Building Natural Language System based on Theoretical Linguistics

理論言語学に基づいた自然言語処理システム

@MiCS 2019/10/23

Masashi Yoshikawa (NAIST D3)

Self Introduction

- NAIST Matsumoto-ken D3
- Like: syntactic/semantic parsing, structured prediction
- Originally from Osaka Univ. (Foreign Studies)
  - mainly worked on Turkish and Arabic languages
- Spent 2.5 years of my Ph.D period at Bekki-sensei's lab (Ochanomizu Univ.), and now back in Nara
  - Surprised to know everyone is working on IE at the lab (no more parsing)

@Kuwait 2012

(Ice Breaker?) Arabic Morphology is

Three Concept Consonant times Syntactic Template

KTB write, DRS study, QRʔ read, ʔKL eat, JDD new
ðHB go, ħML carry, QBL accept, QLL few, QRR decide
ṬLB seek, ɣRB sink, QʕD sit, SJD head down, ...

Three consonants
representing concepts

XaYaZa did, XaaYiZu doer, maXYuuZu is patient to
XaYYaZa made one do, maXYaZa / miXYaZu place to do, XaYiiZu adjective
yaXYaZu do
...

Syntactic Templates
deciding syntactic function

(Ice Breaker?) Arabic Morphology is

Three Concept Consonant times Syntactic Template

KTB write | DRS study | QRʔ read | ʔKL eat | JDD new
ðHB go | ĦML carry | QBL accept | QLL few | QRR decide
ṬLB seek | ɣRB sink | QʕD sit | SJD head down | ...

Three consonants

representing concepts

Fill XYZ with ABC

×

XaYaZa did | XaaYiZu doer | maXYuuZu is patient to
XaYYaZa made one do | maXYaZa miXYaZu place to do | XaYiiZu adjective
yaXYaZu do | ...

Syntactic Templates

deciding syntactic function

KiTaaBu book
KaTaBa wrote
KaaTiBu writer
QaRRaRa decide
QaLiiLu few
ṬaaLiBu student
ĦaaMiLu pregnant
maDRaSa school
maSJiDu mosque
maɣRiDu west
miQʕaD basement

(Ice Breaker?) Arabic Morphology is

Three Concept Consonant times Syntactic Template

KTB write · DRS study · QRʔ read · ʔKL eat · JDD new
ðHB go · ĦML carry · QBL accept · QLL few · QRR decide
ṬLB seek · ɣRB sink · QʕD sit · SJD head down · ...

Three consonants
representing concepts

Fill XYZ
with ABC

XaYaZa did · XaaYiZu doer · maXYuuZu is patient to
XaYYaZa made one do · maXYaZa · miXYaZu place to do · XaYiiZu adjective
yaXYaZu do · ...

Syntactic Templates
deciding syntactic function

Q..RR..Ra · ṬaaLiBu student · maDRaSa school · maɣRiDu west
...aLiiLu few · ĦaaMiLu pregnant · maSJiDu mosque · miQʕaD basement

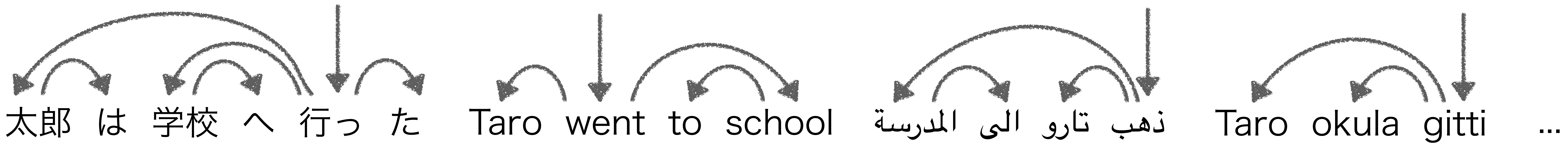
- Sematic languages (Hebrew, Amharic..)
- Implication: recent subword methods are adequate for these languages?
- But its syntax is familiar to us
  - VSO with postpositional modifiers

ايضا	فيها	هاناكو	تدرس	التي	الجديدة	المدرسة	الى	تارو	ذهب
ADV	ADP	PROPN	VERB	PRON	ADJ	NOUN	ADP	PROPN	VERB

Taro went to the new school in which Hanako studies as well

What is Syntactic Theory?

- Provide explanations for phenomena arising from the way words are concatenated
  - PP-attachment: "`John (saw a girl (with a telescope))`"
  - Coordination: "`Wendy (ran 19 miles) and (walked 9 miles)`"
  - control verb, complement, passive/active voice, scope, etc.
- Must be general to cover all languages, while describing language specificities
  - e.g. Universal Dependencies (de Merneffe et al., 2014)



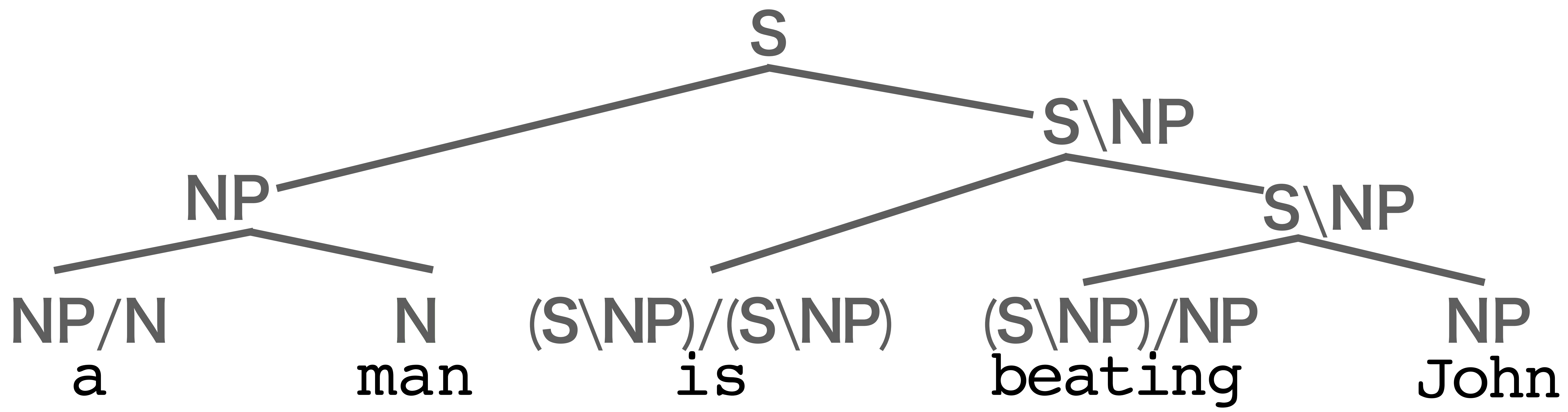
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
  - Meta rules (cf. CFG: $S \rightarrow NP\ VP$)
  - Forward/backward application: $X \rightarrow X/Y\ Y$ $\quad X \rightarrow Y\ X\backslash Y$
  - Forward/backward composition rules: $X/Z \rightarrow X/Y\ \ Y/Z$

X/Y argument

$X\backslash Y$ return value



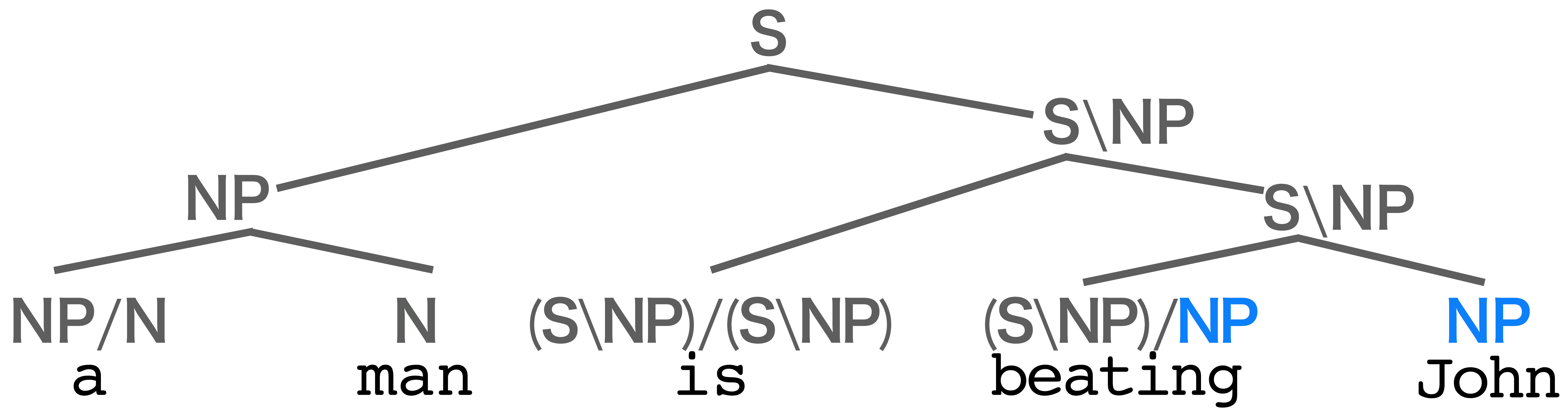
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
  - Meta rules (cf. CFG: `S → NP VP`)
  - Forward/backward application: X → X/Y Y   X → Y X\Y
  - Forward/backward composition rules: X/Z → X/Y Y/Z

X/Y argument

X\Y return value



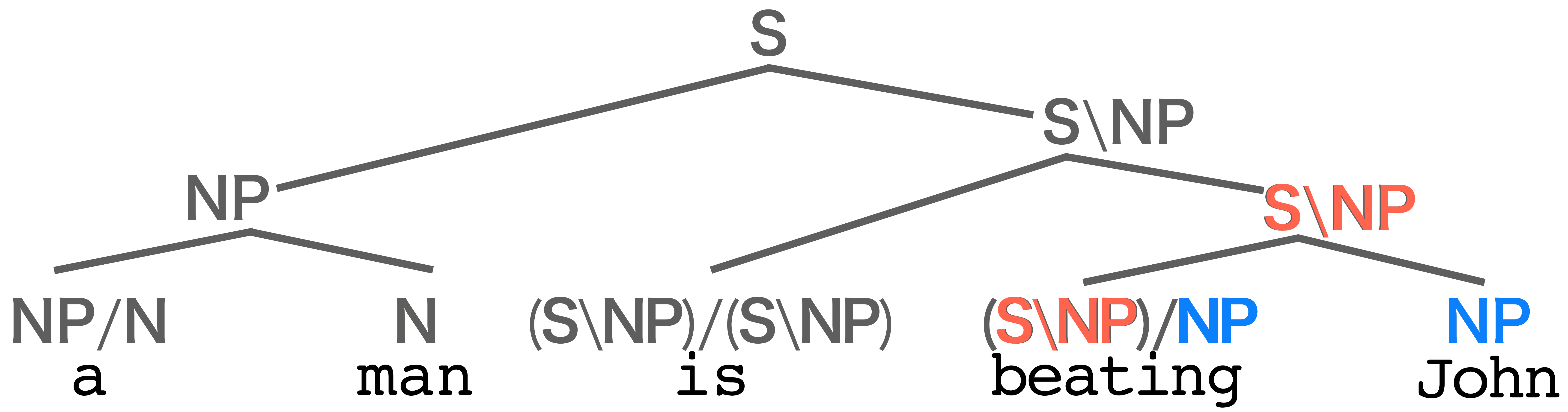
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
    - Meta rules (cf. CFG: S $\rightarrow$ NP VP)
    - Forward/backward application: X $\rightarrow$ X/Y Y   X $\rightarrow$ Y X\Y
    - Forward/backward composition rules: X/Z $\rightarrow$ X/Y Y/Z

X/Y argument

X\Y return value



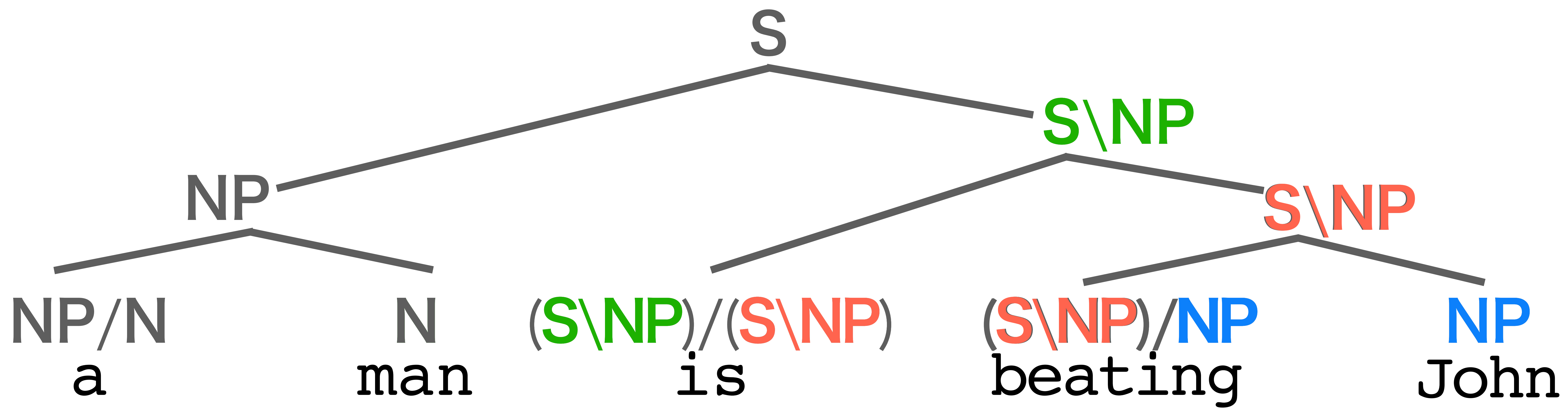
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
  - Meta rules (cf. CFG: `S → NP VP`)
  - Forward/backward application: X → X/Y Y   X → Y X\Y
  - Forward/backward composition rules: X/Z → X/Y Y/Z

X/Y argument

X\Y return value



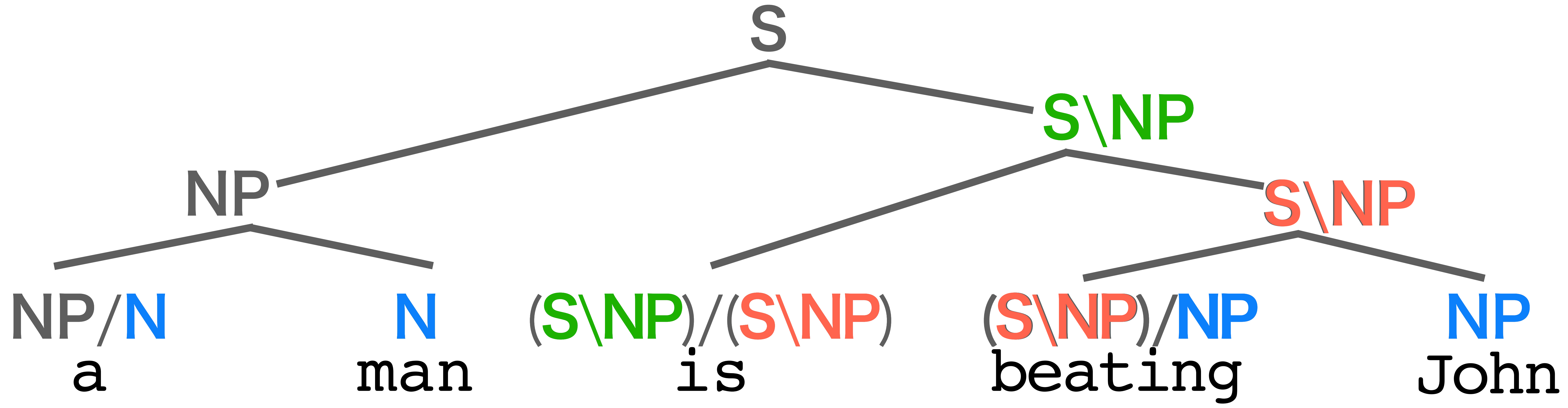
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
    - Meta rules (cf. CFG: `S → NP VP`)
    - Forward/backward application: $X \rightarrow X/Y\ Y$ $\quad X \rightarrow Y\ X\backslash Y$
    - Forward/backward composition rules: $X/Z \rightarrow X/Y\ \ Y/Z$

X/Y argument

$X\backslash Y$ return value



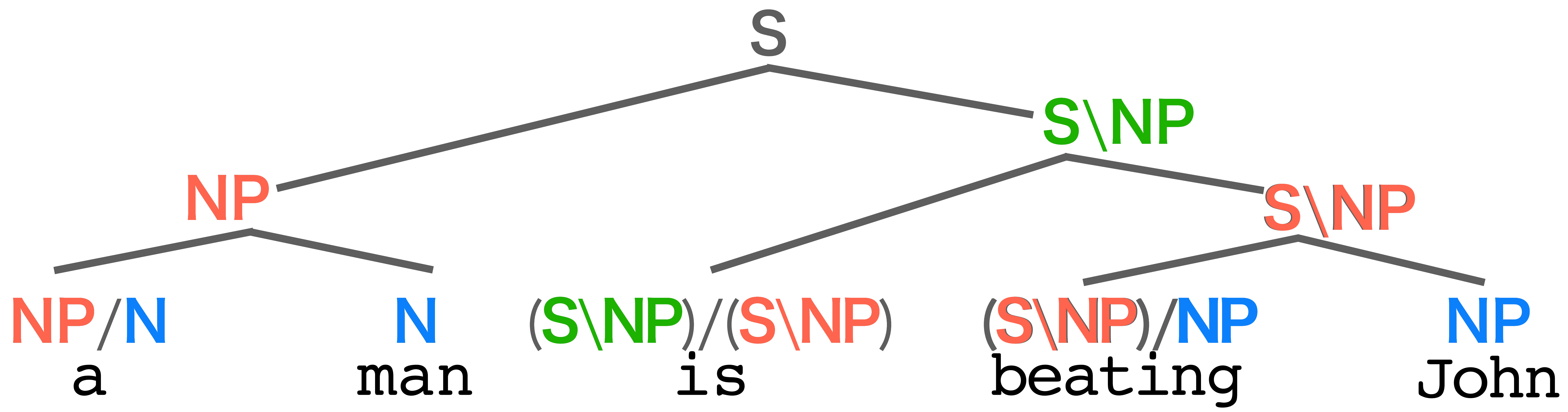
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
    - Meta rules (cf. CFG: `S → NP VP`)
    - Forward/backward application: $X \rightarrow X/Y\ Y \qquad X \rightarrow Y\ X\backslash Y$
    - Forward/backward composition rules: $X/Z \rightarrow X/Y\ \ Y/Z$

X/Y argument

$X\backslash Y$ return value



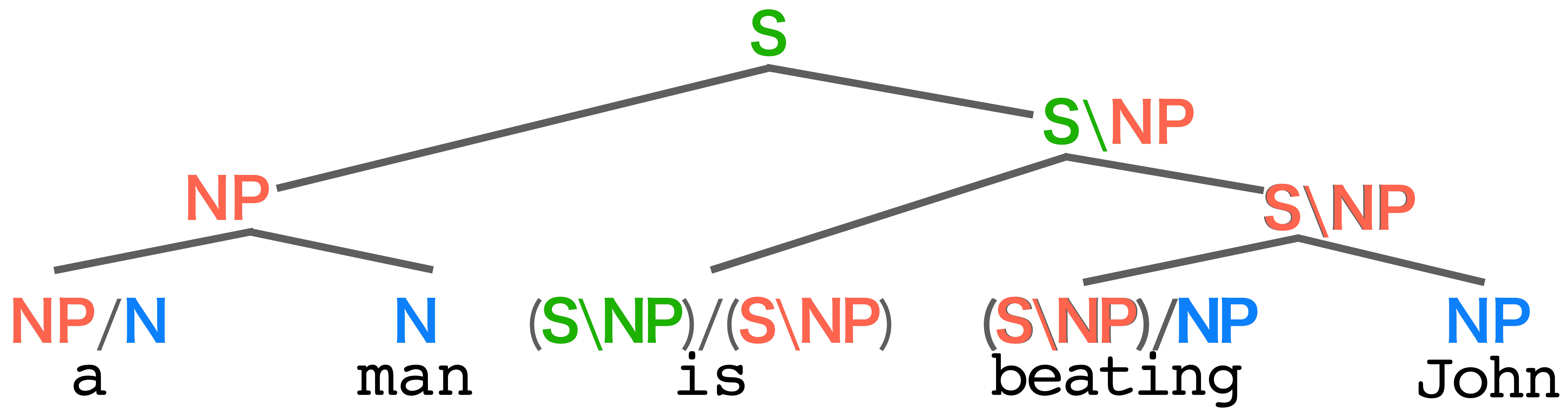
Combinatory Categorial Grammar

Steedman 2000, Bekki 2010

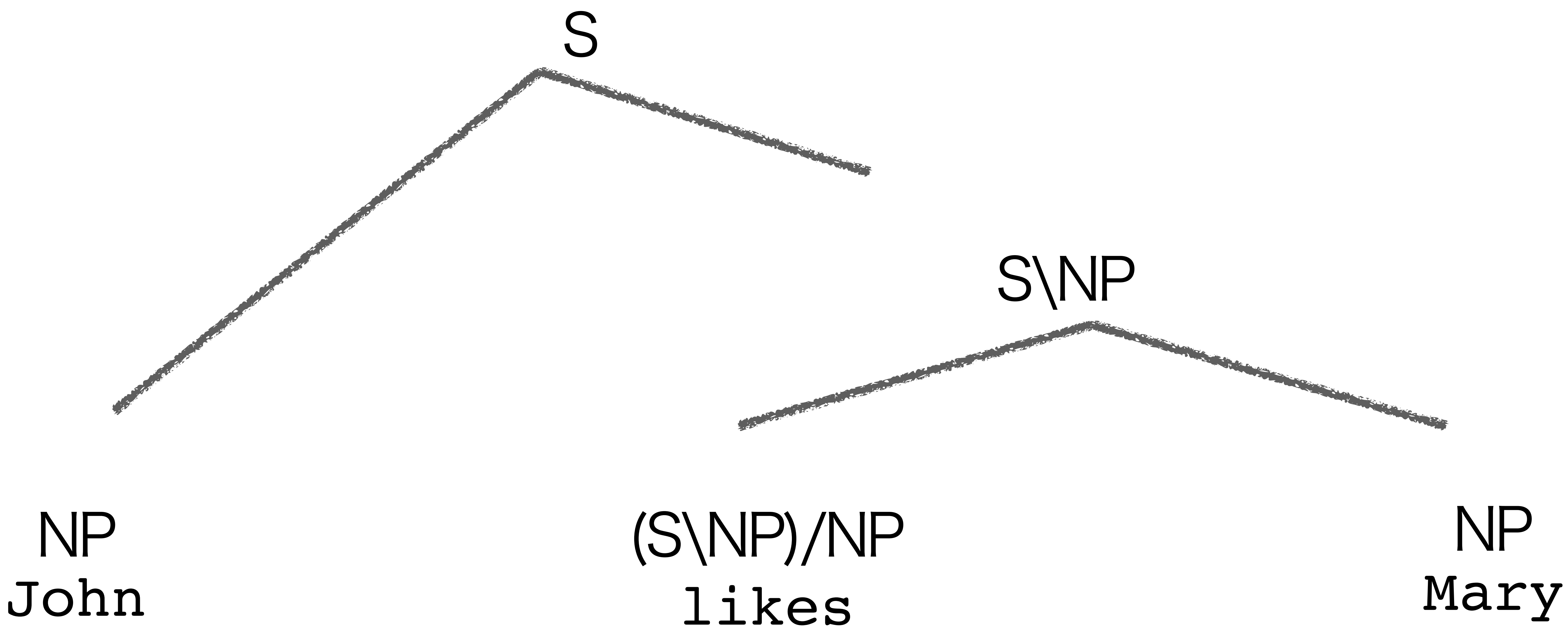
- Categories with recursive function-like structure
- A small number of derivational rules (less than 10)
    - Meta rules (cf. CFG: `S → NP VP`)
    - Forward/backward application: `X → X/Y Y`   `X → Y X\Y`
    - Forward/backward composition rules: `X/Z → X/Y Y/Z`

`X/Y` argument

`X\Y` return value



Basic CCG-based Semantic Parsing




```
• F : NP ⇒ F
• F : N ⇒ \x → F(x)
• F : (S\NP)/NP ⇒ \y x → exist e. F(e) …
• F : S\NP ⇒ \x → exist e. F(e) & A0(0)
…
```

Dictionary

- Imagine functional programming language (e.g., Haskell)

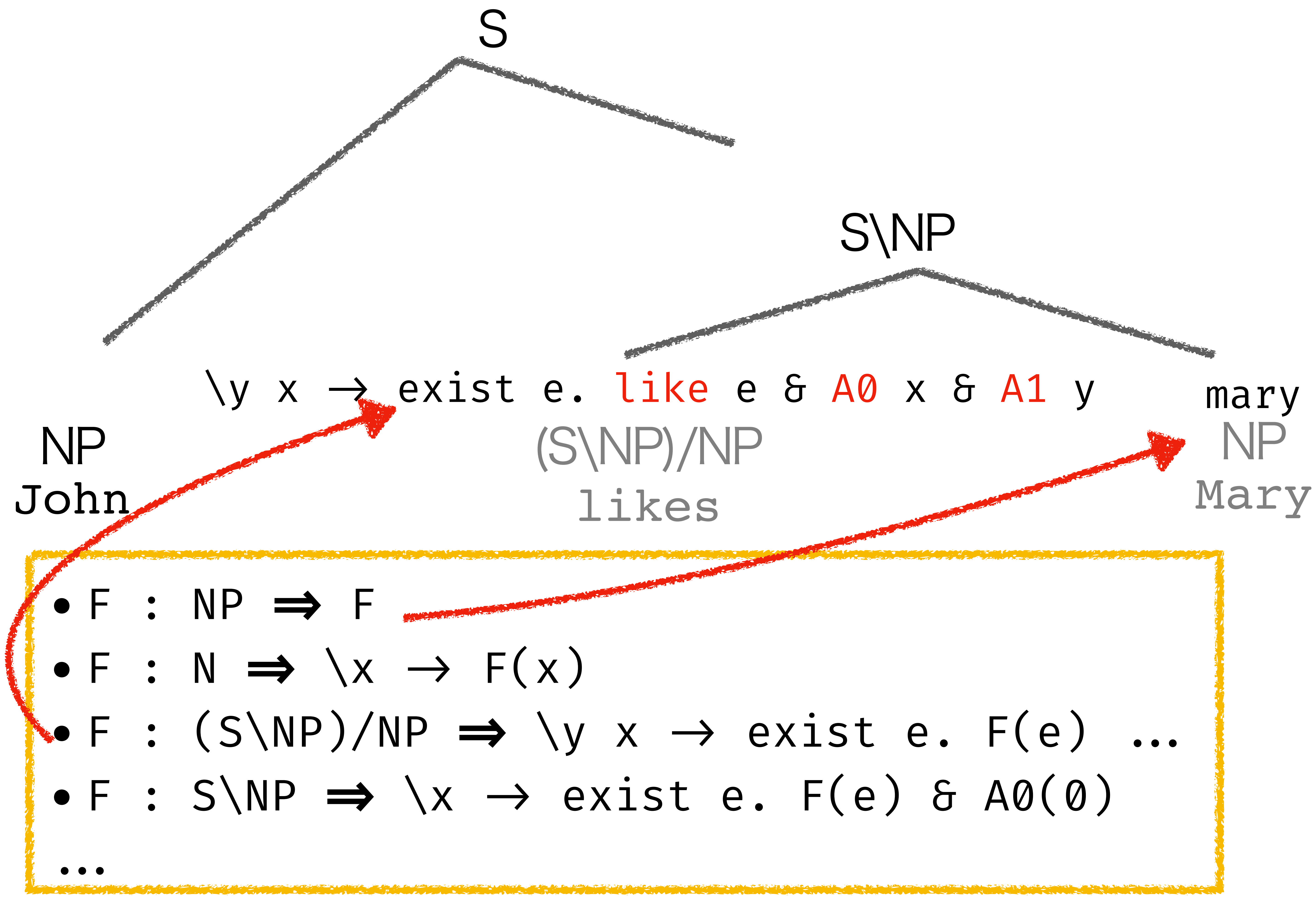
  `\x y → f(x,y)`: lambda term

  `john, mary`: entity term

  `true, false`: truth term

- Hand-crafted dictionary maps (word, category) to a lambda term
- Here we use logical formulas based on event semantics
  - There exists an event e, whose argument 0 is john and …

Basic CCG-based Semantic Parsing



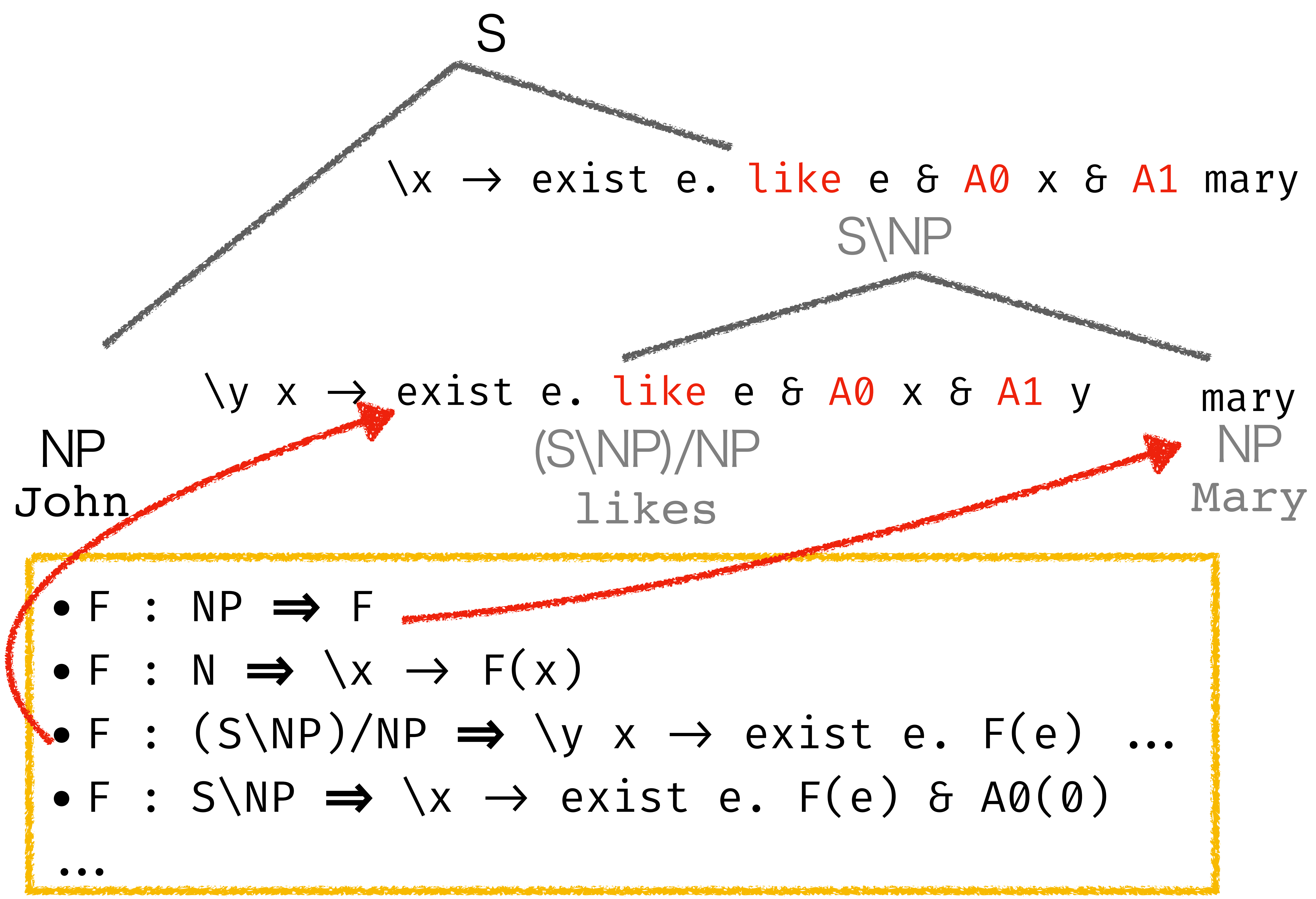
Dictionary

- Imagine functional programming language (e.g., Haskell)

  `\x y → f(x,y)`: lambda term
  `john, mary`: entity term
  `true, false`: truth term

- Hand-crafted dictionary maps (word, category) to a lambda term
- Here we use logical formulas based on event semantics
  - There exists an event e, whose argument 0 is john and …

Basic CCG-based Semantic Parsing



Dictionary

- Imagine functional programming language (e.g., Haskell)

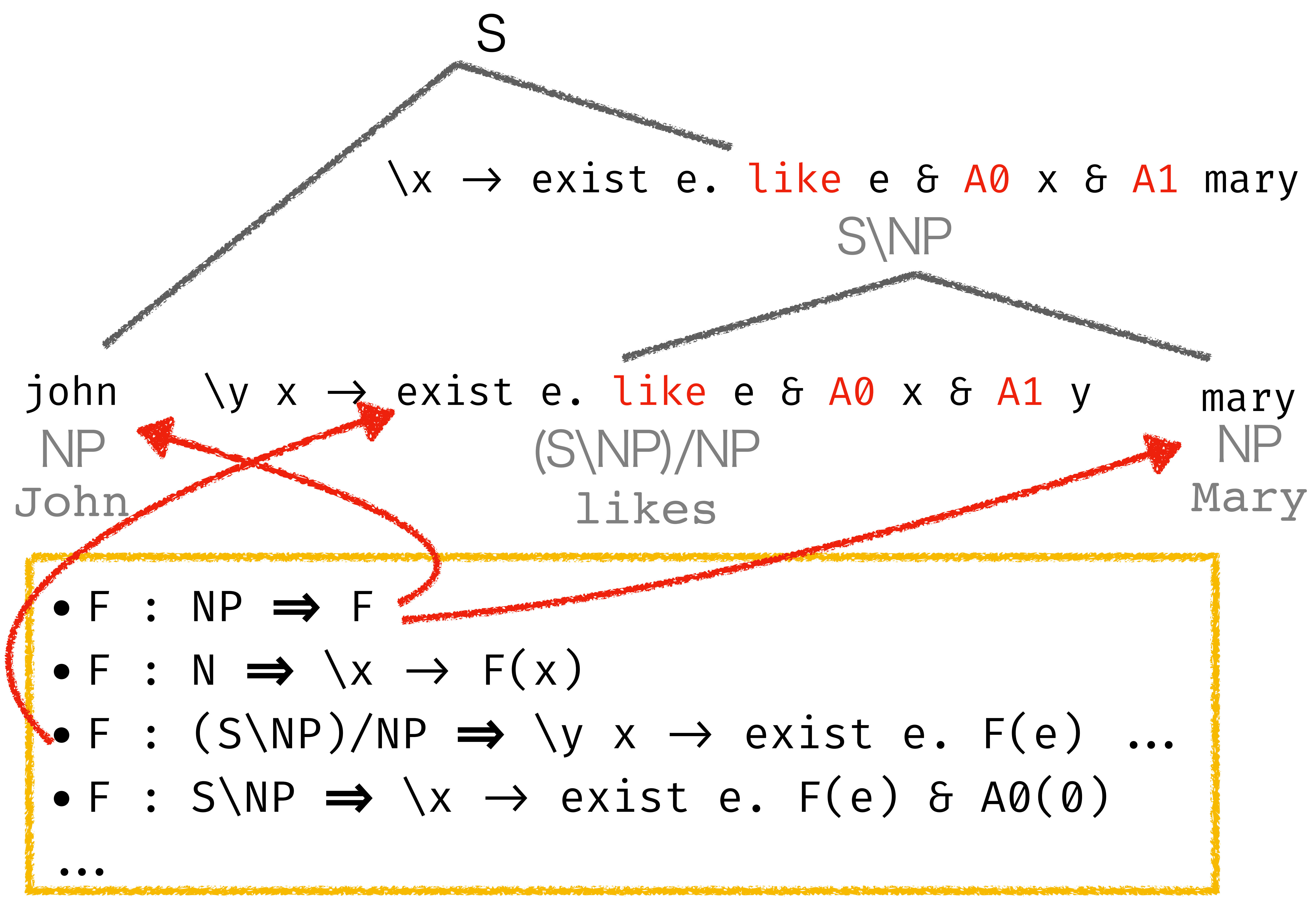
  `\x y → f(x,y)`: lambda term

  `john, mary`: entity term

  `true, false`: truth term

- Hand-crafted dictionary maps (word, category) to a lambda term
- Here we use logical formulas based on event semantics
  - There exists an event e, whose argument 0 is john and …

Basic CCG-based Semantic Parsing



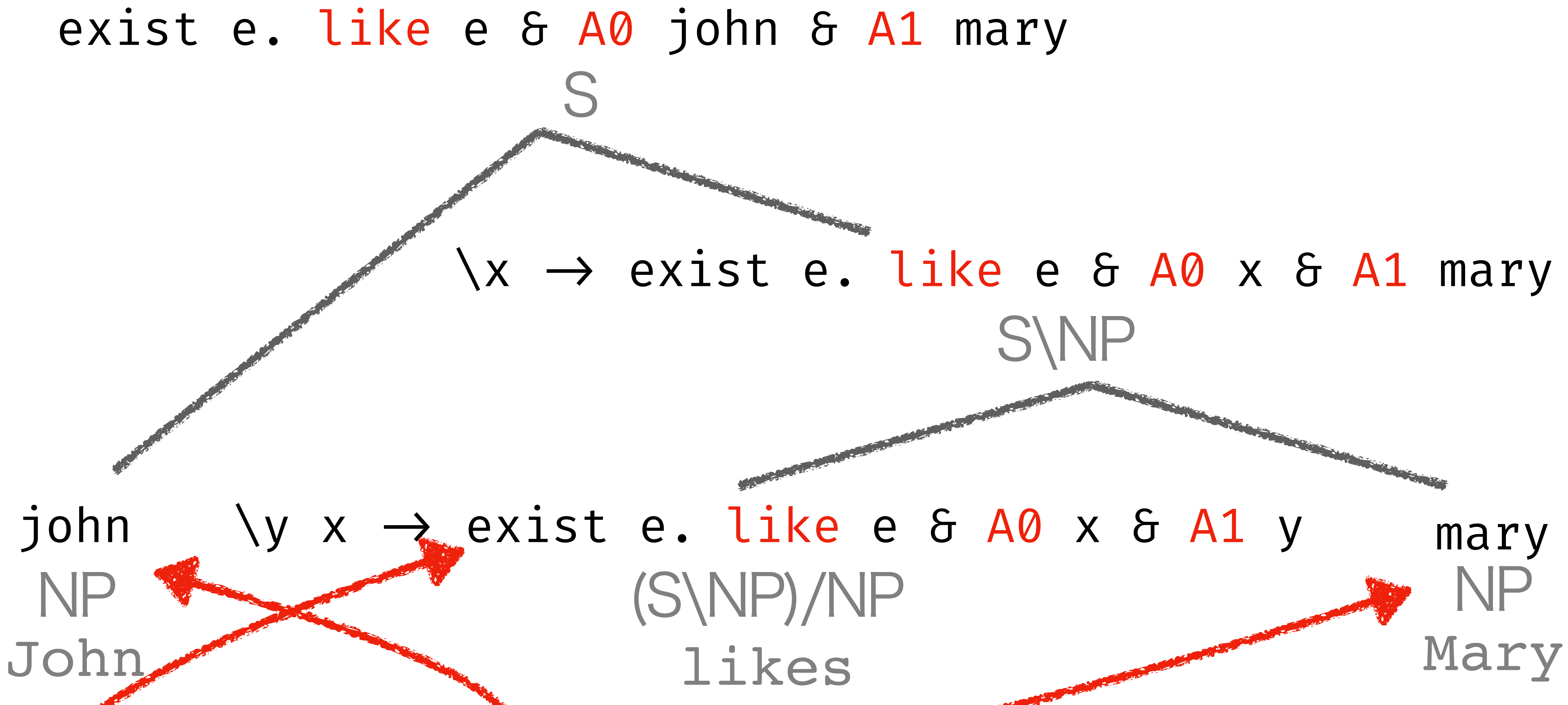
Dictionary

- Imagine functional programming language (e.g., Haskell)

  `\x y → f(x,y)`: lambda term
  `john, mary`: entity term
  `true, false`: truth term

- Hand-crafted dictionary maps (word, category) to a lambda term
- Here we use logical formulas based on event semantics
  - There exists an event e, whose argument 0 is john and …

Basic CCG-based Semantic Parsing




```
• F : NP ⇒ F
• F : N ⇒ \x → F(x)
• F : (S\NP)/NP ⇒ \y x → exist e. F(e) …
• F : S\NP ⇒ \x → exist e. F(e) & A0(0)
…
```

Dictionary

- Imagine functional programming language (e.g., Haskell)

  `\x y → f(x,y)`: lambda term
  `john, mary`: entity term
  `true, false`: truth term

- Hand-crafted dictionary maps (word, category) to a lambda term

- Here we use logical formulas based on event semantics
  - There exists an event e, whose argument 0 is john and …

Semantic Parsing in Real Application

e.g. Mineshima et al., 2015, Abzianidze, 2017

CCG-based Inference System

(latter half of the talk)

e.g. Mineshima et al., 2015, Abzianidze, 2017

Premise (P)
& Hypothesis (H)

Syntactic Parsing

CCG Derivations

Semantic Parsing

Logical Formulas

Theorem Proving

{ yes, no, unknown }

P: A man hikes.

H: A man walks.

S
NP *S\NP*
NP/N *N*
A man hikes

S
NP *S\NP*
NP/N *N*
A man walks

$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{hike}(e) \wedge \mathbf{subj}(e,x)$$

$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{walk}(e) \wedge \mathbf{subj}(e,x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

result: unknown

Coq
theorem
prover

Annotation Criteria for CCG

Q: How do you choose that structure/category? Is it because you like that?

A: No, it is designed to optimize the performance of inference systems built upon it

- e.g. Why are there N and NP?

	syntax	semantics
NP (John)	proper noun	entity (`john`)
N (dog)	common noun	set of entities (`\x → dog x`)

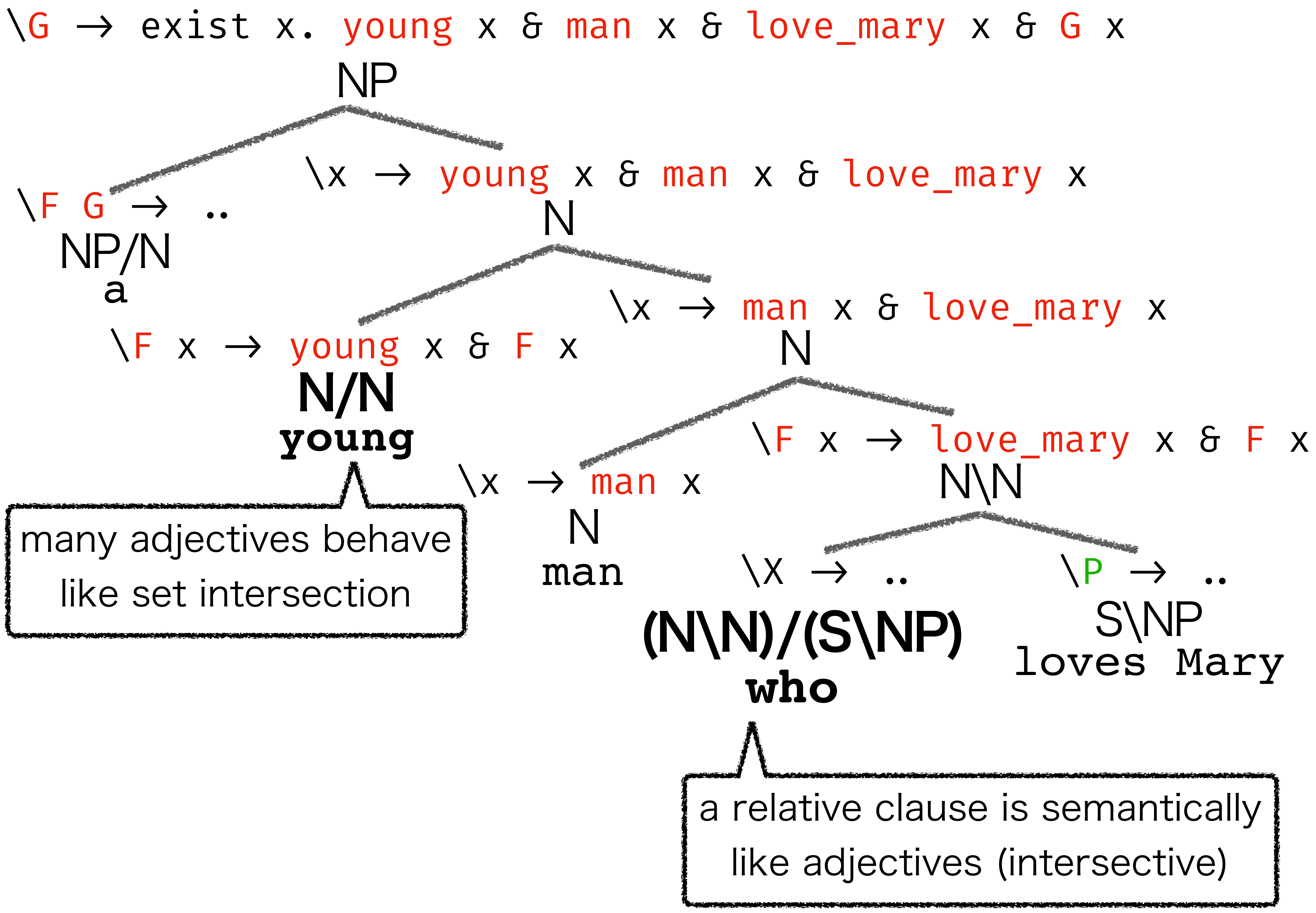
Annotation Criteria for CCG

Q: How do you choose that structure/category? Is it because you like that?

A: No, it is designed to optimize the performance of inference systems built upon it

- e.g. Why are there N and NP?

	syntax	semantics
NP (John)	proper noun	entity (john)
N (dog)	common noun	set of entities (\x → dog x)



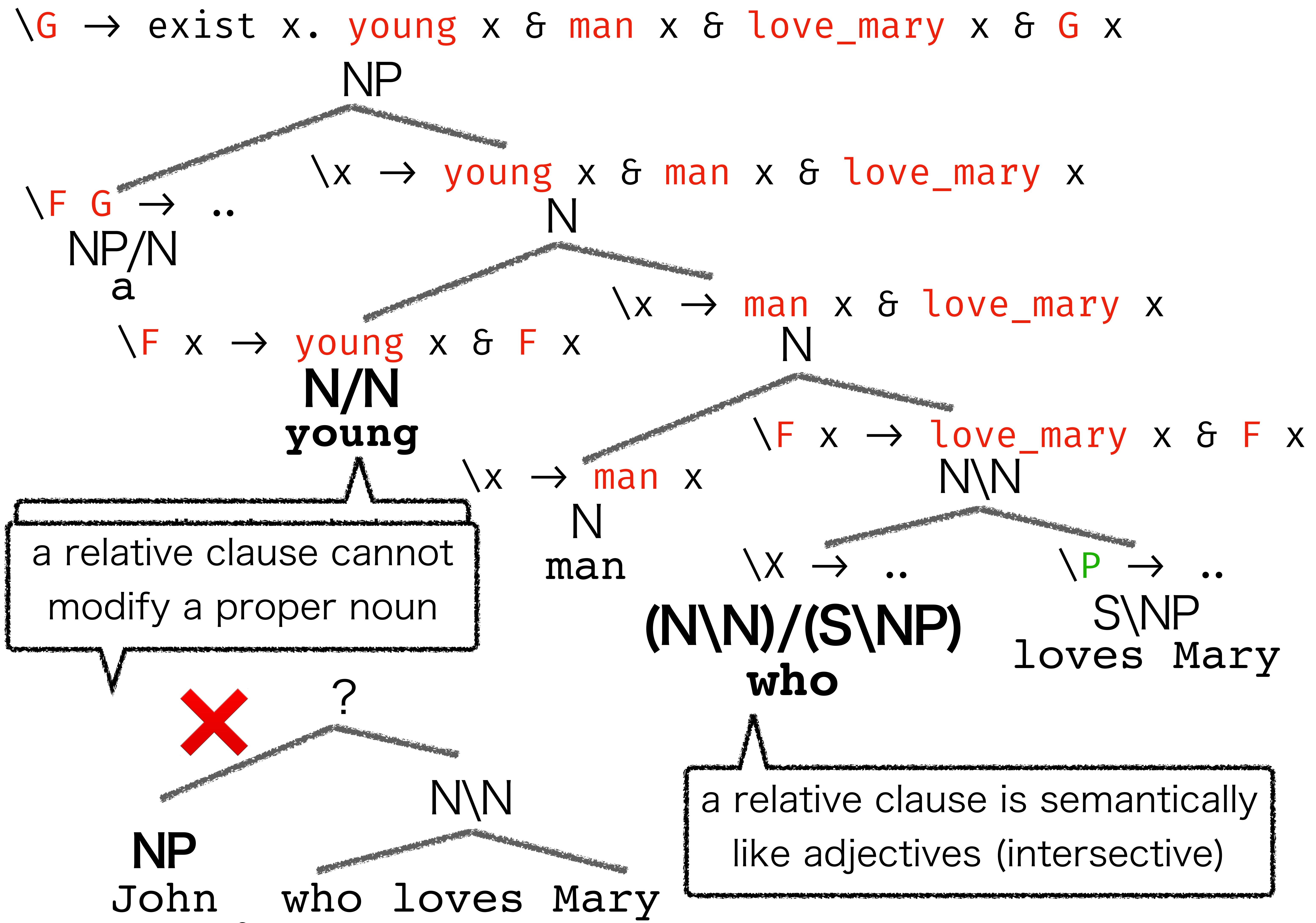
Annotation Criteria for CCG

Q: How do you choose that structure/category? Is it because you like that?

A: No, it is designed to optimize the performance of inference systems built upon it

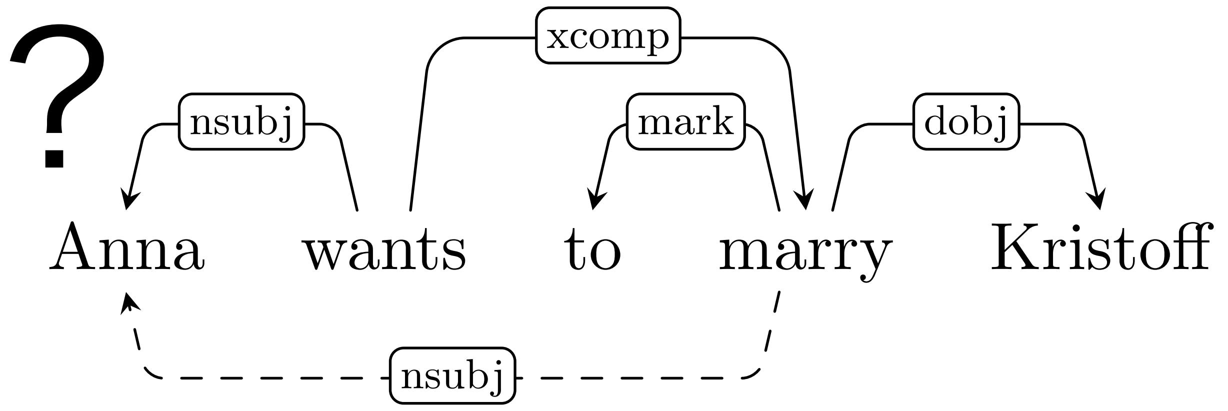
- e.g. Why are there N and NP?

	syntax	semantics
NP (John)	proper noun	entity (john)
N (dog)	common noun	set of entities (\x → dog x)

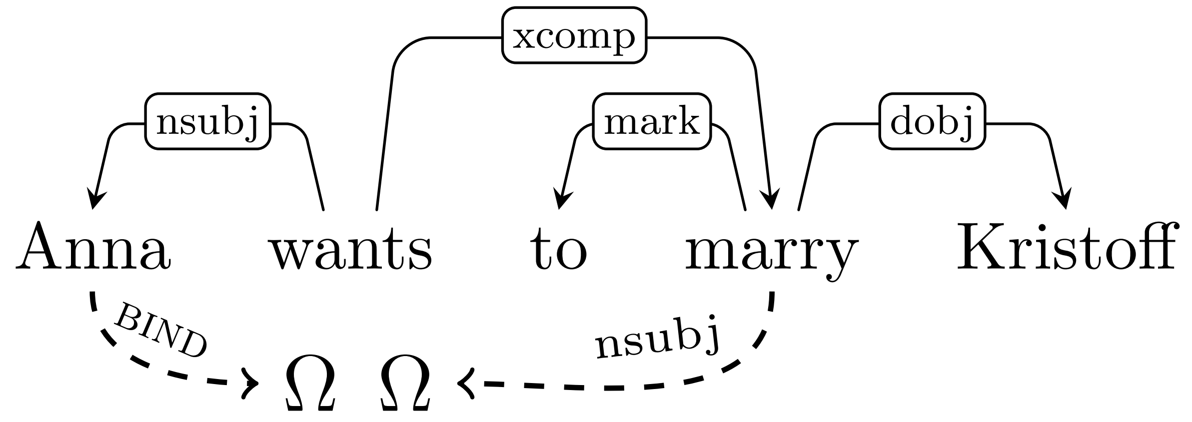


Why CCG? and not dependencies?

- 😊 Gives elegant explanations for complex phenomena
    - leads to better meaning representation
    - cf. semantic parsing based on UD (Reddy et al., 2017)
        - suffers from control verbs, coordination, etc.



(a) With long-distance dependency.

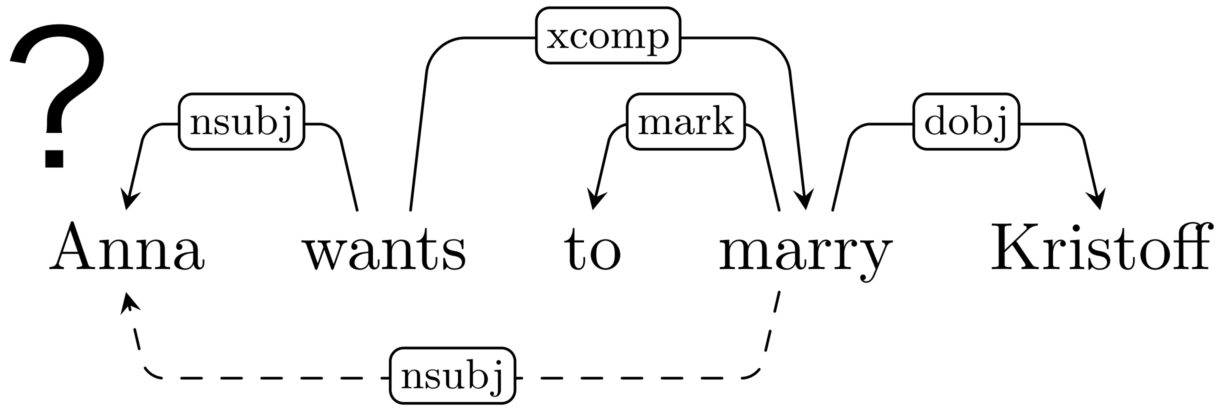


(b) With variable binding.

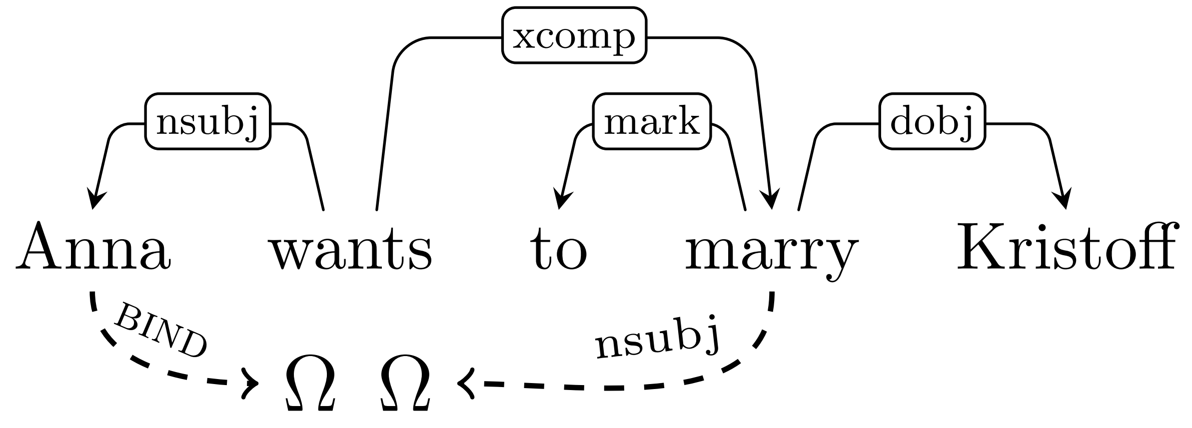
Figure 2: The original and enhanced dependency trees for *Anna wants to marry Kristoff*.

Why CCG? and not dependencies?

- 😊 Gives elegant explanations for complex phenomena
  - leads to better meaning representation
  - cf. semantic parsing based on UD (Reddy et al., 2017)
    - suffers from control verbs, coordination, etc.
- 😊 Collaborate with linguists to address long-tail problems
  - e.g., comparatives (Haruta et al., 2019)

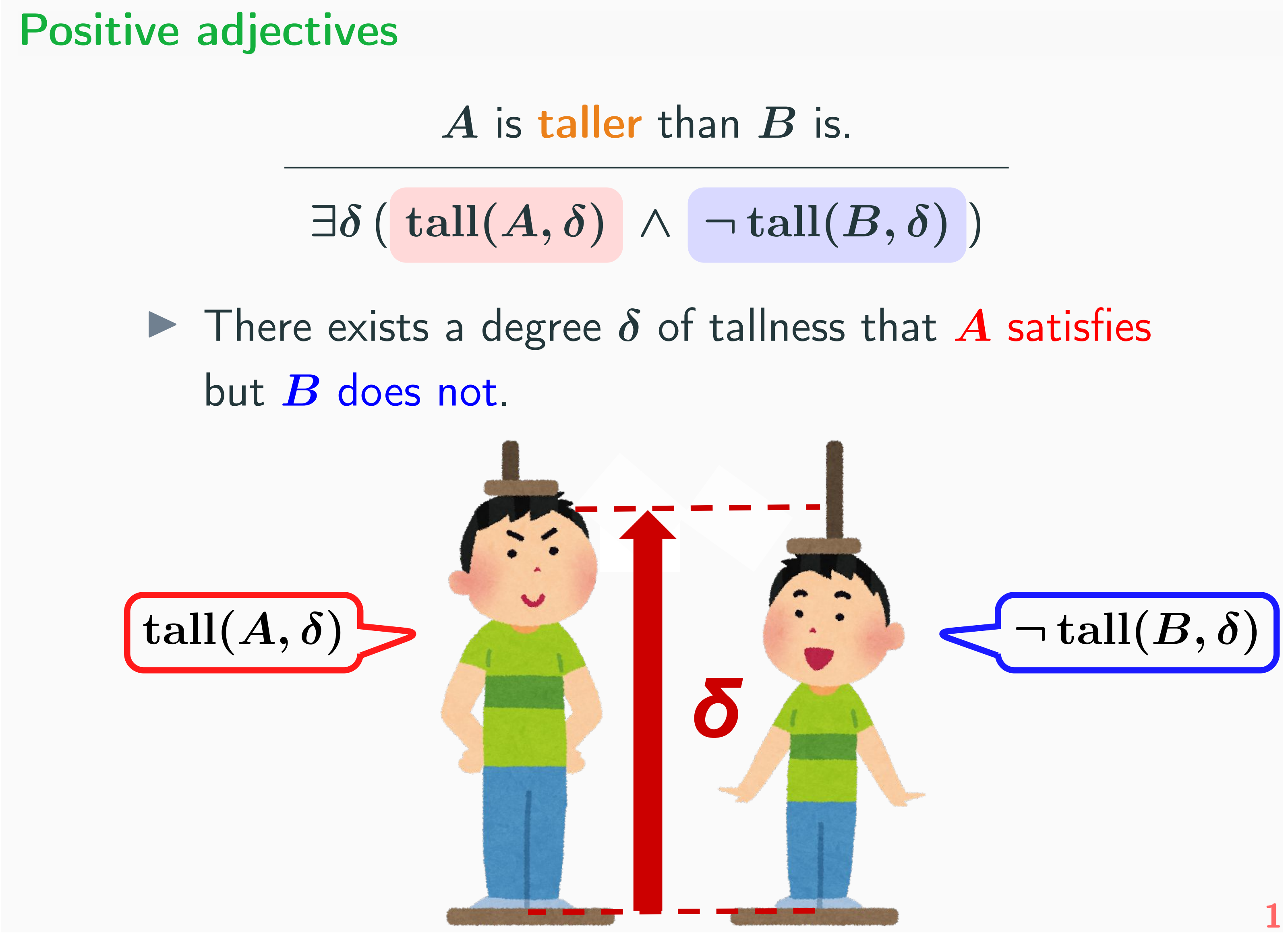


(a) With long-distance dependency.



(b) With variable binding.

Figure 2: The original and enhanced dependency trees for *Anna wants to marry Kristoff*.



Why CCG? and not dependencies?

- 😊 Gives elegant explanations for complex phenomena
  - leads to better meaning representation
  - cf. semantic parsing based on UD (Reddy et al., 2017)
    - suffers from control verbs, coordination, etc.
- 😊 Collaborate with linguists to address long-tail problems
  - e.g., comparatives (Haruta et al., 2019)
- 😊 General to cover many languages, giving detailed description of language specifities

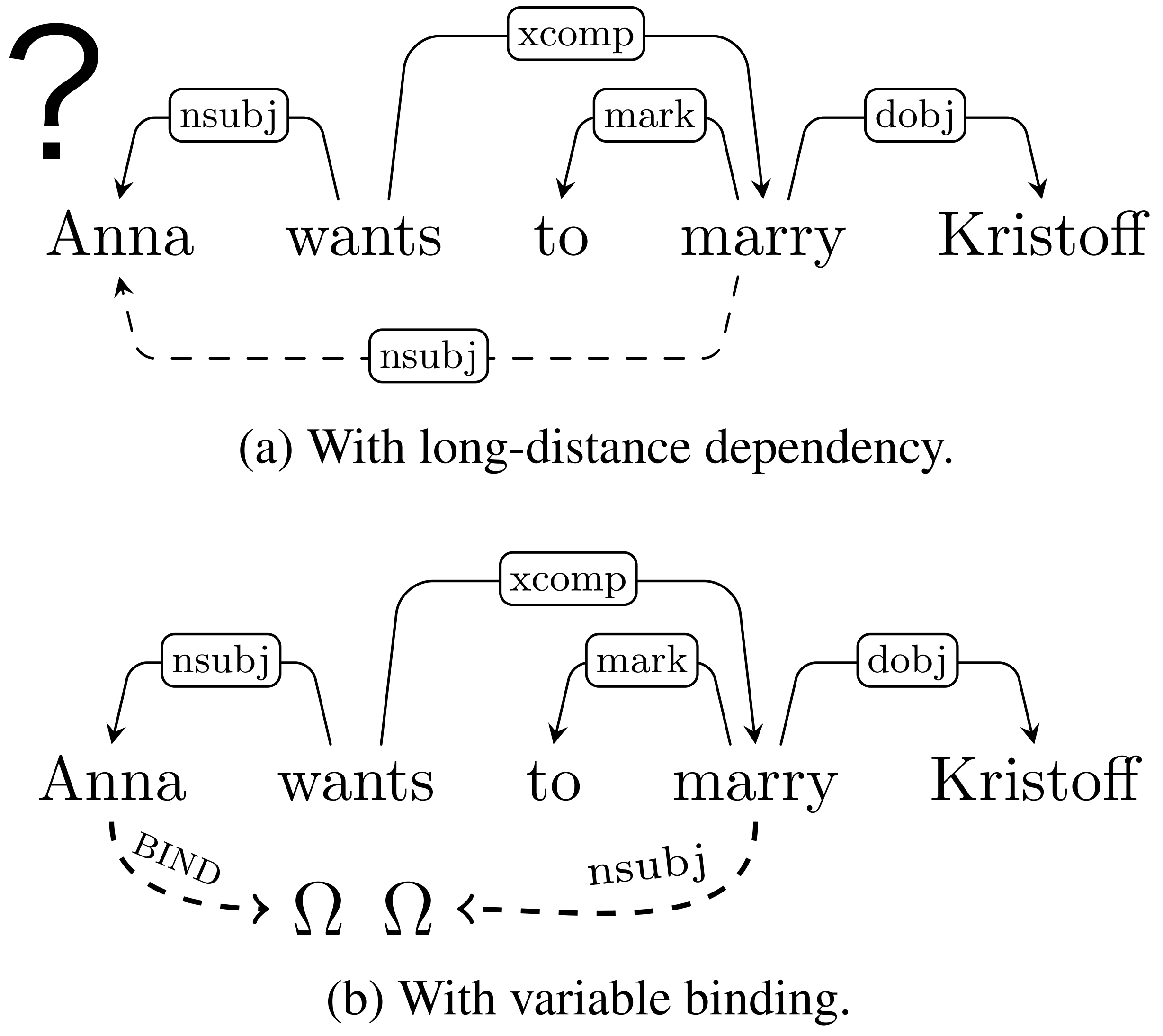
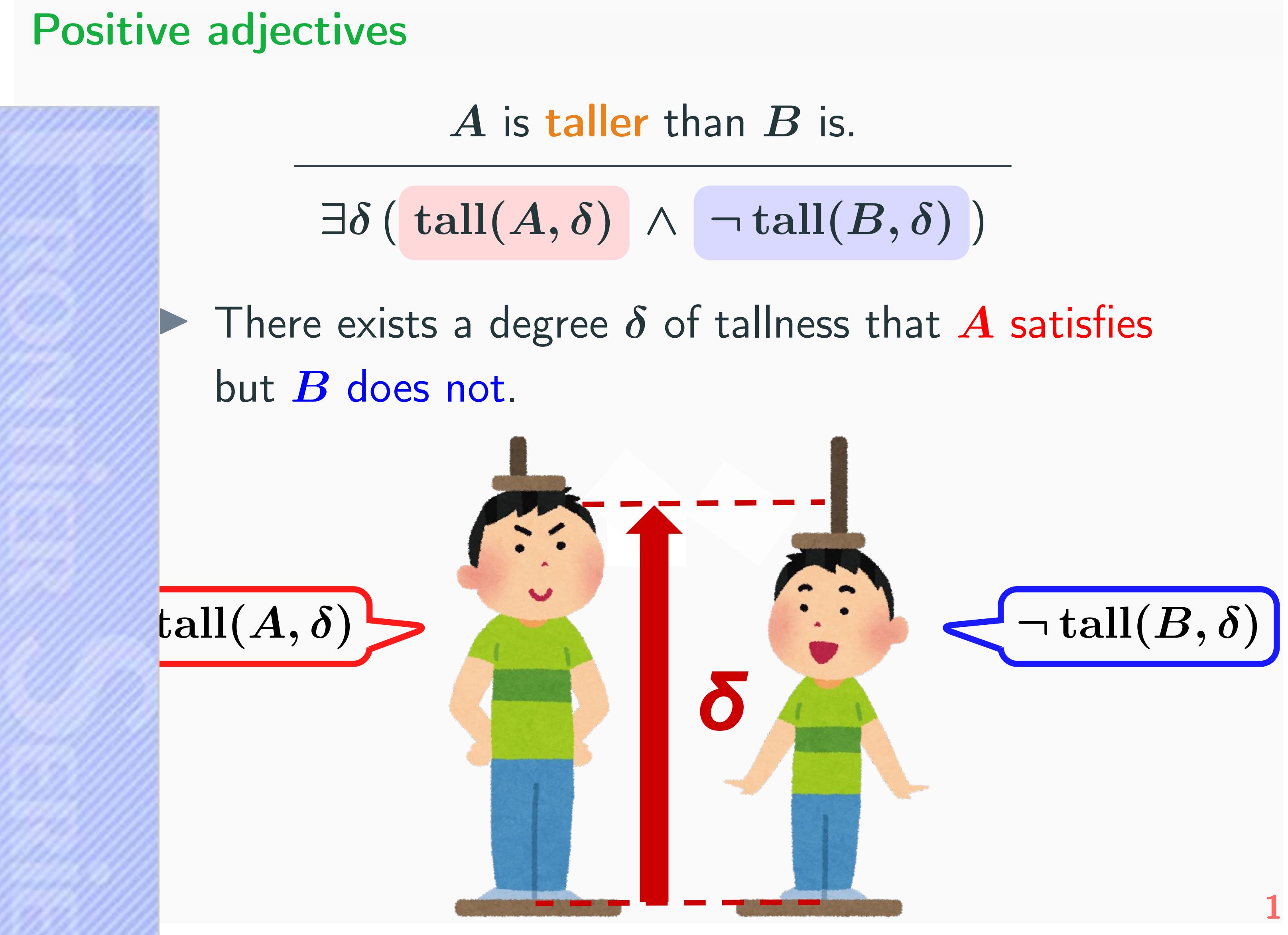


Figure 2: The original and enhanced dependency trees for *Anna wants to marry Kristoff*.

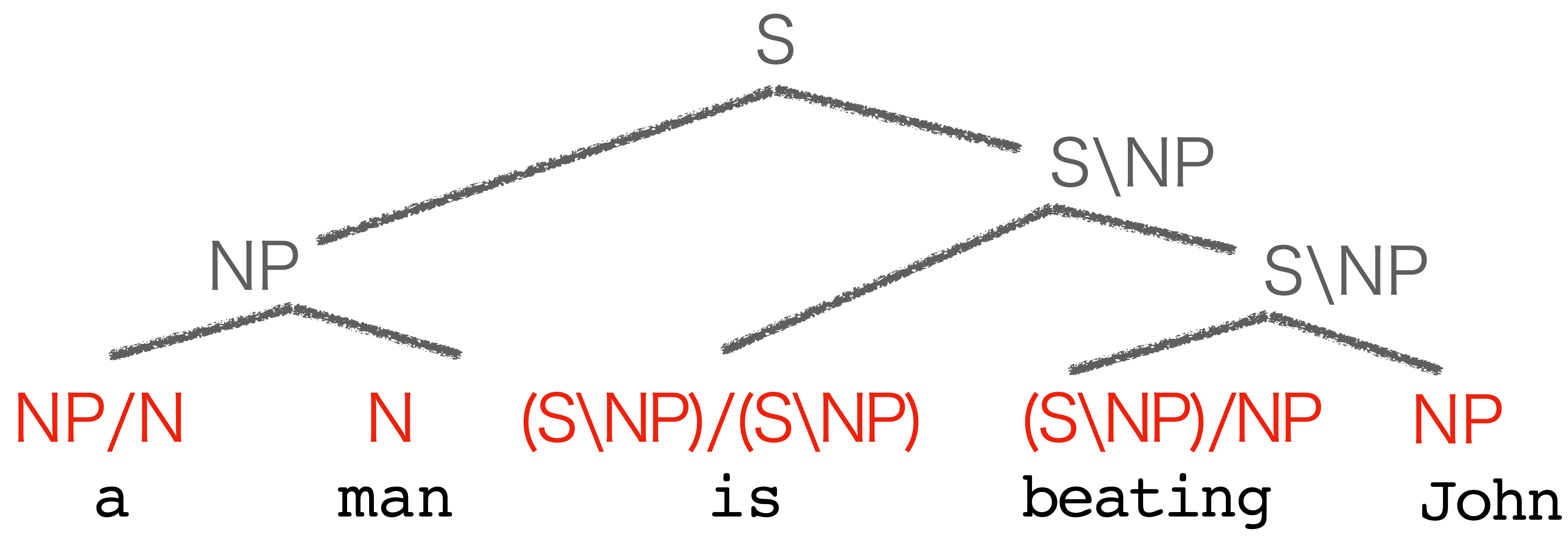


Interesting Model for CCG Parsing

- Category-factored Model (Lewis and Steedman, 2014)
  - Complex categories almost uniquely determine higher-level structure
- Exactly same form as POS tagging, but models the entire tree!
  - Note: computing $\arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}\,|\,\mathbf{x})$ is not trivial (CKY parsing is needed)

set of valid CCG trees

$$p(\mathbf{y}\,|\,\mathbf{x}) = \prod p_{tag}(c_i\,|\,\mathbf{x})$$



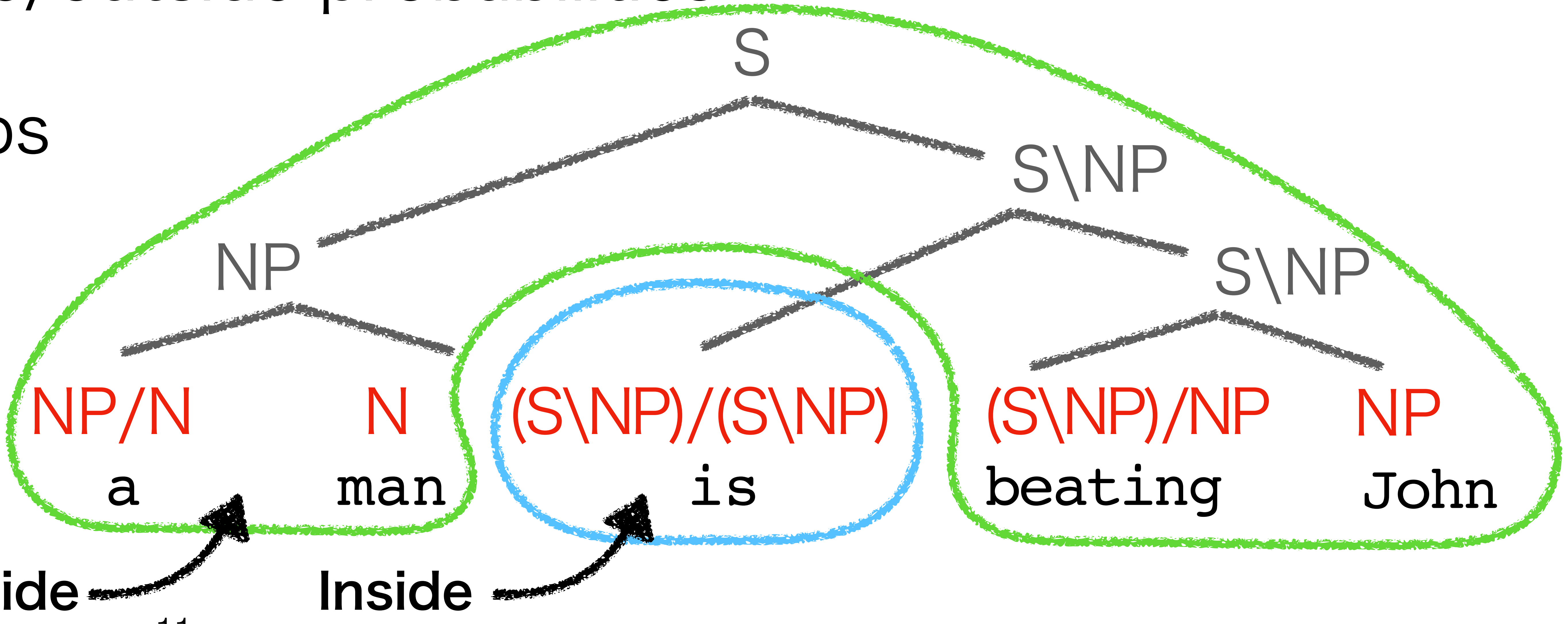
Interesting Model for CCG Parsing

- Category-factored Model (Lewis and Steedman, 2014)
  - Complex categories almost uniquely determine higher-level structure
- Exactly same form as POS tagging, but models the entire tree!
  - Note: computing $\arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}\,|\,\mathbf{x})$ is not trivial (CKY parsing is needed)

set of valid CCG trees

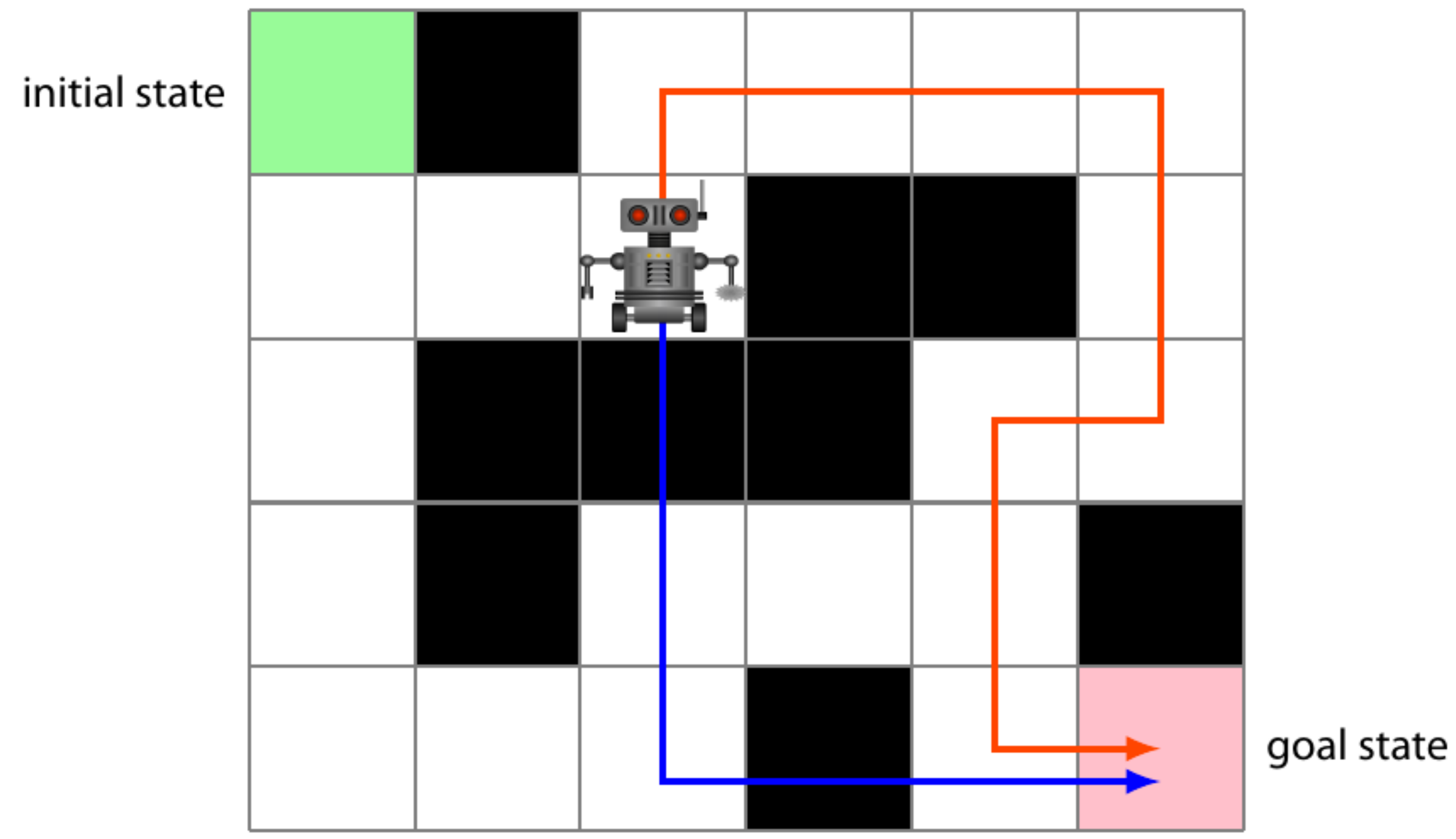
- (Advantage) Easy to compute inside/outside probabilities
  - Even <u>upper bounds</u> on these probs

$$p(\mathbf{y}\,|\,\mathbf{x}) = \prod p_{tag}(c_i\,|\,\mathbf{x})$$



Efficient A* Parsing Klein & Manning, 2003

Shortest Path Problem



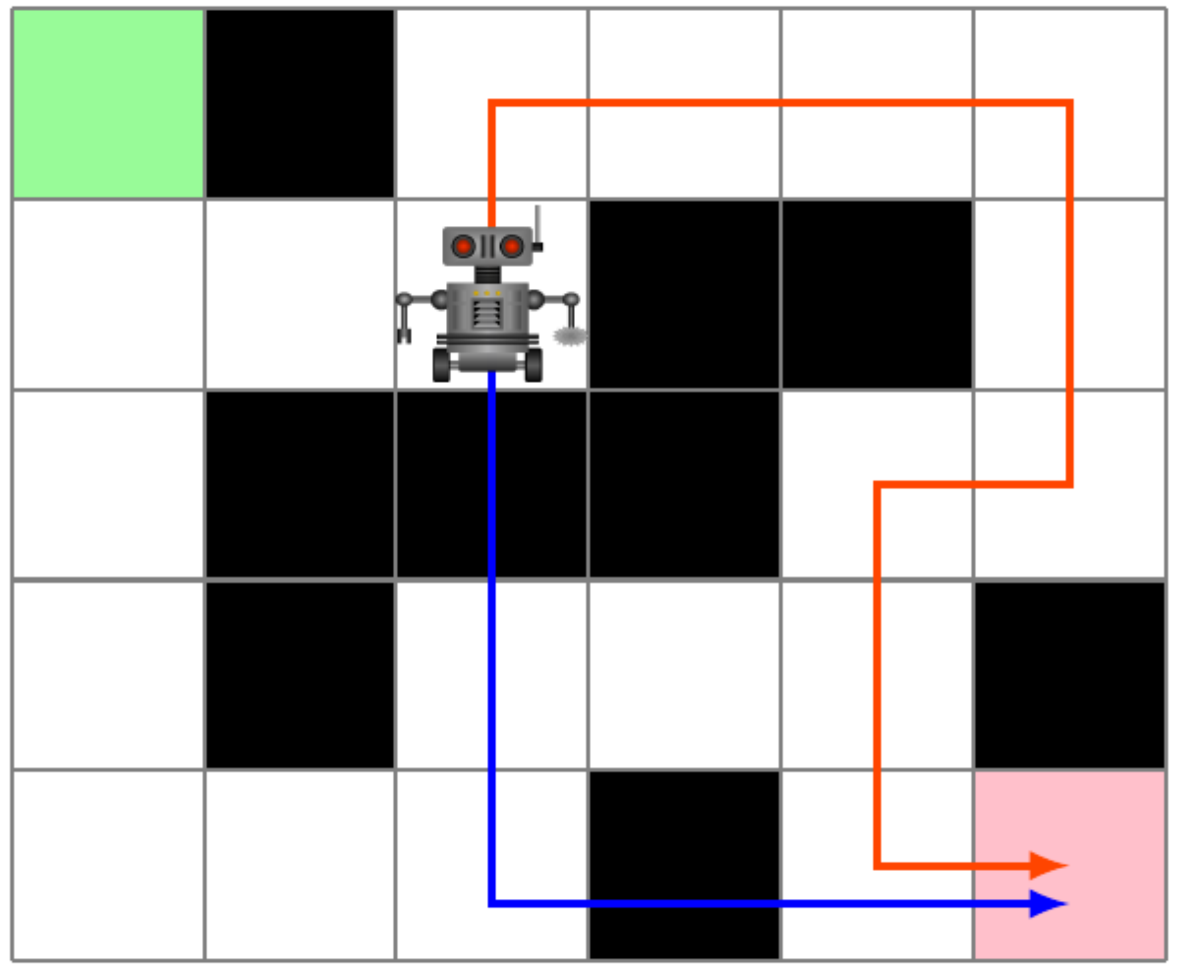
Node	f
(1,1)	0.1
(2,0)	0.1
(0,1)	0.1
(0,2)	0.9
(3,0)	0.99
…	…

PriorityQueue(f)

- Searches based on $f = g + h$
- g: Sum of the cost to the node
- h: Estimate on the cost to the goal
  - e.g. Manhattan distance

Efficient A* Parsing Klein & Manning, 2003

Shortest Path Problem

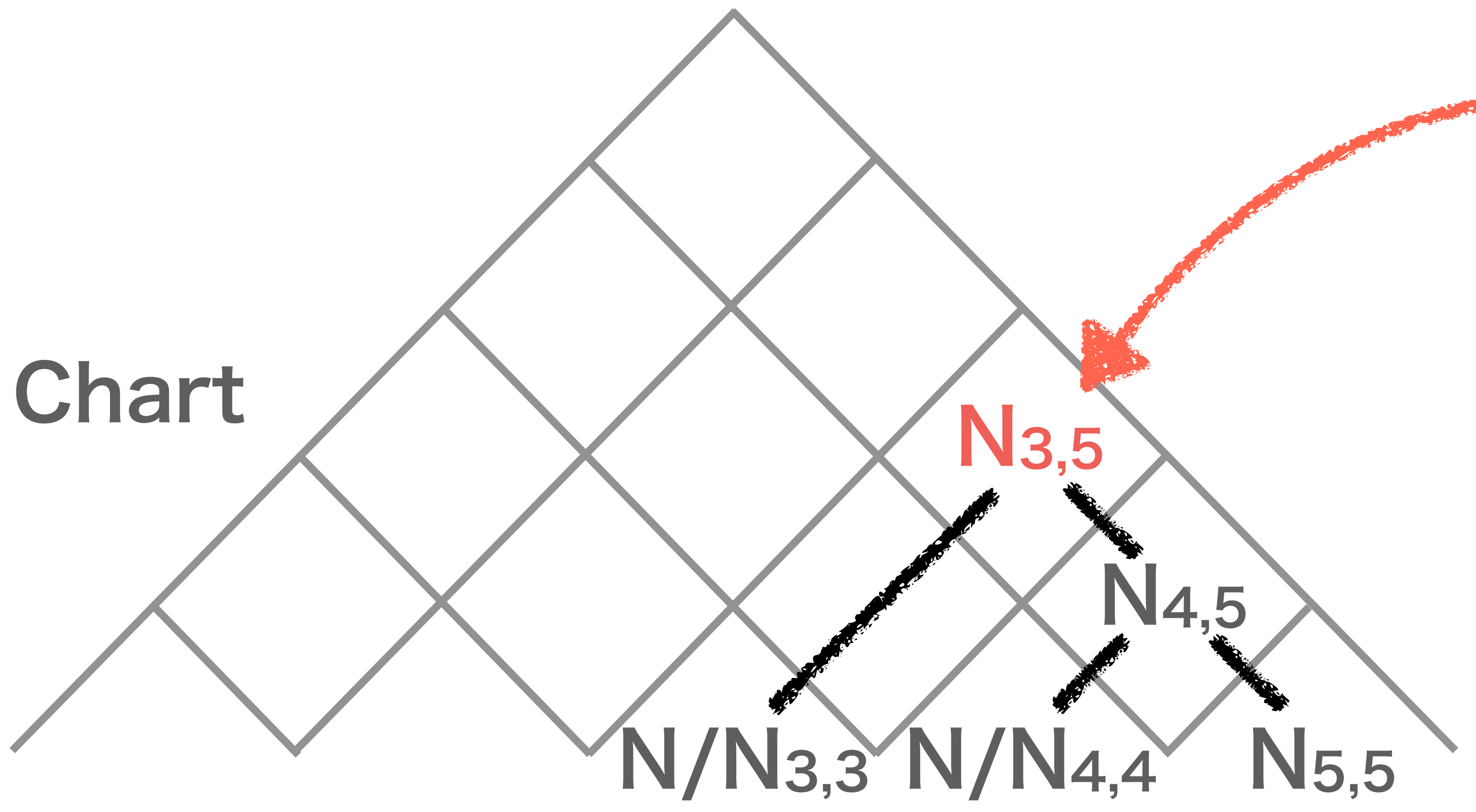


Node	f
(1,1)	0.1
(2,0)	0.1
(0,1)	0.1
(0,2)	0.9
(3,0)	0.99
…	…

PriorityQueue(f)

- Searches based on $f = g + h$
- g: Sum of the cost to the node
- h: Estimate on the cost to the goal
  - e.g. Manhattan distance

A*-based Chart Parsing

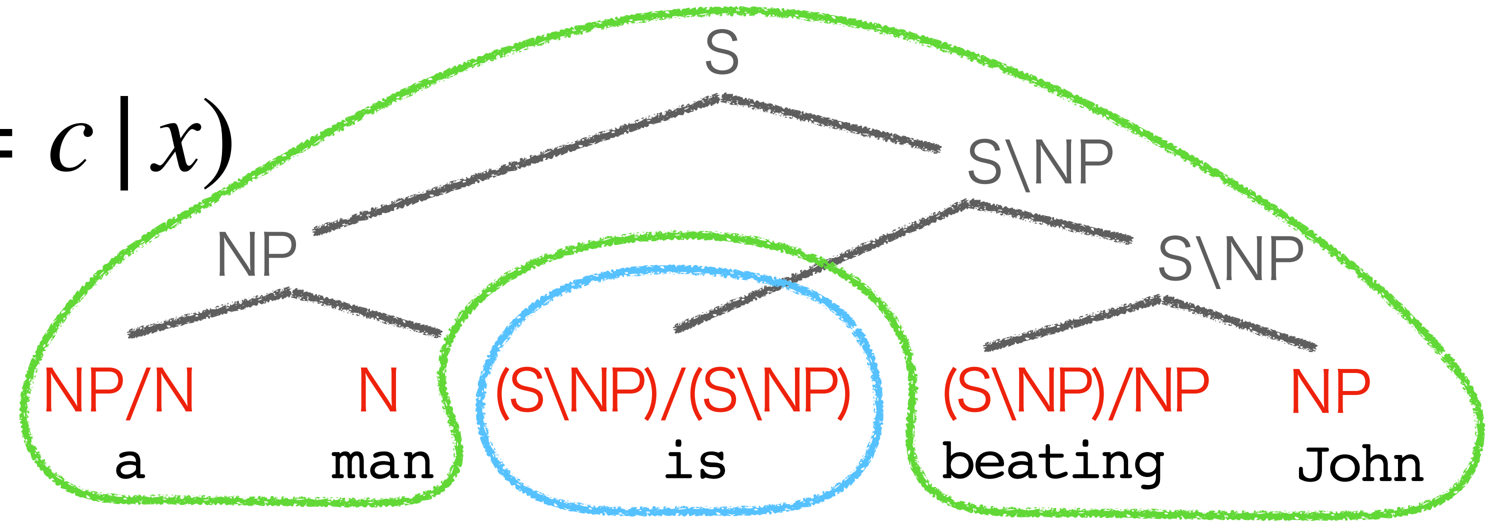


Node	f
$N_{3,5}$	0.1
$N_{1,1}$	0.1
$S\backslash N/N_{2,2}$	0.1
$N_{4,4}$	0.9
$S\backslash N_{2,2}$	0.99
…	…

PriorityQueue(f)

- Searches based on $f = g + h$
- g: Inside probability
- h: Upper bound on outside probability

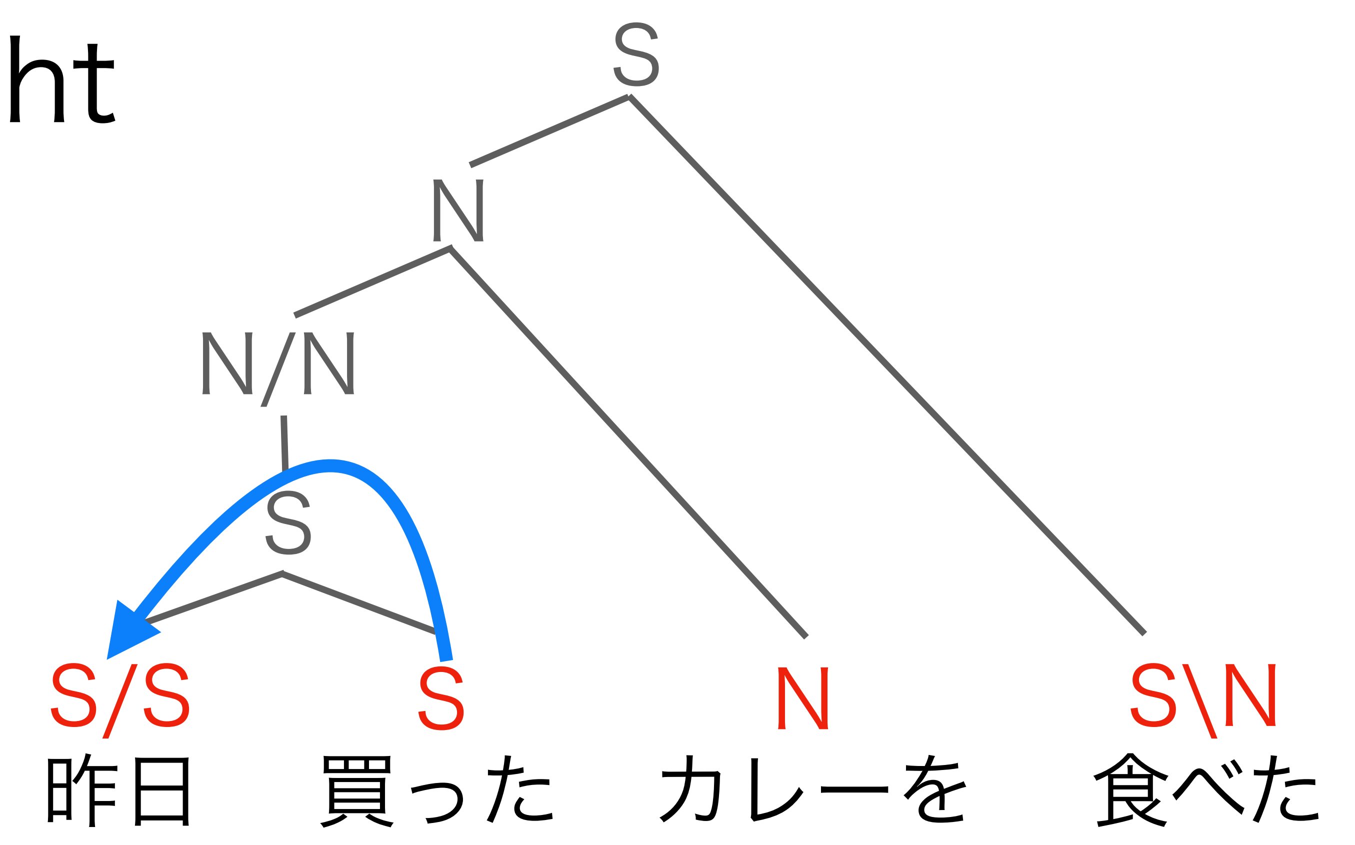
$$\sum_i \max_c p_{tag}(c_i = c \mid x)$$

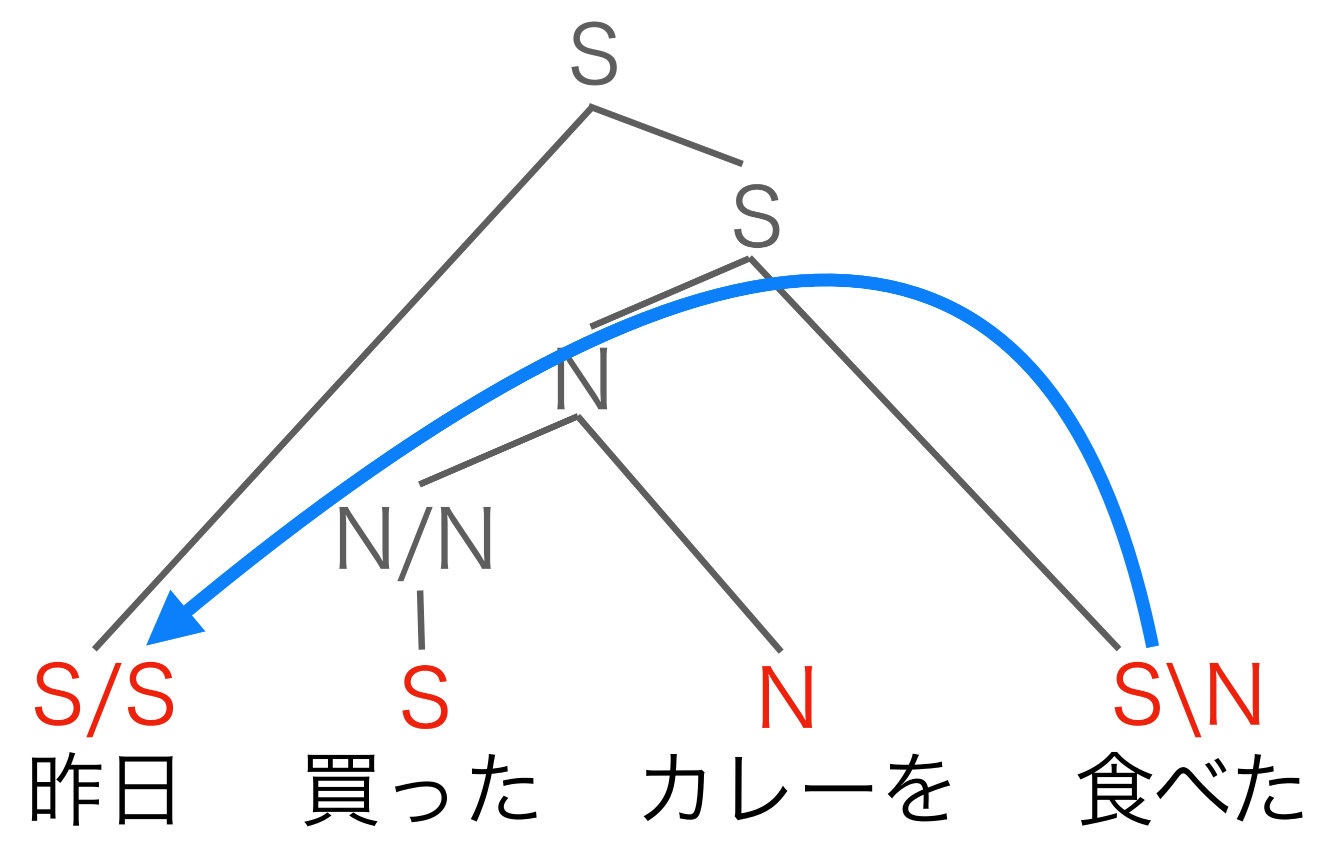


Very efficient while guaranteeing the optimality of the solution!

However..

- Modeling Japanese sentence structures with this model is not so reliable
  - It assigns the exactly same probabilities to the structures right
  - The kind of ambiguities that must be addressed in parsing!
- 🤔 **Dilemma**:
  - **Want** to extend the model to achieve higher expressivity
    - Extension with TreeLSTMs (Lee et al., 2016)
  - **Do not want** to lose the original merits
    - Efficiency and optimality guarantee



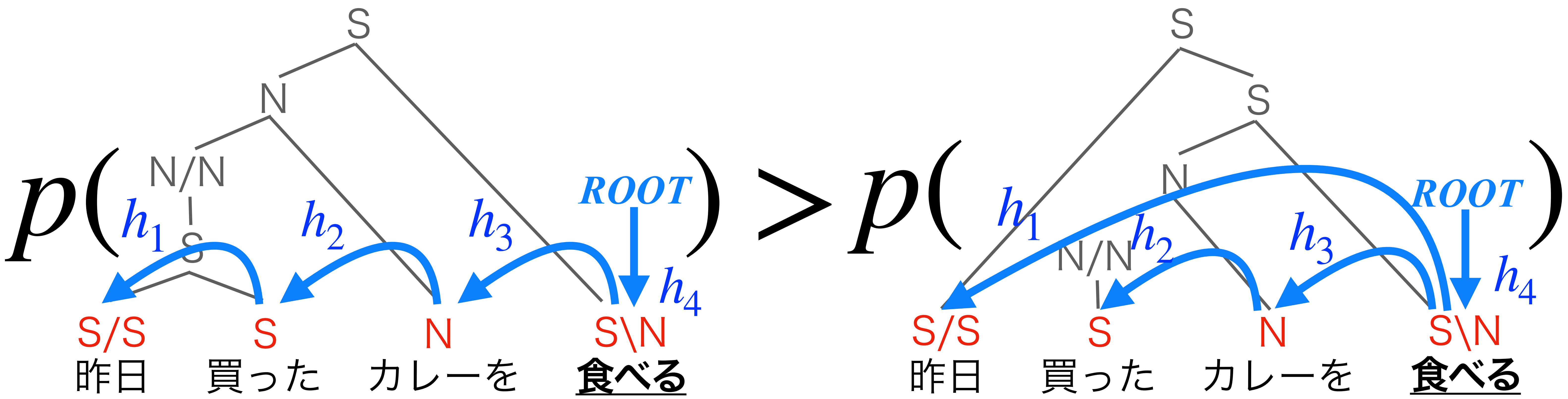


My Previous Contribution

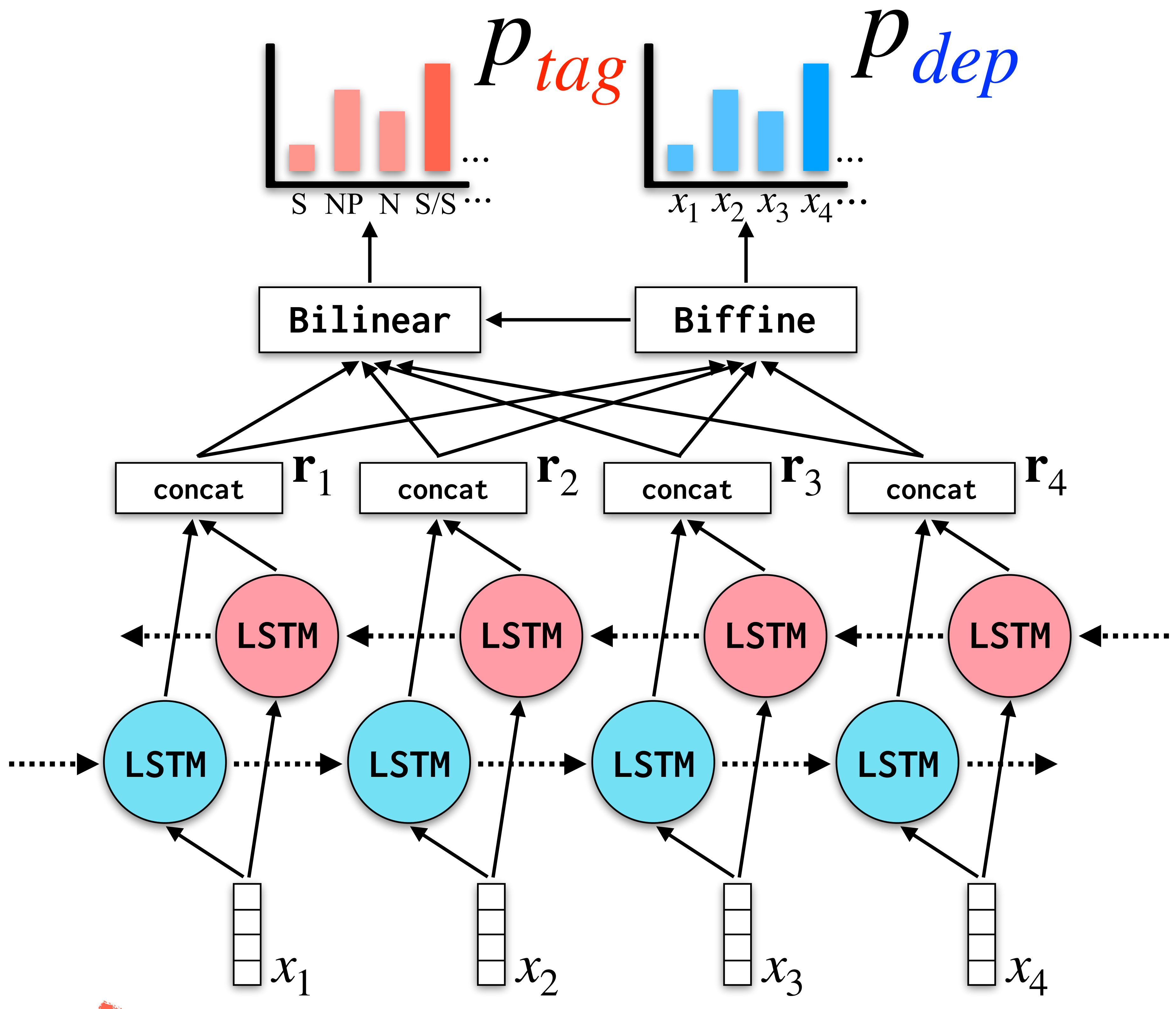
- Category and Dependency-factored Model (Yoshikawa et al., 2017)
    - Model the higher-level structure through <u>dependency edges</u>

$$p(\mathbf{y}\,|\,\mathbf{x}) = \prod p_{tag}(c_i\,|\,\mathbf{x}) \times \underline{\prod p_{dep}(h_i\,|\,\mathbf{x})}$$

- The probability is decomposable: A* parsing is available!
    - The all quantities required in A* search can be pre-computed
    - Efficiency and optimality guarantee



Calculating p_{tag} and p_{dep}



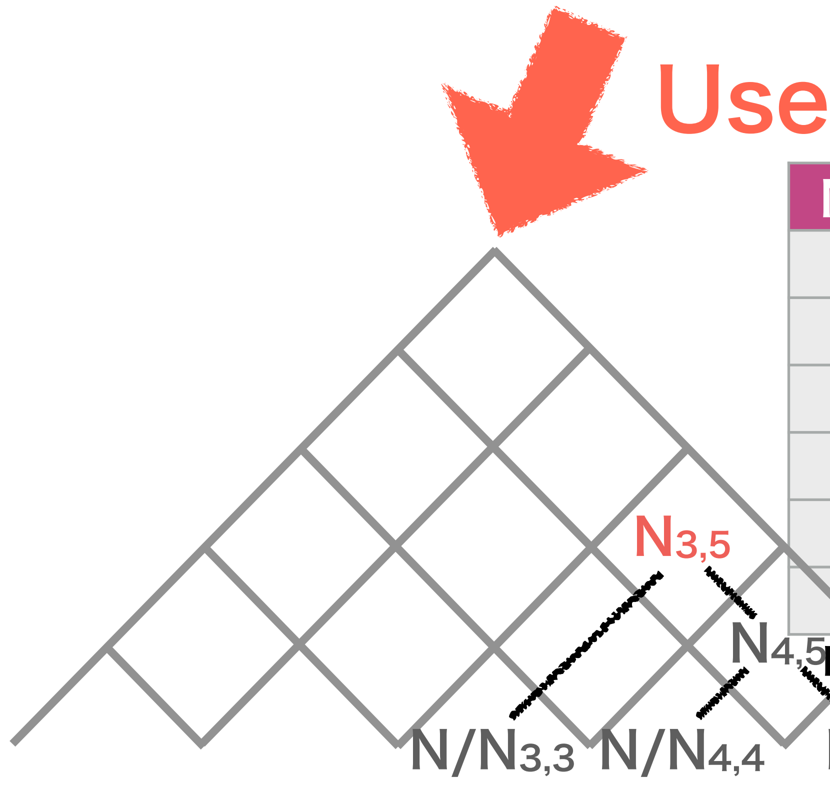
- biLSTM-based vectors: $\mathbf{r}_i$
- Best-performing dependency parsing method (Dozat et al., 2017) is utilized:
- Biaffine layer to model dependencies

$$p_{dep}(x_j \quad x_i) \propto \mathbf{r}_i^{\mathrm{T}} W \mathbf{r}_j + \mathbf{r}_i^{\mathrm{T}} \mathbf{u}$$

- Bilinear layer to model categories

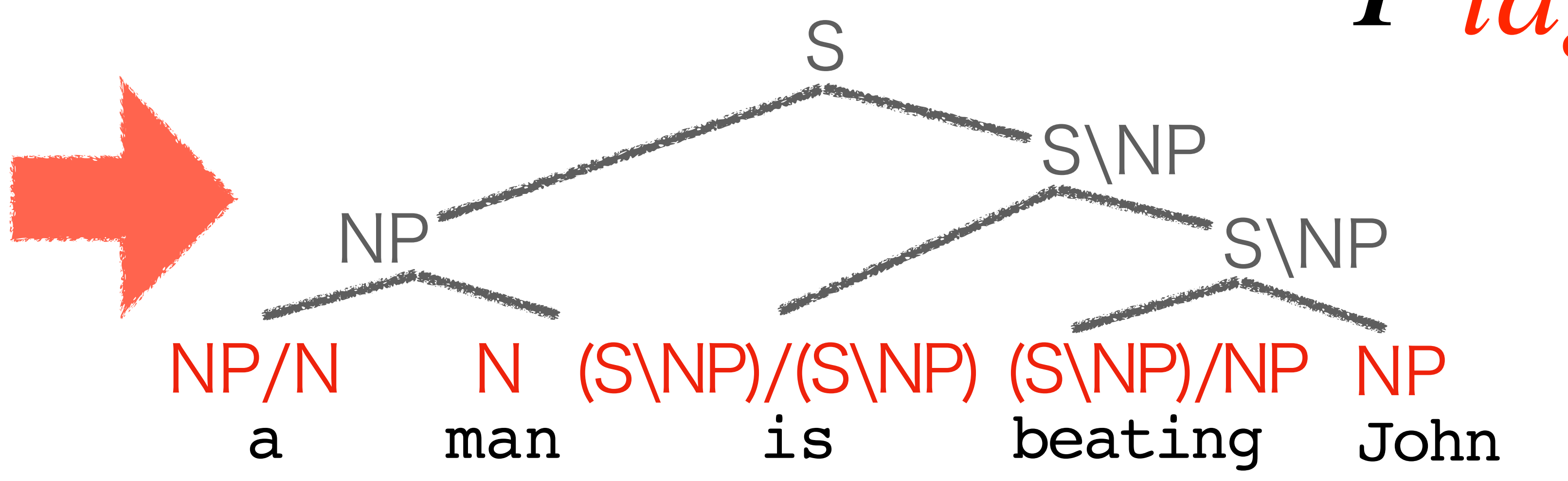
$$p_{tag}(c_i = c) \propto \mathbf{r}_i^{\mathrm{T}} W_c \mathbf{r}_{i\_head}$$

Used as costs in A* search

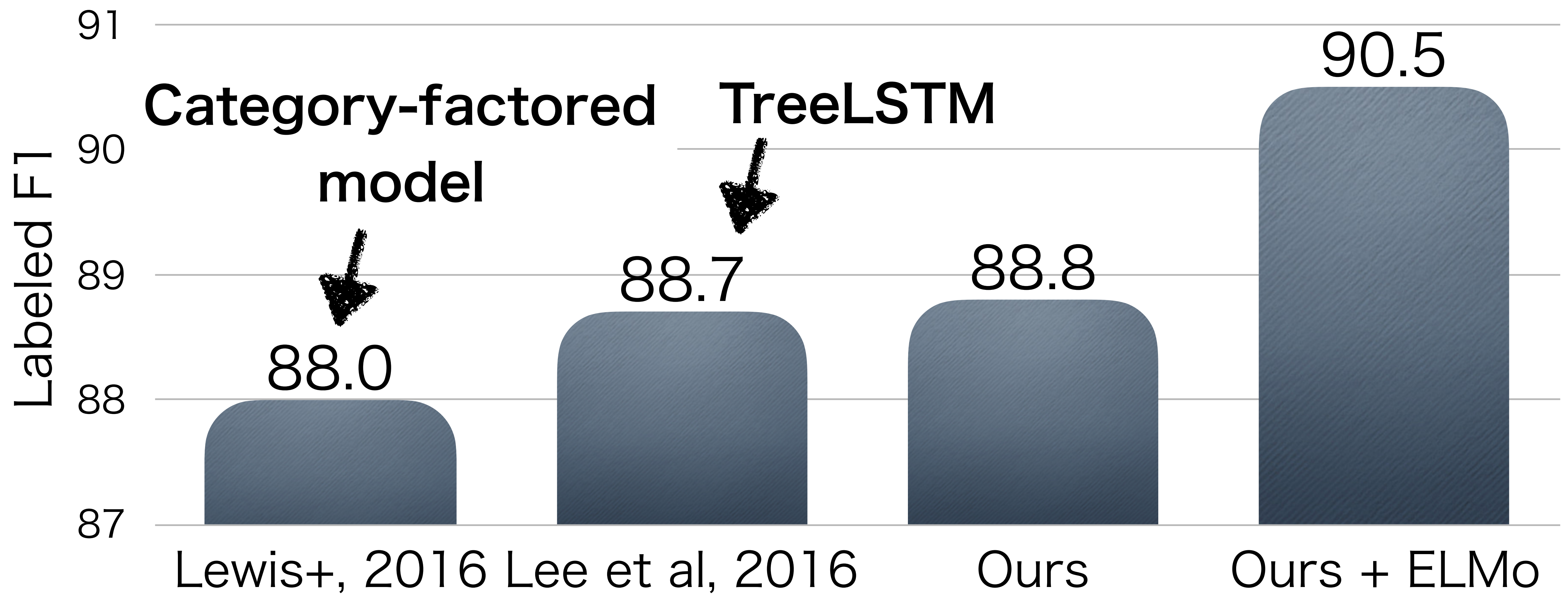


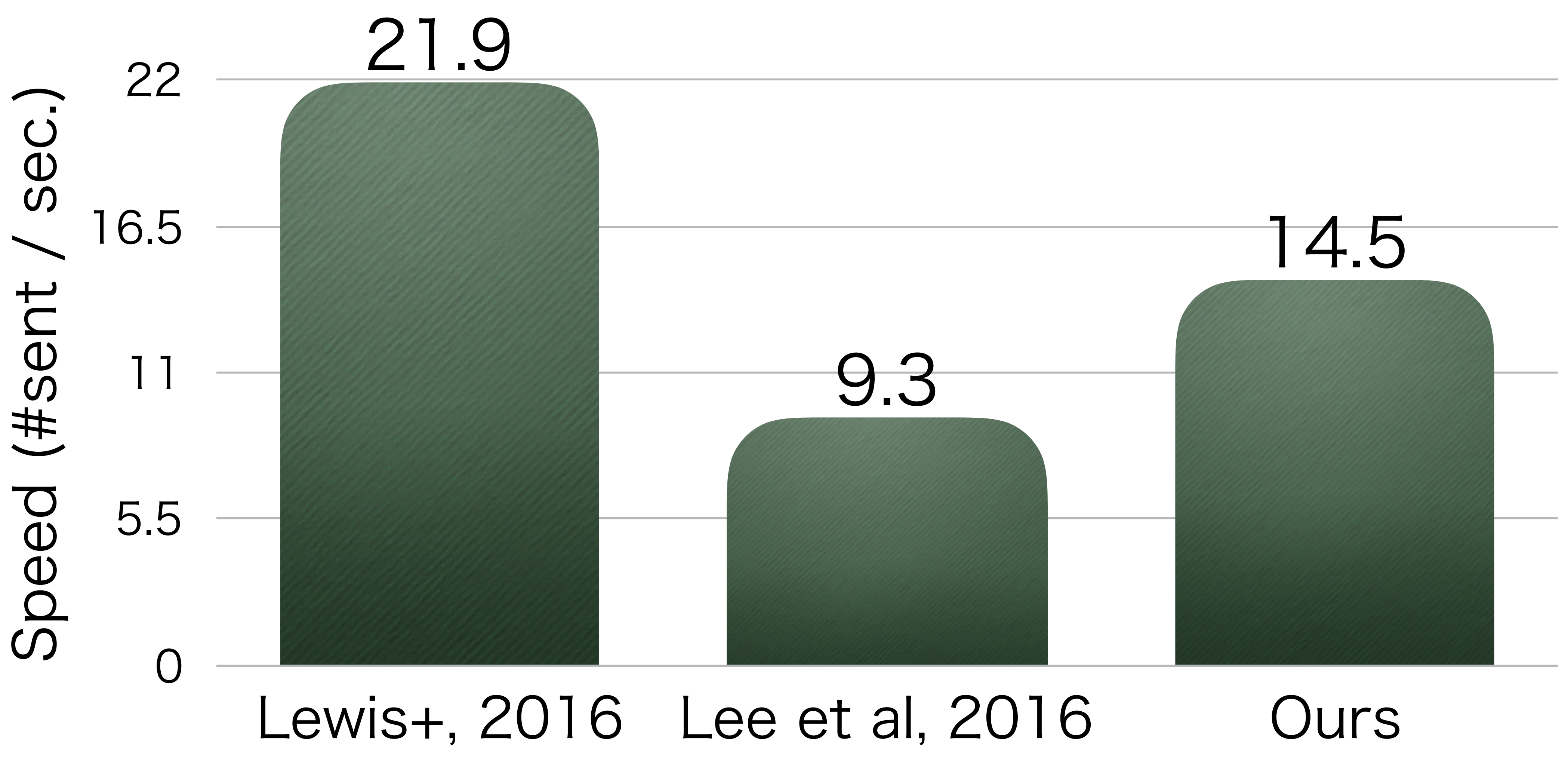
Node	f
(1,1)	0.1
(2,0)	0.1
(0,1)	0.1
(0,2)	0.9
(3,0)	0.99

PriorityQueue(f)



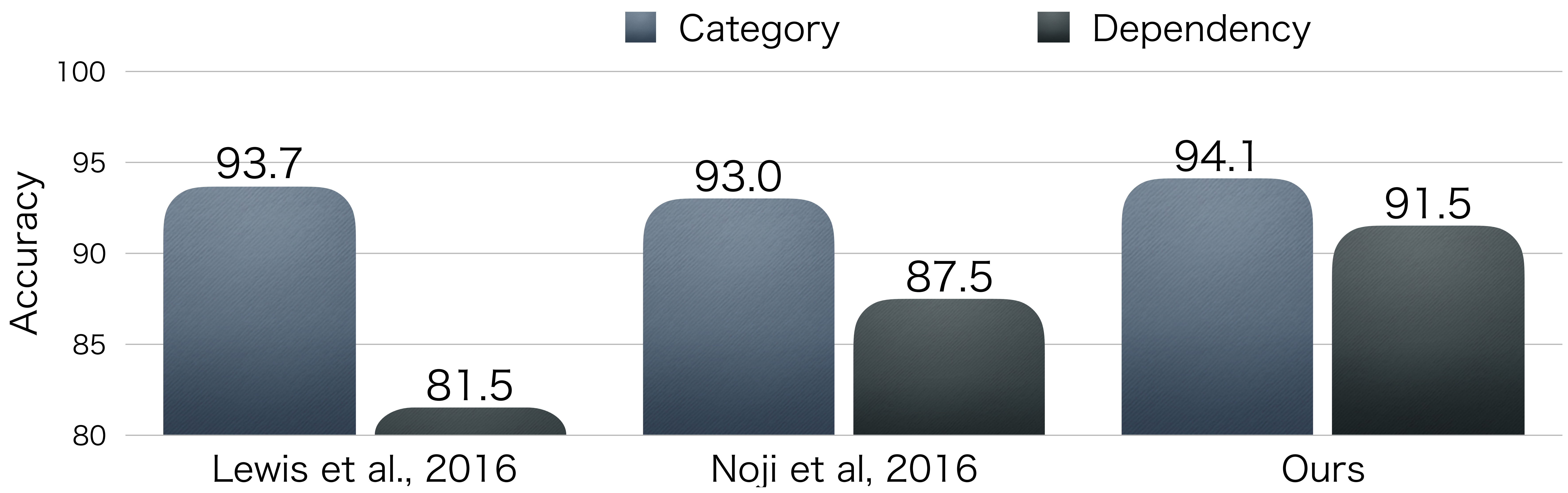
Experiments on English CCGbank





- English CCGbank (Hockenmeier and Steedman, 2007)
  - the same set of sentences as WSJ
- Accuracy: the proposed method achieved the best score
- Speed: it is more efficient than the powerful TreeLSTM-based method

Experiments on Japanese CCGbank



- Japanese CCGbank (Uematsu et al., 2013)
  - the same set as Kyoto University Text Corpus (Mainichi newspaper)
- (Noji et al., 2016): Shift-reduce CCG parser with a linear model
- For Japanese language, modeling the level higher than per-terminal is crucial

Summary so far

- I introduced CCG and my previous work on its parsing algorithm
- CCG provides elegant explanations for linguistic phenomena for various languages
- I proposed an efficient CCG parsing model, utilizing dependencies within a CCG tree
    - The proposed method is especially effective for the Japanese language
- Next, I'd like to talk about an inference system based on CCG, for solving Recognizing Textual Inference task

```
➜ ~ pip install allennlp depccg
➜ ~ allennlp train --include-package depccg.models.my_allennlp -s results supertagger.jsonnet
➜ ~ echo "CCG parsing is fun" | depccg_en --model results/model.tar.gz --format deriv --silent
1..
ID=1, log probability=-0.2000395804643631
 CCG  parsing        is         fun
 N/N     N     (S[dcl]\NP)/NP   N
-------------->
      N
--------------<un>
```

```
$ pip install depccg
$ depccg_en download
```

Part Two:
Combining Axiom Injection and Knowledge Base Completion for Efficient Natural Language Inference

Masashi Yoshikawa[◆], Koji Mineshima[★], Hiroshi Noji[♥], Daisuke Bekki[★]

◆ Nara Institute of Science and Technology

★ Ochanomizu University

♥ Artificial Intelligence Research Center, AIST

*presented at AAAI-33

NAIST®

Ochanomizu University

National Institute of Advanced Industrial Science and Technology AIST

Recognizing Textual Entailment

a.k.a. Natural Language Inference

Premise(s)

P1: Clients at the demonstration were all impressed by the system's performance.

P2: Smith was a client at the demonstration.

Hypothesis

H: Smith was impressed by the system's performance.

{entailment, contradiction, unknown}

- A testbed to evaluate if a machine can reason as we do
  - lexical, logical, syntactic phenomena, etc.
- Elemental technology for improving other NLP tasks
  - Question answering, reading comprehension, etc.

ccg2lambda (Mineshima et al., 2015)

Premise (P)
& Hypothesis (H)

Syntactic Parsing

CCG Derivations

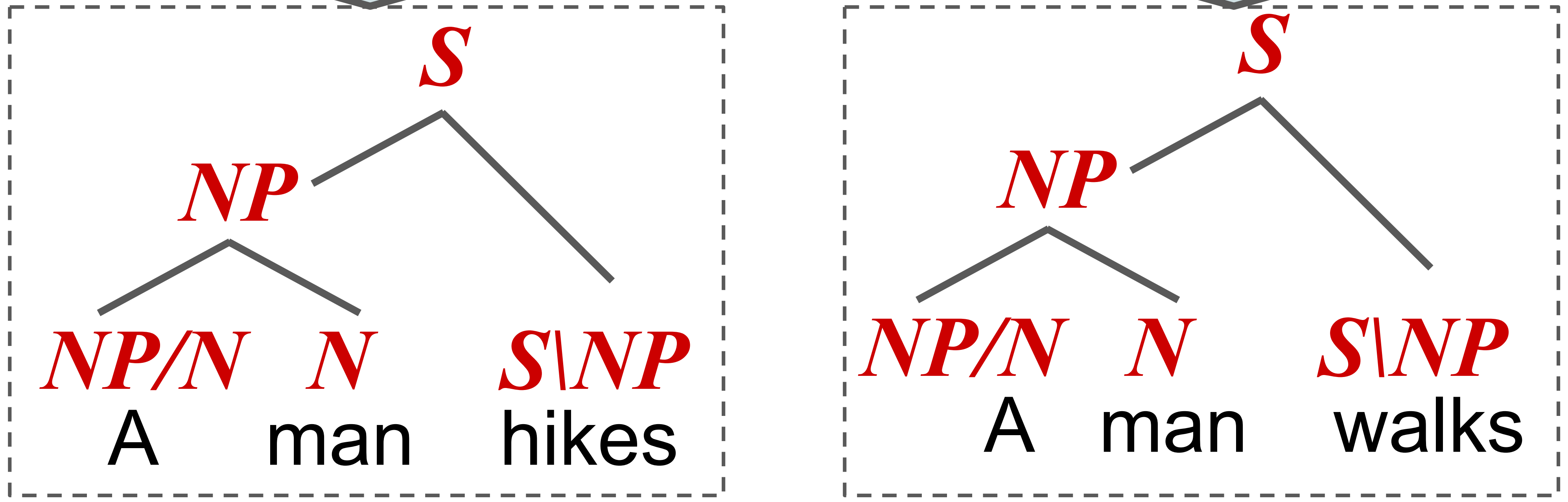
Semantic Parsing

Logical Formulas

Theorem Proving

{ yes, no, unknown }

P: A man hikes.　　H: A man walks.



$$\exists x.\mathtt{man}(x) \wedge \exists e.\mathtt{hike}(e) \wedge \mathtt{subj}(e,x)$$

$$\exists x.\mathtt{man}(x) \wedge \exists e.\mathtt{walk}(e) \wedge \mathtt{subj}(e,x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

result: unknown

Coq
theorem
prover

ccg2lambda (Mineshima et al., 2015)

Premise (P)
& Hypothesis (H)

Syntactic Parsing

CCG Derivations

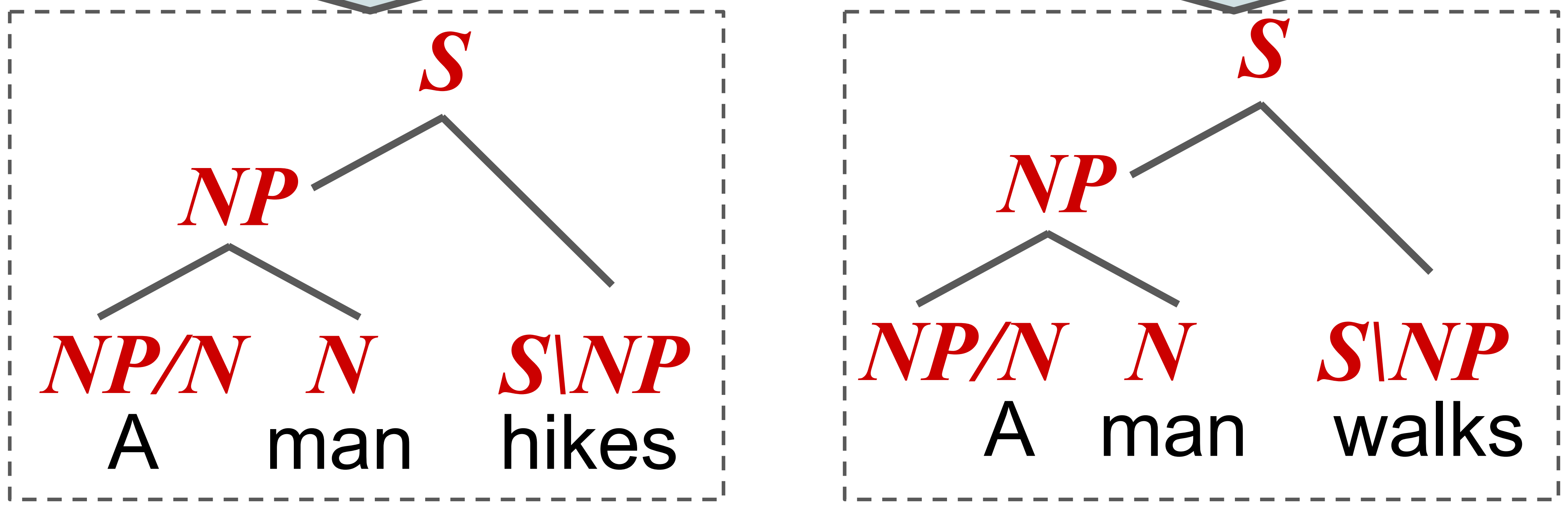
Semantic Parsing

Logical Formulas

Theorem Proving

{ yes, no, unknown }

P: A man hikes.　　H: A man walks.

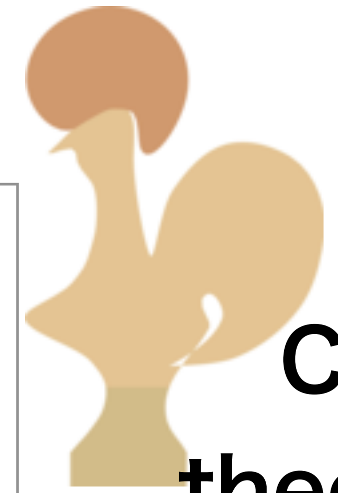


$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{hike}(e) \wedge \mathbf{subj}(e,x)$$

$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{walk}(e) \wedge \mathbf{subj}(e,x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

result: unknown



Coq theorem prover

👍 Unsupervised

👍 Captures linguistic phenomena
- 83.6 % accuracy in SICK

ccg2lambda (Mineshima et al., 2015)

Premise (P) & Hypothesis (H)

↓

Syntactic Parsing

↓

CCG Derivations

↓

Semantic Parsing

↓

Logical Formulas

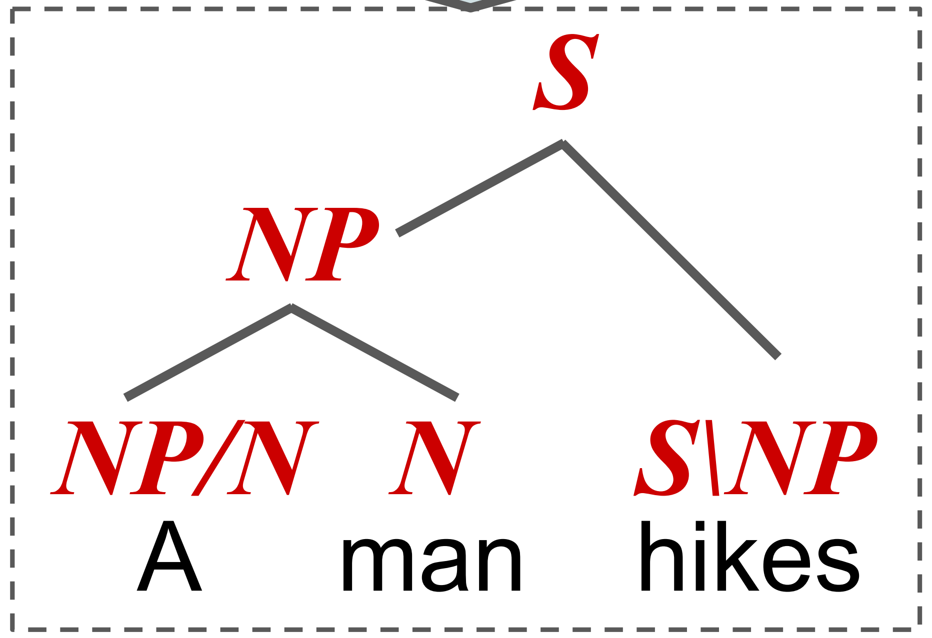
↓

Theorem Proving

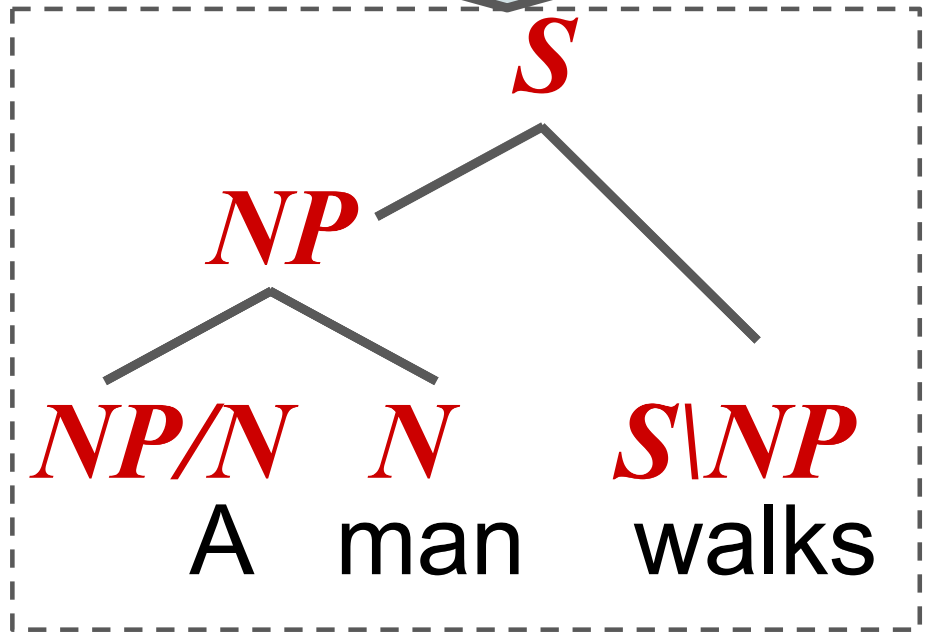
↓

{ yes, no, unknown }

P: A man hikes.



H: A man walks.



$$\exists x.\,\mathbf{man}(x) \wedge \exists e.\,\mathbf{hike}(e) \wedge \mathbf{subj}(e, x)$$

$$\exists x.\,\mathbf{man}(x) \wedge \exists e.\,\mathbf{walk}(e) \wedge \mathbf{subj}(e, x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

Coq theorem prover

result: unknown

👍 Unsupervised

👍 Captures linguistic phenomena
- 83.6 % accuracy in SICK

🤔 How to handle external knowledge?
e.g. $\forall x\,.\,\mathbf{hike}(x) \rightarrow \mathbf{walk}(x)$
- Use WordNet as axioms blows up the search space of theorem proving!

"Abduction" mechanism (Martínez-Gómez et al., 2017)

Premise (P)
& Hypothesis (H)

Syntactic Parsing

CCG Derivations

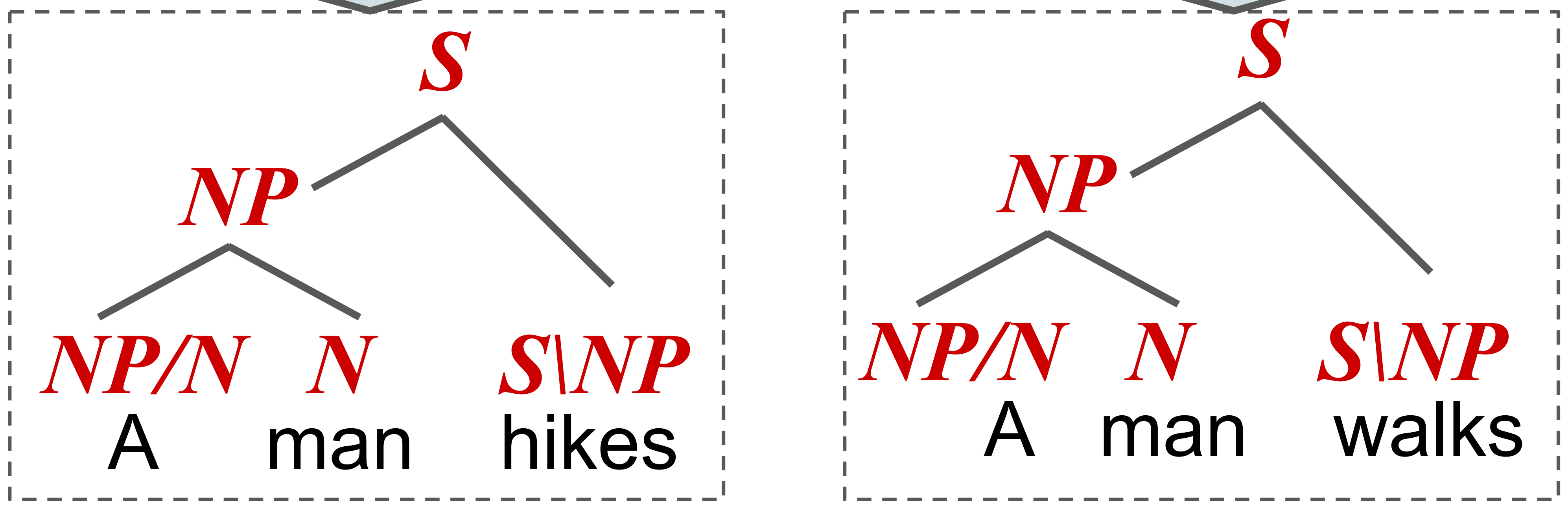
Semantic Parsing

Logical Formulas

Theorem Proving

{ yes, no, unknown }

P: A man hikes. H: A man walks.



$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{hike}(e) \wedge \mathbf{subj}(e,x)$$

$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{walk}(e) \wedge \mathbf{subj}(e,x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

result: unknown

Coq theorem prover

"Abduction" mechanism (Martínez-Gómez et al., 2017)

$\exists x.\,\mathbf{man}(x) \wedge \exists e.\,\mathbf{walk}(e) \wedge \mathbf{subj}(e, x)$

Logical Formulas

Theorem Proving

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

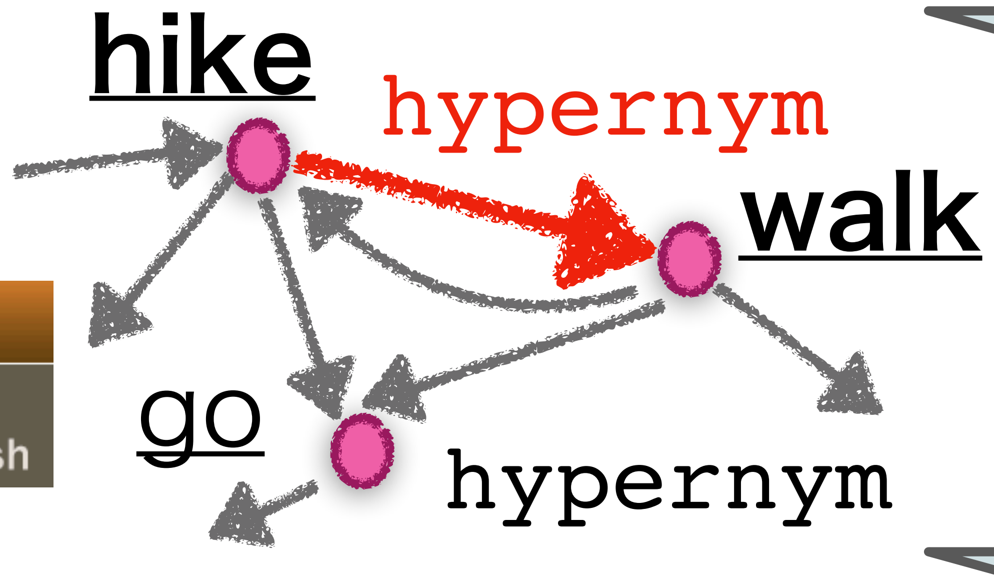
Coq theorem prover

{ yes, no, unknown }

result: unknown

Search on KBs

PRINCETON UNIVERSITY
WordNet
A lexical database for English



$\forall e.\,\mathbf{hike}(e) \rightarrow \mathbf{walk}(e)$

New Axioms

Theorem Proving

```
Coq < Axiom ax1: forall x: Event, hike e -> walk e.
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

Coq theorem prover

{ yes, no, unknown }

result: yes

More steps when the 1st theorem proving is unsuccessful

1. Search KBs (e.g. WordNet) for useful lexical relations
2. Rerun Coq with additional axioms

"Abduction" mechanism (Martínez-Gómez et al., 2017)

- Promising approach to handling external knowledge within a logic-based system

"Abduction" mechanism (Martínez-Gómez et al., 2017)

- Promising approach to handling external knowledge within a logic-based system
- (However,) **Practical issues**:
  - We want to **add more knowledge** to increase the coverage of reasoning
  - We want the **KBs to be compact** for efficient inference & memory usage

"Abduction" mechanism (Martínez-Gómez et al., 2017)

- Promising approach to handling external knowledge within a logic-based system
- (However,) **Practical issues**:
  - We want to **add more knowledge** to increase the coverage of reasoning
  - We want the **KBs to be compact** for efficient inference & memory usage
- Do not want to run Coq again and again for real applications 😣
  - Ideally, the mechanism should be tightly integrated with the inference for effciency

Freebase™

"Abduction" mechanism (Martínez-Gómez et al., 2017)

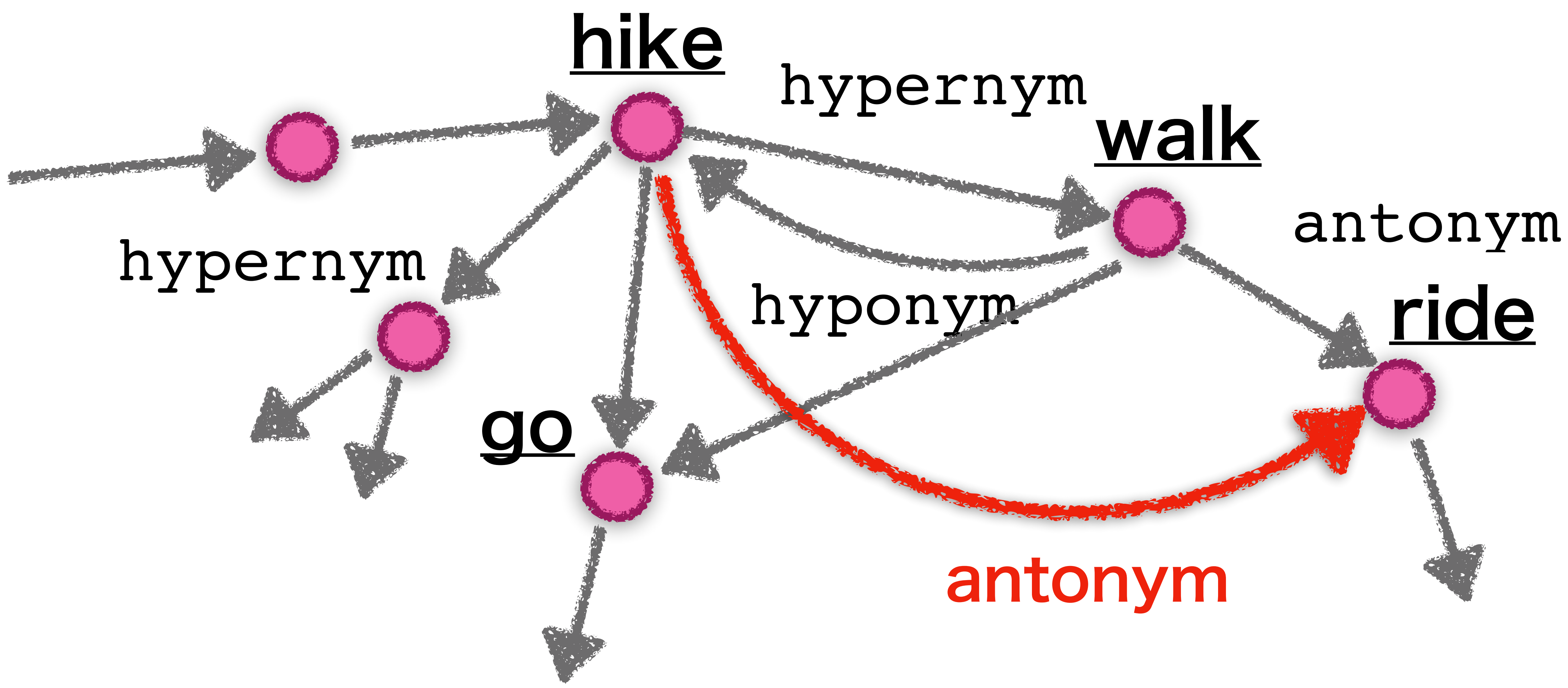
- Promising approach to handling external knowledge within a logic-based system
- (However,) **Practical issues**:
  - We want to **add more knowledge** to increase the coverage of reasoning
  - We want the **KBs to be compact** for efficient inference & memory usage
- Do not want to run Coq again and again for real applications 😣
  - Ideally, the mechanism should be tightly integrated with the inference for effciency

👉 **We solve these issues by:**

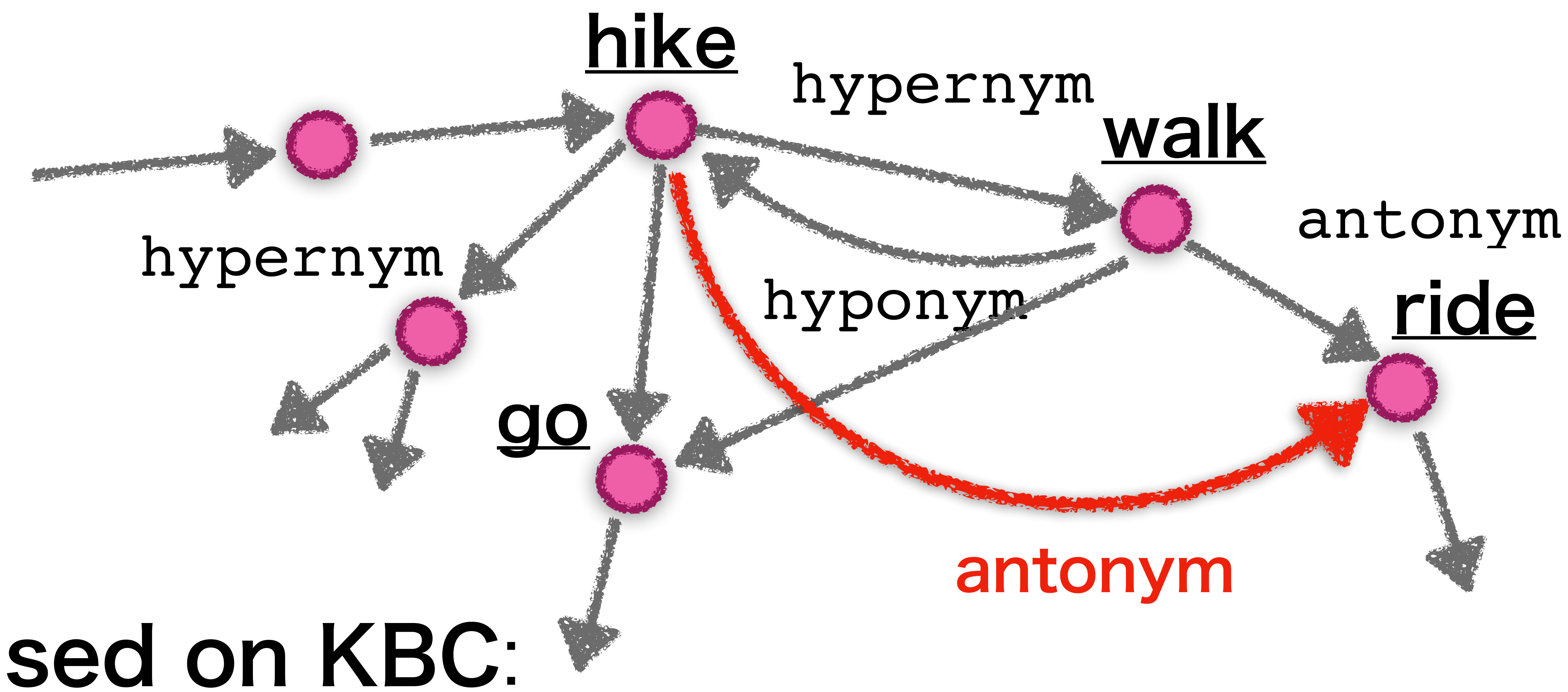
1. **Replacing search on KBs by techniques of "Knowledge Base Completion"**
2. **Developing "`abduction`" Coq plugin**

1. Extending Abduction Mechanism with KBC

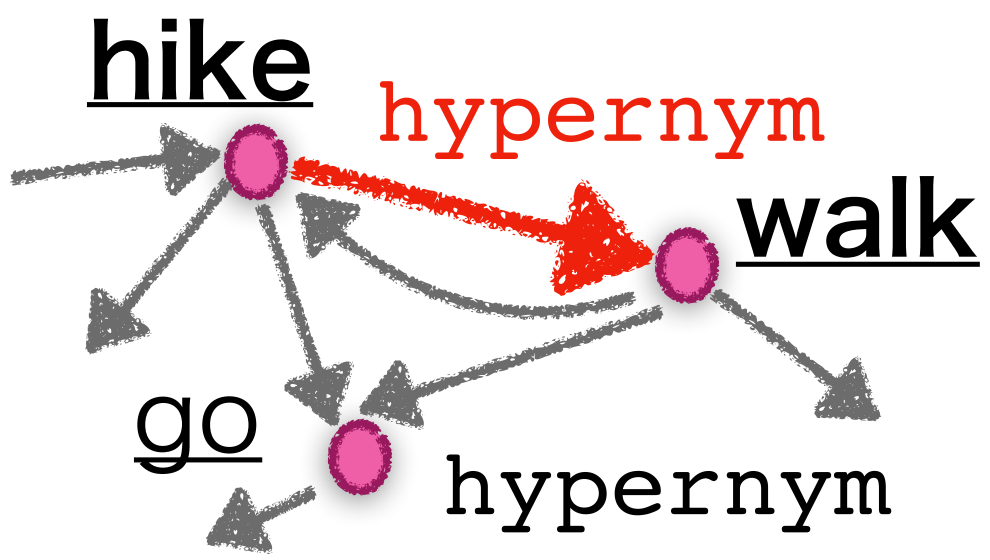
- **Knowledge Base Completion**:
  - A task to complement missing relations
  - recent huge advancement

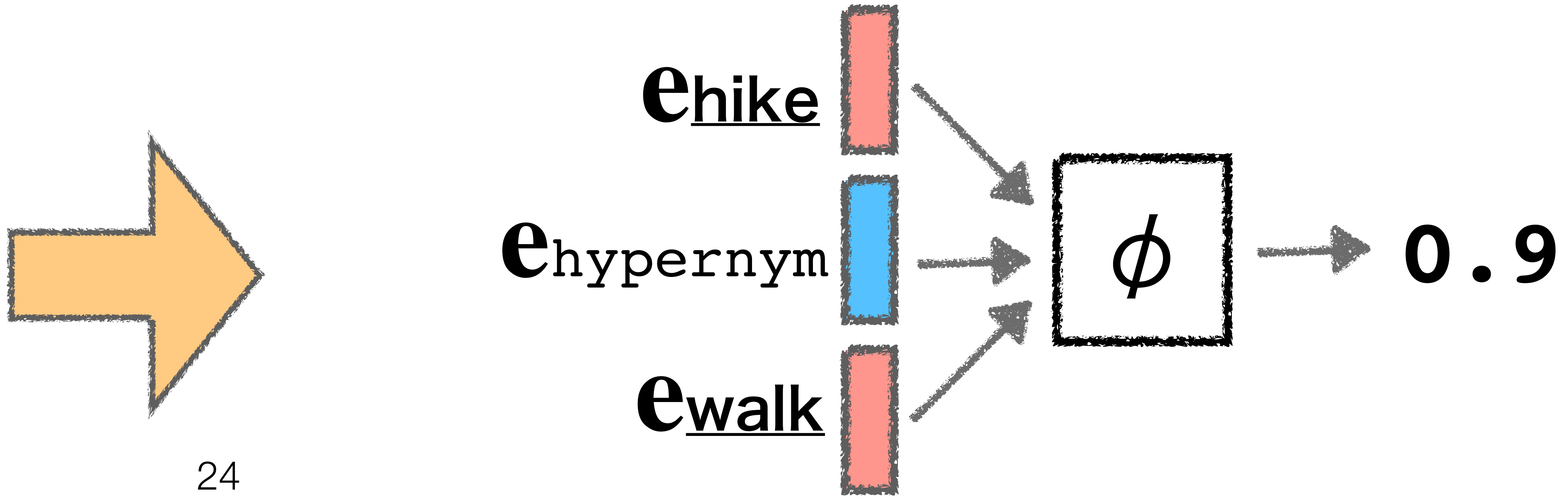


1. Extending Abduction Mechanism with KBC



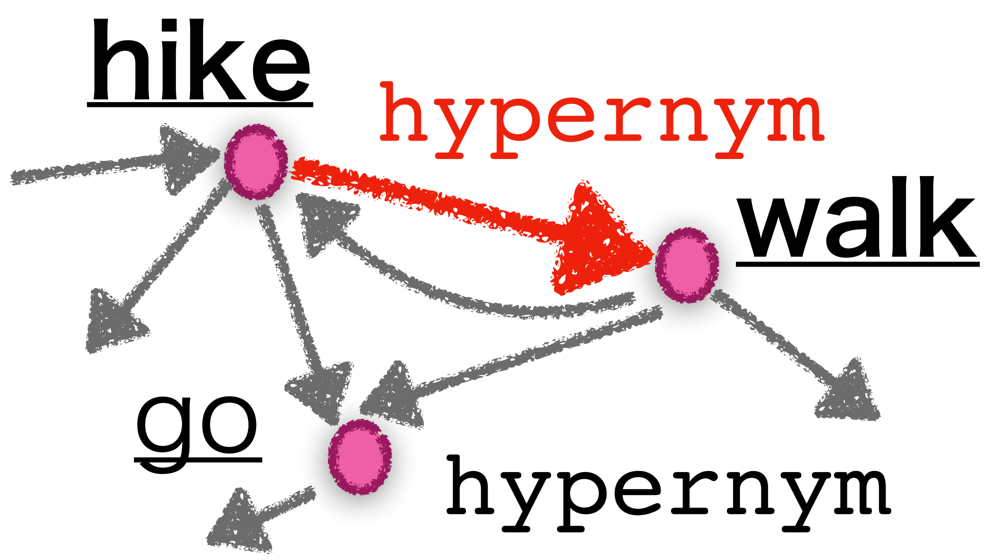
- **Knowledge Base Completion**:
  - A task to complement missing relations
  - recent huge advancement
- We propose an **abduction mechanism based on KBC**:
  - If (s, r, o) is missing, use it as axiom if $\phi(s, r, o) \geq \delta$ (threshold)
  - ComplEx (Trouillon et al., 2016): $\phi(s, r, o) = \sigma(Re(\langle \mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o \rangle)), \forall \mathbf{e}_v \in \mathbb{C}^n$

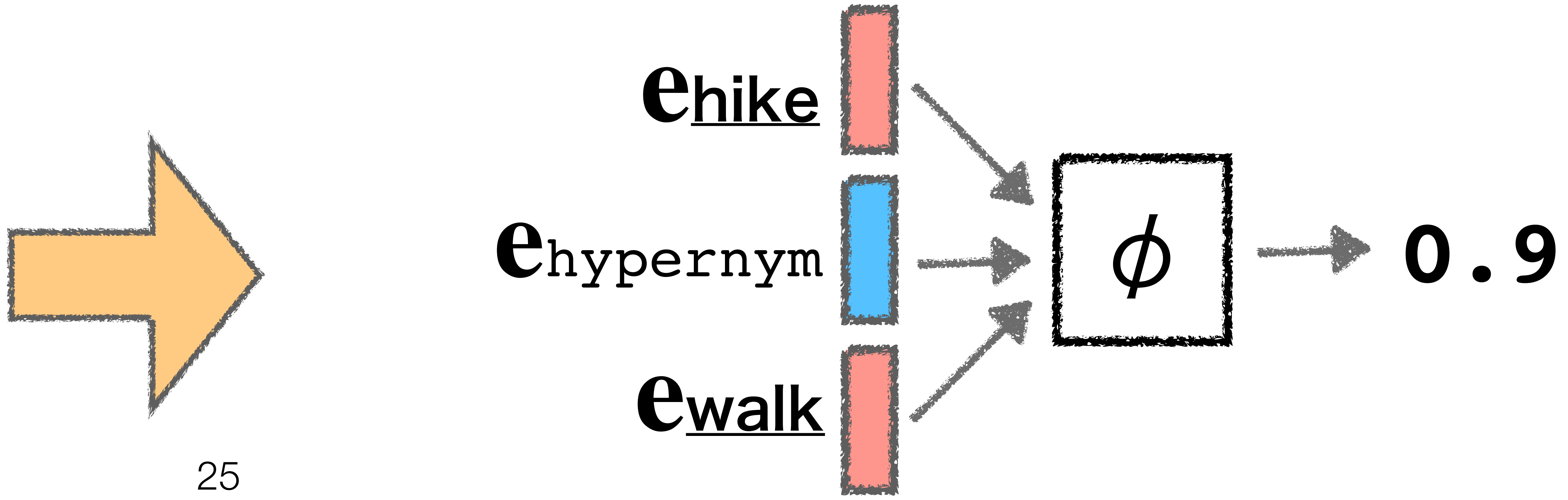




1. Extending Abduction Mechanism with KBC

	Search on KB	KBC
Latent Knowledge	Hand-crafted rules (e.g. transitive closure of `hypernym`)	KBC models learn accurately
Efficiency	Multi-hop reasoning takes time	One dot product (ComplEx)
Scalability	Adding more knowledge harms the search time	Knowledge from VerbOcean (Chklovski et al., 2004) are added for free





2. Faster Reasoning with "**abduction**" Coq plugin

Coq Interactive Session

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 ============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x)
```

2. Faster Reasoning with "**abduction**" Coq plugin

Coq Interactive Session

Lexical gap!

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 ============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x)
```

2. Faster Reasoning with "abduction" Coq plugin

Coq Interactive Session

Lexical gap!

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 ============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x)

t < abduction.
```

2. Faster Reasoning with "abduction" Coq plugin

Coq Interactive Session

Lexical gap!

(man, walk)
(man, hike)
(hike, walk)

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 ===============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x)

t < abduction.
```

2. Faster Reasoning with "abduction" Coq plugin

Construct a list of predicate pairs from context and goal

Coq Interactive Sessio

Lexical gap!

(man, walk)
(man, hike)
(hike, walk)

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 ============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x)

t < abduction.
```

2. Faster Reasoning with "abduction" Coq plugin

Coq Interactive Sessio

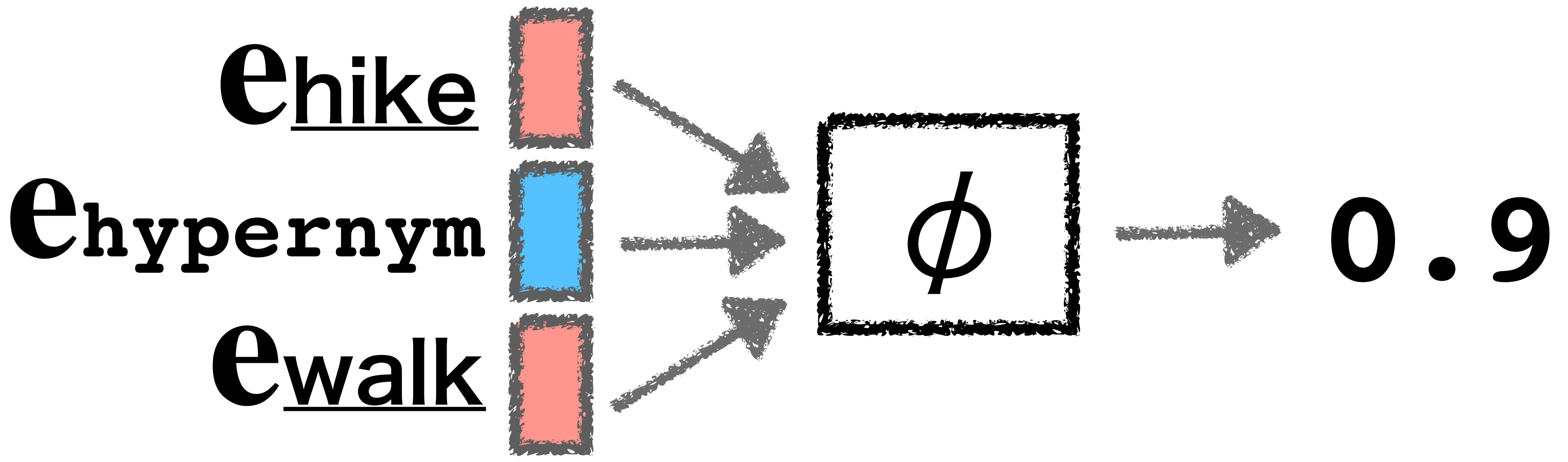
```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj
 ================================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x

t < abduction.
```

Lexical gap!

(man, walk)
(man, hike)
(hike, walk)

Construct a list of predicate pairs from context and goal

Evaluate all the predicate pairs using ComplEx



Filter them by score

2. Faster Reasoning with "**abduction**" Coq plugin

Coq Interactive Sessio

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj
 ================================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x

t < abduction.
```

Lexical gap!

(man, walk)
(man, hike)
(hike, walk)

Construct a list of predicate pairs from context and goal

Evaluate all the predicate pairs using ComplEx

$\mathbf{e}_{\text{hike}}$ $\mathbf{e}_{\text{hypernym}}$ $\mathbf{e}_{\text{walk}}$ → ϕ → 0.9

Filter them by score

Add them as axioms

(hike, hypernym, walk)

$$\forall x \,.\, \mathbf{hike}(x) \rightarrow \mathbf{walk}(x)$$

2. Faster Reasoning with "abduction" Coq plugin

Coq Interactive Sessio

Lexical gap!

(man, walk)
(man, hike)
(hike, walk)

```
1 subgoal

 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj
 ==============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x

t < abduction.
1 subgoal
 H : exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)
 NLax1 : forall x : Event, hike x -> walk x
 ==============================
 exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x
```

(hike, hypernym, walk)

$$\forall x\,.\,\textbf{hike}(x) \rightarrow \textbf{walk}(x)$$

Construct a list of predicate pairs from context and goal

↓

Evaluate all the predicate pairs using ComplEx

$\mathbf{e}_{\text{hike}}$, $\mathbf{e}_{\text{hypernym}}$, $\mathbf{e}_{\text{walk}}$ → ϕ → 0.9

↓

Filter them by score

↓

Add them as axioms

Summary so far…

Syntactic Parsing

CCG Derivations

NP

NP/N N $S\backslash NP$

A man hikes

NP

NP/N N $S\backslash NP$

A man walks

Semantic Parsing

Logical Formulas

Theorem Proving

{ yes, no, unknown }

$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{hike}(e) \wedge \mathbf{subj}(e, x)$$

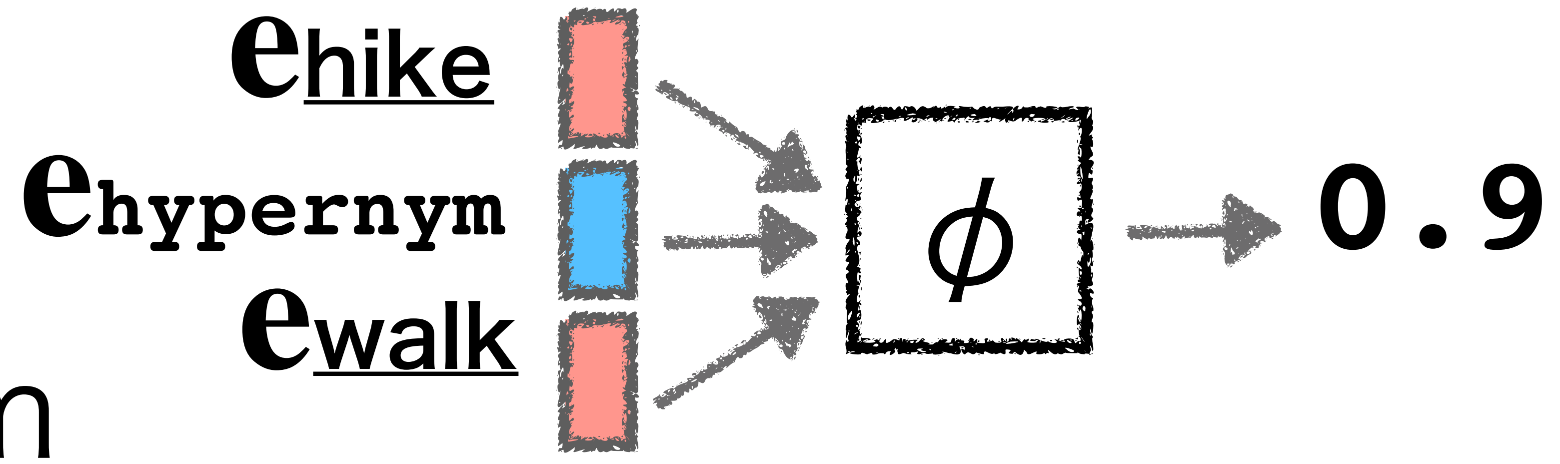
$$\exists x.\mathbf{man}(x) \wedge \exists e.\mathbf{walk}(e) \wedge \mathbf{subj}(e, x)$$

```
Coq < Theorem t1:
(exists x : Entity, man x /\ (exists e : Event, hike e /\ subj e x)) ->
  exists x : Entity, man x /\ (exists e : Event, walk e /\ subj e x).
Coq < Proof. ccg2lambda. Qed.
```

Coq

+abduction

result: yes



- 👍 Efficient and scalable abduction mechanism
- 👍 No need to rerun Coq in abduction
- Our method is applicable to other logic-based systems
  - e.g. Modern Type Theory (Bernandy and Chatzikyriakidis, 2017)

Experiments

- SICK RTE dataset (Marelli et al., 2014)
- Evaluation metrices: accuracy and processing time
- ComplEx is trained on logistic loss: $\sum_{((s,r,o),t)\in\mathcal{D}} t \log f(s,r,o) + (1-t)\log(1-f(s,r,o))$
- The training data is constructed using WordNet
  - synonym, antonym, hyponym, hypernyms, etc.
  - The trained ComplEx model achieves MRR of 77.68%

P: A flute is being played in a lovely way by a girl.

H: One woman is playing a flute.

logical

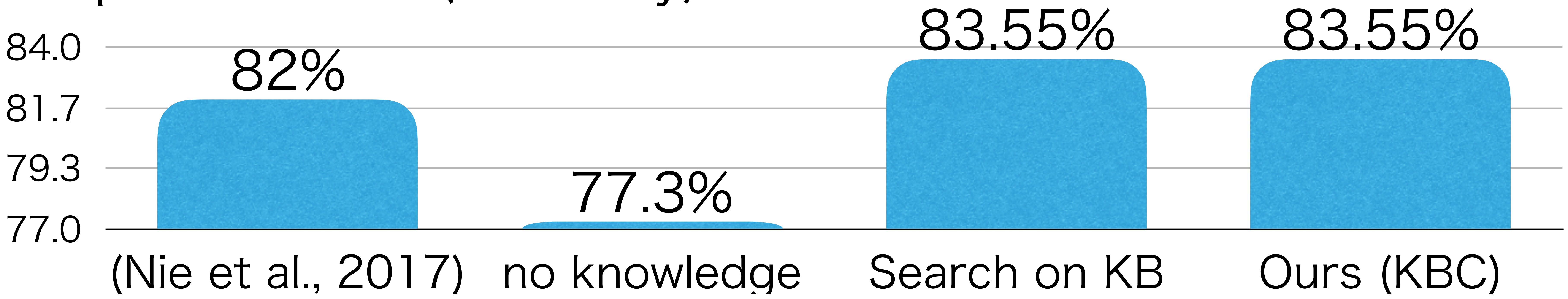
lexical phenomena

syntactic

entailment

Experimental Results on SICK

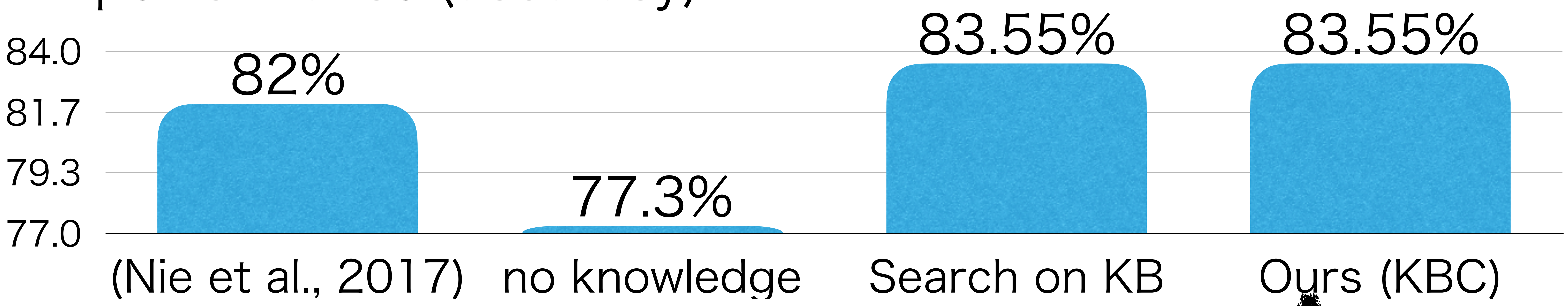
- RTE performance (accuracy)



- Baselines: Search on KB (Martínez-Gómez et al., 2017), NN-based (Nie et al., 2017)

Experimental Results on SICK

- RTE performance (accuracy)

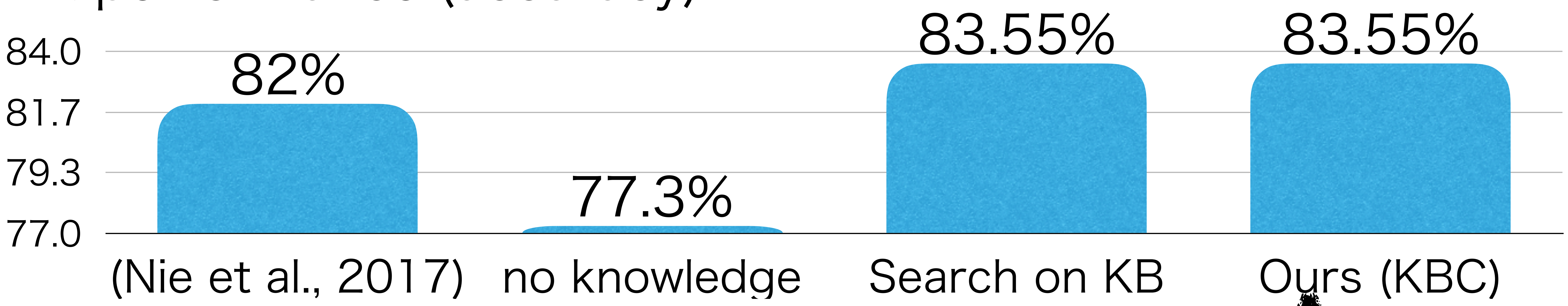


Achieves the same accuracy, improving significantly from "no knowledge" case

- Baselines: Search on KB (Martínez-Gómez et al., 2017), NN-based (Nie et al., 2017)

Experimental Results on SICK

- RTE performance (accuracy)



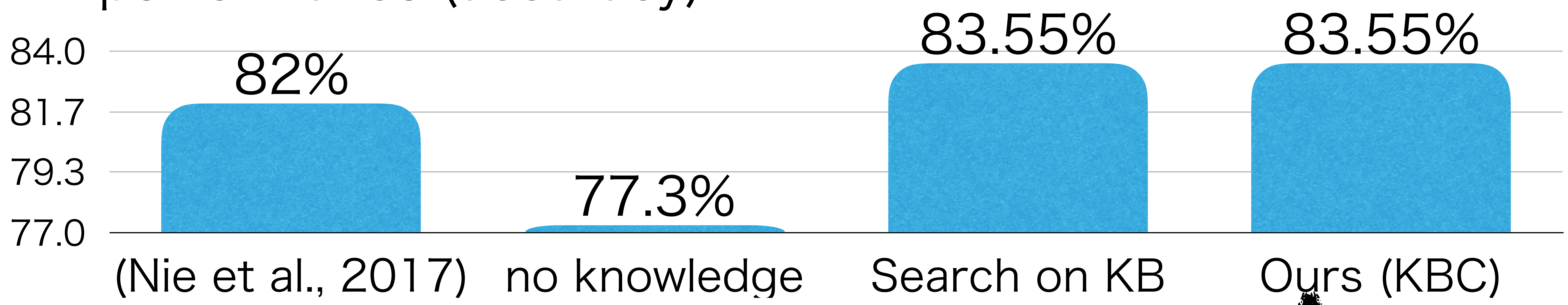
Achieves the same accuracy, improving significantly from "no knowledge" case

- Processing speed (second per a problem)

- Baselines: Search on KB (Martínez-Gómez et al., 2017), NN-based (Nie et al., 2017)

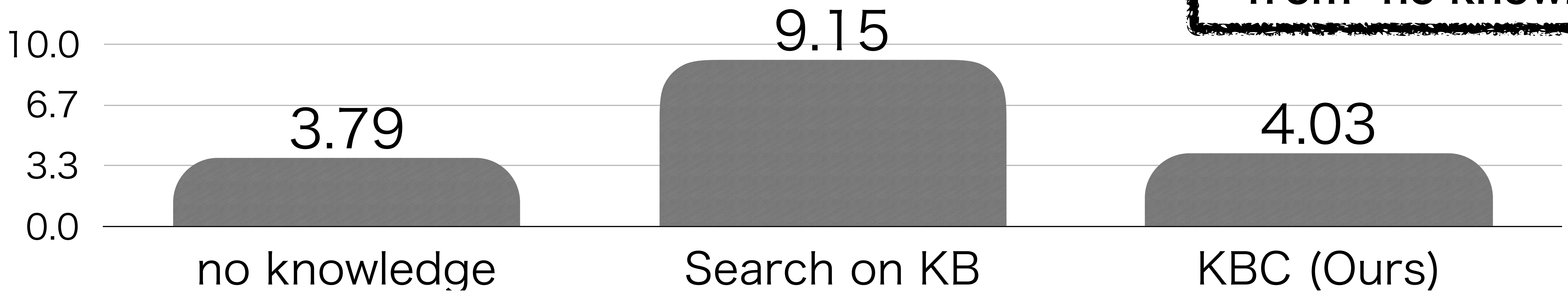
Experimental Results on SICK

- RTE performance (accuracy)



Achieves the same accuracy, improving significantly from "no knowledge" case

- Processing speed (second per a problem)



Our method halves the time to process an RTE problem!

- Baselines: Search on KB (Martínez-Gómez et al., 2017), NN

Summary of Part Two

- A KBC-based axiom injection for logic-based RTE systems
    - Efficient, scalable, and it provides latent knowledge
- `abduction` tactic for further faster reasoning
- **Other topics**:
    - Adding other KB (VerbOcean) without losing efficiency
    - Evaluating learned latent knowledge in terms of RTE (LexSICK dataset)
- **All the codes, dataset and slides are available**:
    - https://github.com/masashi-y/abduction_kbc

The performance of ccg2lambda on various datasets

- SICK (Marelli et al., 2014): Accuracy 82,3%

```
P: A flute is being played in a lovely way by a girl.
H: One woman is playing a flute
```

passive voice, quantifier
lexical semantics

- FraCaS (Cooper et al., 1992): Accuracy 69%

```
P: Smith believed that ITEL had won the contract in 1992.
H: ITEL won the contract in 1992.
```

Quantifier, Plurals,
Adjectives, Comparatives,
Verbs, Attitudes

```
P: ITEL won more orders than APCOM did.
H: APCOM won some orders.
```

(Haruta et al., 2019)
Adjectives (22 problems): 100%
Comparatives (31): 94%

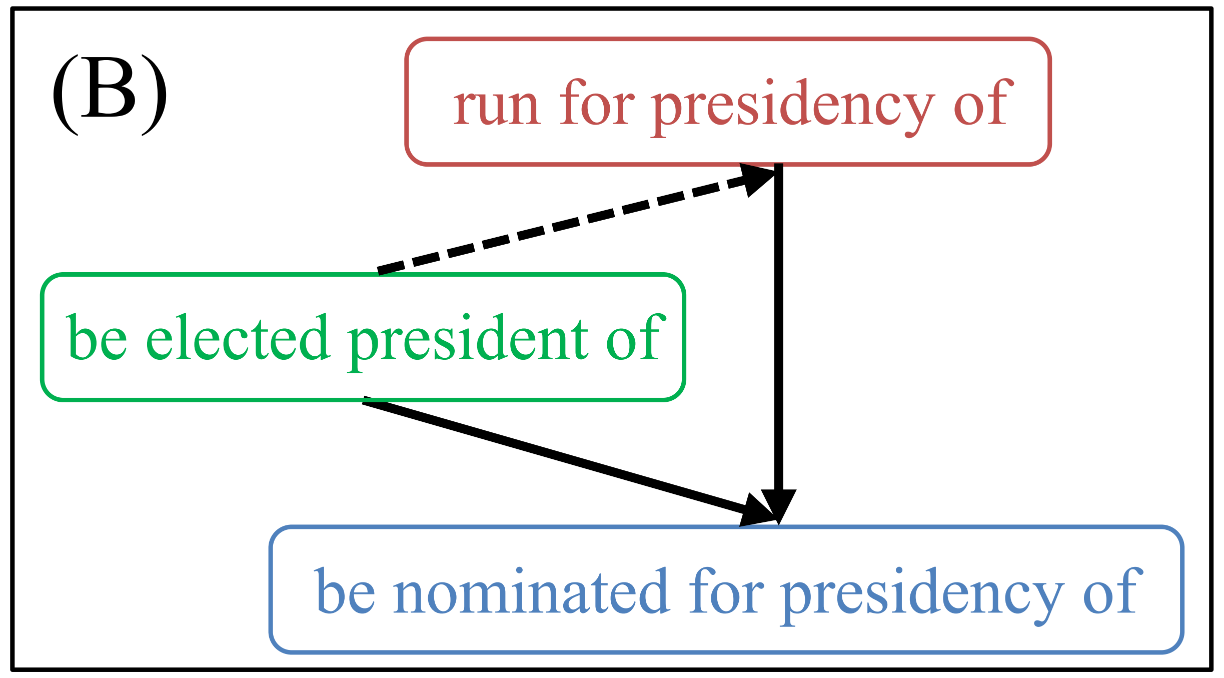
- SNLI (Bowman et al., 2015): No result

```
P: A black race car starts up in front of a crowd of people
H: A man is driving down a lonely road.
```

"a crowd" relates to "lonely",
"car starts up" relates to "driving",

Summary

- A CCG-based system has some advantages in handling complex linguistic phenomena
  - They reside in the long tail of distribution, and have been the focus of linguistics
  - It is unlikely that a neural method understands passive voice, though it achieves the similar accuracy on SICK using 5,000 sents …
- Difficulties at handling similarities between phrases, which is much easier for neural methods
- Some promising approaches:
  - Learning Entailment Graph (e.g., Hosseini et al., 2018, 2019)
  - Vector-based Semantics (e.g., Wijnholds and Sadrzadeh, 2018)



$$(\overrightarrow{John} \otimes \overrightarrow{subj}) \otimes \overrightarrow{likes} \otimes (\overrightarrow{Mary} \otimes \overrightarrow{obj})$$

Hosseini et al., Learning Typed Entailment Graphs with Global Soft Constraints, TACL 2018
Hosseini et al., Duality of Link Prediction and Entailment Graph Induction, ACL 2019
Wijnholds and Sadrzadeh, Evaluating Composition Models for Verb Elliptic Sentence Embeddings, NAACL 2019