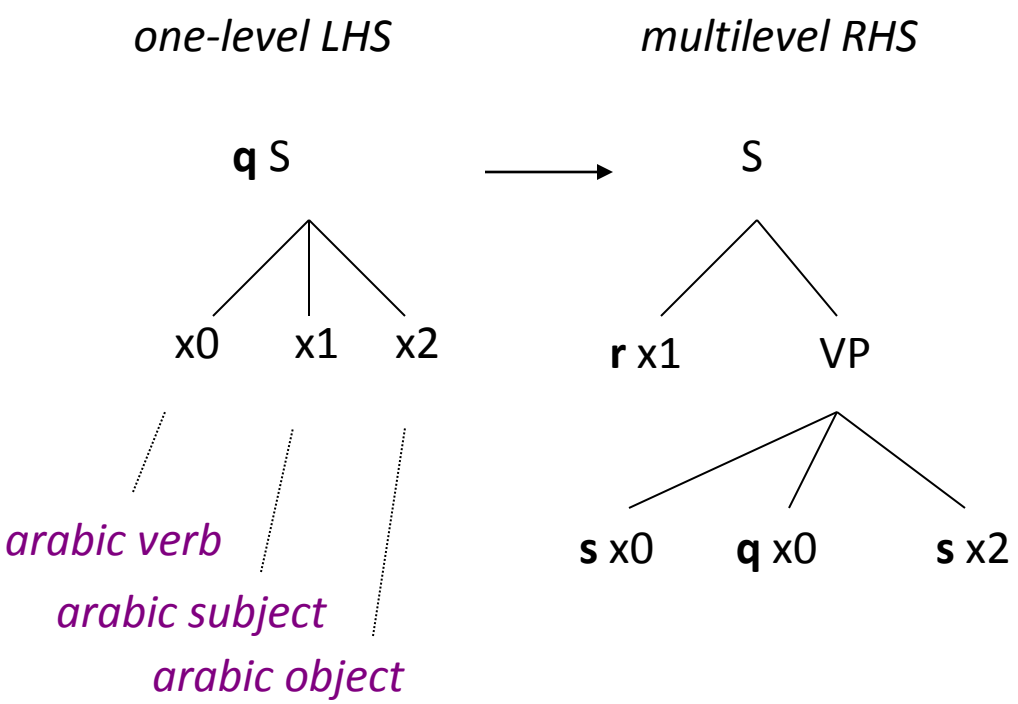


T – top-down
L – linear (non-copying)
N – non-deleting

T

can copy & delete subtrees

Tree Transducers



T – top-down
L – linear (non-copying)
N – non-deleting

T LT LNT

all employ **states**

Choices

- There are many different tree transducers
- What is good for NLP?
- Investigate formal properties related to:
  - expressiveness
  - modularity
  - inclusiveness
  - learnability

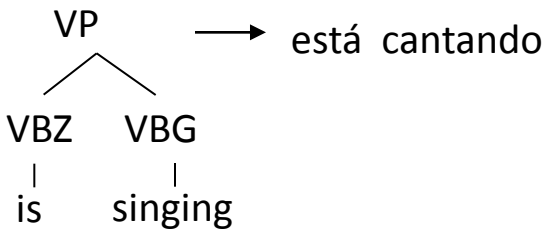
Choices

- There are many different tree transducers
- What is good for NLP?
- Investigate formal properties related to:
  - expressiveness (suggestions coming)
  - modularity (closed under composition)
  - inclusiveness (does everything FST does)
  - learnability (polynominal time EM)

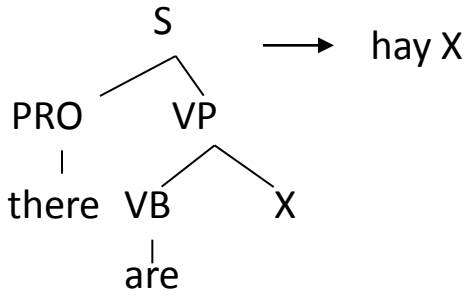
Expressiveness

some necessary things for machine translation

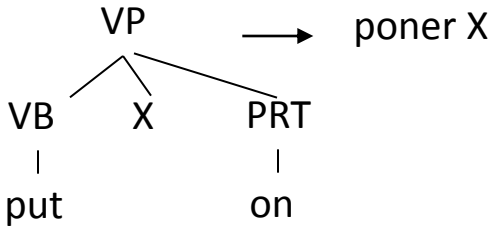
Phrasal Translation



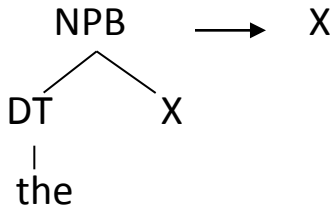
Non-constituent Phrases



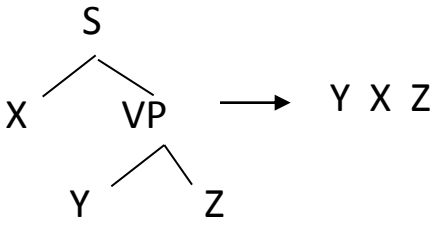
Non-contiguous Phrases



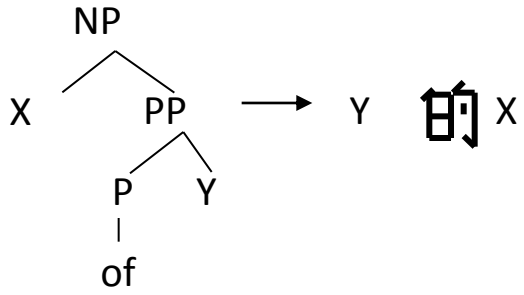
Context-Sensitive Word Insertion



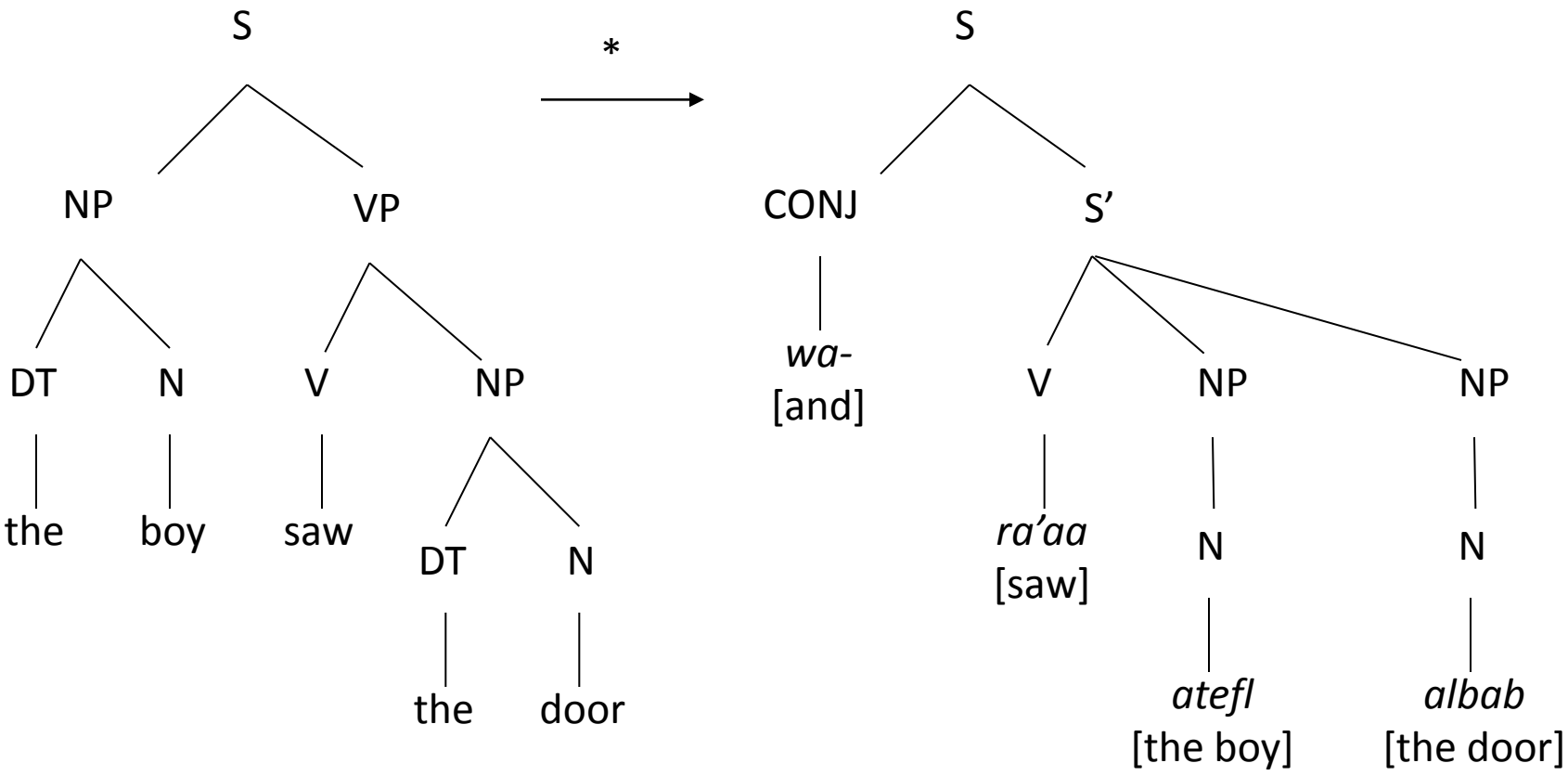
Re-Ordering



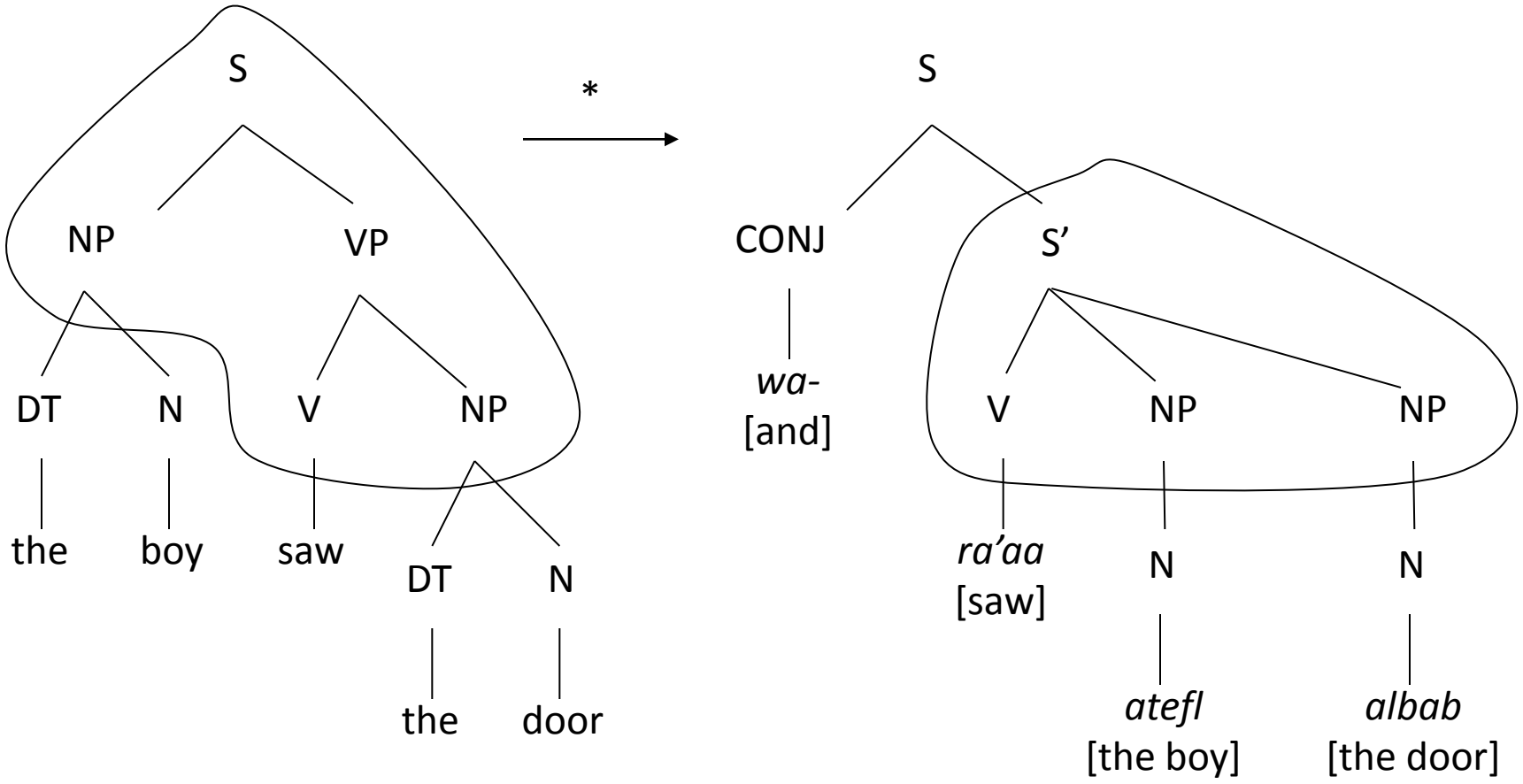
Lexicalized Re-Ordering



Expressiveness

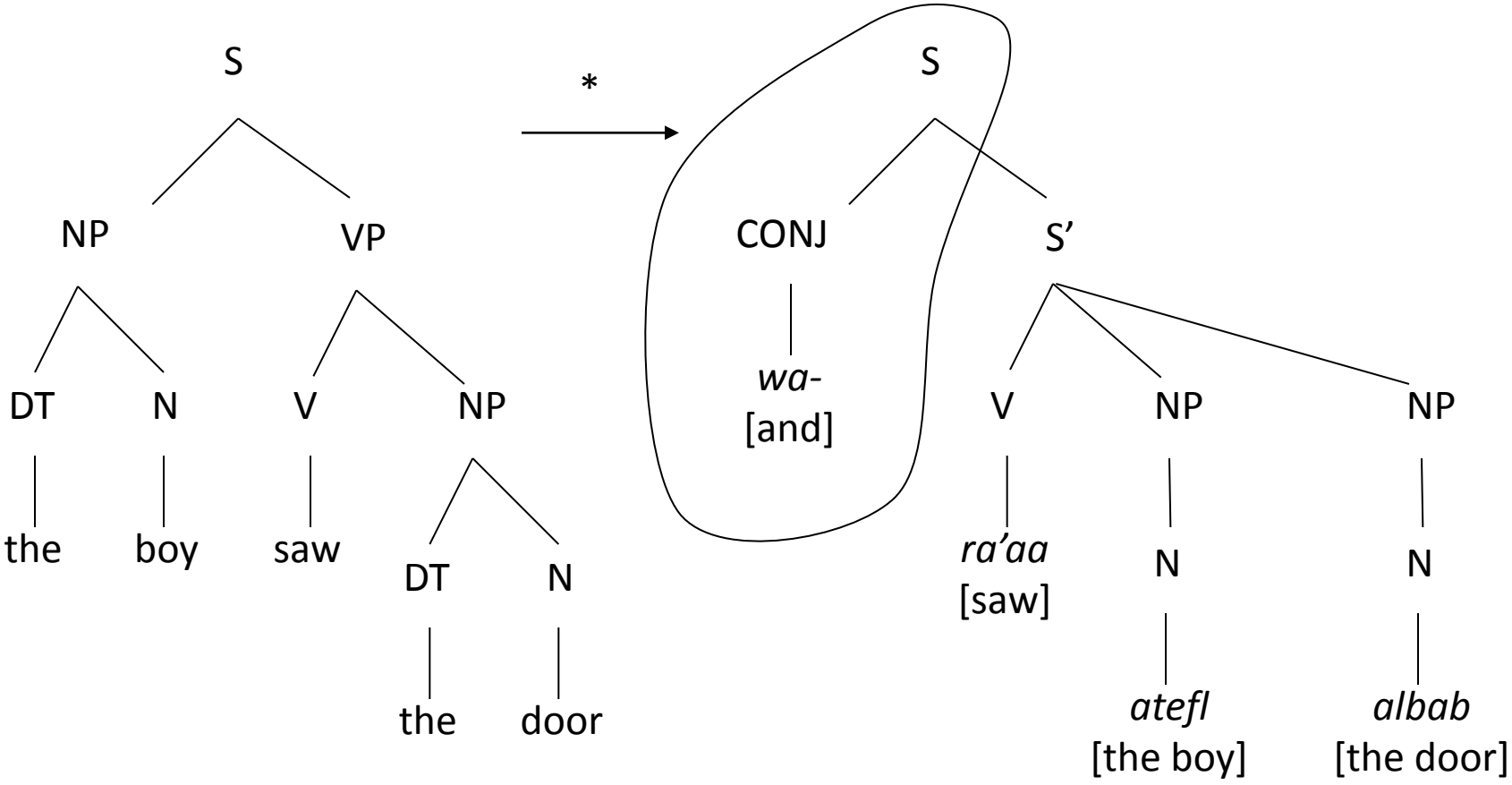


Expressiveness



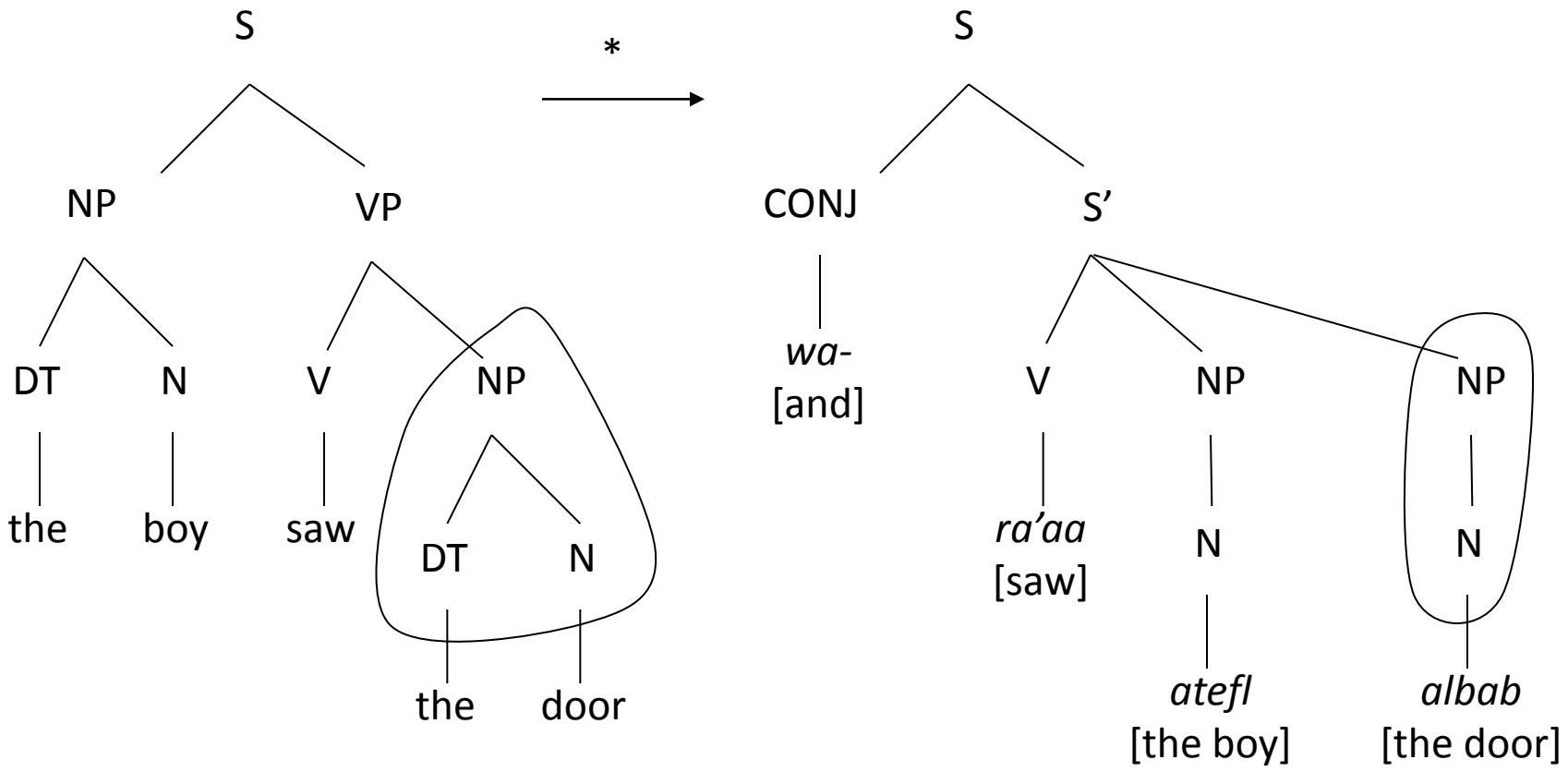
Local rotation

Expressiveness



Produce output without consuming input

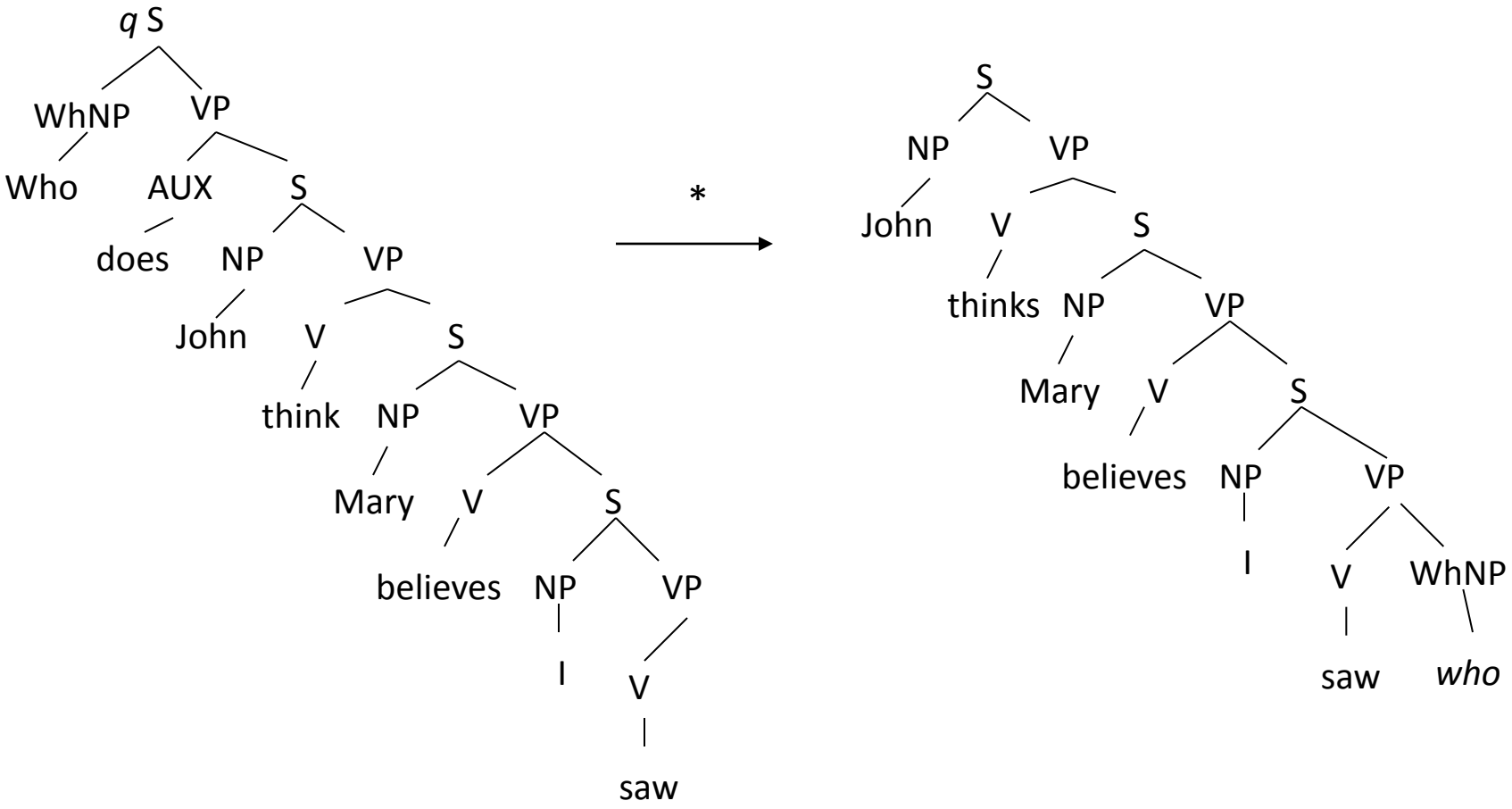
Expressiveness



Lookahead

Expressiveness

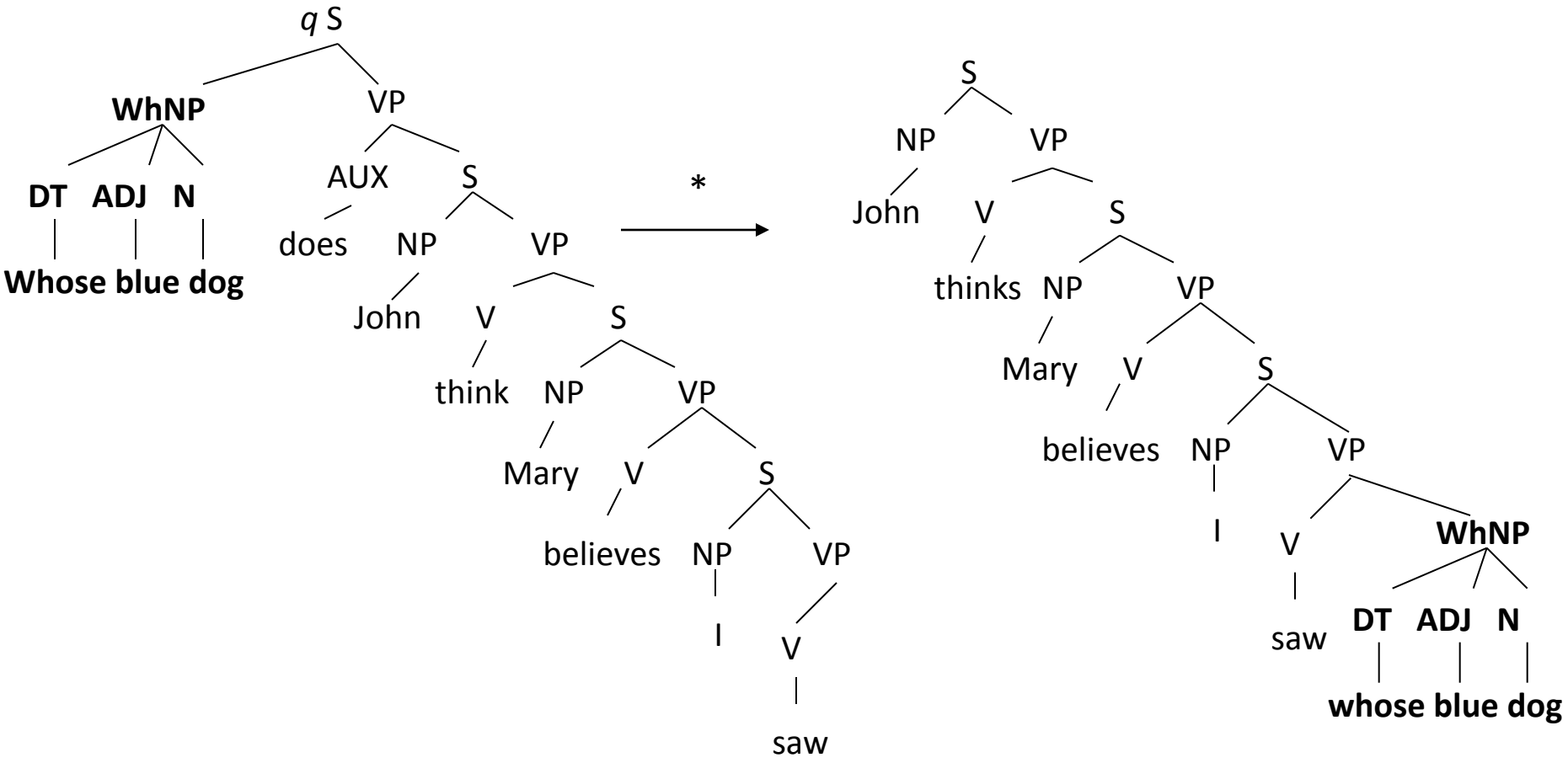
Who does John think Mary believes I saw? → John thinks Mary believes I saw ***who***?



Long distance movement
of bounded material

Expressiveness

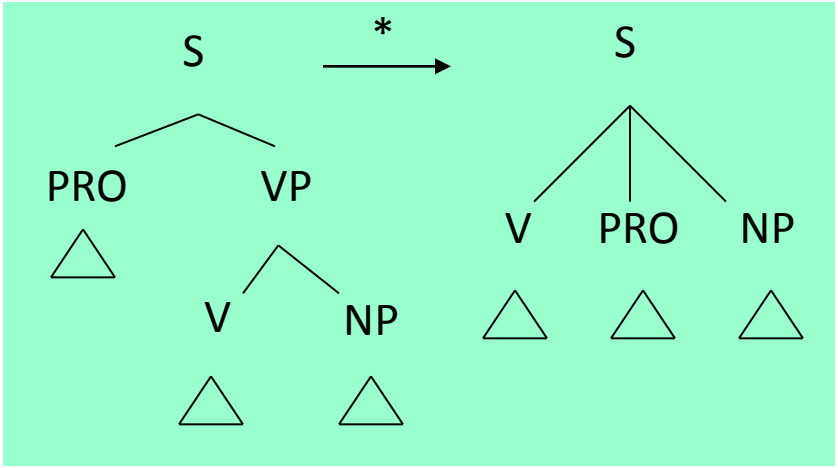
Whose blue dog does John think Mary believes I saw**?** → John thinks Mary believes I saw ***whose blue dog***?



Long distance movement
of unbounded material

Expressiveness

Let’s mainly focus on local rotation:



T – top-down
L – linear (non-copying)
N – non-deleting

T – top-down
L – linear (non-copying)
N – non-deleting

copying

non-copying

deleting

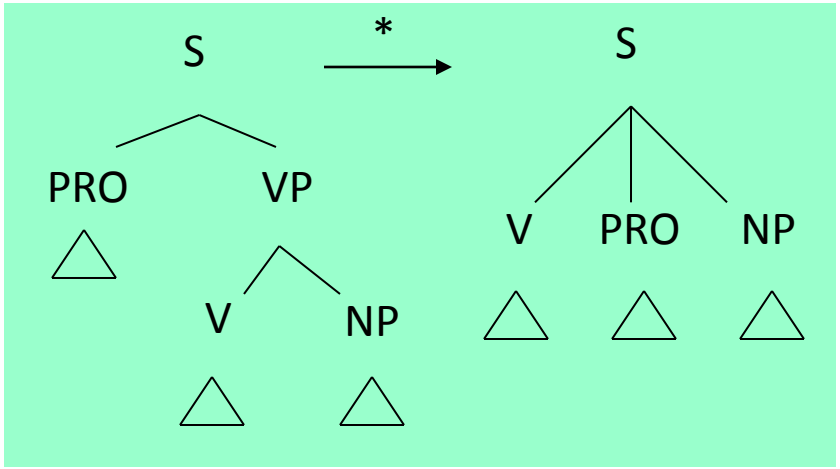
non-deleting

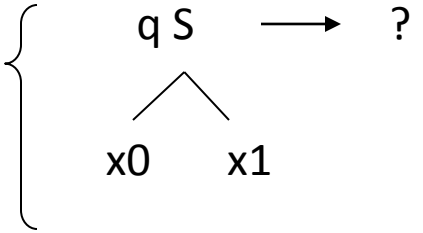
T

LT

LNT

Expressiveness:





T – top-down
L – linear (non-copying)
N – non-deleting

copying

non-copying

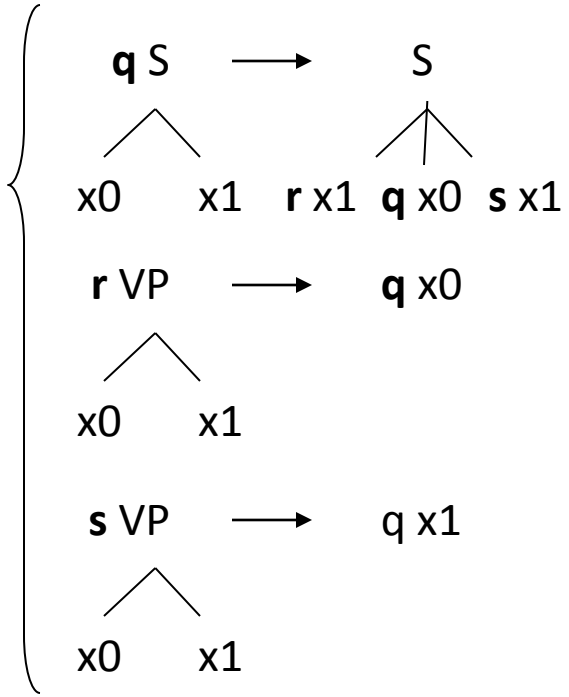
deleting

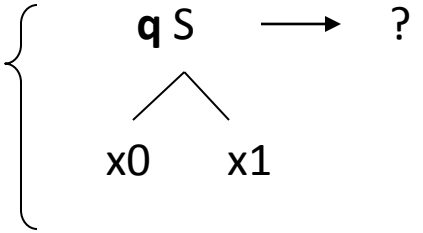
non-deleting

T

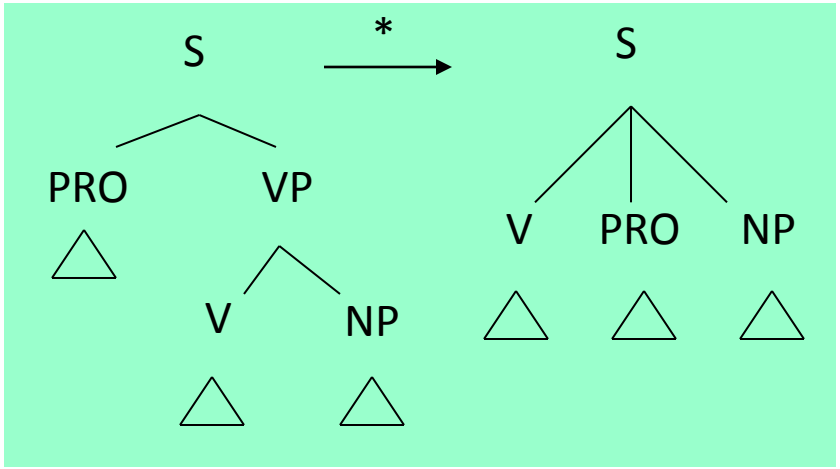
LT

LNT

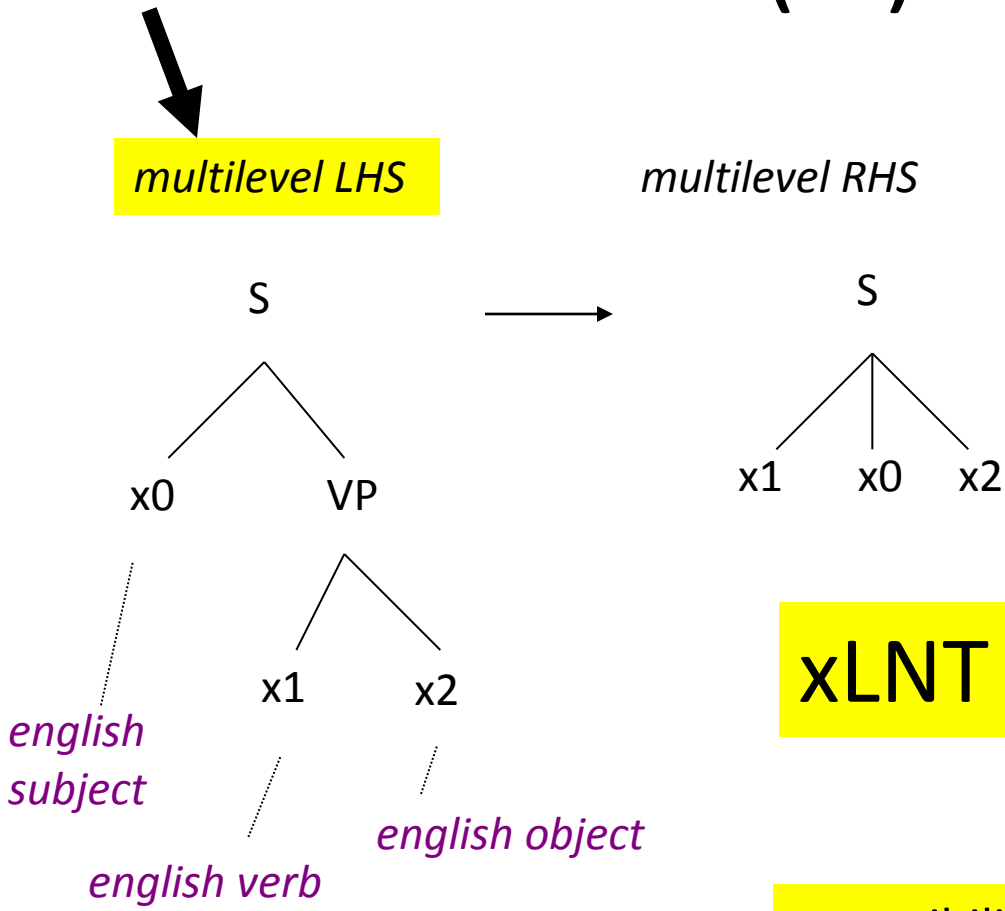




Expressiveness:



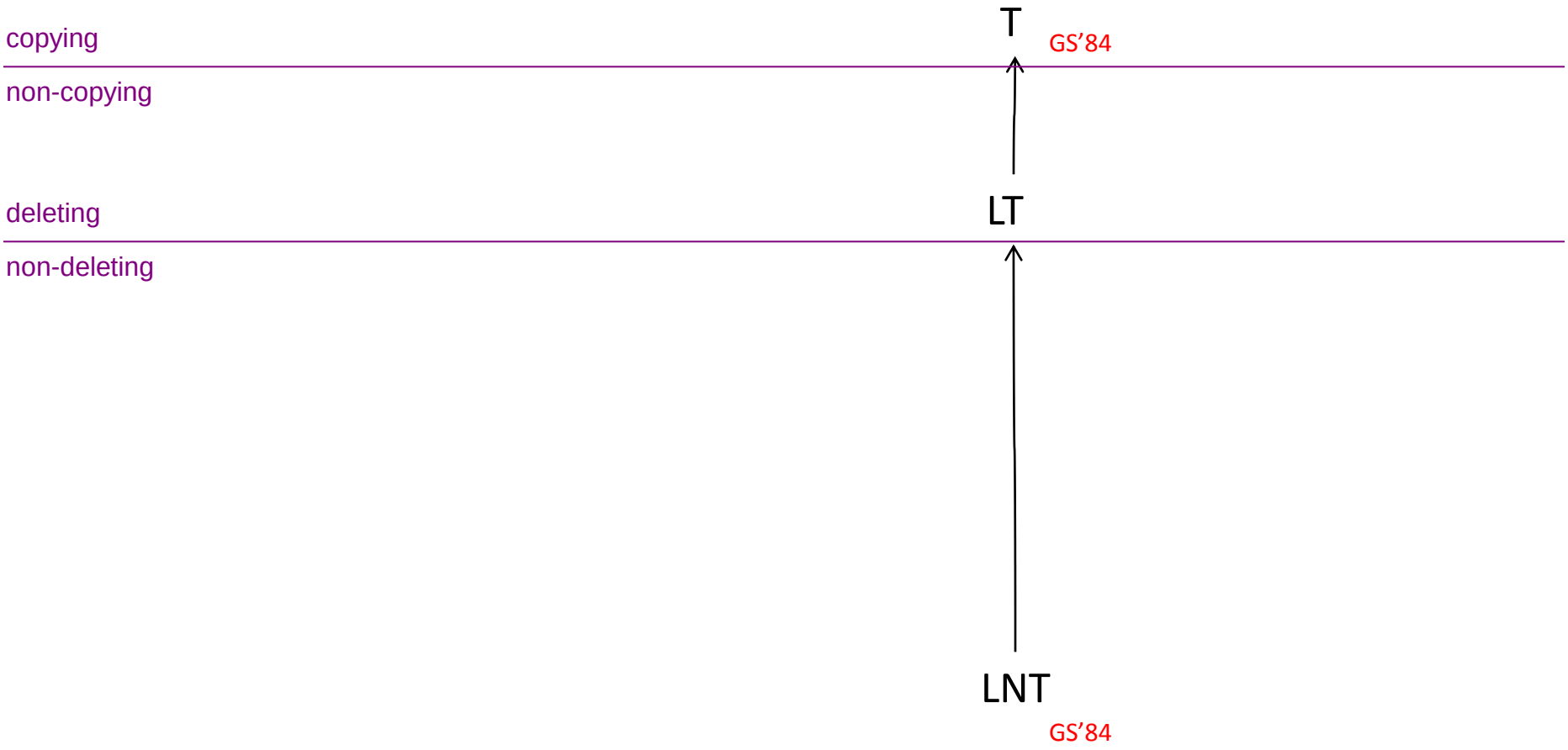
Extended (x-) Transducers

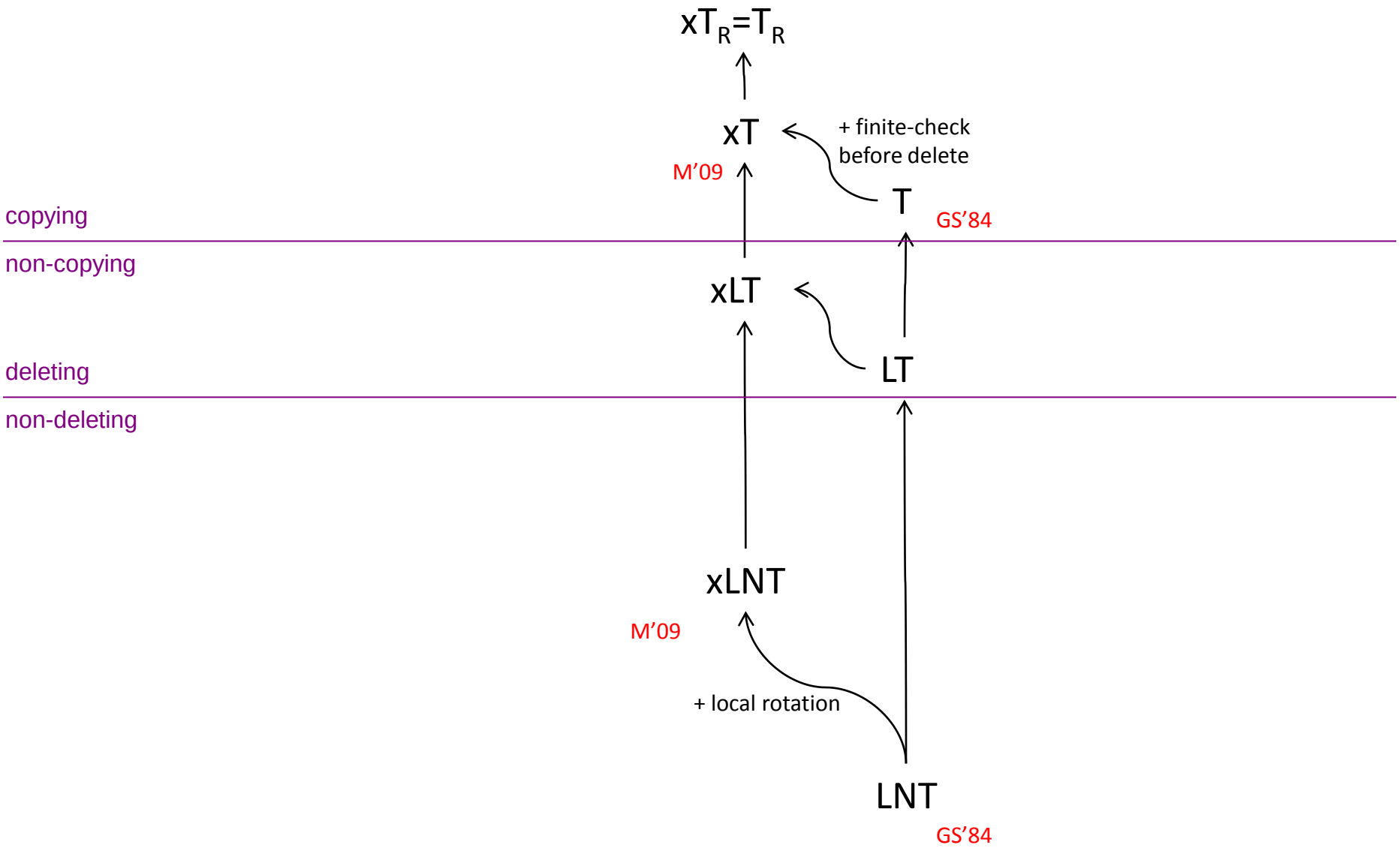


T – top-down
L – linear (non-copying)
N – non-deleting
x – extended LHS

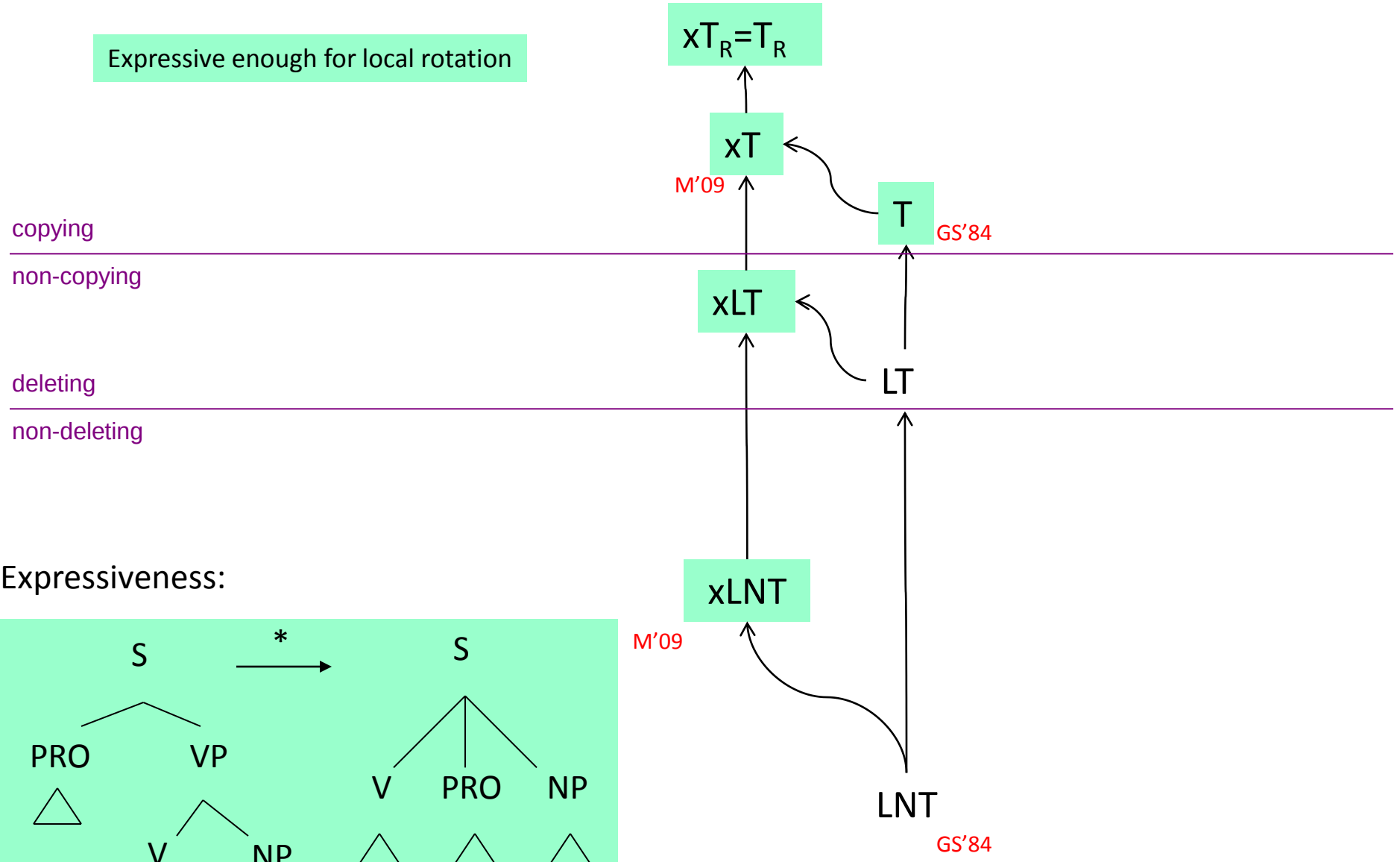
xLNT can grab more structure

- possibility mentioned in [Rounds 70]
- variant defined in [Dauchet 76]
- used for practical MT by [Galley et al 04, 06]
- studied formally by [Maletti et al 09]


copying
non-copying
T
GS'84
deleting
non-deleting
LT
LNT
GS'84


$xT_R=T_R$
xT
M'09
+ finite-check
before delete
T
GS'84
copying
non-copying
xLT
LT
deleting
non-deleting
xLNT
M'09
+ local rotation
LNT
GS'84

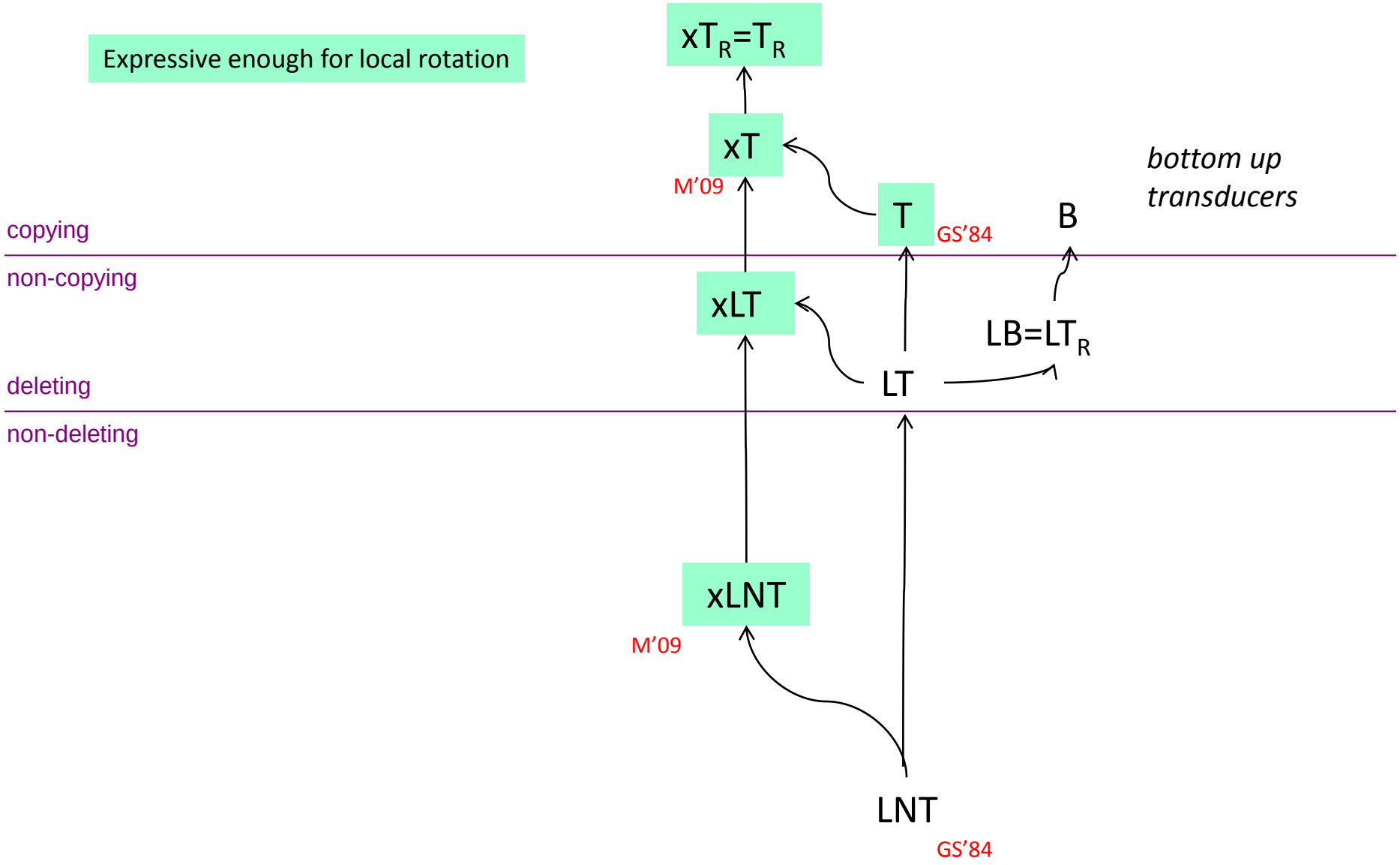
Expressive enough for local rotation

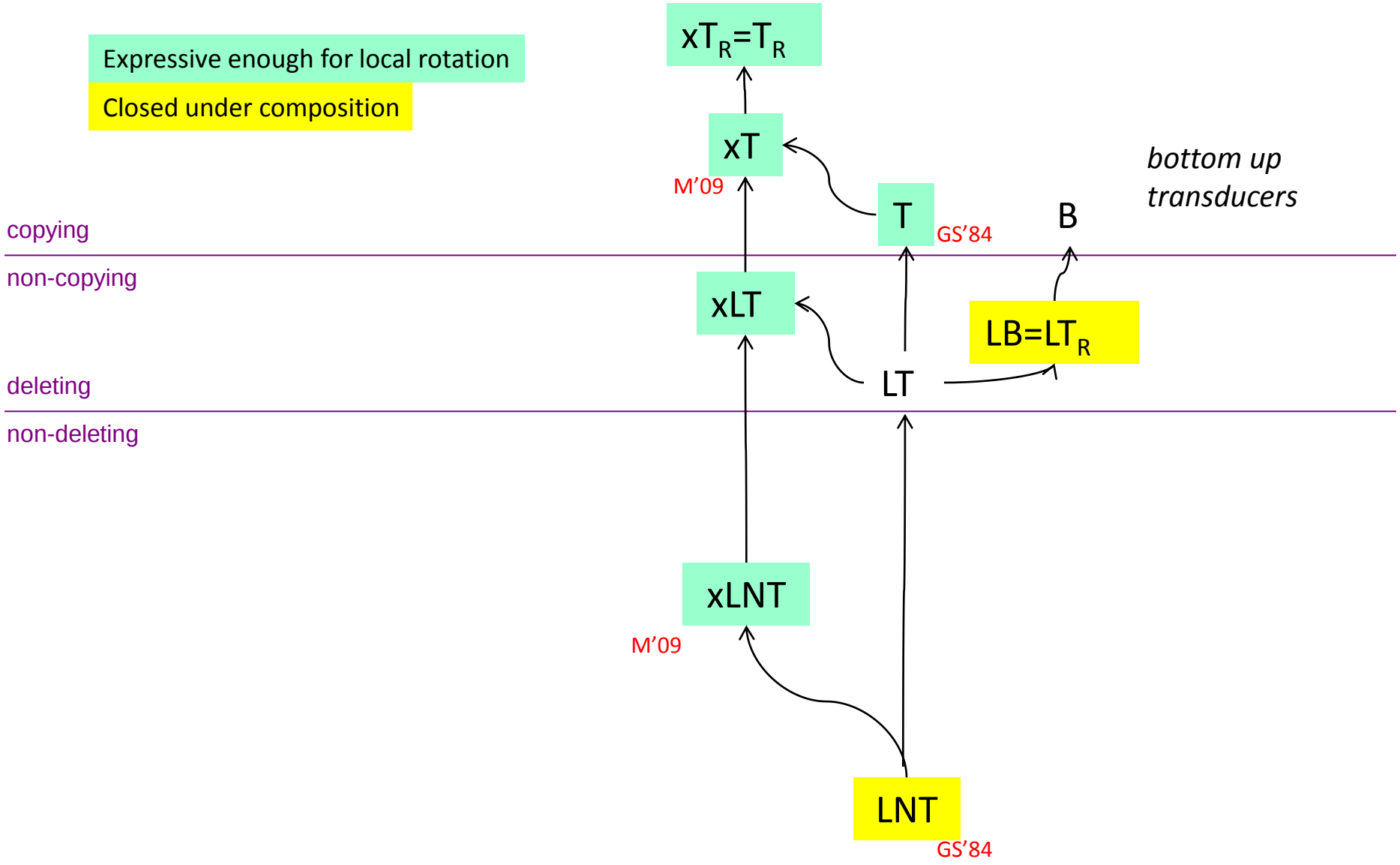


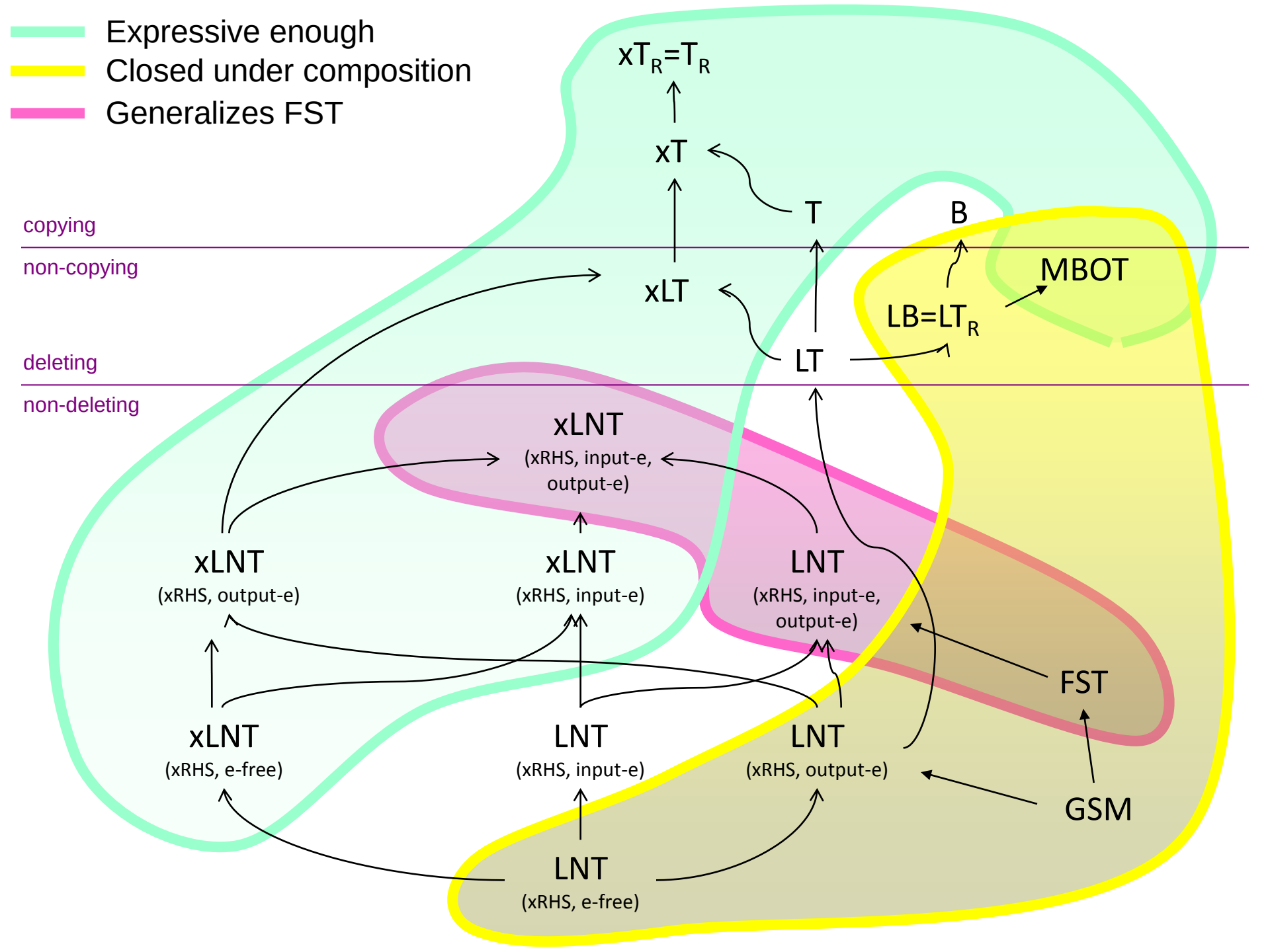
Expressiveness:

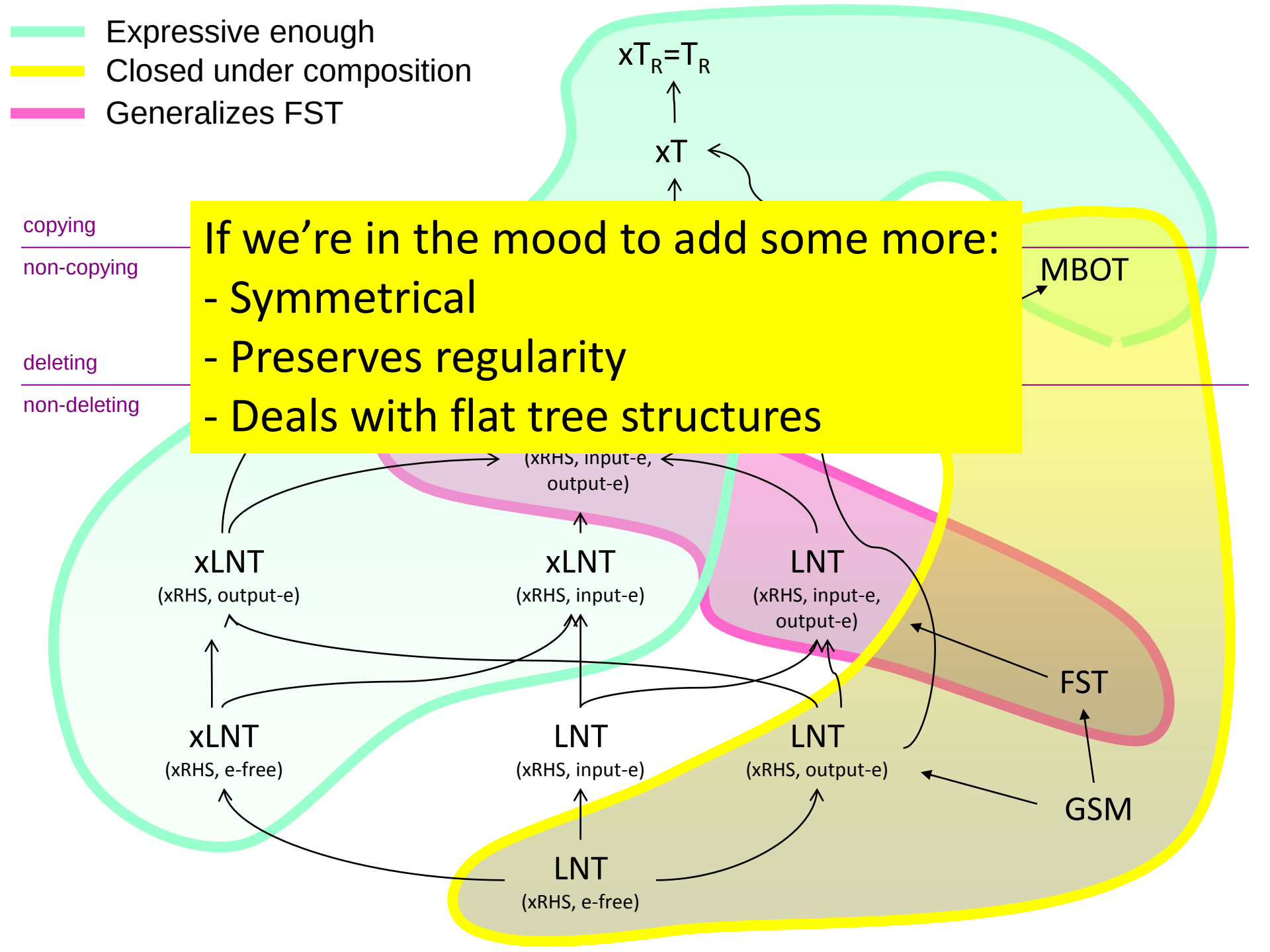
Expressive power theorems in [Maletti, Graehl, Hopkins, Knight, *SIAM J. Comput*]

Expressive enough for local rotation




Expressive enough for local rotation
Closed under composition
xT_R=T_R
xT
M'09
T
GS'84
B
bottom up
transducers
copying
non-copying
xLT
LB=LT_R
deleting
LT
non-deleting
xLNT
M'09
LNT
GS'84


Expressive enough
Closed under composition
Generalizes FST
copying
non-copying
deleting
non-deleting
xT_R=T_R
xT
T
B
MBOT
xLT
LB=LT_R
LT
xLNT
(xRHS, input-e, output-e)
xLNT
(xRHS, output-e)
xLNT
(xRHS, input-e)
LNT
(xRHS, input-e, output-e)
FST
xLNT
(xRHS, e-free)
LNT
(xRHS, input-e)
LNT
(xRHS, output-e)
GSM
LNT
(xRHS, e-free)


Expressive enough
Closed under composition
Generalizes FST
xT_R=T_R
xT
copying
non-copying
deleting
non-deleting
If we're in the mood to add some more:
- Symmetrical
- Preserves regularity
- Deals with flat tree structures
MBOT
(xRHS, input-e, output-e)
xLNT
(xRHS, output-e)
xLNT
(xRHS, input-e)
LNT
(xRHS, input-e, output-e)
FST
xLNT
(xRHS, e-free)
LNT
(xRHS, input-e)
LNT
(xRHS, output-e)
GSM
LNT
(xRHS, e-free)

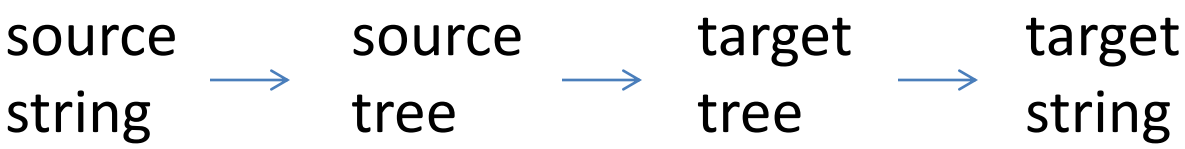
General-Purpose Algorithms for Tree Automata

	String Automata Algorithms	Tree Automata Algorithms
N-best …	… paths through an WFSA (Viterbi, 1967; Eppstein, 1998)	… trees in a weighted forest (Jiménez & Marzal, 2000; Huang & Chiang, 2005)
EM training	Forward-backward EM (Baum/Welch, 1971; Eisner 2003)	Tree transducer EM training (Graehl & Knight, 2004)
Determinization …	… of weighted string acceptors (Mohri, 1997)	… of weighted tree acceptors (Borchardt & Vogler, 2003; May & Knight, 2005)
Intersection	WFSA intersection	Tree acceptor intersection
Applying transducers	string → WFST → WFSA	tree → TT → weighted tree acceptor
Transducer composition	WFST composition (Pereira & Riley, 1996)	Many tree transducers not closed under composition (Maletti et al 09)
General-purpose tools	FSM, Carmel, OpenFST	Tiburon (May & Knight 10)

Machine Translation

Phrase-based MT

Syntax-based MT



Meaning-based MT

Equivalent Semantic Representations

"The boy wants to go."

LOGICAL FORM

∃ w, b, g : instance(w, WANT) ^
instance(g, GO) ^
instance(b, BOY) ^
agent(w, b) ^
patient(w, g) ^
agent(g, b)

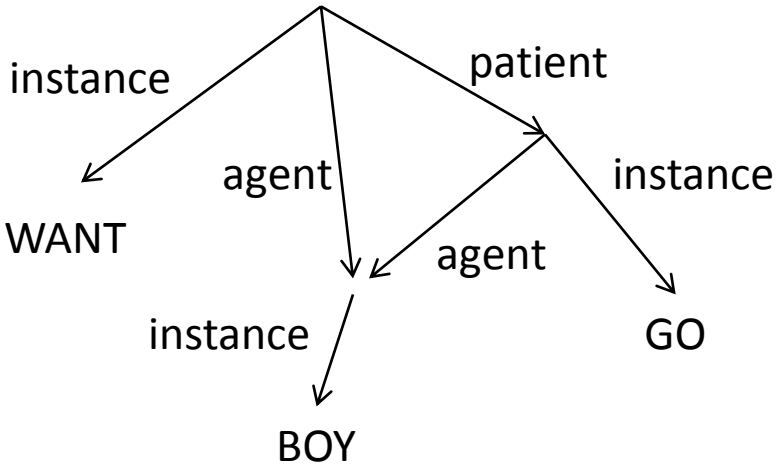
PENMAN

(w / WANT
:agent (b / BOY)
:patient (g / GO
:agent b)))

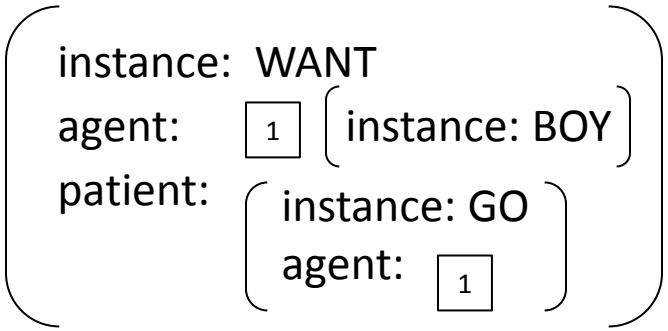
PATH EQUATIONS

((x0 instance) = WANT
((x1 instance) = BOY
((x2 instance) = GO
((x0 agent) = x1
((x0 patent) = x2
((x2 agent) = x1

DIRECTED ACYCLIC GRAPH



FEATURE STRUCTURE



Equivalent Semantic Representations

"The boy wants to go."

LOGICAL FORM

∃ w, b, g : instance(w, WANT) ^
instance(g, GO) ^
instance(b, BOY) ^
agent(w, b) ^
patient(w, g) ^
agent(g, b)

EASY FOR PEOPLE

PENMAN

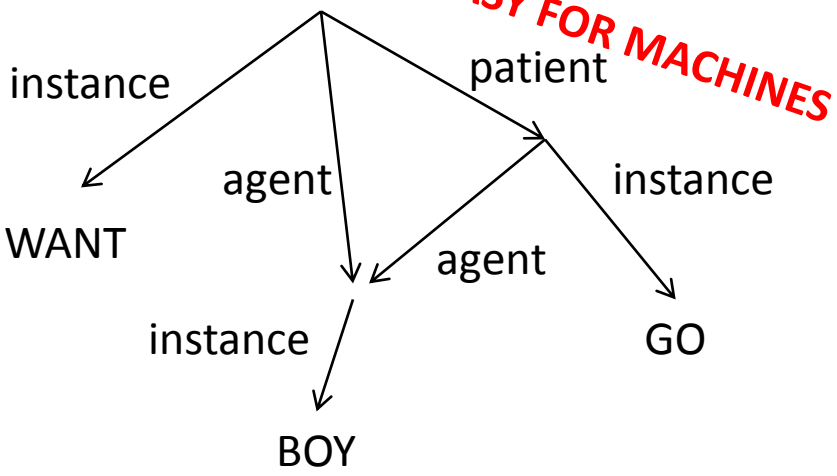
(w / WANT
:agent (b / BOY)
:patient (g / GO
:agent b)))

PATH EQUATIONS

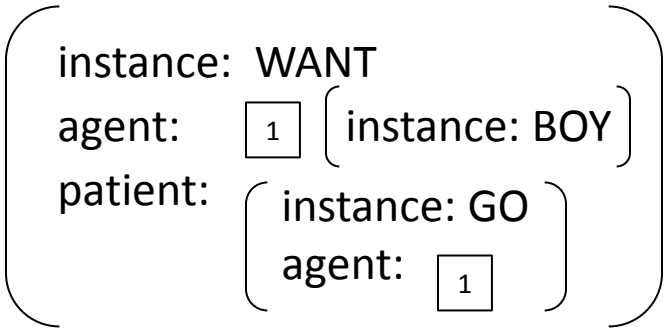
((x0 instance) = WANT
((x1 instance) = BOY
((x2 instance) = GO
((x0 agent) = x1
((x0 patent) = x2
((x2 agent) = x1

DIRECTED ACYCLIC GRAPH

EASY FOR MACHINES



FEATURE STRUCTURE



Example

"Government forces closed on rebel outposts on Thursday, showering the western mountain city of Zintan with missiles and attacking insurgents holed up near the Tunisian border, according to rebel sources."

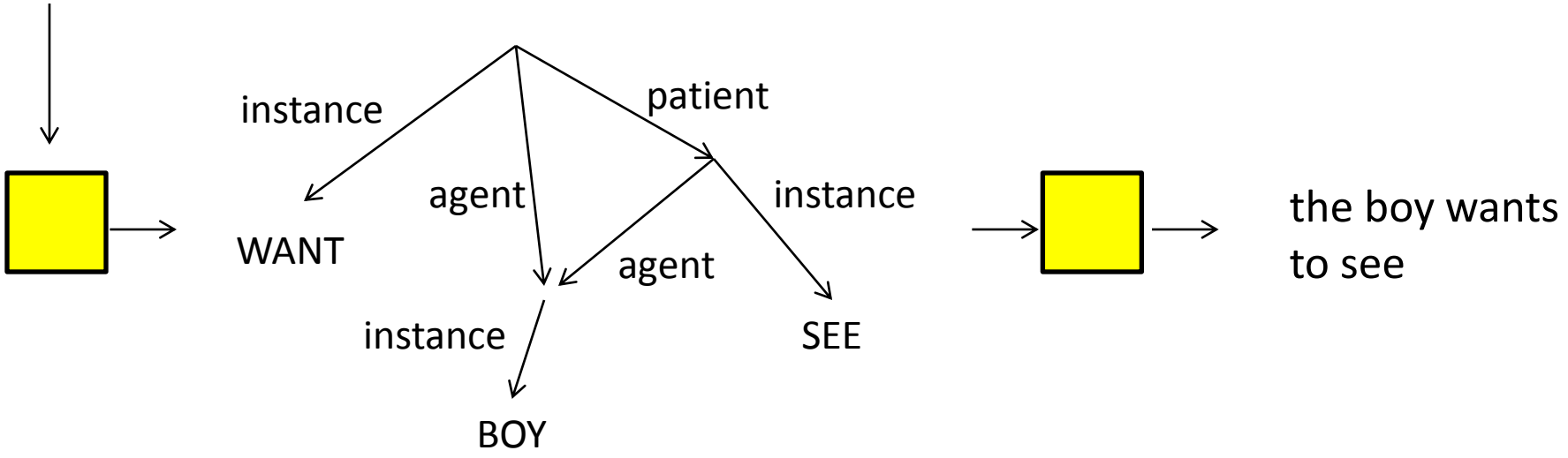
```
(s / say
  :agent (s2 / source :mod (r / rebel))
  :patient (a / and
            :op1 (c / close-on
                   :agent (f / force :mod (g / government))
                   :patient (o / outpost :mod (r2 / rebel))
                   :temporal-locating (t / thursday))
            :op2 (s / shower
                   :agent f
                   :patient (c2 / city
                              :mod (m / mountain)
                              :mod (w / west)
                              :name "Zintan")
                   :instrument (m2 / missile))
            :op3 (a2 / attack
                   :agent f
                   :patient (i / insurgent
                              :agent-of (h / hole-up
                                          :pp-near (b / border :poss (c3 / country
                                                                       :name "Tunisia"]
```

"more logical than a parse tree"

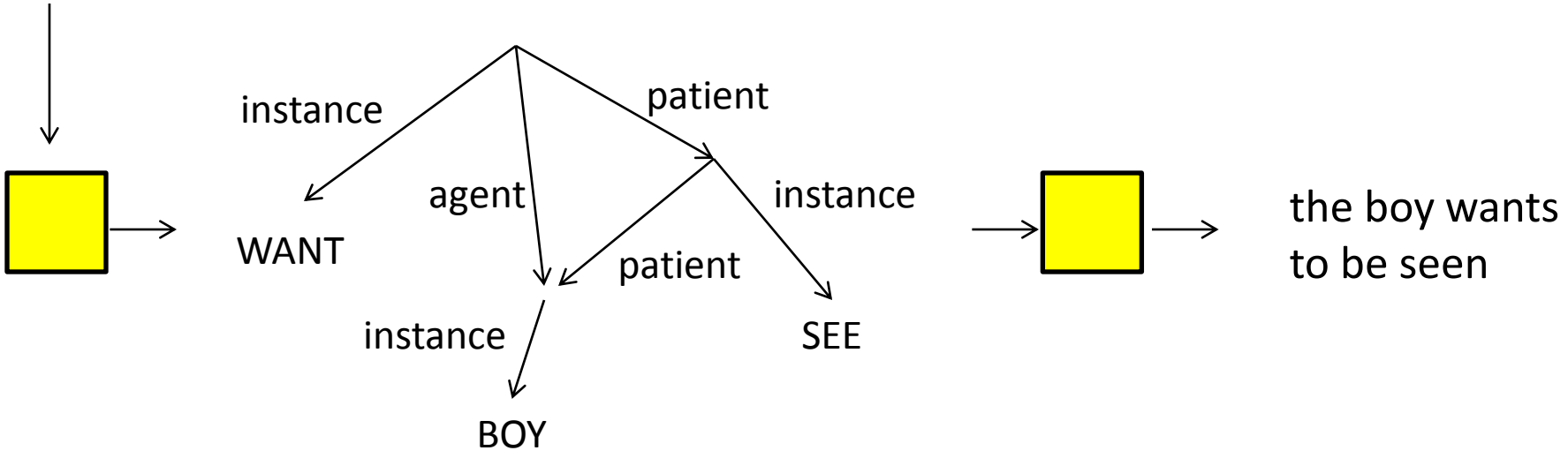
General-Purpose Algorithms for Feature Structures (Graphs)

	String Automata Algorithms	Tree Automata Algorithms	Graph Automata Algorithms?
N-best …	… paths through an WFSA (Viterbi, 1967; Eppstein, 1998)	… trees in a weighted forest (Jiménez & Marzal, 2000; Huang & Chiang, 2005)	
EM training	Forward-backward EM (Baum/Welch, 1971; Eisner 2003)	Tree transducer EM training (Graehl & Knight, 2004)	
Determinization…	… of weighted string acceptors (Mohri, 1997)	… of weighted tree acceptors (Borchardt & Vogler, 2003; May & Knight, 2005)	
Intersection	WFSA intersection	Tree acceptor intersection	
Applying transducers	string → WFST → WFSA	tree → TT → weighted tree acceptor	
Transducer composition	WFST composition (Pereira & Riley, 1996)	Many tree transducers not closed under composition (Maletti et al 09)	
General tools	FSM, Carmel, OpenFST	Tiburon (May & Knight 10)	

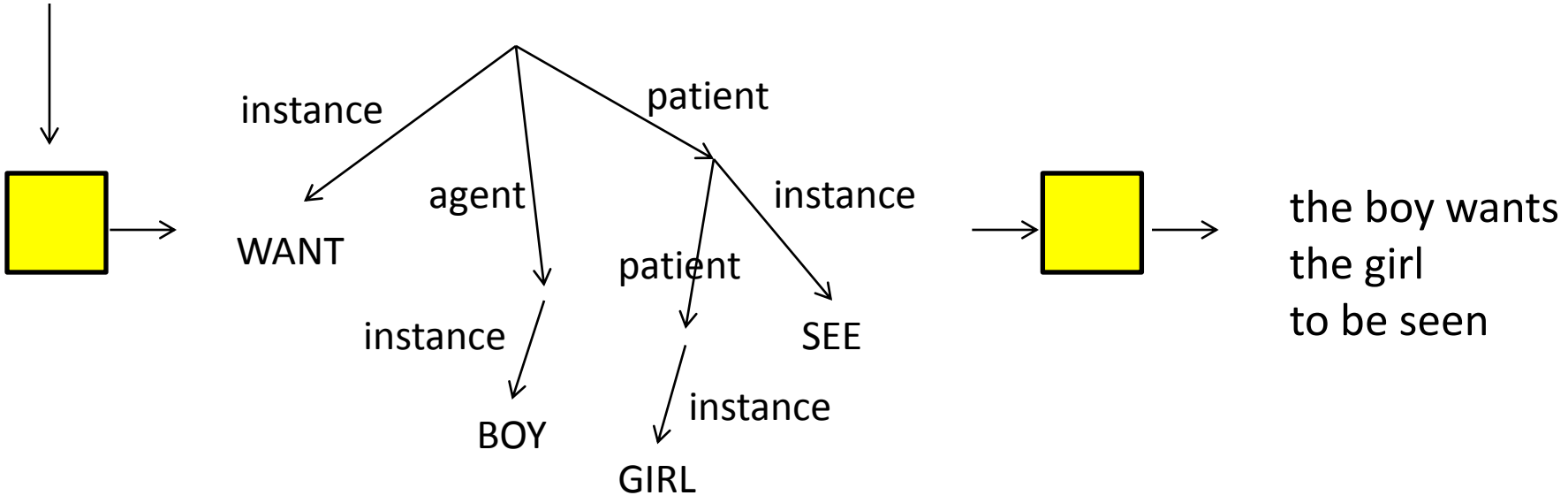
Mapping Between Meaning and Text



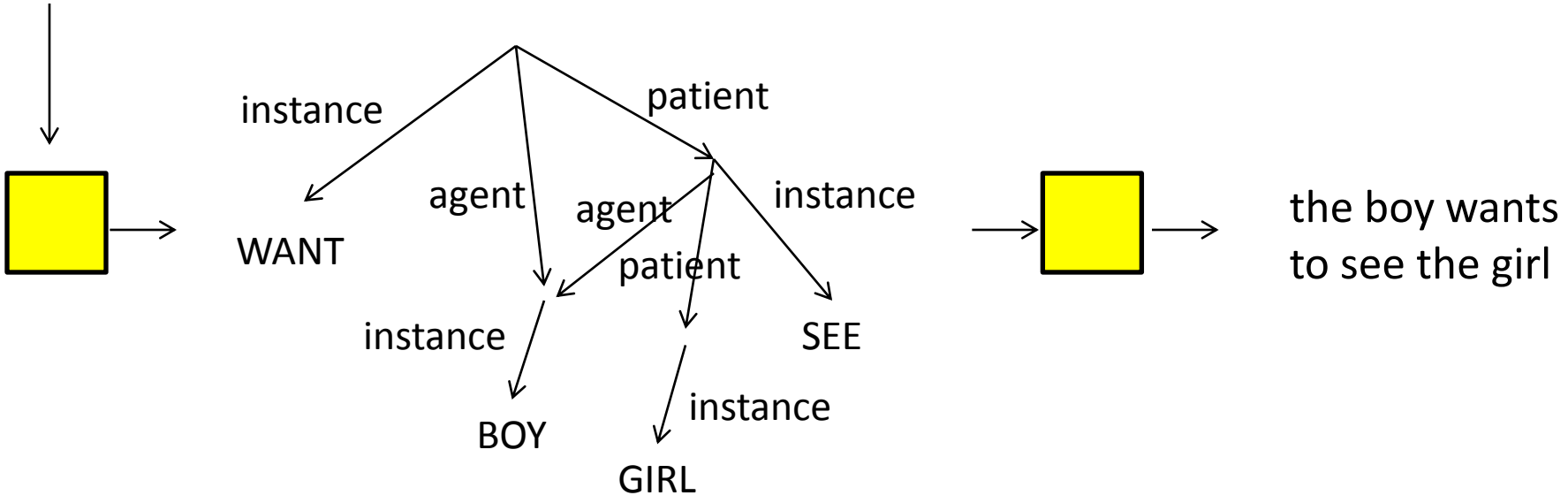
Mapping Between Meaning and Text



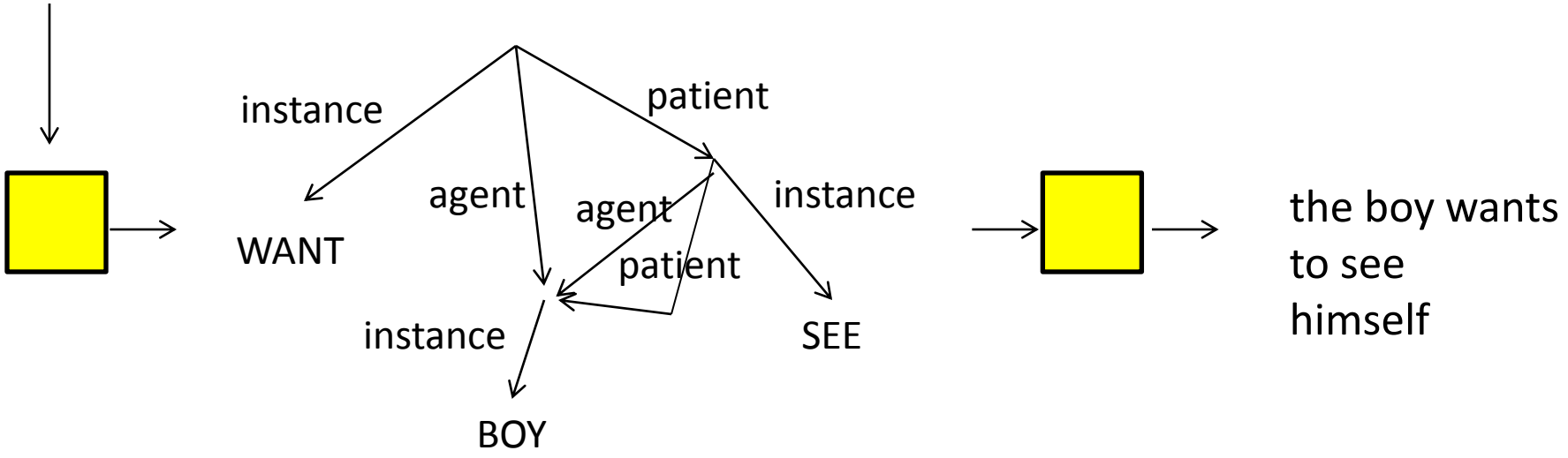
Mapping Between Meaning and Text



Mapping Between Meaning and Text



Mapping Between Meaning and Text

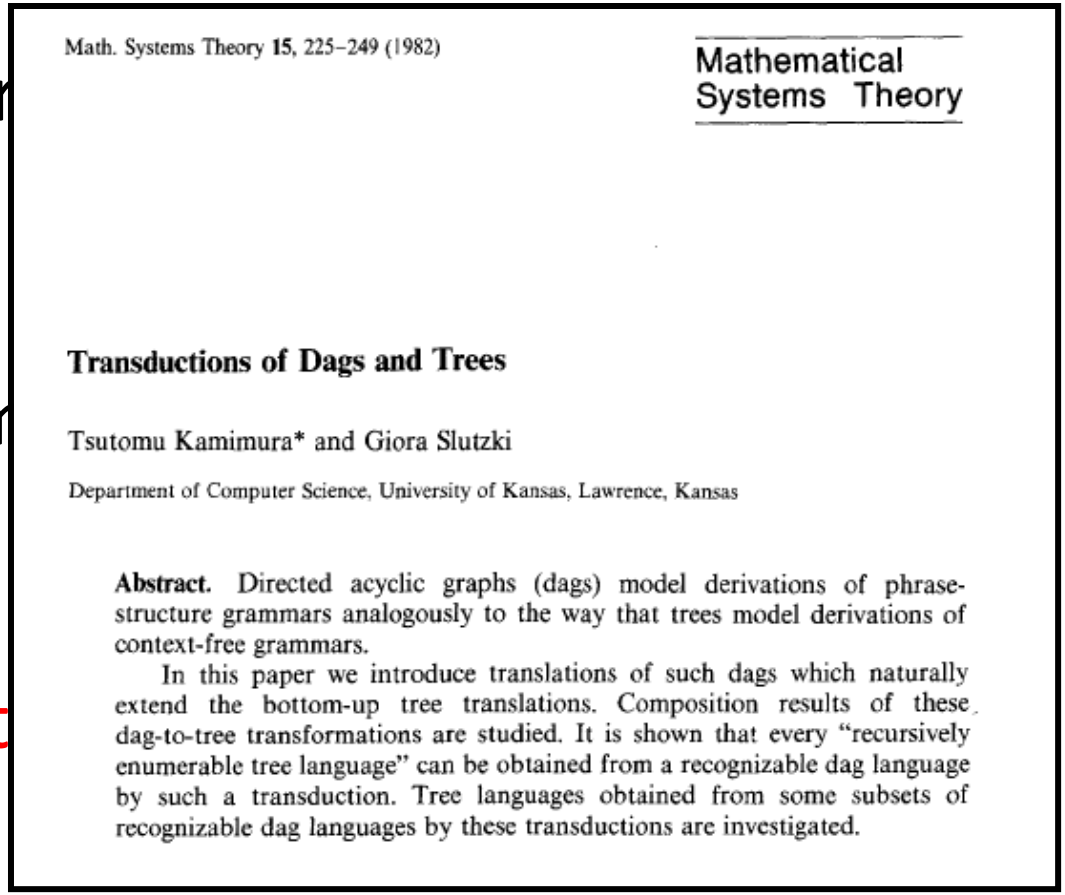


Automata Frameworks

- Unification grammar: string-to-semantics (Moore 89)
- Hyperedge-replacement graph grammars (Drewes et al 97)
- DAG acceptors (Hart 75)
- DAG-to-tree transducers (Kamimura & Slutski 82)

Automata Frameworks

- Unification grammar (Moore 89)
- Hyperedge-replacem (Drewes et al 97)
- DAG acceptors (Hart
- DAG-to-tree transducers (Kamimura & Slutski 82)



Math. Systems Theory 15, 225–249 (1982)

Mathematical Systems Theory

Transductions of Dags and Trees

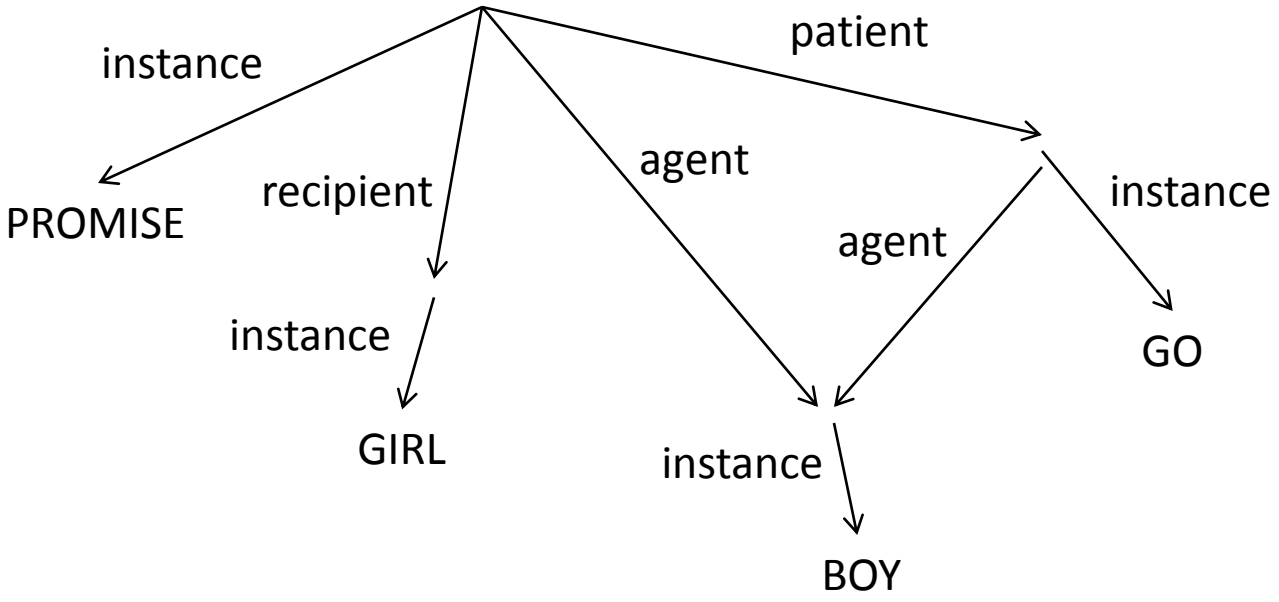
Tsutomu Kamimura* and Giora Slutzki

Department of Computer Science, University of Kansas, Lawrence, Kansas

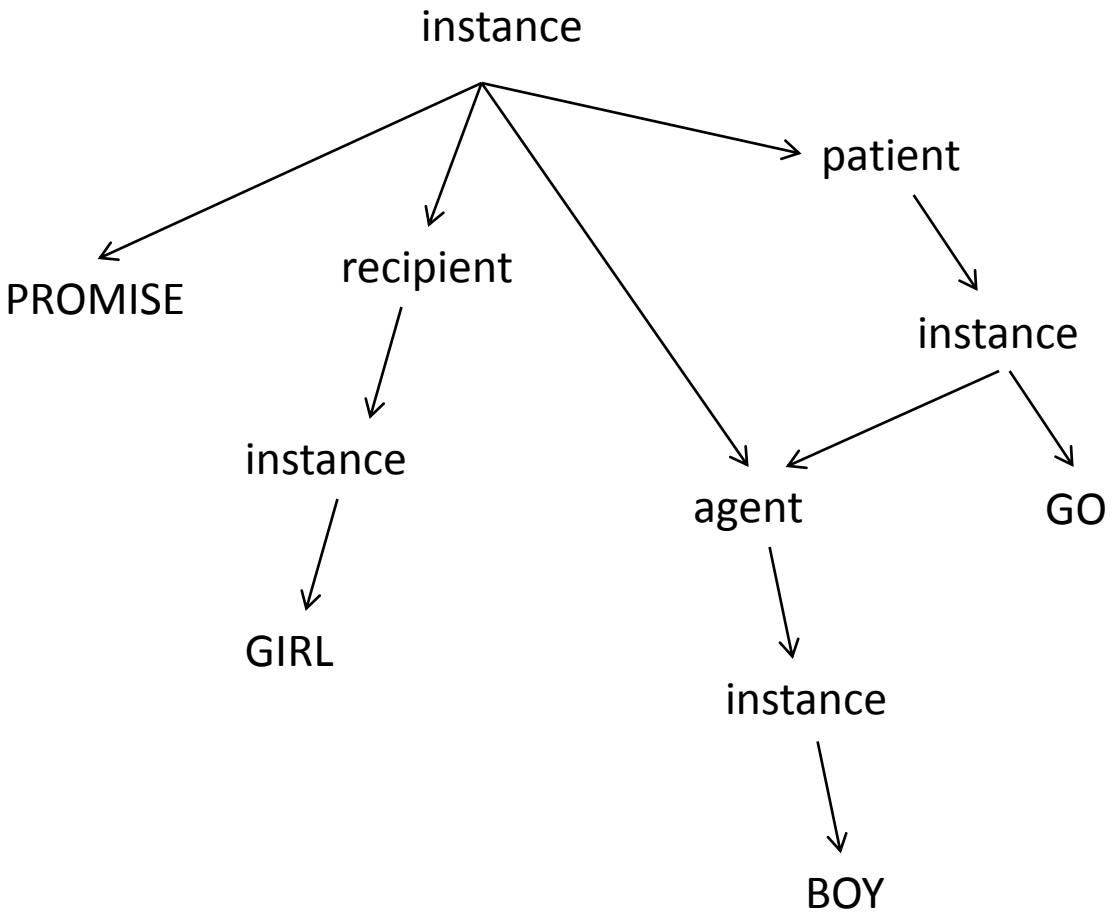
Abstract. Directed acyclic graphs (dags) model derivations of phrase-structure grammars analogously to the way that trees model derivations of context-free grammars.

In this paper we introduce translations of such dags which naturally extend the bottom-up tree translations. Composition results of these dag-to-tree transformations are studied. It is shown that every "recursively enumerable tree language" can be obtained from a recognizable dag language by such a transduction. Tree languages obtained from some subsets of recognizable dag languages by these transductions are investigated.

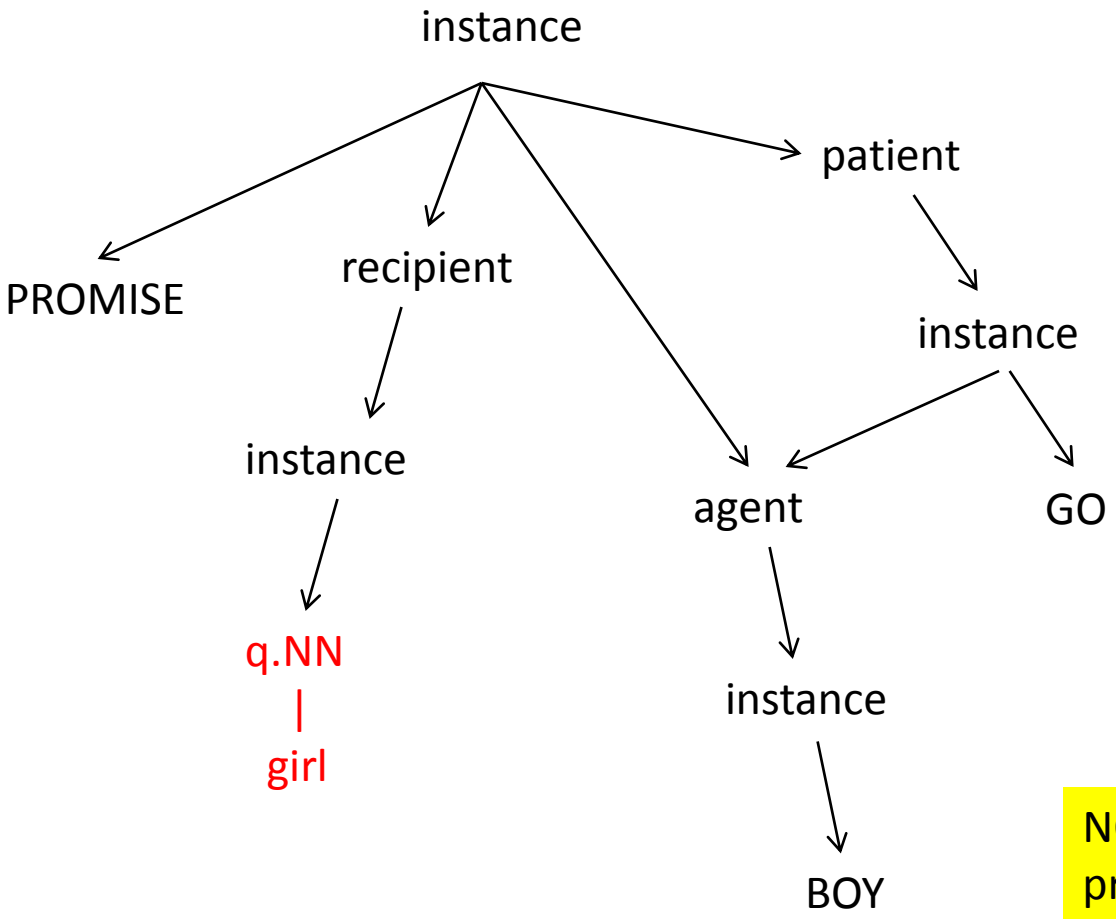
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

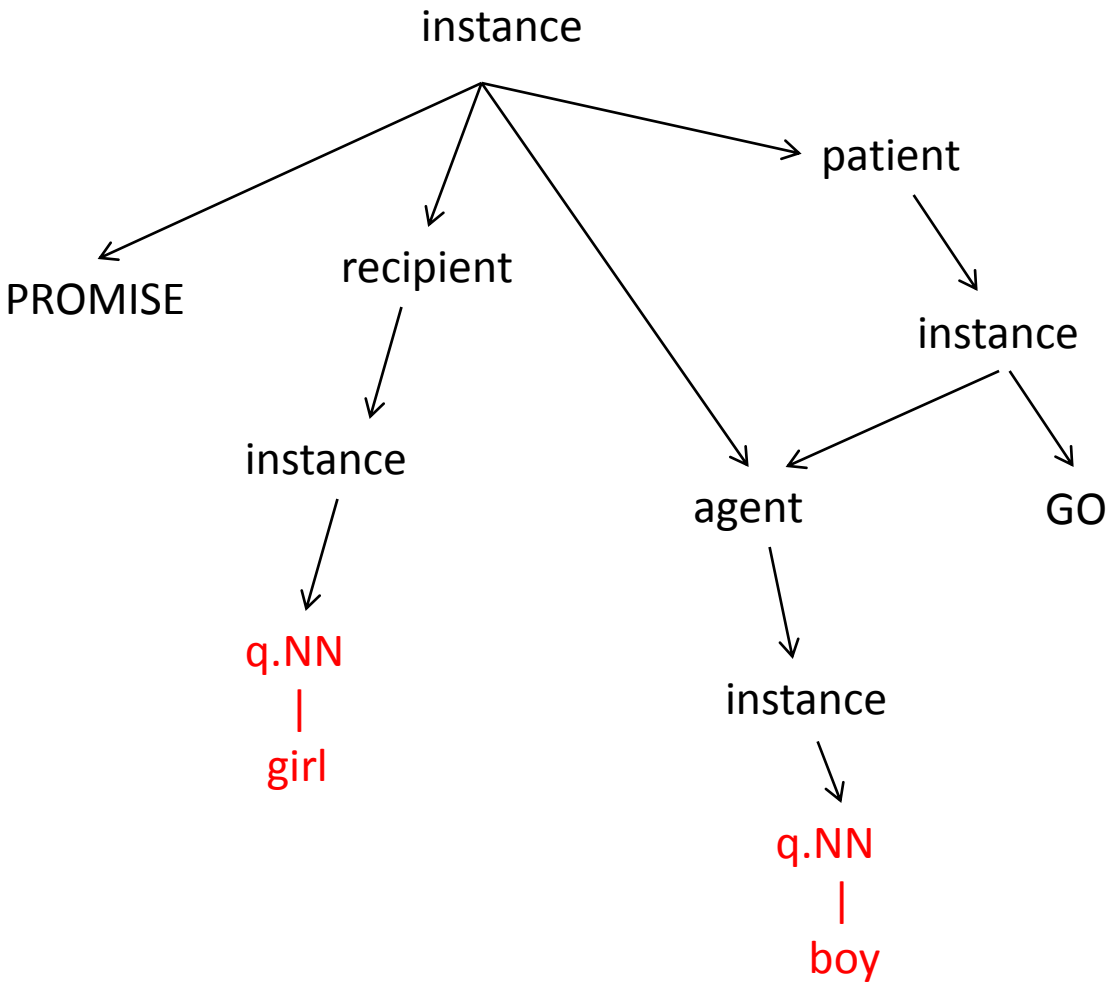


Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

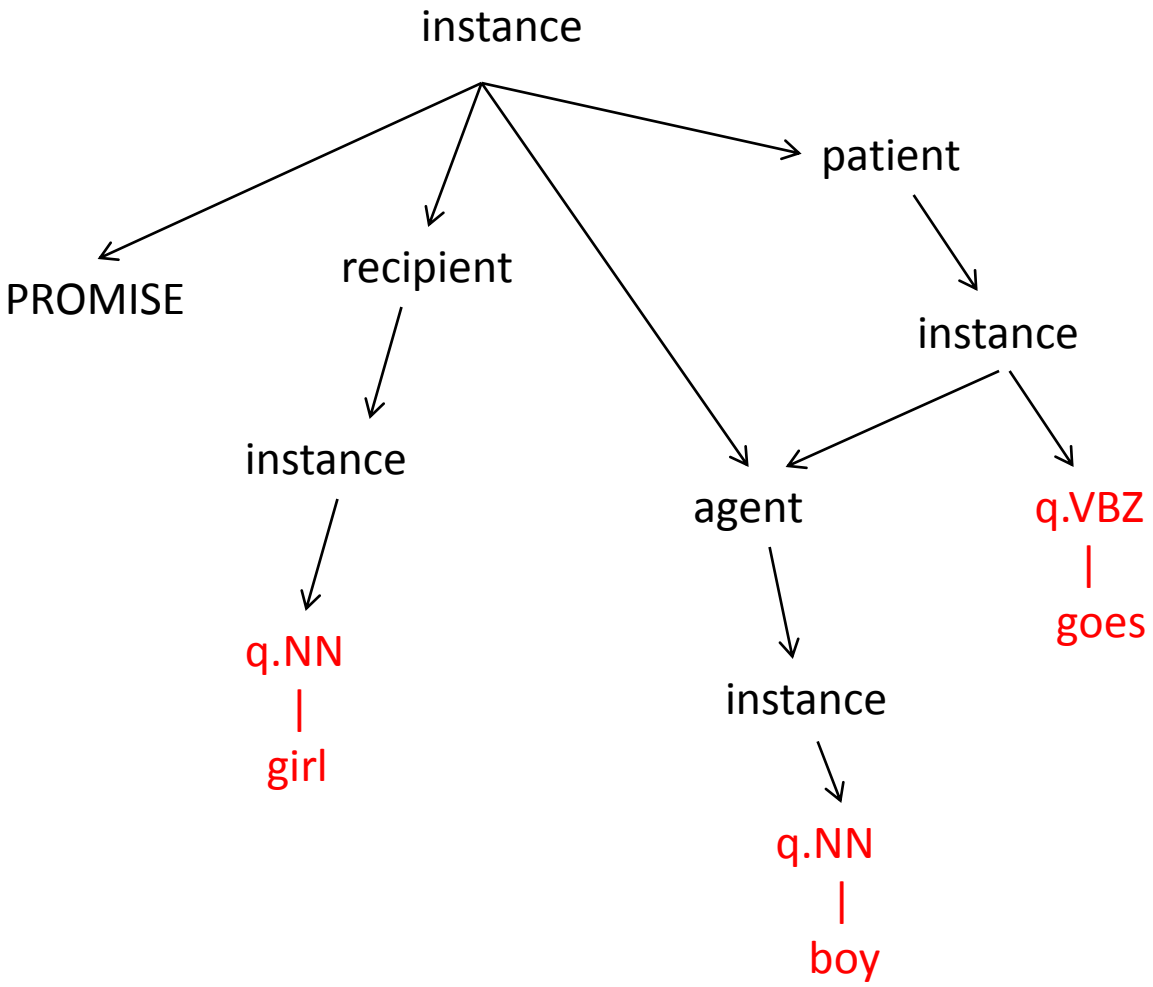


NOTE: States not precise! Rules not shown!

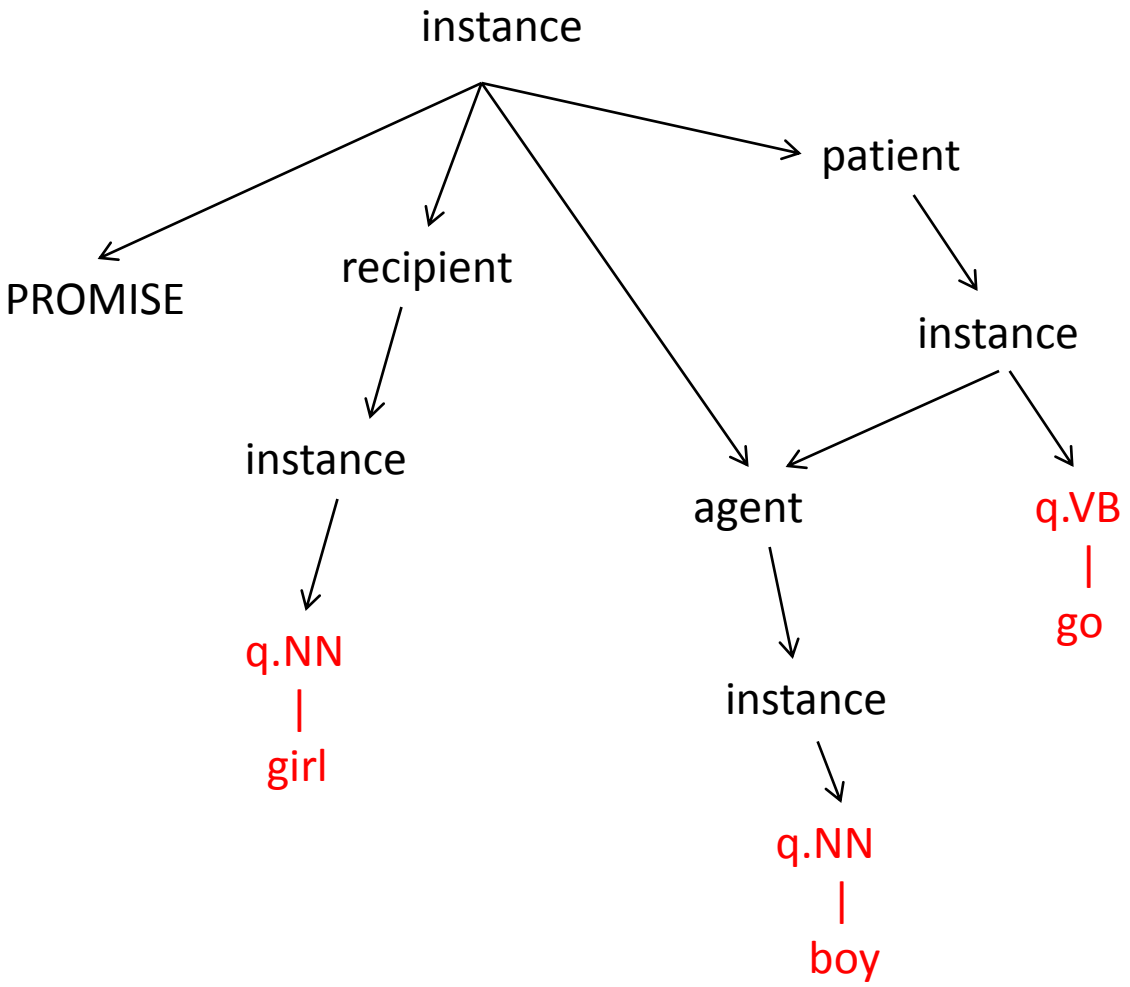
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



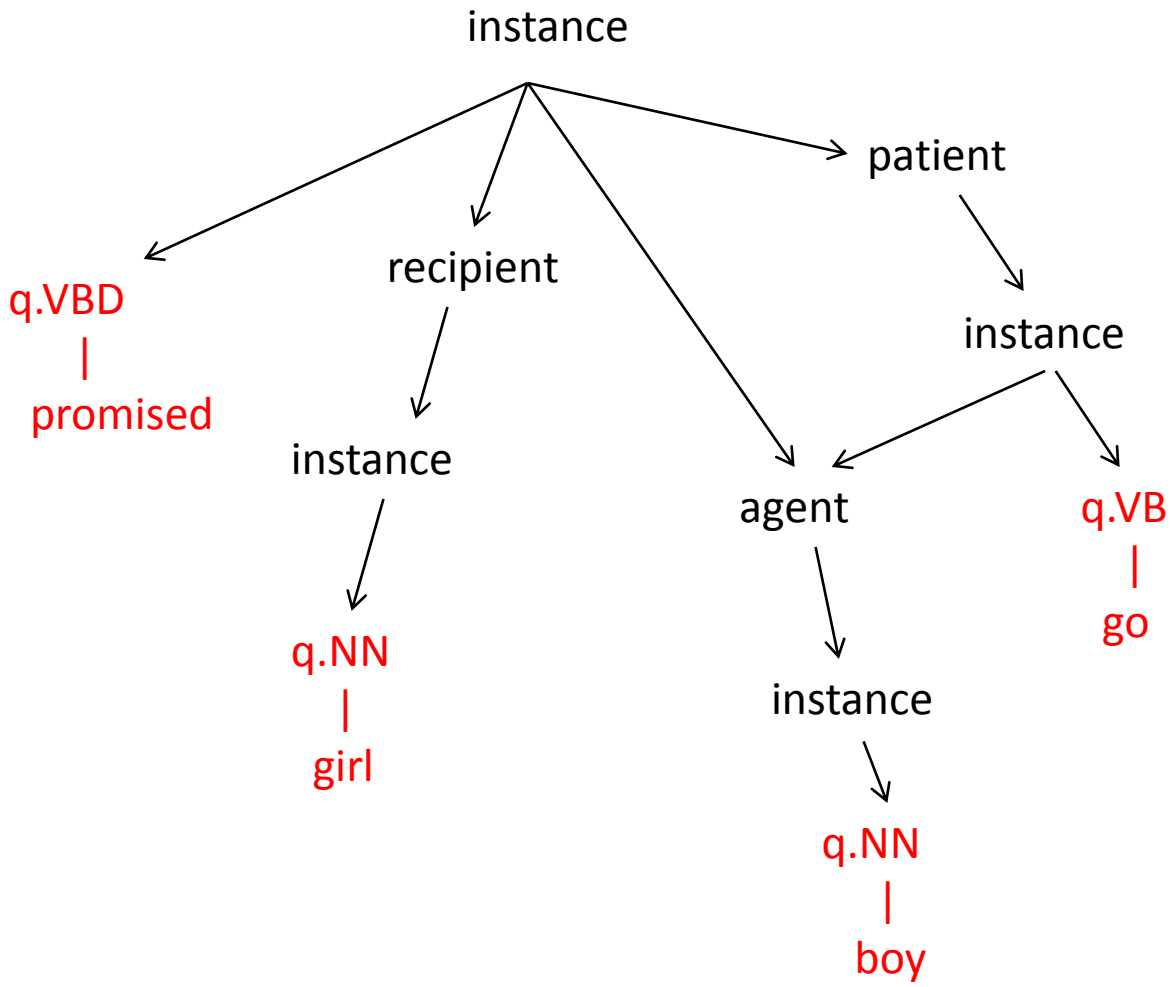
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



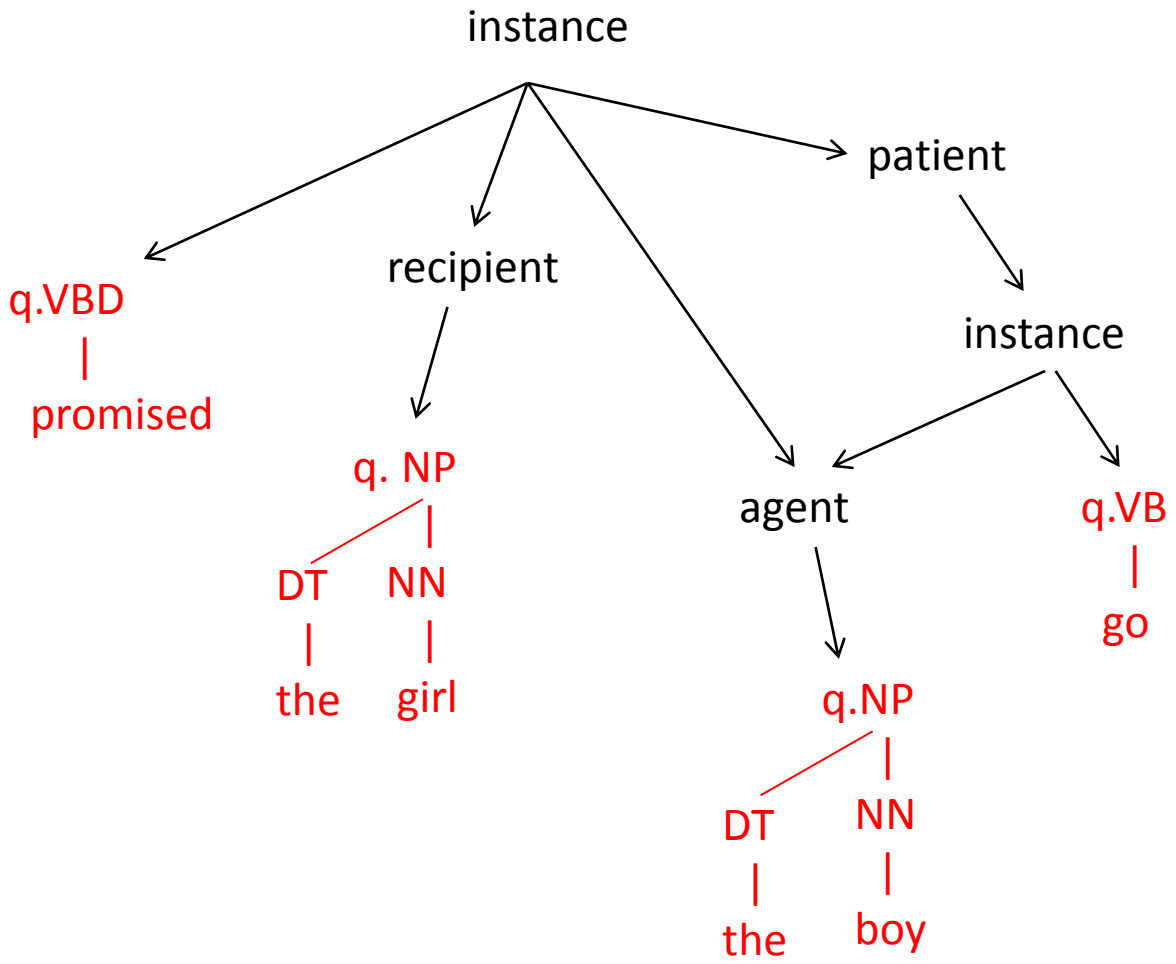
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



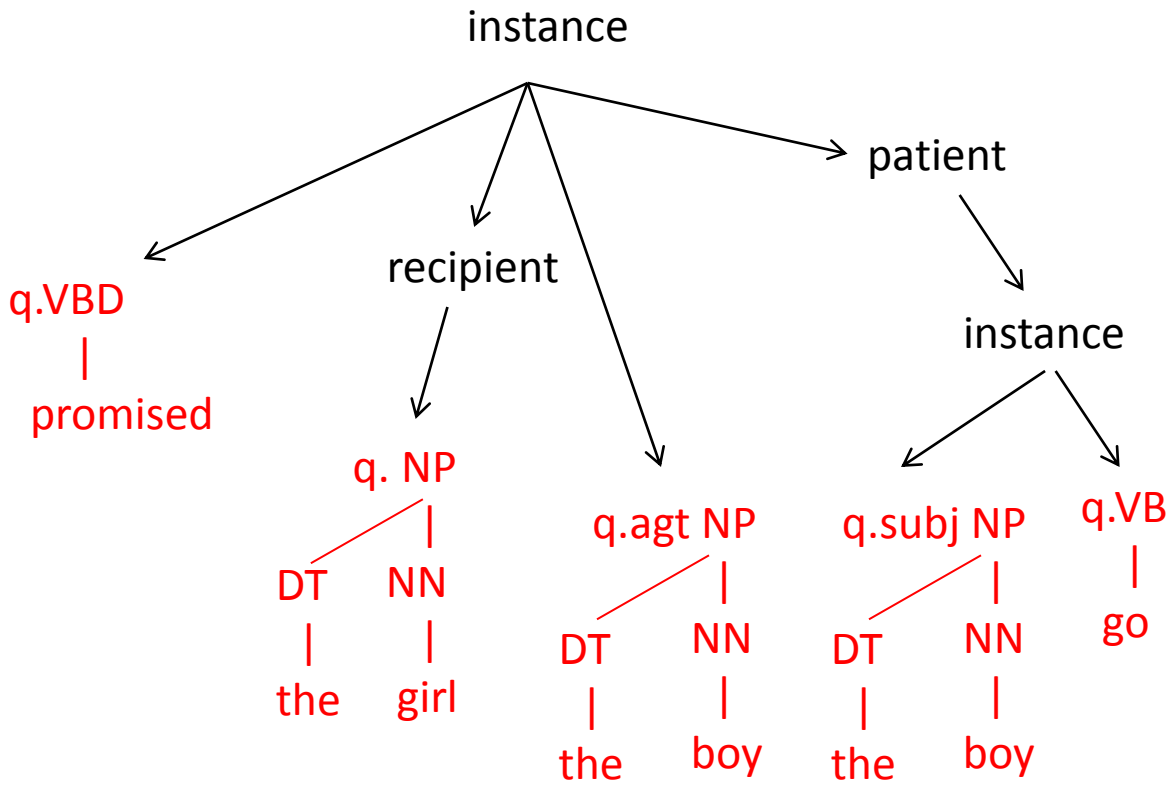
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



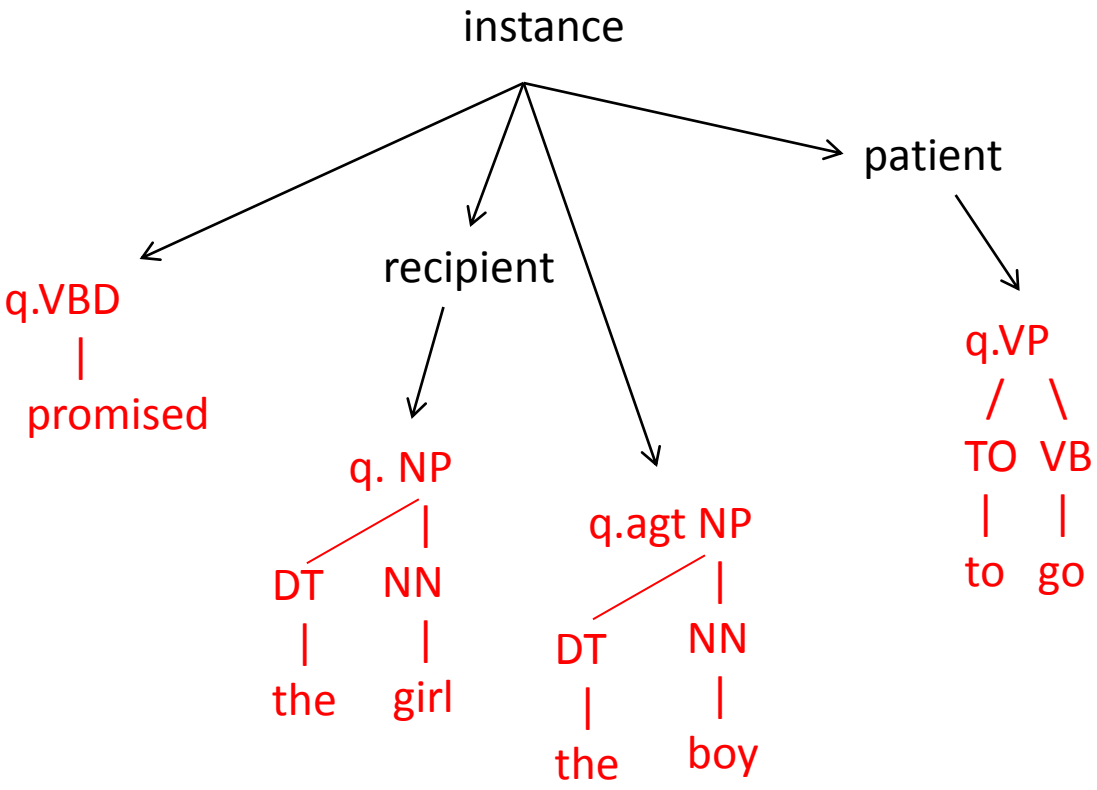
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



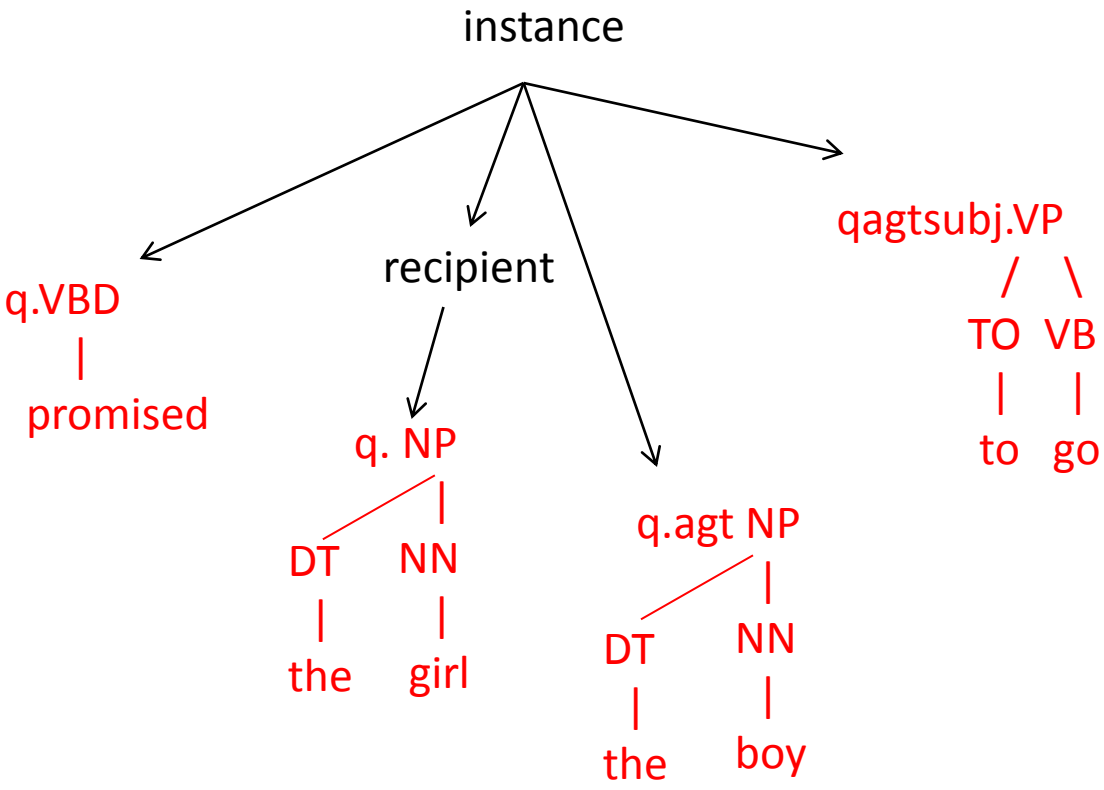
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



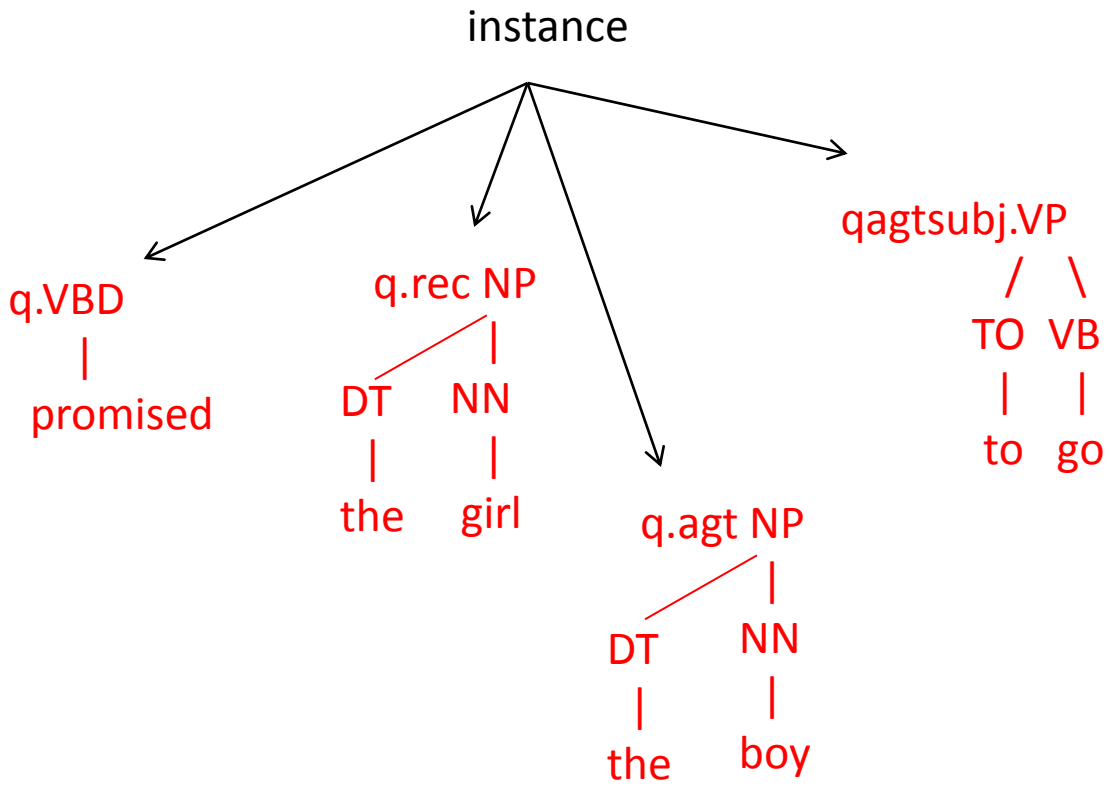
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



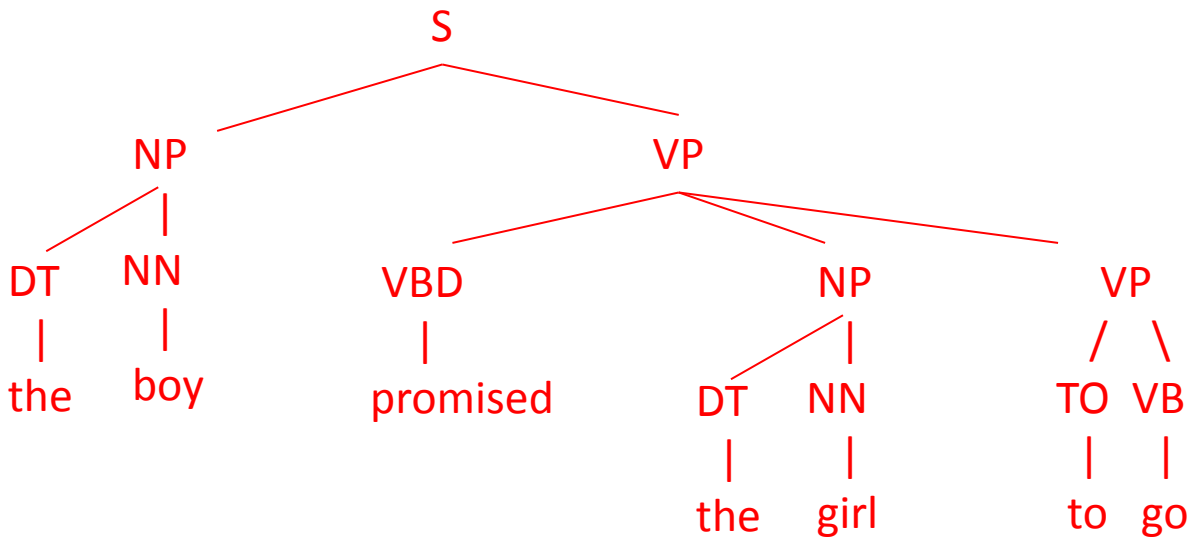
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



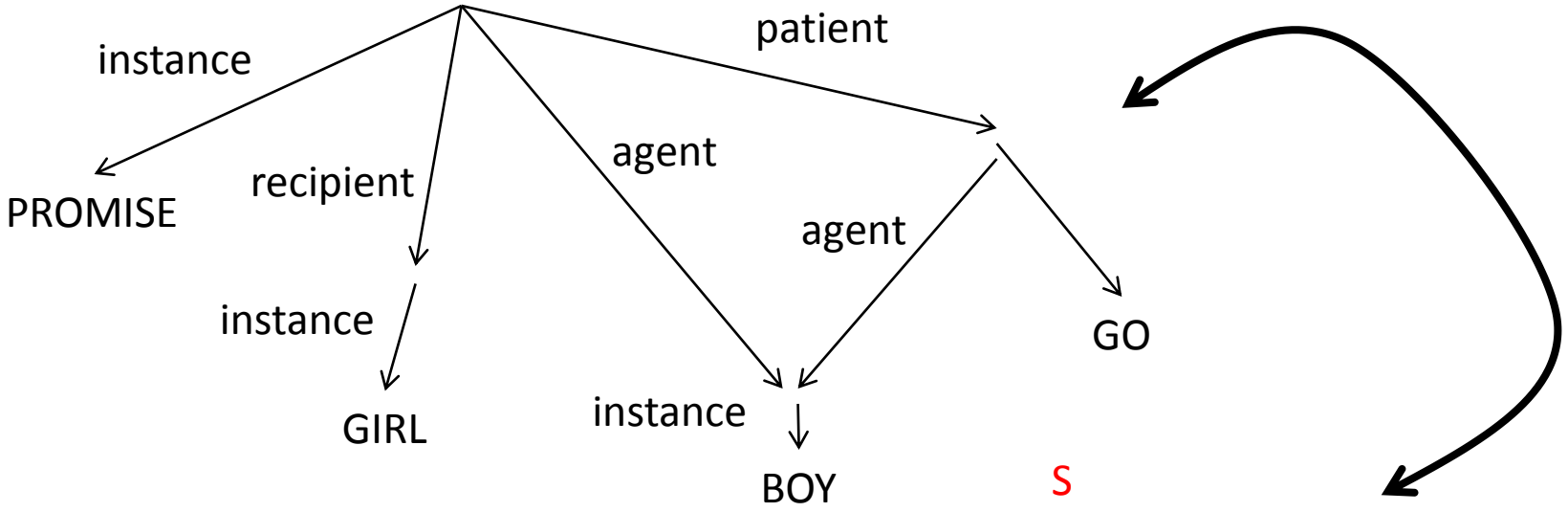
Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

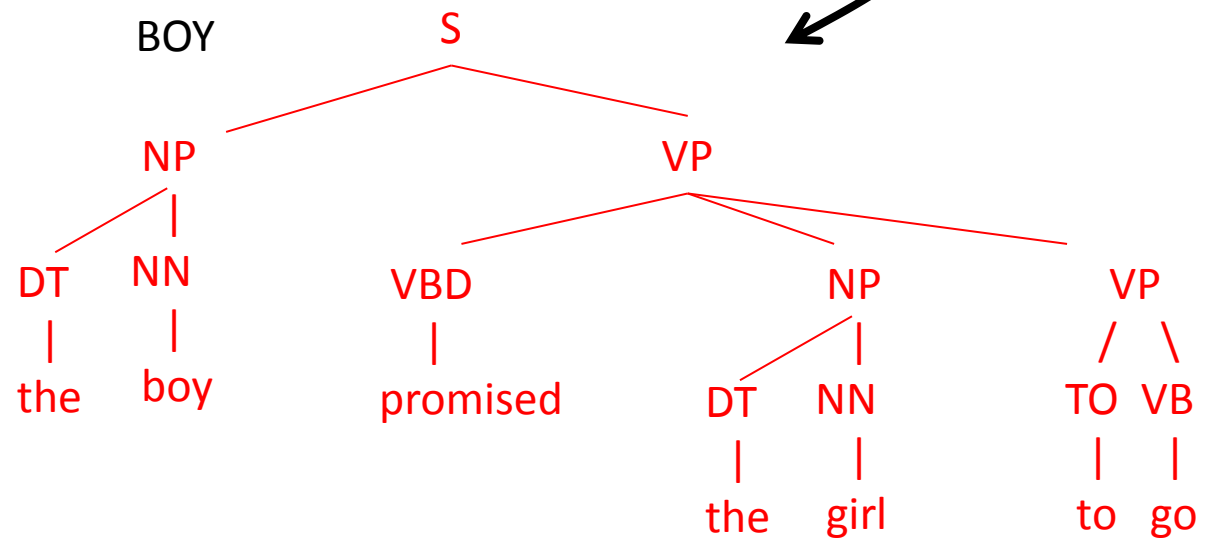


Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

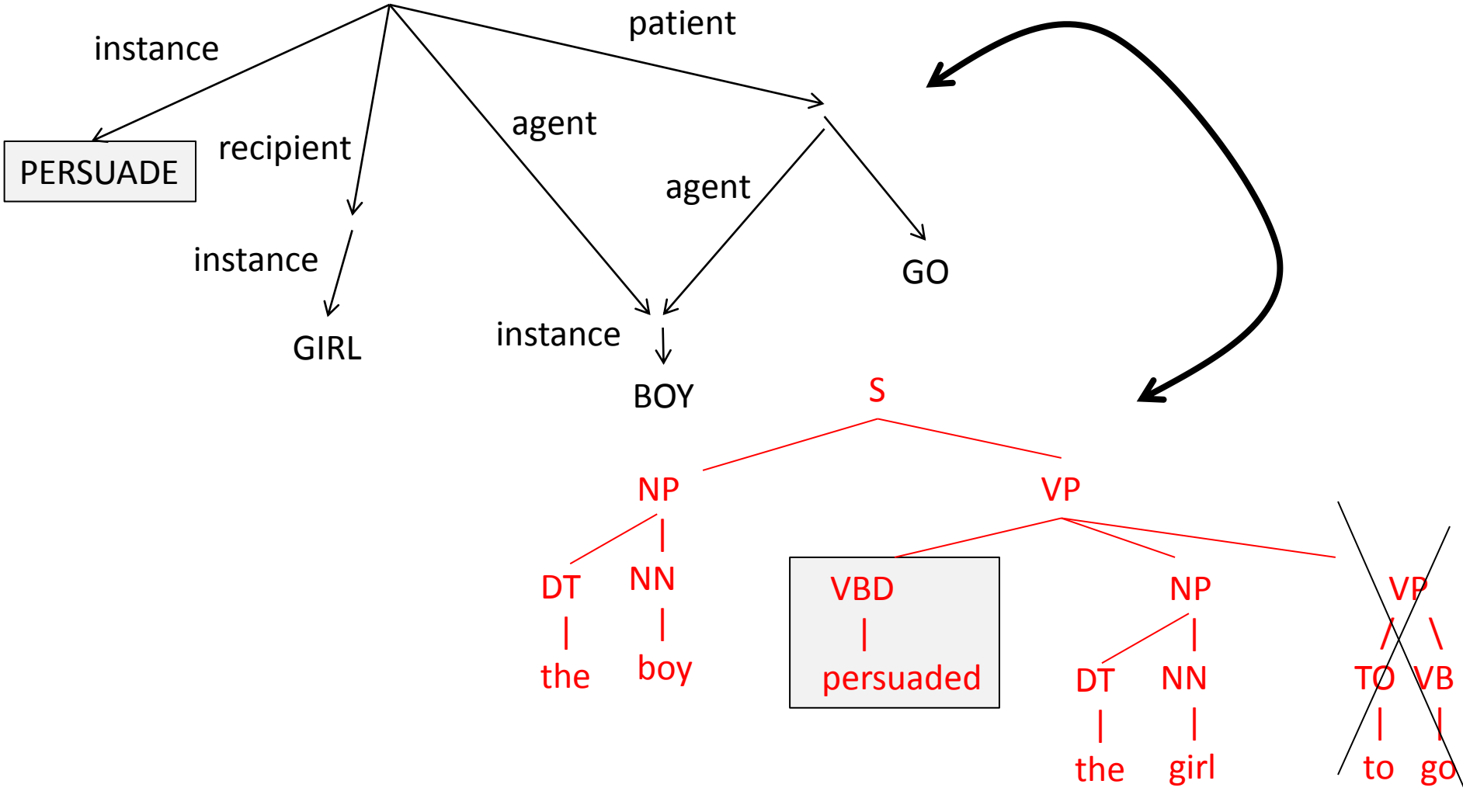


Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

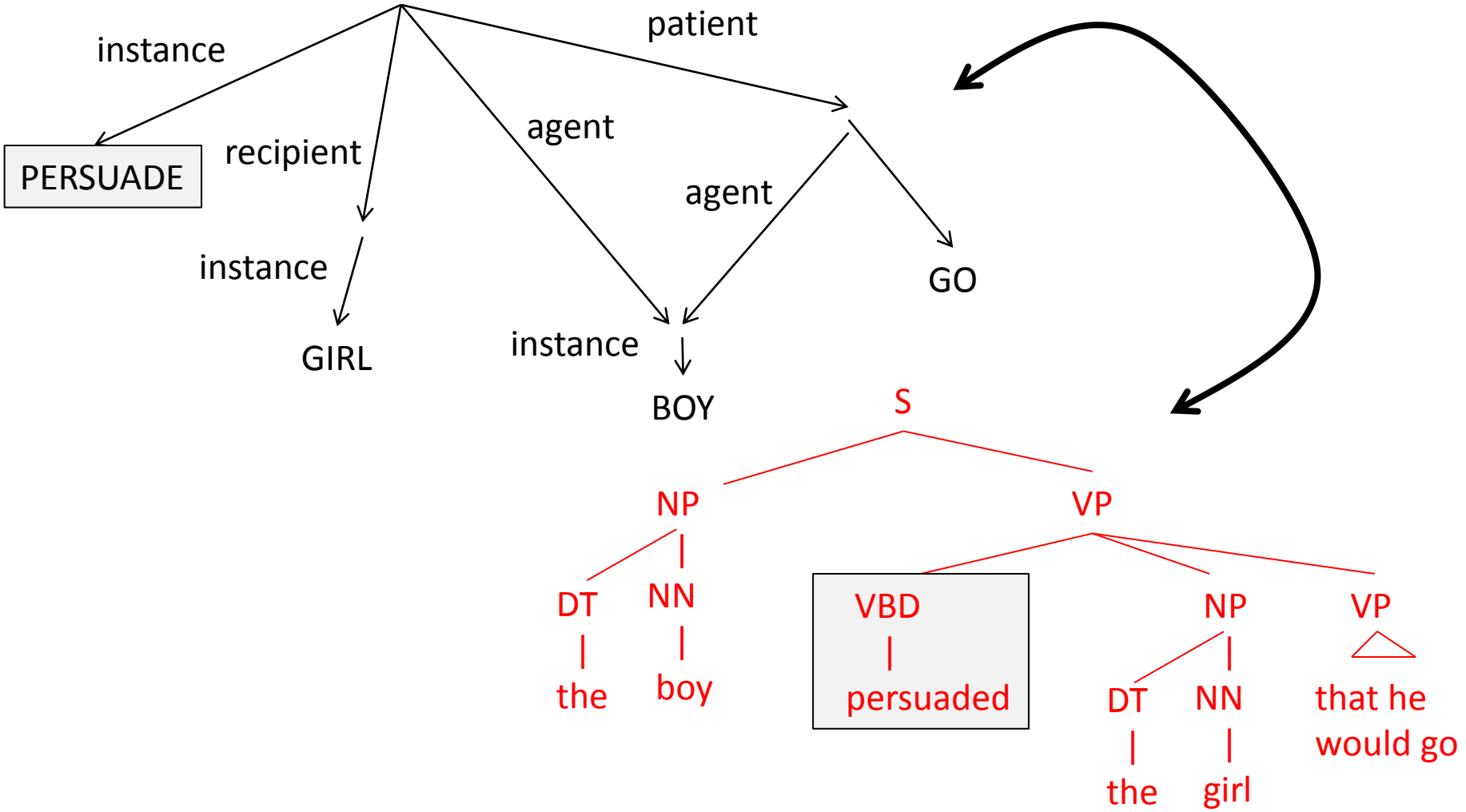




Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)



Bottom-Up DAG-to-Tree Transduction (Kamimura & Slutski 82)

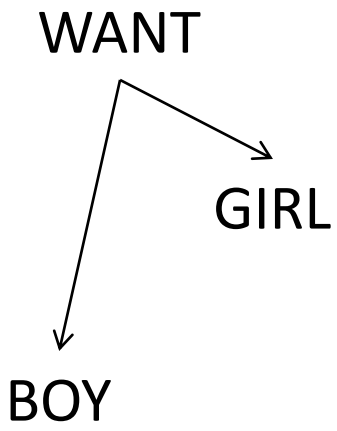


A Semantic Microworld

Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}

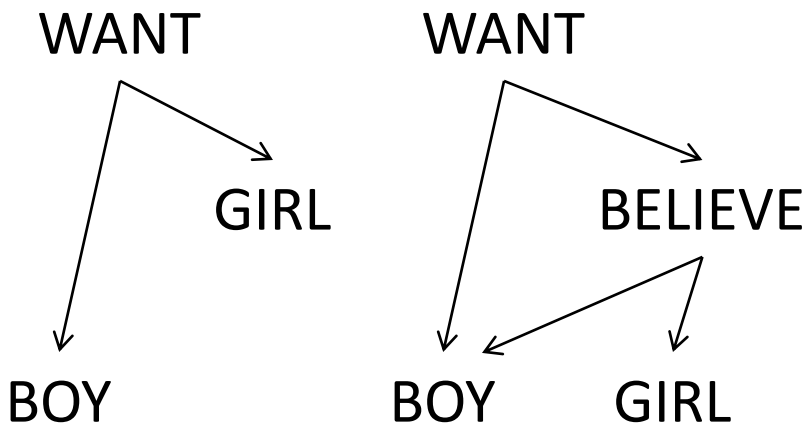
A Semantic Microworld

Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}



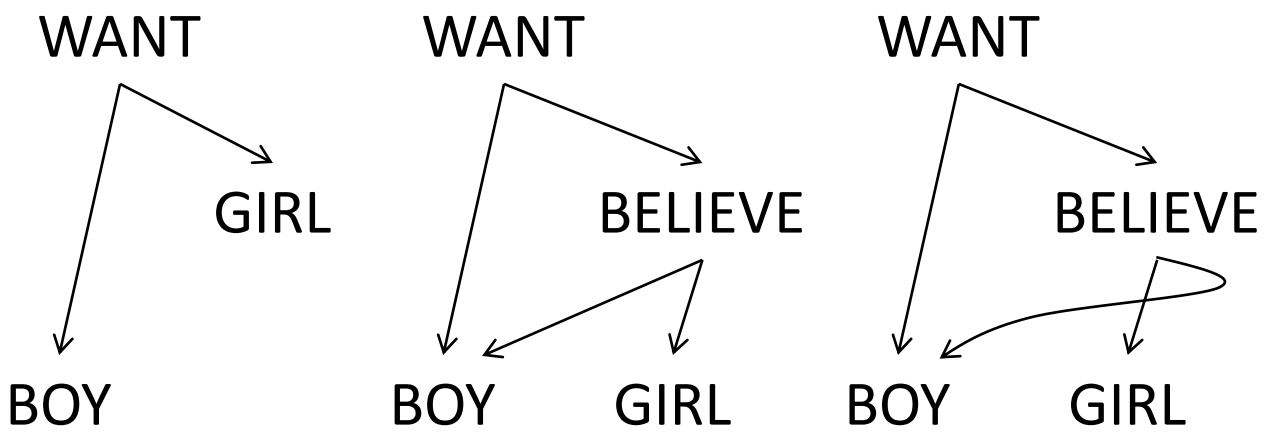
A Semantic Microworld

Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}



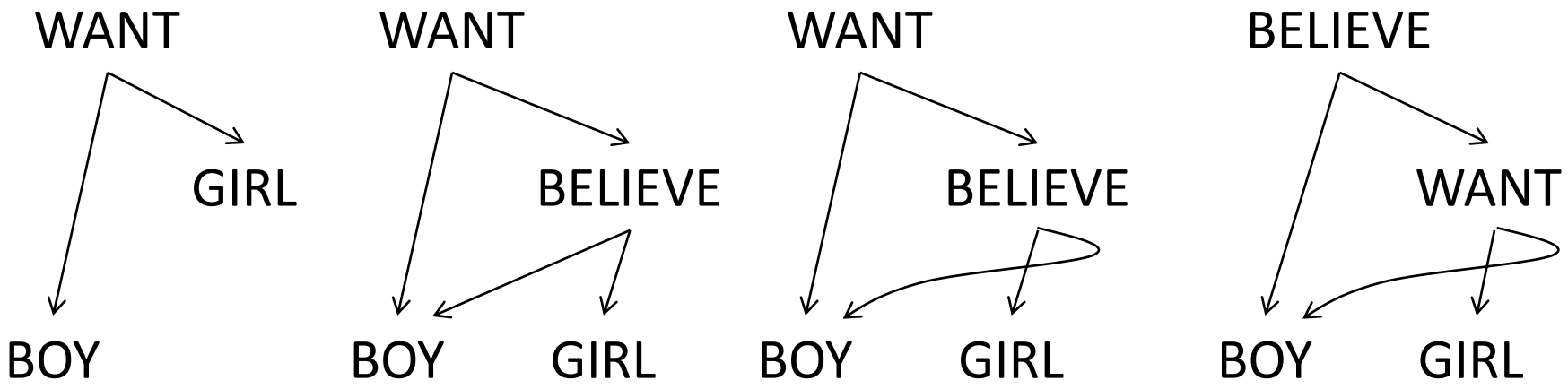
A Semantic Microworld

Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}



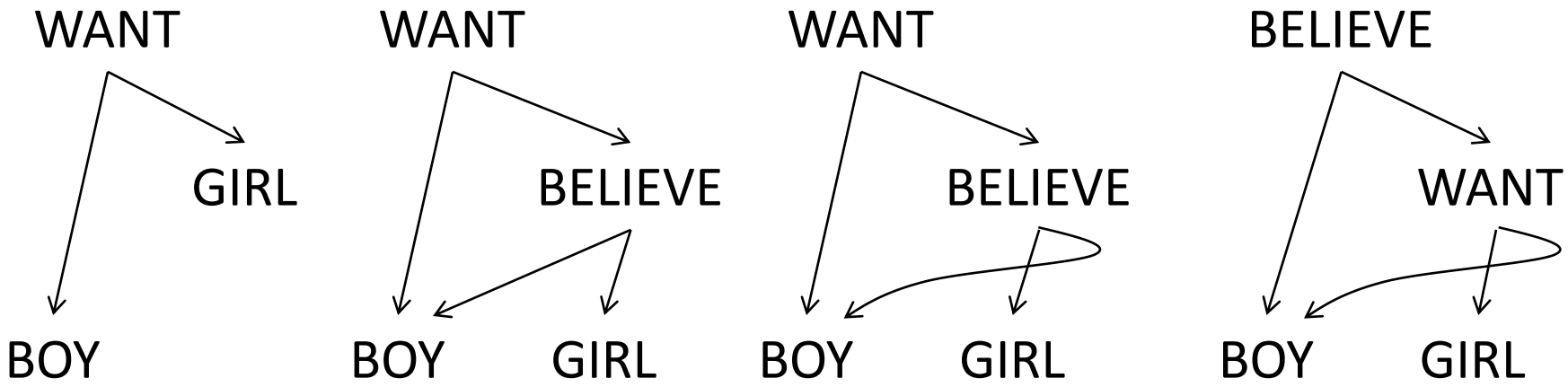
A Semantic Microworld

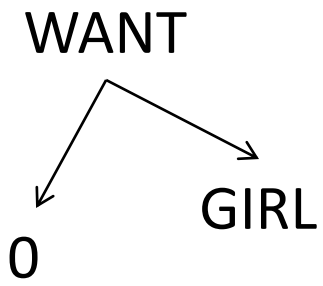
Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}



A Semantic Microworld

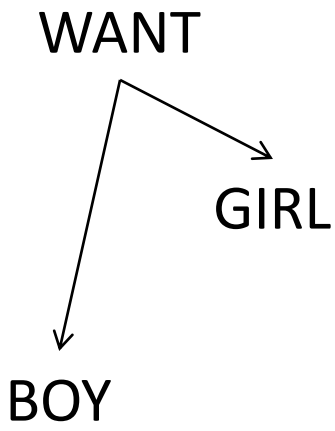
Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}

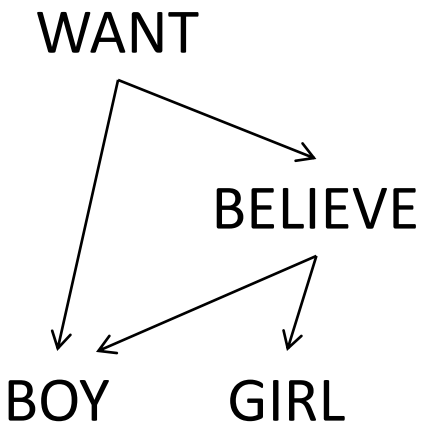


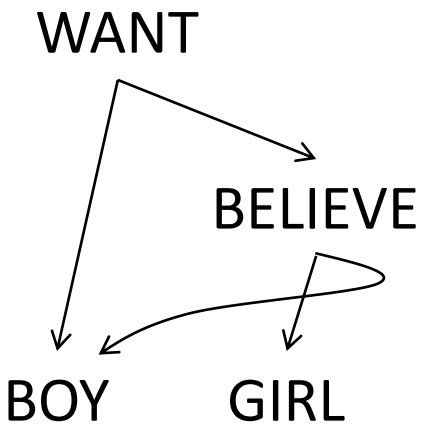


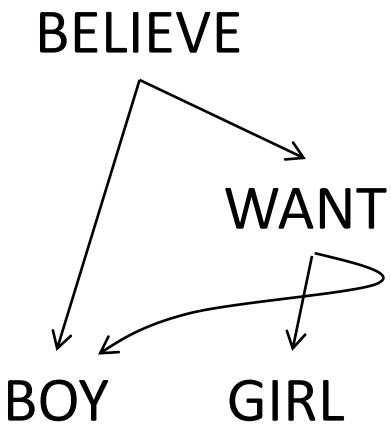
A Semantic Microworld

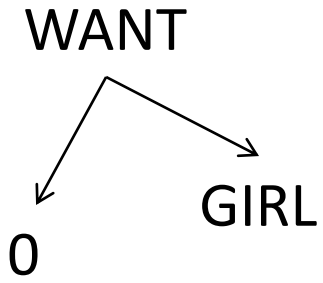
Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}

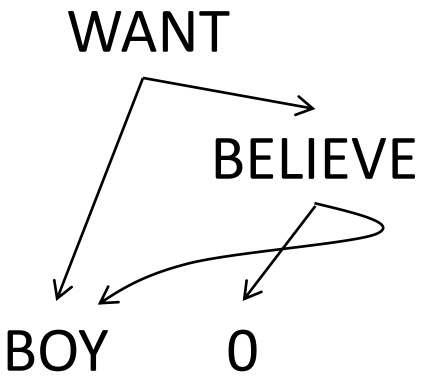






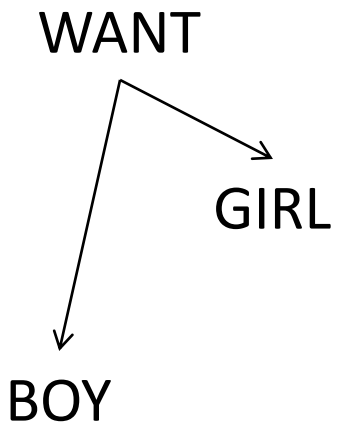


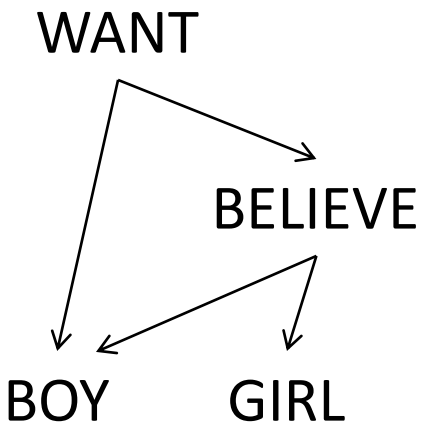


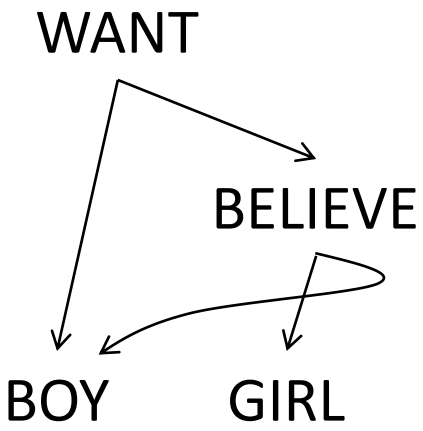


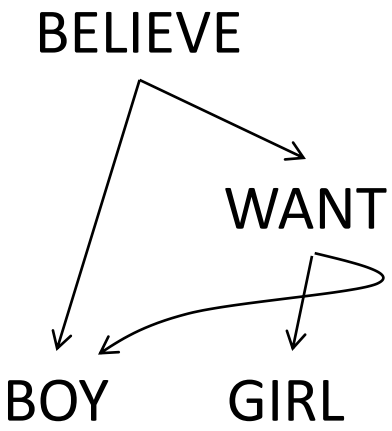
A Semantic Microworld

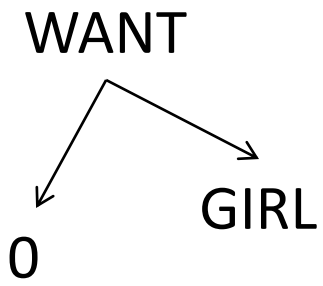
Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}

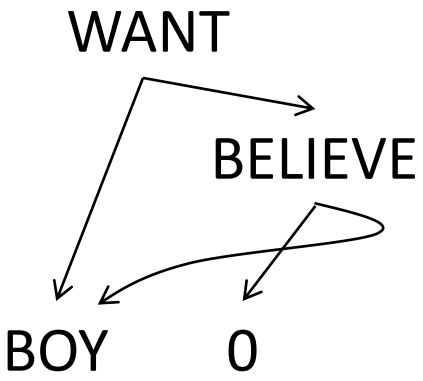


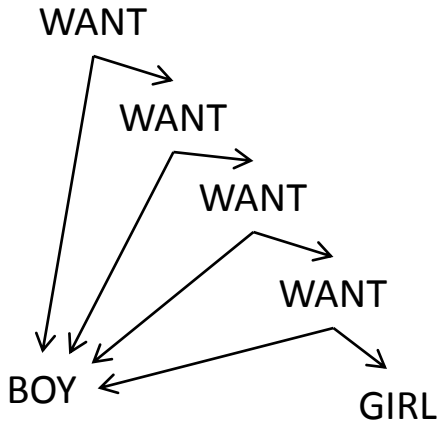






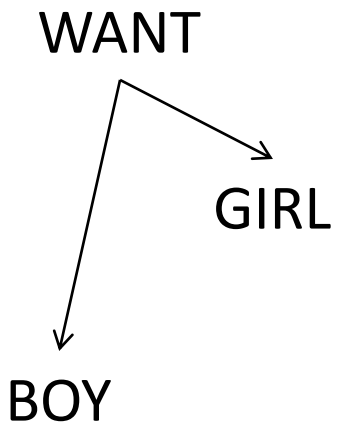


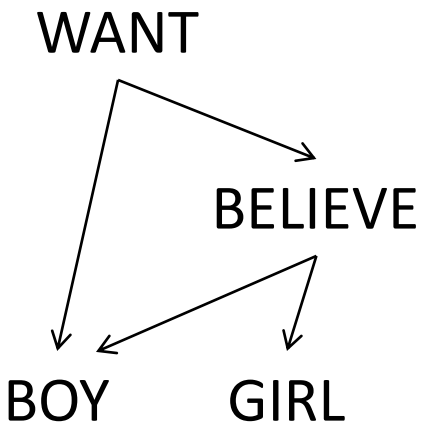


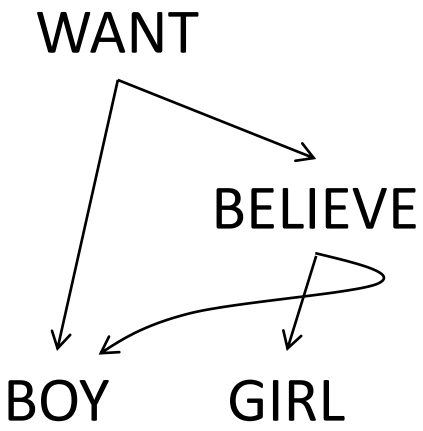


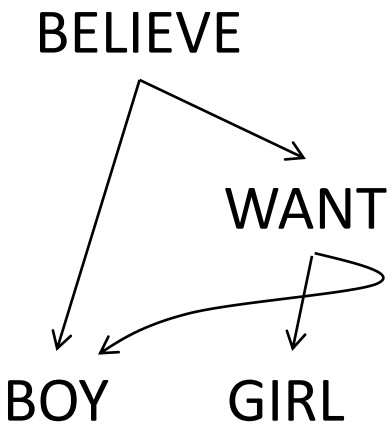
A Semantic Microworld

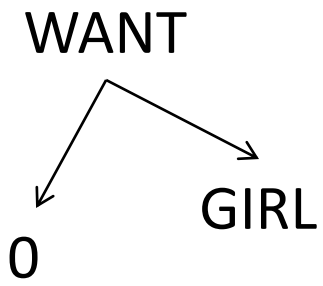
Node Labels: {WANT, BELIEVE, BOY, GIRL, 0}

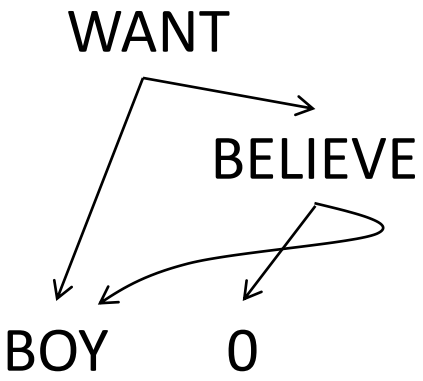


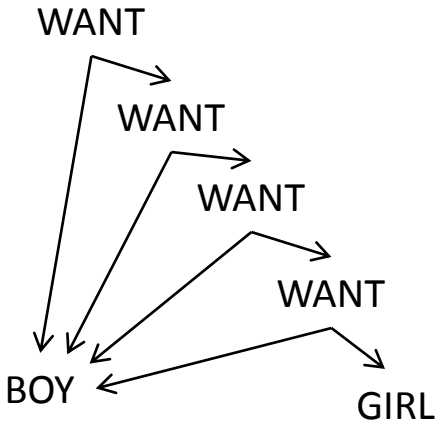


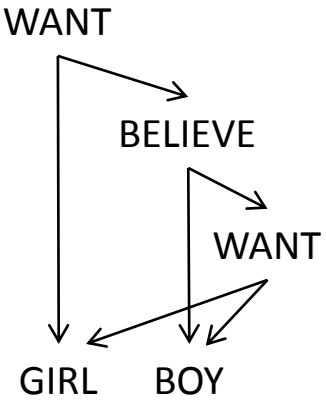












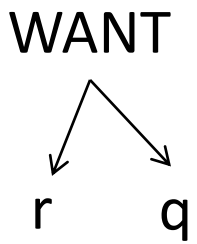
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r

q

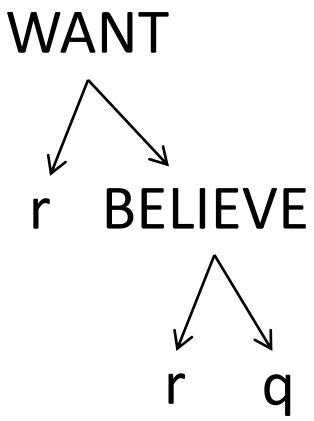
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



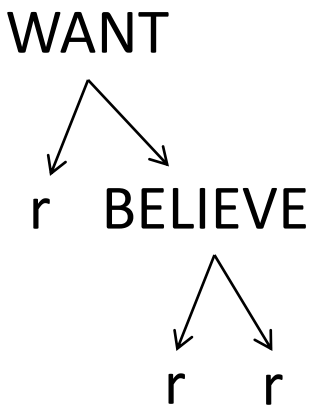
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



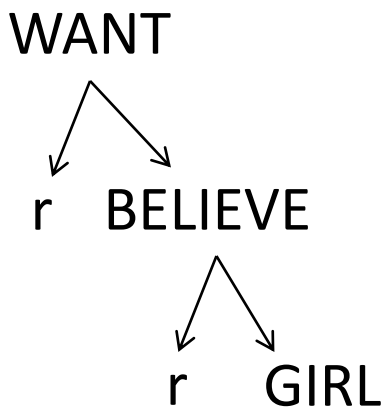
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



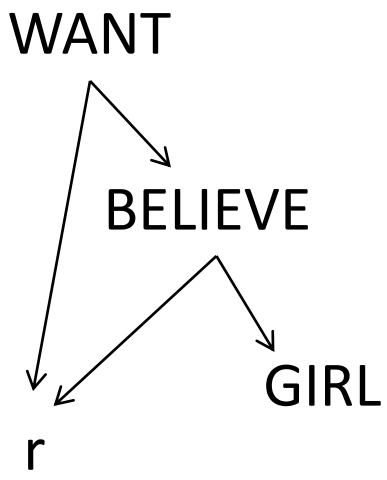
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



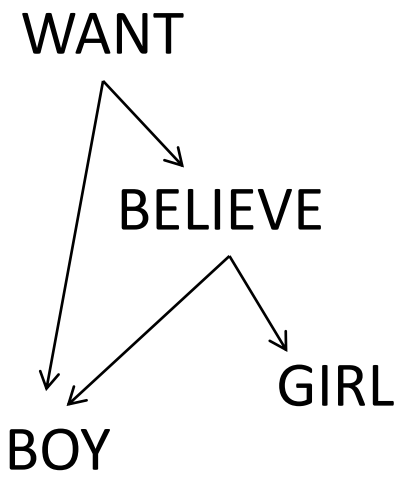
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



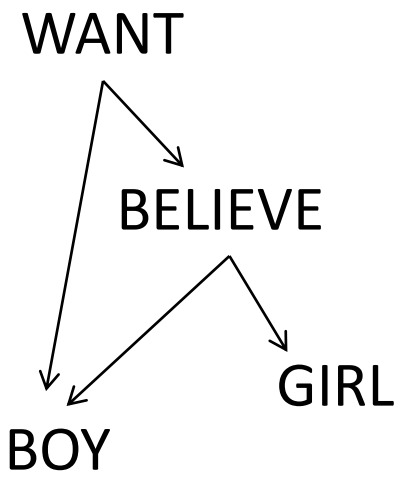
DAG Acceptor

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r

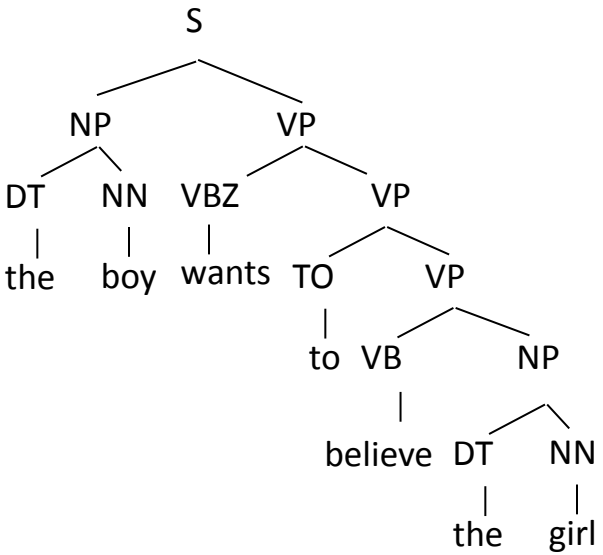


DAG Transducer

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r

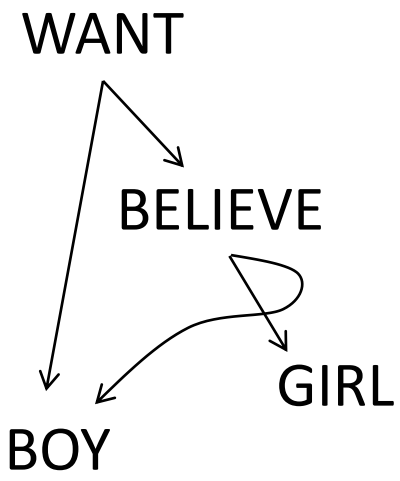


→

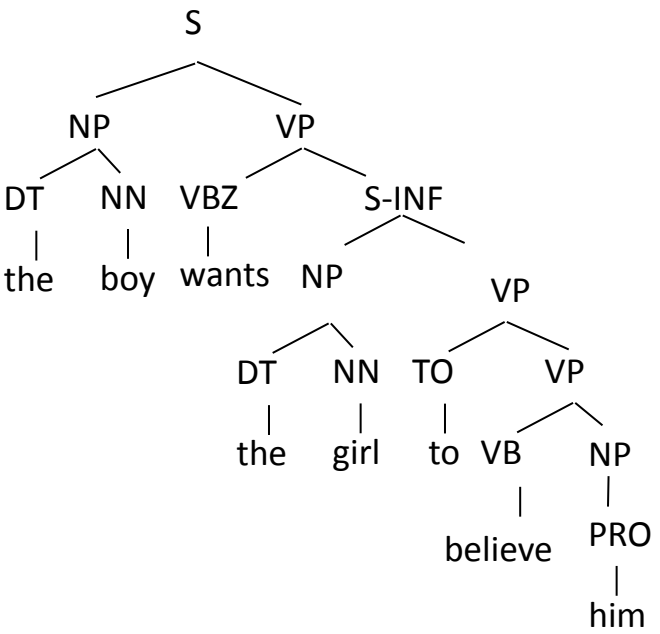


DAG Transducer

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r



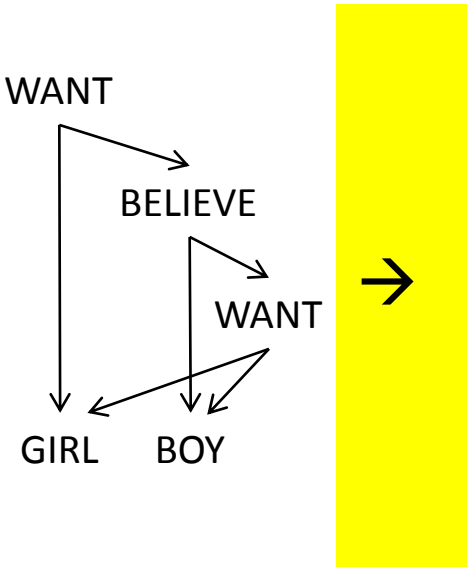
→

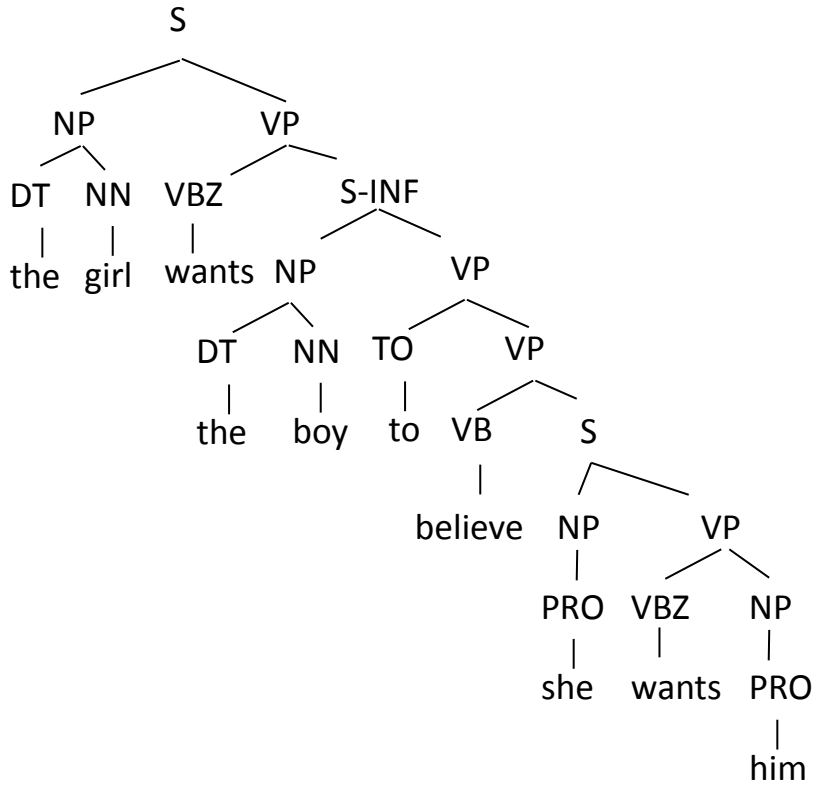


or, “the boy wants to be believed by the girl”

DAG Transducer

q
q → WANT(r q)
q → BELIEVE(r q)
q → r | 0
r → BOY | GIRL | 0
[r r] → r
[r r r] → r





DAG Transducer

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{wants}, q_{accg}.y)$

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{wants}, q_{reflb}.y)$

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{wants}, q_{accb}.y)$

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{wants}, q_{reflg}.y)$

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{wants}, q_{infb}.y)$

$q_s.\text{WANT}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{wants}, q_{infg}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to want}, q_{accg}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to want}, q_{reflb}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to want}, q_{accb}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to want}, q_{reflg}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to want}, q_{infb}.y)$

$q_{infb}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to want}, q_{infg}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to want}, q_{accb}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to want}, q_{reflg}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to want}, q_{accg}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to want}, q_{reflg}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to want}, q_{infg}.y)$

$q_{infg}.\text{WANT}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to want}, q_{infb}.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{believes}, q_{accg}.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{believes}, q_{reflb}.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{believes}, q_{accb}.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{believes}, q_{reflg}.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomb}.x, \text{believes that}, q_s.y)$

$q_s.\text{BELIEVE}(x, y) \rightarrow \text{S}(q_{nomg}.x, \text{believes}, q_s.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to believe}, q_{accg}.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to believe}, q_{reflb}.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to believe}, q_{accb}.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to believe}, q_{reflg}.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerob}.x, \text{to believe}, thatq_s.y)$

$q_{infb}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accg}.x, \text{to believe}, thatq_s.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to believe}, q_{accb}.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to believe}, q_{reflg}.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to believe}, q_{accg}.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to believe}, q_{reflb}.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{zerog}.x, \text{to believe that}, qs.y)$

$q_{infg}.\text{BELIEVE}(x, y) \rightarrow \text{INF}(q_{accb}.x, \text{to believe that}, qs.y)$

$q_{nomb}.\text{BOY} \rightarrow \text{NP(the boy)}$

$q_{accb}.\text{BOY} \rightarrow \text{NP(the boy)}$

$[q_{nomb}, q_{nomb}].\text{BOY} \rightarrow \text{NP(the boy), NP(he)}$

$[q_{nomb}, q_{accb}].\text{BOY} \rightarrow \text{NP(the boy), NP(him)}$

$[q_{nomb}, q_{reflb}].\text{BOY} \rightarrow \text{NP(the boy), NP(himself)}$

$[q_{nomb}, q_{zerob}].\text{BOY} \rightarrow \text{NP(the boy), 0}$

$q_{nomg}.\text{GIRL} \rightarrow \text{NP(the girl)}$

$q_{accg}.\text{GIRL} \rightarrow \text{NP(the girl)}$

$[q_{nomg}, q_{nomg}].\text{GIRL} \rightarrow \text{NP(the girl), NP(she)}$

$[q_{nomg}, q_{accg}].\text{GIRL} \rightarrow \text{NP(the girl), NP(her)}$

$[q_{nomg}, q_{reflg}].\text{GIRL} \rightarrow \text{NP(the girl), NP(herself)}$

$[q_{nomg}, q_{zerog}].\text{GIRL} \rightarrow \text{NP(the girl), 0}$

Two DAG Transducers

foreign text

→-1

```
(s / say
  :agent (s2 / source :mod (r / rebel))
  :patient (a / and
           :op1 (c / close-on
                  :agent (f / force :mod (g / government))
                  :patient (o / outpost :mod (r2 / rebel))
                  :temporal-locating (t / thursday))
           :op2 (s / shower
                  :agent f
                  :patient (c2 / city
                             :mod (m / mountain)
                             :mod (w / west)
                             :name "Zintan")
                  :instrument (m2 / missile))
           :op3 (a2 / attack
                  :agent f
                  :patient (i / insurgent
                             :agent-of (h / hole-up
                                         :pp-near (b / border
                                                    :poss (c3 / country
                                                            :name "Tunisia"]
```

→

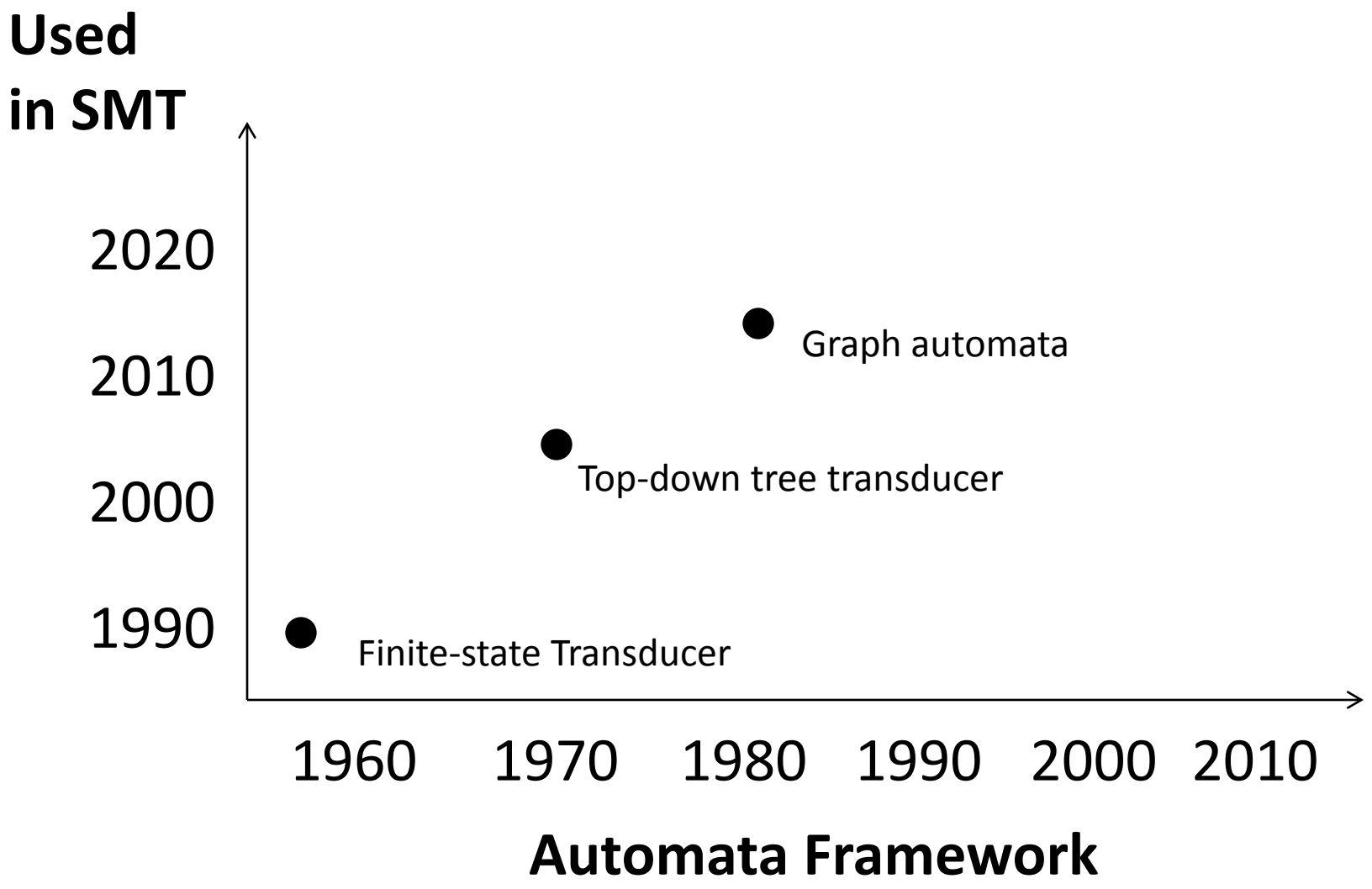
Government forces closed on rebel outposts on Thursday, showering the western mountain city of Zintan with missiles and attacking insurgents holed up near the Tunisian border, according to rebel sources.

DAG Transducer applies in reverse

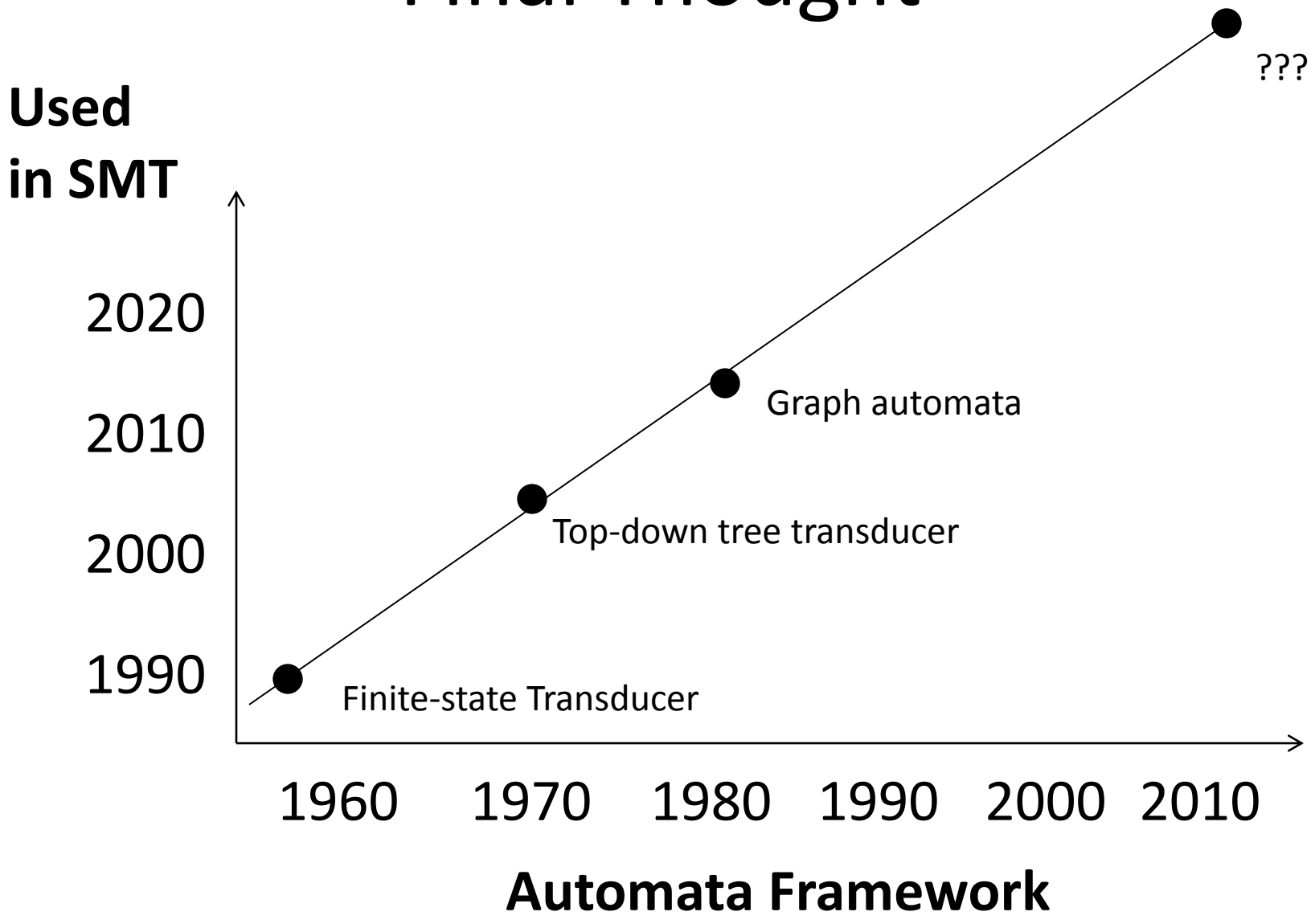
General-Purpose Algorithms

	String Automata Algorithms	Tree Automata Algorithms	Graph Automata Algorithms?
N-best …	… paths through an WFSA (Viterbi, 1967; Eppstein, 1998)	… trees in a weighted forest (Jiménez & Marzal, 2000; Huang & Chiang, 2005)	
EM training	Forward-backward EM (Baum/Welch, 1971; Eisner 2003)	Tree transducer EM training (Graehl & Knight, 2004)	
Determinization…	… of weighted string acceptors (Mohri, 1997)	… of weighted tree acceptors (Borchardt & Vogler, 2003; May & Knight, 2005)	
Intersection	WFSA intersection	Tree acceptor intersection	
Applying transducers	string → WFST → WFSA	tree → TT → weighted tree acceptor	
Transducer composition	WFST composition (Pereira & Riley, 1996)	Many tree transducers not closed under composition (Maletti et al 09)	
General tools	FSM, Carmel, OpenFST	Tiburon (May & Knight 10)	

Final Thought



Final Thought

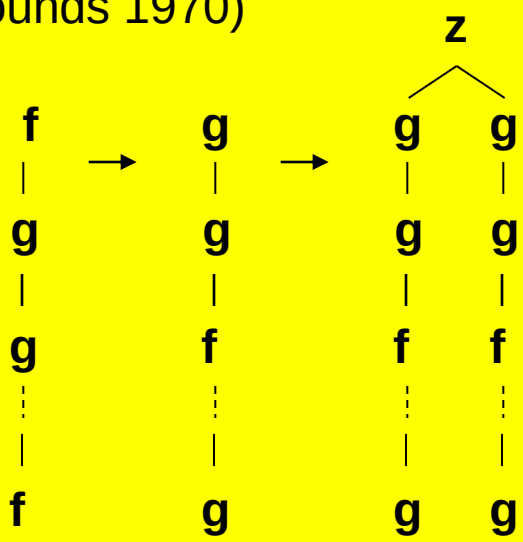


end

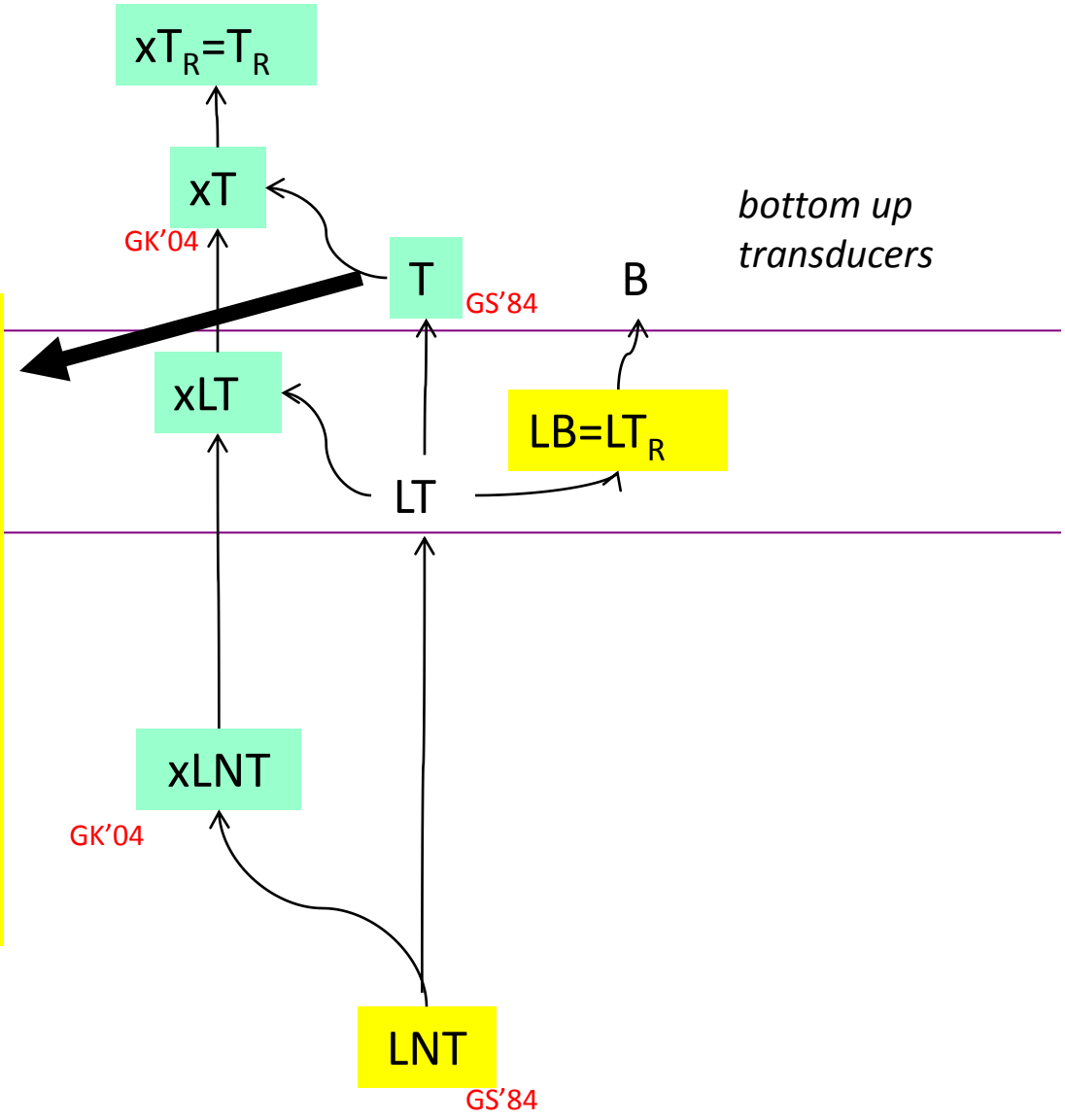
Expressive enough for local rotation

Closed under composition

T is **not closed** under composition (Rounds 1970)

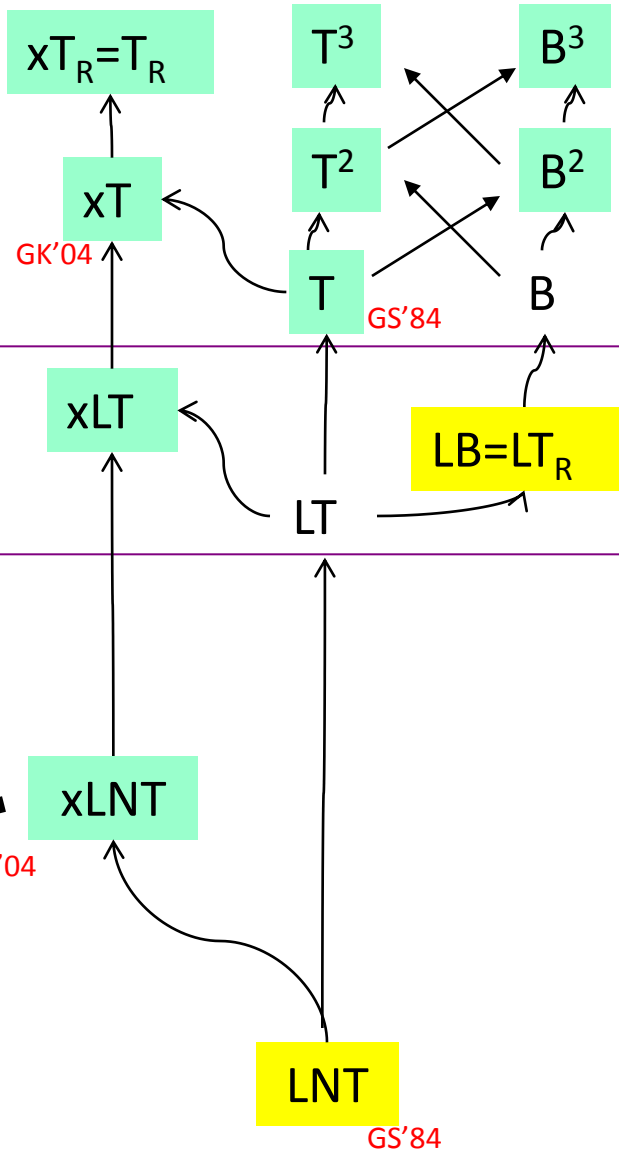


T cannot *Nprocess-then-copy*

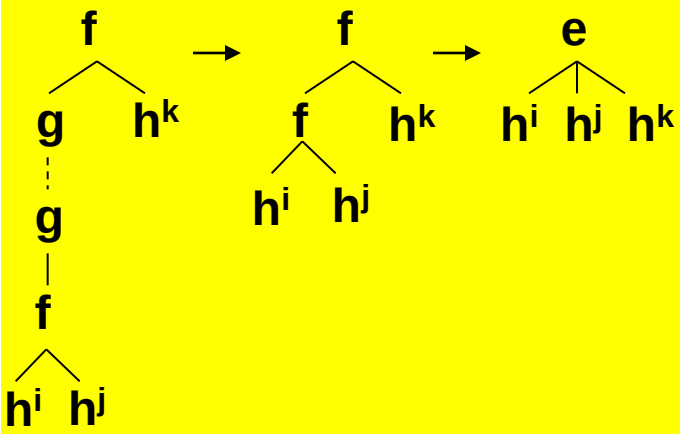


Expressive enough for local rotation

Closed under composition



xLNT is **not closed** under composition

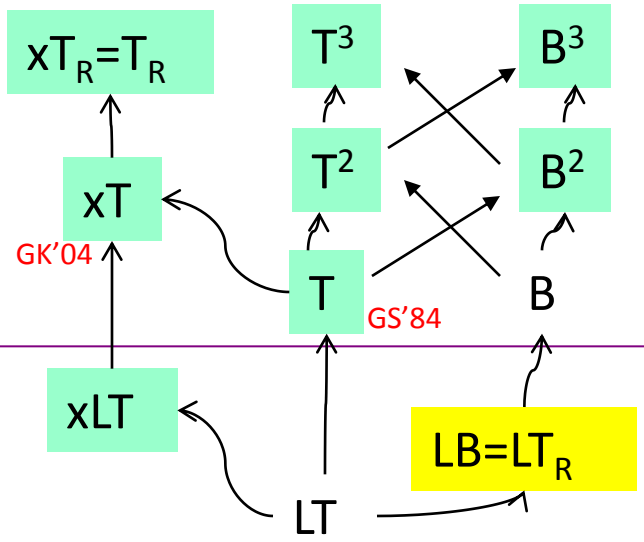


Composition theorems for x-transducers in Maletti, Graehl, Hopkins, Knight (submitted)

Expressive enough for local rotation

Closed under composition

Generalizes FST

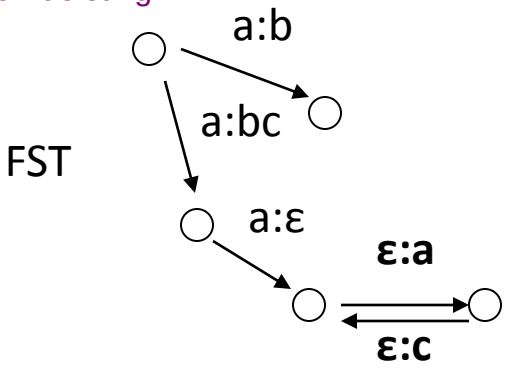


copying

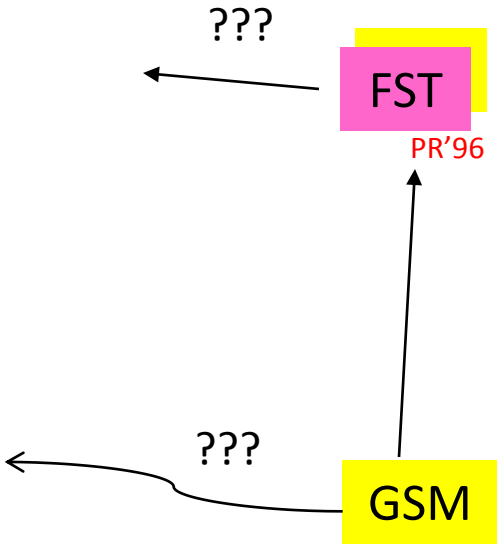
non-copying

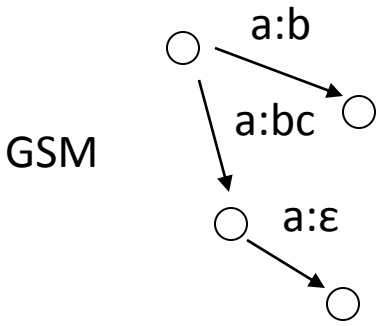
deleting

non-deleting

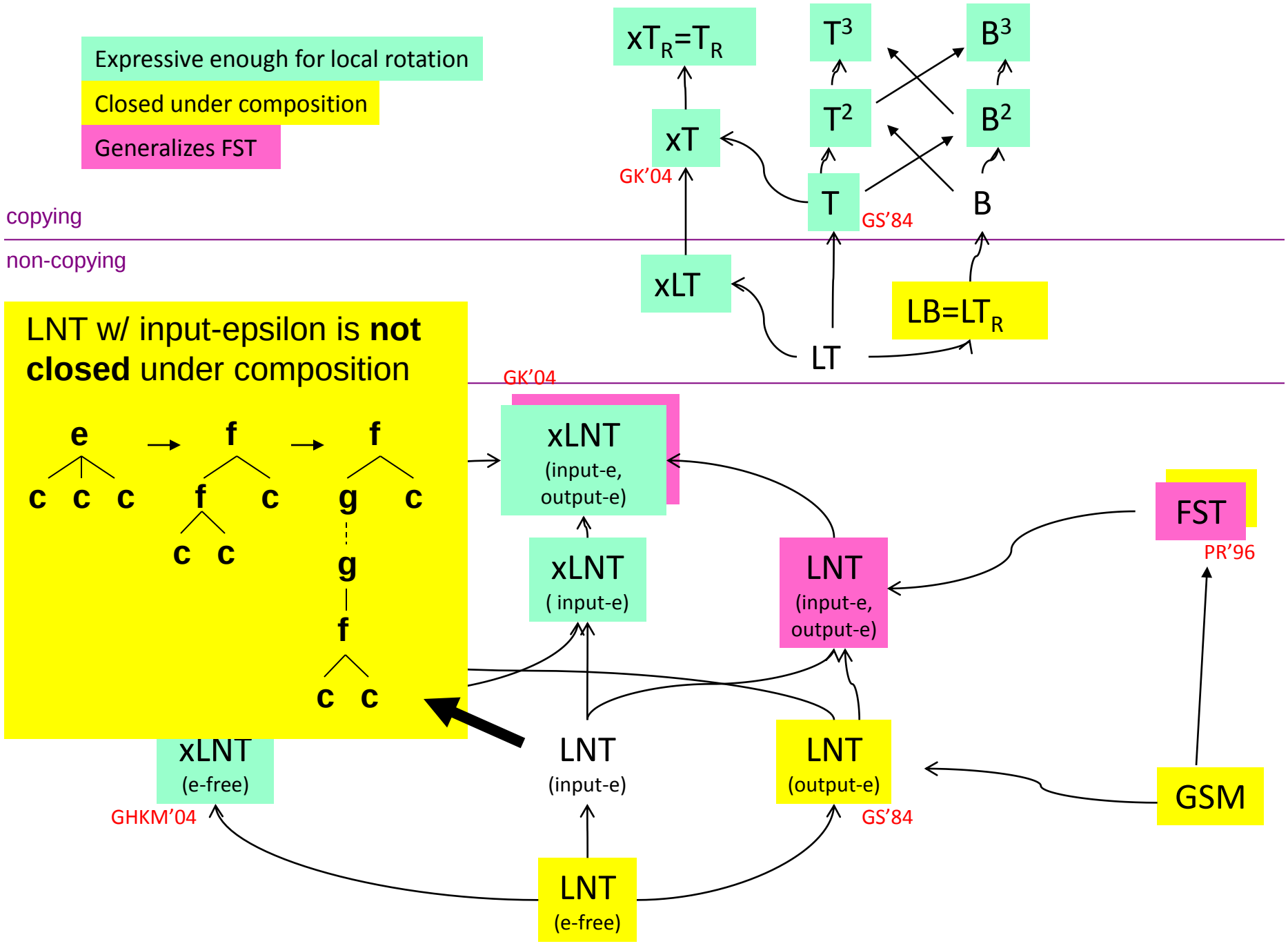


power to generate unbounded amounts of output





every transition must eat up one input symbol.


Expressive enough for local rotation
Closed under composition
Generalizes FST
copying
non-copying
xT_R=T_R
T^3
B^3
xT
T^2
B^2
GK'04
T
GS'84
B
xLT
LB=LT_R
LT
GK'04
LNT w/ input-epsilon is **not closed** under composition
e
c c c
f
f c
c c
f
g c
g
f
c c
xLNT
(input-e, output-e)
xLNT
(input-e)
LNT
(input-e, output-e)
FST
PR'96
xLNT
(e-free)
GHKM'04
LNT
(input-e)
LNT
(output-e)
GS'84
GSM
LNT
(e-free)

Expressive enough for local rotation

Closed under composition

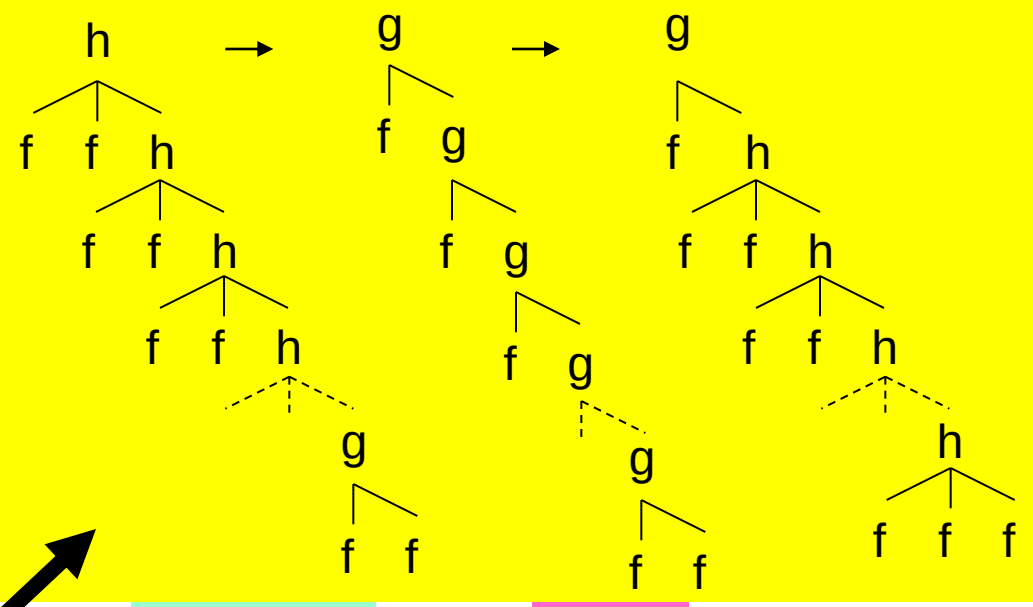
Generalizes FST

$xT_R=T_R$

T^3

B^3

xLNT (e-free) is **not closed** under composition



copying

non-copying

deleting

non-deleting

xLNT
(output-e)

xLNT
(e-free)

GHKM'04

LNT
(input-e)

LNT
(e-free)

output-e)

LNT
(output-e)

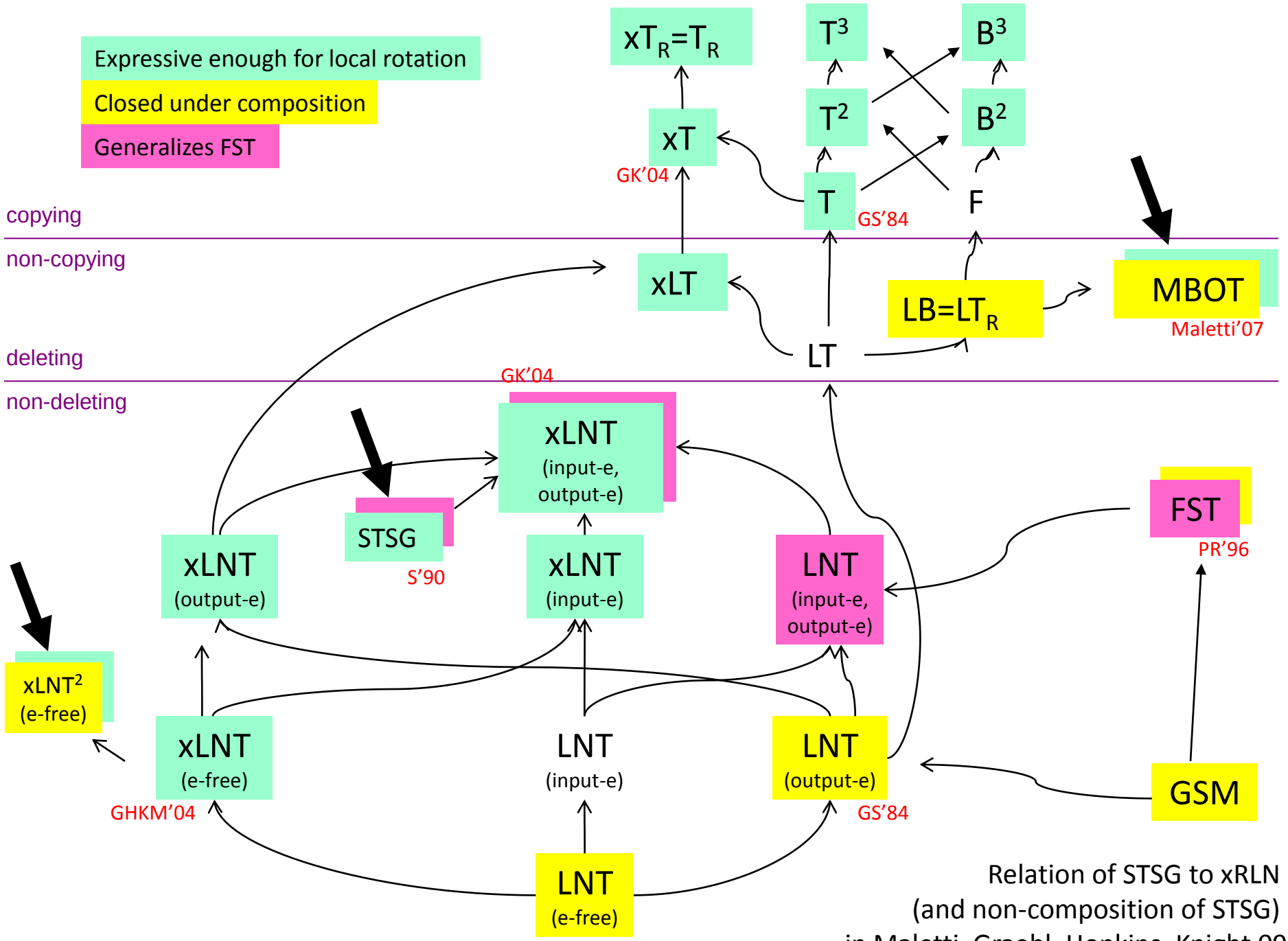
GS'84

FST

PR'96

GSM

Composition theorems for e-free in Arnold & Dauchet 82



Relation of STSG to xRLN
(and non-composition of STSG)
in Maletti, Graehl, Hopkins, Knight 09